ALIBABA CLOUD

Alibaba Cloud

Apsara Stack Enterprise

User Guide - Analytics and Artificial Intelligence

Product Version: V3.15.0

Document Version: 20220210

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 amounts of data. MaxCompute provides channels for data uploads and downloads, a range of computing and analysis features including MaxCompute SQL and MaxCompute MapReduce, and comprehensive security solutions.

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

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

MaxCompute aims to provide easy analysis and processing of large amounts of data. You can analyze big data without a deep knowledge of distributed computing. MaxCompute is widely implemented within Alibaba Group for multiple scenarios. The scenarios include 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 channels
  - Tunnel: provides highly concurrent uploads and downloads of offline data. MaxCompute Tunnel enables you to upload or download large amounts of data to or from MaxCompute. MaxCompute Tunnel provides a Java API.
  - DataHub: provides real-time data uploads and downloads. Data uploaded by using DataHub is available immediately, while data uploaded by using 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 capable of processing terabytes and 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 seamlessly migrated 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 a deep knowledge 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 you use MapReduce. MapReduce provides a Java API.
  - 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. MaxCompute Graph iteratively edits and evolves graphs to obtain analysis results.

- Unstructured data access and processing in integrated computing scenarios: MaxCompute SQL cannot directly process external data, such as unstructured data from Object Storage Service (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 handle this issue.

  MaxCompute allows you to create external tables to process data from the following data sources:

  - Internal data sources: OSS, Tablestore, AnalyticDB, ApsaraDB RDS, Alibaba Cloud HDFS, and TDDL
  - External data sources: open source HDFS, MongoDB, and HBase

- Unstructured data access and processing in MaxCompute: MaxCompute reads and writes volumes to store and process unstructured data, which otherwise must be stored in an external storage system.

- Spark on MaxCompute: a big data analytics engine developed by Alibaba Cloud to provide big data processing capabilities for Alibaba Group, government agencies, and enterprises. For more information, see Spark on MaxCompute.
- SDK: a toolkit provided for developers.
- Security solution: MaxCompute provides powerful security features to ensure data security.

1.2. Usage notes

You can selectively read topics in this document based on your requirements. This topic provides reading recommendations based on your roles.

MaxCompute beginners

If you are a beginner in MaxCompute, we recommend that you first familiarize yourself with the following topics:

- What is MaxCompute: This topic provides the MaxCompute overview and describes its features. It helps you obtain a general knowledge of MaxCompute.
- Quick start: This topic describes how to download and configure the client, create a table, grant permissions, import data, and export data. It also describes how to run SQL jobs, user-defined functions (UDFs), and MapReduce programs.
- Terms and common statements: This topic introduces the basic terms of MaxCompute and commonly used statements in MaxCompute. It helps you familiarize yourself with operations on MaxCompute.
- Frequently used tools: This topic describes how to download, configure, and use the commonly used tools in MaxCompute before you analyze data.

Data analysts

If you are a data analyst, we recommend that you familiarize yourself with the following topics:

MaxCompute SQL: The topic describes how to query and analyze large amounts of data stored in MaxCompute.

MaxCompute SQL provides the following features:

- Allows you to use the CREATE, DROP, and ALERT DDL statements to manage tables and partitions.
- Allows you to execute the following DML statements:
  - You can use the SELECT statement to select data records in a table and the WHERE clause to query data records that meet specific conditions. These statements help you search for data records.
  - You can join two tables by using equi-joins.
  - You can use the GROUP BY clause to aggregate columns.
  - You can use the INSERT OVERWRITE or INSERT INTO statement to insert data records into another table.
  - You can use SELECT TRANSFORM to simplify the reference of script code.
- Allows you to use built-in functions and UDFs to complete a variety of computations.

- Allows you to use user-defined types (UDTs) to reference classes or objects of third-party programming languages in SQL statements and therefore to obtain data or call methods.
- Allows you to use UDJs to implement flexible cross-table and multi-table custom operations and reduce the detailed operations on the underlying distributed systems by using methods such as MapReduce.
- Allows you to collect statistics on tables and configure table lifecycles.
- Supports regular expressions.

Users with development experience

If you have development experience, understand the distributed architecture, and want to obtain data analytics capabilities that SQL cannot deliver, we recommend that you read the following advanced functional topics of MaxCompute:

- MaxCompute MapReduce: a MapReduce programming model for Java. You can use the Java API provided by MapReduce to write MapReduce programs and process MaxCompute data.
- MaxCompute Graph: a processing framework for iterative graph computing. A graph consists of vertices and edges, both of which contain values. MaxCompute Graph iteratively edits and evolves graphs to obtain analysis results.
- Eclipse plug-in: an integrated development environment (IDE) to help you complete development by using MapReduce, UDFs, and Graph SDK for Java.
- SDK for Java: a toolkit provided for developers.
- MaxCompute Tunnel: enables you to upload or download large amounts of data to or from MaxCompute at a time.

Project owners or administrators

If you are a project owner or an administrator, we recommend that you familiarize yourself with the following topics:

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

1.3. Preparations

1.3.1. Log on to the Apsara Uni-manager Management Console

This topic describes how to log on to the Apsara Uni-manager Management Console.

Prerequisites

- The URL of the Apsara Uni-manager Management Console is obtained from the deployment personnel before you log on to the Apsara Uni-manager Management Console.
- We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL of the Apsara Uni-manager Management Console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password that you can use to log on to the console from the operations administrator.

> **Note** When you log on to the Apsara Uni-manager Management Console for the first time, you must change the password of your username. Your password must meet complexity requirements. The password must be 10 to 32 characters in length and must contain at least two of the following character types:
>
> - Uppercase or lowercase letters
> - Digits
> - Special characters, which include ! @ # $ %

3. Click **Log On**.
4. If your account has multi-factor authentication (MFA) enabled, perform corresponding operations in the following scenarios:
   - It is the first time that you log on to the console after MFA is forcibly enabled by the administrator.
     a. On the Bind Virtual MFA Device page, bind an MFA device.
     b. Enter the account and password again as in Step 2 and click **Log On**.
     c. Enter a six-digit MFA verification code and click **Authenticate**.
   - You have enabled MFA and bound an MFA device.

     Enter a six-digit MFA authentication code and click **Authenticate**.

> **Note** For more information, see the *Bind a virtual MFA device to enable MFA* topic in *Apsara Uni-manager Operations Console User Guide*.

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 Uni-manager Management Console as an administrator.
2. In the top navigation bar, choose **Products > Big Data > MaxCompute** to go to the **MaxCompute** homepage.
3. In the left-side navigation pane of the page that appears, click **Task Accounts**. In the upper-right corner of the page that appears, click **Create Task Account**.
4. In the dialog box that appears, configure the parameters and click **OK**.

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 Uni-manager Management Console as an administrator.
2. In the top navigation bar, choose **Products > Big Data > MaxCompute** to go to the **MaxCompute** homepage.
3. In the left-side navigation pane, click **Projects**. In the upper-right corner of the page that appears, click **Create MaxCompute Project**.
4. On the page that appears, configure the parameters and click **Submit**.

   Configurations in the Region section

Parameter	Description
Organization	The organization that you select for the project.
Resource Set	The resource set of the selected organization.
Cross-domain Enhancement Pack	The region with which the project you create can associate.
VPC	The virtual private cloud (VPC) in the region. The VPC is used to isolate networks.
Cluster	The information about the cluster of the project.

   Configurations in the Security Enhancement Switch section

Parameter	Description
Enhancement Pack for Joint Computing	By default, this switch is turned off. You can turn on the switch to support joint computing features.
Encrypted	Specifies whether to encrypt data. Default value: No.
Encryption Method	By default, this parameter is not displayed. It is displayed when you set Encrypted to Yes. Valid values: AESCTR, AES256, RC4, and SM4.
Encryption Key	By default, this parameter is not displayed. It is displayed when you set Encrypted to Yes. Default value: Generate Automatically.

   Configurations in the Resource Configurations section

Parameter	Description
Quota Group	The quota group that corresponds to the cluster. If no groups are available, click **Create Quota Group** to create one. For more information about the configurations, see Create a quota group.
Storage	You need only to specify the Requested (GB) parameter. The Requested (GB) parameter specifies the storage space that is requested for the project. The value cannot be greater than the total capacity of the disk.

   Configurations in the Basic Settings section

Parameter	Description
Project Name	The name of the project you want to create.
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.
Task Account	Optional. The account of the task. If no accounts are available, click **Create Task Account** to create one. For more information about the creation process and detailed configurations, see Prepare an Apsara Stack tenant account.
Rule	The description of the project. This parameter can be empty.

1.3.4. Create a quota group

This topic describes how to create a quota group in MaxCompute.

Procedure

1. Log on to the Apsara Uni-manager Management Console as an administrator.
2. In the top navigation bar, choose **Products > Big Data > MaxCompute** to go to the **MaxCompute** homepage.
3. In the left-side navigation pane, click **Quota Groups**. In the upper-right corner of the page that appears, click **Create**.
4. On the page that appears, configure the parameters and click **Submit**.

   Configurations in the Region section

Parameter	Description
Organization	The organization that you select for the quota group.
Resource Set	The resource set in the selected organization.
Region	The region of the cluster where the quota group resides.
Cluster	The information about the cluster of the quota group.

   Configurations in the Basic Settings section

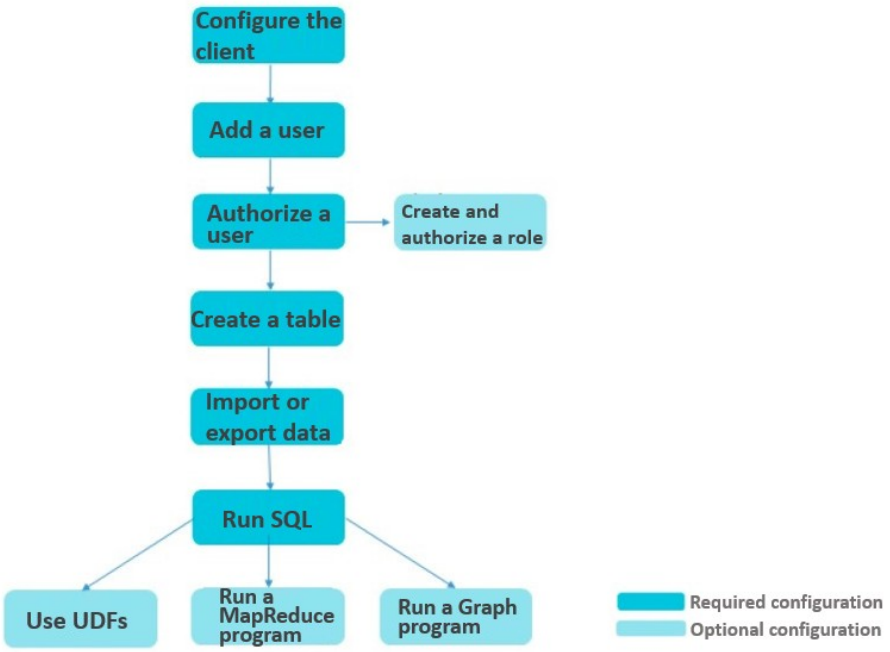
Parameter	Description
Quota Group Name	The name of the quota group that you want to create.
Sharing Scope	The sharing scope of the quota group. Valid values: Current Resource Set, Current Organization and Subordinate Organizations, and Current Organization.
Quantity	The number of resources to allocate to the current quota group. The value must be greater than 0.

1.4. Quick start

1.4.1. Overview

This topic describes the procedure to use MaxCompute. It provides you with step-by-step instructions on basic MaxCompute operations.

The following figure shows the procedure to use MaxCompute.



1. Configure the client.

   You must install and configure the client to access MaxCompute projects and use all the features of MaxCompute. For more information, see MaxCompute client.

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 perform operations on the project only after the user is authorized.

4. (Optional) Create and authorize a role.

   It is time-consuming to authorize each user in a project that 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 who assume this role are granted the same permissions.

5. Create a table.

   After a user is added to a project and authorized, the user can 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 write your own Java programs and import or export data.

7. Execute SQL statements.

   You can familiarize yourself with the limits on a few common SQL statements. For more information about how to execute SQL statements, see MaxCompute SQL.

8. Perform the following optional operations:
   - Use UDFs.

     After you install the MaxCompute client, you can try to use user-defined functions (UDFs). MaxCompute supports user-defined scalar functions, user-defined aggregate functions (UDAFs), and user-defined table-valued functions (UDTFs).
   - 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 download, configure, and run the client.

Prerequisites

JRE 1.8 is installed on your on-premises machine because the MaxCompute client is developed in Java. In addition, you have an Apsara Stack tenant account and have obtained AccessKey ID and AccessKey secret.

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

Procedure

1. Download the client package to your on-premises machine.
2. Decompress the package to the folder in which you want to store the client. The package contains the following folders:

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

3. Edit the odps_config.ini file in the conf folder. The following code shows the configuration information:

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

   - access_id and access_key are the AccessKey pair of the Apsara Stack tenant account. access_id corresponds to AccessKey ID and access_key corresponds to AccessKey secret.
   - 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.
4. After the modification, run the odpscmd file in Linux or the odpscmd.bat file in Windows in the bin directory to execute SQL statements and related commands. 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 a user

Except for a project owner, all users must be manually added to the project and granted permissions before they can perform operations on the project. This topic describes how to add users to or delete users from a project as a project owner.

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

Add a user

Syntax:

```
add user <full_username>;
```

Examples:

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

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

```
list users;
```

Note

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

Delete a user

Syntax:

```
remove user <full_username> ;
```

Examples:

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

> **Note**
>
> - Before you delete a user, make sure that you have revoked all the permissions from the user. If you delete a user before you revoke the permissions from the user, the permissions are retained. If the user is added to the project again, the user still has the permissions that were previously granted.
> - For more information about how to add or delete users, see User management.

1.4.4. Grant and view permissions

1.4.4.1. Overview

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

Authorization is a process of granting permissions to perform operations on objects, such as tables, tasks, and resources, in MaxCompute. The permissions include read, write, and view permissions.

The following topics are intended for project administrators. If you are a regular MaxCompute user, make sure that you have obtained the required permissions.

MaxCompute provides two authorization mechanisms. For more information, see ACL authorization and Policy authorization.

1.4.4.2. ACL-based authorization

This topic describes the statements for ACL-based authorization and provides examples.

ACL-based authorization in MaxCompute applies to the following objects: projects, tables, functions, resources, instances, and tasks. Each type of object requires different operation permissions. For more information, see ACL-based authorization actions.

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 |
           resource res_name
subject ::= user full_username | role role_name
```

Examples:

alice@aliyun.com is newly added to the test_project_a project. Allen is a RAM user of bob@aliyun.com. An Alibaba Cloud account can execute the following statements to grant permissions, including the permissions to submit jobs, create data tables, and query existing objects in a project:

```
-- Enter a project.
use test_project_a;
-- Add a member to the project.
add user aliyun$alice@aliyun.com;
-- Add a RAM user.
add user ram$bob@aliyun.com:Allen;
-- Create a role named worker.
create role worker;
-- Assign the worker role to the added members.
grant worker TO aliyun$alice@aliyun.com;
grant worker TO ram$bob@aliyun.com:Allen;
-- Grant the CREATE INSTANCE, CREATE RESOURCE, CREATE FUNCTION, CREATE TABLE, and LIST permissions t
o the worker role.
grant CreateInstance, CreateResource, CreateFunction, CreateTable, List ON PROJECT test_project_a TO
ROLE worker;
-- Grant all instance permissions to the worker role.
grant all on instance instance_name to Role worker;
```

1.4.4.3. Policy-based access control

This topic describes the commands that you can use for policy-based access control and provides examples on how to configure a policy.

Policy-based access control is subject-based authorization. For more information, see Authorization policies.

Commands:

```
GET POLICY;
-- Read the project policy.
PUT POLICY <policyFile>;
-- Specify or overwrite the project policy.
GET POLICY ON ROLE <roleName>;
-- Read the policy of a role in the project.
PUT POLICY <policyFile> ON ROLE <roleName>;
-- Specify or overwrite the policy of a role in the project.
```

Example on how to configure a policy:

```
{
"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 the user the LIST, READ, DESCRIBE, and SELECT permissions only on the project and table a in the project.

1.4.4.4. Query permissions

You can execute a statement to query user permissions in MaxCompute.

Execute the following statement to query the permissions of a user:

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

Note For more information, 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 grant users a specified set of permissions. After you authorize a role, all users who assume this role are granted the same permissions. This topic describes how to create and authorize a role.

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

Create a role

Syntax:

```
create role <roleName>;
```

Examples:

```
create role player;
```

Assign a role to a user

Syntax:

```
grant <roleName> to <full_username>;
```

Examples:

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

Delete a role

Syntax:

```
drop role <roleName>;
```

Examples:

```
drop role player;
```

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

Authorize a role

Role authorization is similar to user authorization. For more information about how to authorize roles, see User authorization. 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.

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]
-- You can also execute the following statement to create a table:
create table [if not exists] table_name
like existing_table_name
```

Examples:

```
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 that has a lifecycle specified.
create table test4 like test3;
-- Create a table named test4 that has the same attributes, such as the field type and partition type, as those of test3. The only difference between the two tables is the lifecycle.
create table test5 as select * from test2;
-- Create a table named test5. However, the partitioning and lifecycle attributes are not replicated to the destination table. Only data of test2 is replicated to test5.
```

You can specify partitions and 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 lifecycle settings, see Modify the lifecycle of a table.

1.4.6.2. Query table information

This topic describes how to query table information.

Syntax:

```
desc <table_name>;
```

Examples:

```
desc test3;
-- Query information about test3.
desc test4;
-- Query information about test4.
```

1.4.6.3. Delete a table

This topic describes how to delete a table.

Syntax:

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

Examples:

```
drop table test2;
```

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

1.4.7. Import or export data

This topic describes how to compile Java programs by using the MaxCompute Tunnel SDK and import data to or export data from MaxCompute.

For more information about data import and export, see Tunnel SDK example.

1.4.8. Run SQL

1.4.8.1. Overview

This topic describes the limits on the execution of SQL statements.

For more information about how to execute SQL statements, see MaxCompute SQL.

Take note of the following points when you use 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 directly 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

This topic describes limits of the SELECT statements.

The following limits apply to the SELECT statements:

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

```
select substr(col2, 2) from tbl group by substr(col2, 2);
-- This statement can be executed because the key of the GROUP BY clause is the expression compose
d of columns in the input table.
select col2 from tbl group by substr(col2, 2);
-- This statement cannot be executed because the key of the GROUP BY clause is not included in the
columns of the SELECT operation.
select substr(col2, 2) as c from tbl group by c;
-- This statement cannot be executed because the key of the GROUP BY clause is the alias of an out
put column from the SELECT operation.
```

> **Note** In most cases, the GROUP BY clause precedes the SELECT operation during the parsing of SQL statements. Therefore, GROUP BY uses only the column names in the input table or expressions of columns as keys.

- The DISTRIBUTE BY clause must be placed before the SORT BY clause.
- The key of the ORDER BY, SORT BY, or DISTRIBUTE BY clause must be the alias of an output column from the SELECT operation.

```
select col2 as c from tbl order by col2 limit 100;
-- This statement cannot be executed because the key of the ORDER BY clause is not the alias of an
output column from the SELECT operation.
select col2 from tbl order by col2 limit 100;
-- This statement can be executed. If an output column from the SELECT operation does not have an
alias, the column name is used as the alias.
```

> **Note** In most cases, 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 of these clauses.

For more information about the SELECT statements, see SELECT.

1.4.8.3. INSERT

This topic describes the limits of the INSERT statements.

The following limits apply to the INSERT statements:

- If you execute an INSERT statement to insert data into a partition, the partition key columns cannot be included in the SELECT clause.

```
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. The sale_date and region columns are partition key columns and cannot be
included in the INSERT statement that is used to insert data into static partitions.
```

- If you execute an INSERT statement to insert data into dynamic partitions, the dynamic partition key columns must be included in the SELECT clause.

```
insert overwrite table sale_detail_dypart partition (sale_date='2017', region) select shop_name,cu
stomer_id,total_price from sale_detail;
-- An error is returned. If you execute an INSERT statement to insert data into dynamic partitions
, the dynamic partition key columns must be included in the SELECT clause.
```

For more information about the INSERT statements, see INSERT.

1.4.8.4. JOIN

This topic describes the limits on JOIN statements.

Limits:

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

For more information, see JOIN.

1.4.8.5. Other limits

This topic describes other limits on MaxCompute SQL.

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

1.4.9. Compile and use UDFs

1.4.9.1. Overview

This topic describes how to write and run a MaxCompute user-defined function (UDF).

MaxCompute supports user-defined scalar functions, user-defined aggregate functions (UDAFs), and user-defined table-valued functions (UDTFs).

> **Note**
> - UDFs support only Java APIs. To write a UDF program, you can upload UDF code to your project as a resource and execute the required statement to create a UDF.
> - This topic provides code examples of user-defined scalar functions, UDAFs, and UDTFs.

1.4.9.2. UDF example

This topic describes how to create a user-defined scalar function (UDF) and provides an example.

Procedure

1. Write code.

   You must develop and compile functions in compliance with the MaxCompute UDF framework.

   ```
   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 you run a UDF. User code must be added to MaxCompute as resources. Java UDFs must be packaged as JAR packages and added to MaxCompute as JAR resources. The UDF framework automatically loads the JAR packages and runs the UDFs.

   Example command to add JAR resources:

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

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

3. Register the UDF.

   After your JAR package is uploaded, MaxCompute has no information about this UDF. Therefore, you must register a unique function name in MaxCompute and associate the function name with the JAR resource and function.

   Example command to register the UDF:

   ```
   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 the code example of a user-defined aggregate function (UDAF) for your reference.

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

UDAF code example:

```
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 = (LongWritable) 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 = (LongWritable) 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 user-defined table-valued function (UDTF) code.

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. Write and run a MapReduce program

This topic describes how to write and run a MapReduce program.

Prerequisites

- JDK 1.8 is installed.
- The MaxCompute client is 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 statements that are used to create tables, see Create a table.

2. Insert data into the wc_in table.

   You can use one of the following methods to insert data:

   - Run Tunnel commands to upload data.

     The following code shows the data that you want to insert into the table. You must create a kv.txt file on your computer and save the data to the file. In this example, the kv.txt file is saved in D:\.

     ```
     238,val_238
     186,val_86
     186,val_86
     ```

     Run the following command to upload the data:

     ```
     Tunnel upload D:\kv.txt wc_in;
     ```

   - Execute the following SQL statement to insert the data:

     ```
     INSERT INTO TABLE wc_in VALUES ('238',' val_238'),('186','val_86'),('186','val_86');
     ```

3. Write a MapReduce program and compile it.

   - MaxCompute provides the Eclipse plug-in to help you develop MapReduce programs and debug them on your computer.

- You must create a MaxCompute project in Eclipse. Then, write a MapReduce program in this project. After local debugging succeeds, upload the compiled program (JAR package) to MaxCompute.

4. Add the JAR package as a project resource.

   In this example, the JAR package is named word-count-1.0.jar.

   ```
   add jar word-count-1.0.jar;
   ```

5. Run the JAR command on the MaxCompute client.

   ```
   jar -resources word-count-1.0.jar -classpath /home/resources/word-count-1.0.jar com.taobao.jingf
   an.wordcount wc_in wc_out;
   ```

   > **Note** If the Java program uses resources, use the -resources option to specify the resources.

6. View the command output on the MaxCompute client.

   ```
   select * from wc_out;
   ```

1.4.11. Write and run a Graph job

This topic describes how to write and run a Graph job.

Procedure

1. Create input and output tables.

   Execute the following statements:

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

   > **Note** For more information about the statements that are used to create tables, see Create a table.

2. Upload data to the sssp_in table.

   We recommend that you create an sssp.txt file that contains the following data and save the file to a local directory, such as D:\:

   ```
   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
   ```

   Run the Tunnel command to upload data to the sssp_in table. Use spaces as the delimiter between columns.

   ```
   tunnel u -fd " " D:\sssp.txt sssp_in;
   ```

3. Compile a Single Source Shortest Path (SSSP) example.

   Compile and debug SSSP algorithm examples in local mode based on the Graph development process. For more information, see SSSP.

   > **Note** In this example, the code is packaged as *odps-graph-example-sssp.jar*. You need only to package the SSSP code. You do not need to package the SDK into this JAR package.

4. Add a JAR resource.

Examples:

```
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 a resource.

5. Run the SSSP.

   Examples:

```
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. You can use this command for Graph jobs in the same way as MapReduce jobs.

   The commands used to run Graph jobs list the instance ID, execution progress, and result summary of the job. Example:

```
ID = 20130730160742915g******
2013-07-31 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]_records=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. Use Logview to view job running information

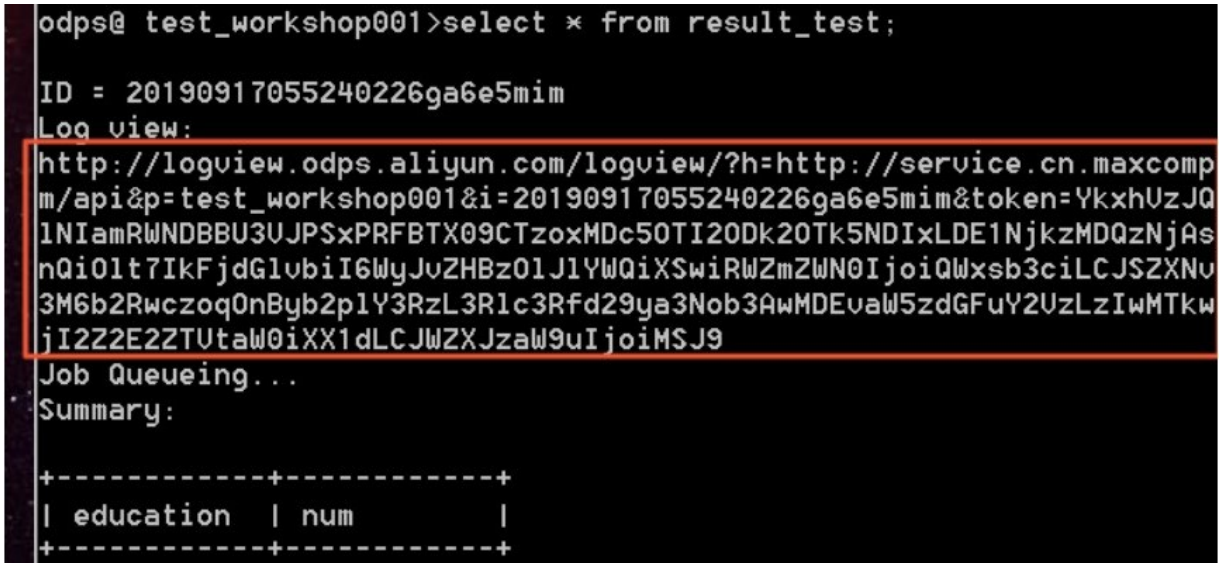
This topic describes how to use Logview to view job running information.

Logview is a tool that you can use to view and debug jobs after you submit the jobs to MaxCompute. You can use Logview to view the following information of a job:

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

> **Note** When you deploy MaxCompute, Logview is deployed by default.

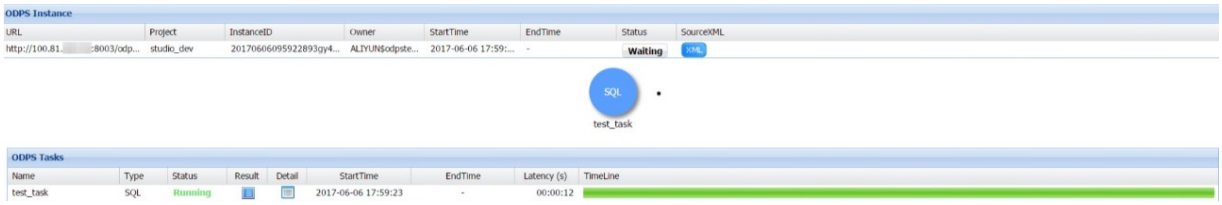
After you submit a job to MaxCompute, the system generates a Logview URL. You can enter the Logview URL in the search engine and press Enter to view job information.



> **Note** The Logview page of each job is valid for seven days.

UI elements

This section describes elements on the Logview UI.



The Logview page consists of two sections:

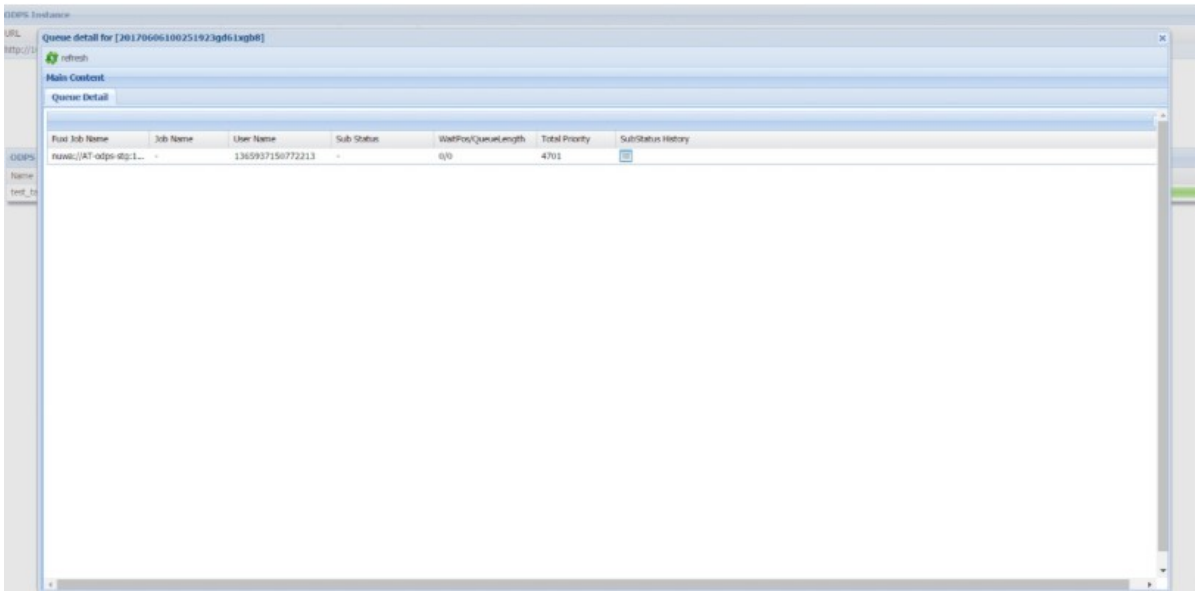
- ODPS Instance
- ODPS Tasks

The ODPS Instance section shows the information of the instance that corresponds to the submitted SQL job. The information includes URL, Project, InstanceID, Owner, StartTime, EndTime, and Status.

- You can click the value in the **Status** column to view queue information. Valid values of Status:
  - 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 queued. n indicates the order number of the job in the queue.
  - Running: The job is running in Job Scheduler.

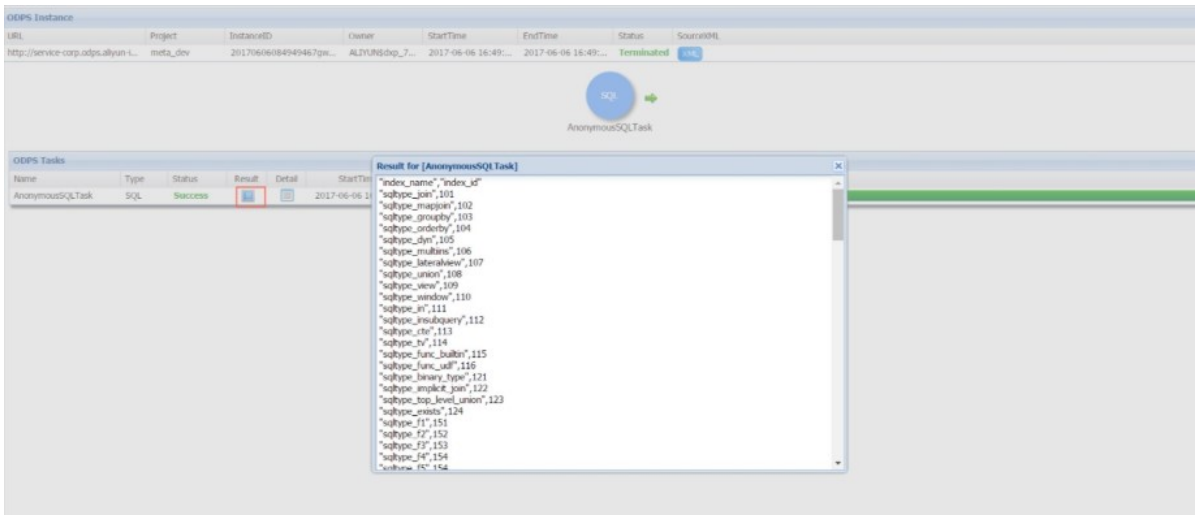
> **Note** If the value of Status is Terminated, the job is complete and has no queue information.

- After you click the value of Status, the following queue information is displayed:
  - Sub Status: the current sub-status information.
  - WaitPos: the position of the job in the queue. The value 0 indicates that the job is running. The value - indicates that 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 the status history. This includes the status code, status description, start time, and duration of a state. The information is unavailable in some versions.

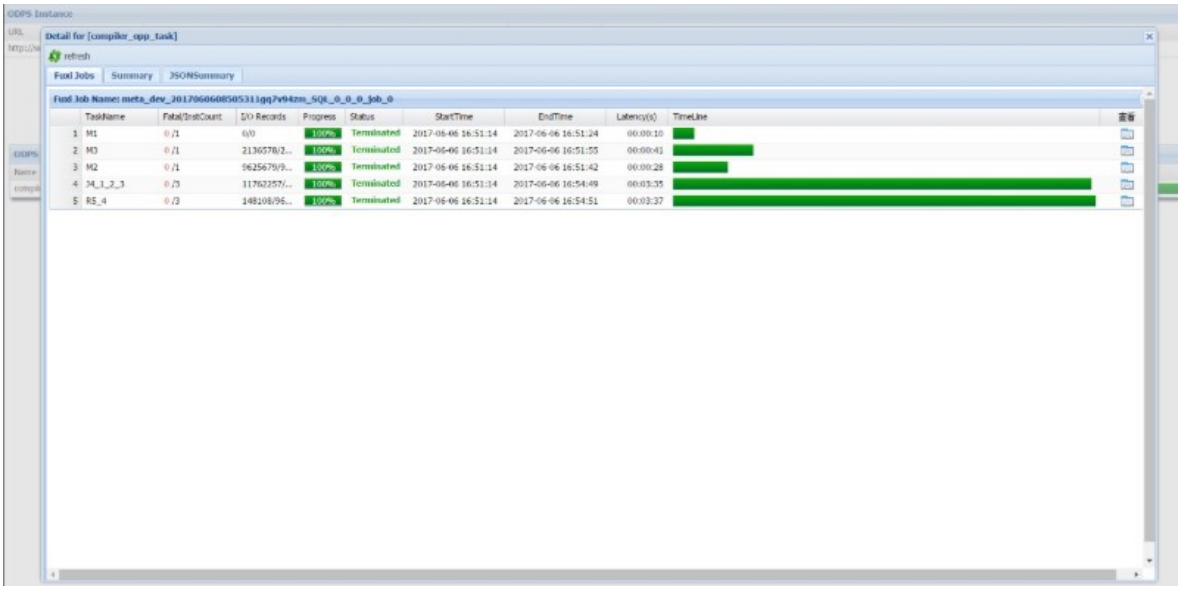


The ODPS Tasks section shows the information of the task that corresponds to the instance. The information includes Name, Type, Status, Result, Detail, StartTime, EndTime, Latency (s), and TimeLine. Like on other pages, Latency (s) indicates the running duration.

- Result: After a job is complete, you can click the icon in the Result column to view execution results. The following figure shows the execution results of a SELECT statement.



- Detail: You can click the icon in the **Detail** column to view running details. You can view the details of both running and completed jobs.



- In the dialog box that shows details about a MaxCompute job, you can view the following information:
  - A MaxCompute job consists of one or more Fuxi jobs. For example, if a complex SQL job is submitted, MaxCompute automatically submits multiple Fuxi jobs to Job Scheduler.
  - Each Fuxi job consists of one or more Fuxi tasks. For example, a simple MapReduce job generates two Fuxi tasks: map task (M1) and reduce task (R2). If an SQL job is complex, multiple Fuxi tasks may be generated.
  - You can view the name of a Fuxi task. In most cases, a task name consists of letters and digits. A letter represents the type of a task, such as M for a map task. The digits that follow the letter represent the ID and dependencies of a task. For example, R5_4 indicates that the reduce task can be executed only after the J4 task is complete. J4_1_2_3 indicates that the join task can be executed only after the M1, M2, and M3 tasks are complete.
  - I/O Records indicates the numbers of the input and output records of the Fuxi task.
- To view the information of the instance that corresponds to a Fuxi task, you can click the icon in the Show Detail column of the Fuxi task or double-click the Fuxi task.

  Note Each Fuxi task consists of one or more Fuxi instances. If the amount of input data for a Fuxi task increases, MaxCompute starts more nodes for the task to process the data. A node is equivalent to a Fuxi instance.

In the lower part of the dialog box, the status information about Fuxi instances at different stages 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 standard output information. You can also click the icon in the StdErr column to view standard error information.

Note To-be-displayed information written in the submitted MaxCompute job is also displayed in the standard output information and standard error information.

Use Logview to handle issues

Tasks with errors: If an error occurs during a task, find the task and click the icon in the Result column in the lower part of the Logview page to view the error information. You can also click the icon in the StdErr column of a Fuxi instance in the details dialog box to view the error information of the instance.

Data skew: The long tail of one or more Fuxi instances may decelerate the execution of a Fuxi task. The long tail is caused by uneven data distribution in a task. After the task is complete, you can view the running results on the Summary tab of the details 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 column.

1.4.13. Use Logview V2.0 to view job running information

This topic describes the entry and features of Logview V2.0. You can use Logview V2.0 to view job running information.

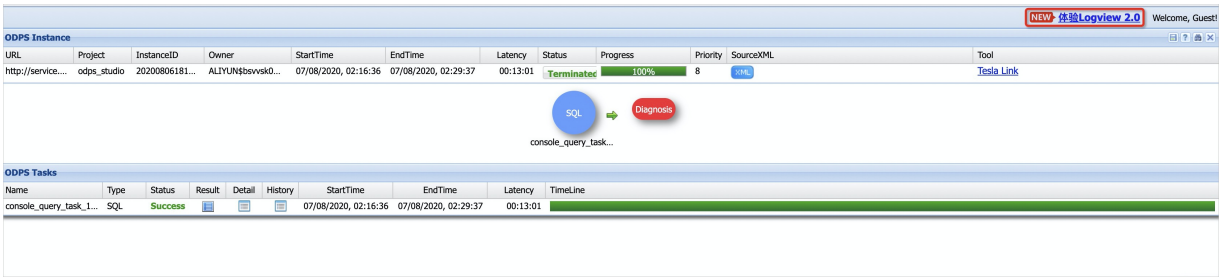
Overview

Logview V2.0 provides a newly designed UI, increases the loading speed, and delivers the following new features:

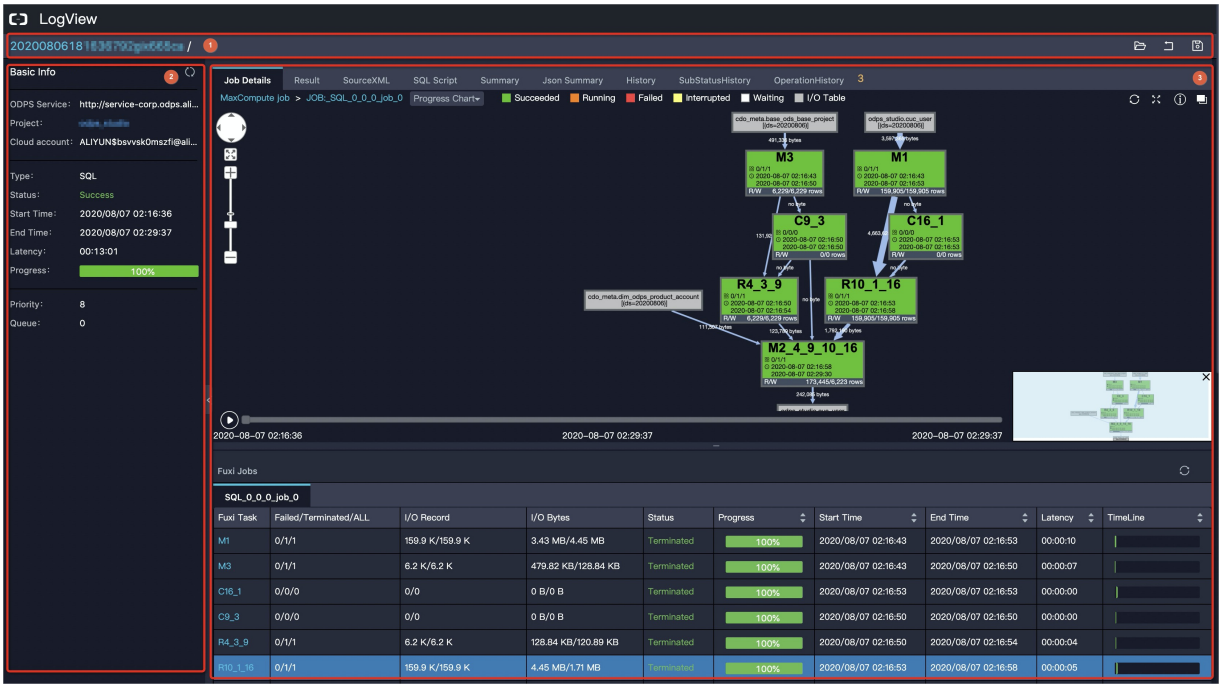
- Provides an interactive directed acyclic graph (DAG) to display the logical architecture of job processing. You can also view the operators of a job.
- Supports job execution playback.
- Allows you to view memory usage and CPU utilization by using Fuxi Sensor.

Entry

After you use the MaxCompute client (odpscmd) to submit a job, the system generates a Logview URL. Enter the Logview URL in the search engine and press Enter. On the Logview UI, click **Go to Logview V2.0**.



Logview V2.0 page



No.	Section
①	The title and function section.
②	The Basic Info section.
③	The job details section.

Title and function section

This section shows the job ID and job name. The job ID uniquely identifies a MaxCompute job. The job ID is generated when you submit the job. The job name is displayed only if the job is submitted by using SDKs. You can also click the icons on the right of this section to perform operations.

Icon	Description
	Open the Logview_detail.txt file that contains job details. The file is saved on your computer.
	Return to the original Logview UI.
	Save the job details as a file to your computer.

Basic Info

This section shows the basic information about a job.

Parameter	Description
MaxCompute Service	The endpoint of MaxCompute on which the job is running.
Project	The name of the MaxCompute project to which the job belongs.

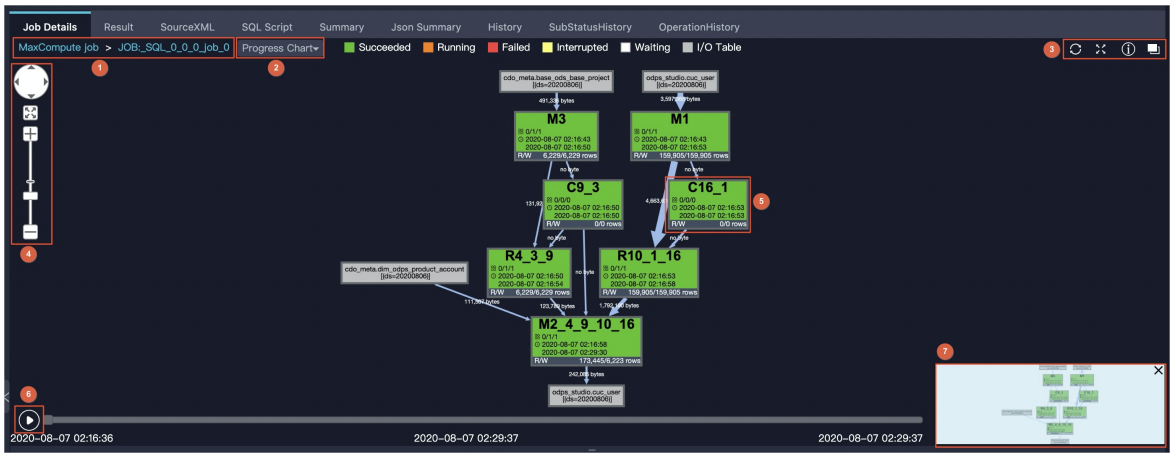
Parameter	Description
Cloud account	The Alibaba Cloud account that is used to submit the job.
Type	The type of the job. Valid values: SQL, SQLRT, LOT, XLib, CUPID, AlgoTask, and Graph.
Status	The status of the job. Valid values: • Success: The job succeeds. • Failed: The job fails. • Canceled: The job is canceled. • Waiting: The job is being processed in MaxCompute but is not submitted to Job Scheduler. • Running: The job is being processed in Job Scheduler. • Terminated: The job is complete.
Start Time	The time when the job is submitted.
End Time	The time when the job is complete.
Latency	The period during which the job is executed.
Progress	The progress of the job.
Priority	The priority of the job.
Queue	The position of the job in the queue in the resource quota group.

Job details

In the job details section, you can query details about a job. This section consists of the following tabs:

- **Job Details**
  - Progress chart

    In the upper part of the **Job Details** tab, the progress chart of a job is displayed. The progress chart shows the subtask dependencies from three dimensions: Fuxi jobs, Fuxi tasks, and operators. It also provides a series of tools to help you troubleshoot issues. The following figure shows the upper part of the Job Details tab.



No.	Description

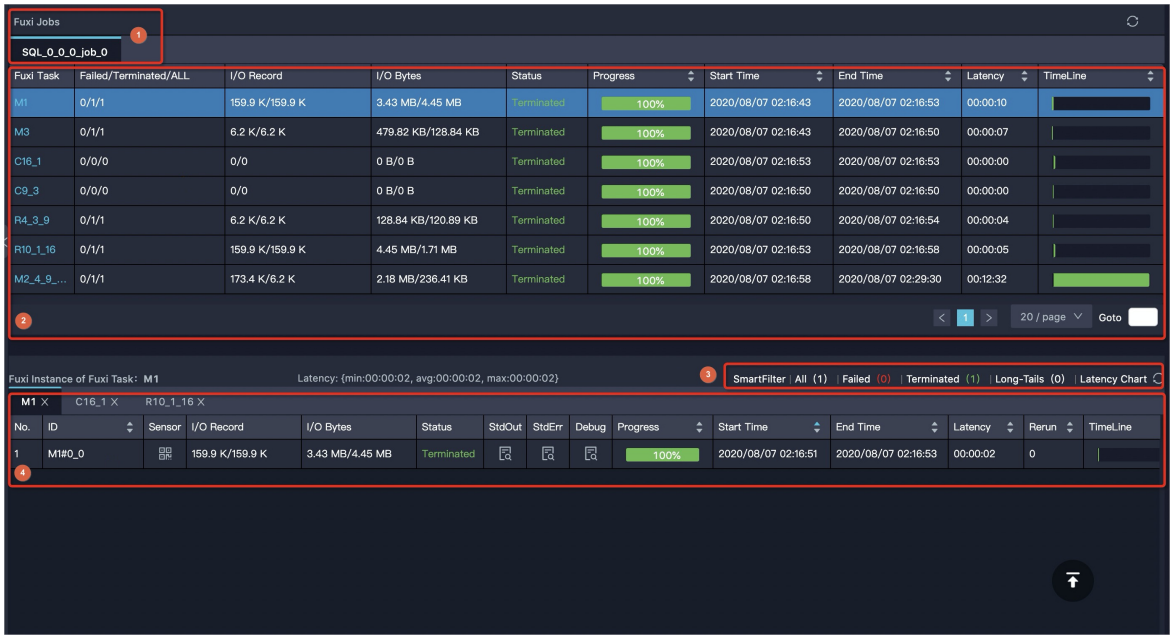
No.	Description
①	The breadcrumb navigation that is used to switch Fuxi jobs. JOB:_SQL_0_0_0_job_0 is the name of a Fuxi job.
②	The troubleshooting tool. You can use Progress Chart, Input Heat Chart, Output Heat Chart, TaskTime Heart Chart, and InstanceTime Heart Chart for troubleshooting.
③	You can click the icon to refresh the job status and the icon to zoom in or out the progress chart. You can also click the icon to obtain MaxCompute Studio documentation and the icon to switch to the upper level of the job.
④	The zoom tool.
⑤	The Fuxi task. A MaxCompute job consists of one or more Fuxi jobs. Each Fuxi job consists of one or more Fuxi tasks. Each Fuxi task consists of one or more Fuxi instances. If the amount of input data increases, MaxCompute starts more nodes for each task to process the data. A node is equivalent to a Fuxi instance. For example, a simple MapReduce job generates two Fuxi tasks: map task (M1) and reduce task (R2). If an SQL statement is complex, multiple Fuxi tasks may be generated. You can view the name of each Fuxi task on the execution page. For example, M1 indicates a map task. The 3 and 9 fields in R4_3_9 indicate that the map task can be executed only after M3 and C9_3 are complete. Similarly, M2_4_9_10_16 indicates that the M2 task can be executed only after R4_3_9, C9_3, R10_1_16, and C16_1 are complete. R/W indicates the numbers of rows that the task reads and writes. Click or right-click a task to view operator dependencies and operator graphs of the task. Logview V2.0 provides table dependencies so that you can quickly view the input and output tables.
⑥	The Fuxi task playback. You can click the icon to start or stop the playback. You can drag the progress bar to play at a specific time. The start time and end time are displayed on the sides of the progress bar. The playing time is displayed in the middle.
⑦	EagleEye.

Note

- The playback feature is not applicable to Fuxi tasks that are in the Running state.
- An AlgoTask job, such as a Machine Learning Platform for AI (PAI) job, contains only one Fuxi task. Therefore, no progress charts are provided for these jobs.
- For non-SQL jobs, only Fuxi jobs and Fuxi tasks are displayed.
- If only one Fuxi job exists, the progress chart shows the dependencies among Fuxi tasks. If multiple Fuxi jobs exist, the progress chart shows the dependencies among the Fuxi jobs.

- Running status of jobs

  In the lower part of the **Job Details** tab, the detailed running information about the job is displayed. The following figure shows the lower part of the Job Details tab.

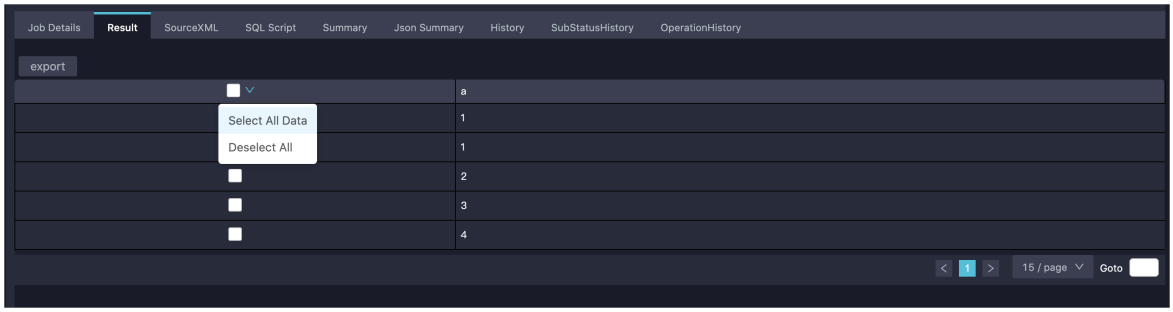


No.	Description
①	The Fuxi Jobs tab. You can switch Fuxi jobs on this tab.
②	The details about Fuxi tasks of the Fuxi job. Click a Fuxi task to display information about the Fuxi instance of this task. By default, information about the Fuxi instance of the first Fuxi task for the first Fuxi job is displayed.
③	Logview divides instances into groups based on their status. You can click the number next to Failed to query information about faulty nodes.
④	Fuxi Sensor. This feature is provided only for AlgoTask jobs, such as PAI jobs. You can use Fuxi Sensor to view the memory usage and CPU utilization of Fuxi instances.
⑤	StdOut and StdErr. You can view output messages, error messages, and information to be displayed. You can also download the information.
⑥	Debug. You can debug and troubleshoot errors.

- **Result**

  This tab shows the result of a job. If a job failed, this tab is displayed and shows the cause of the failure. The system displays data in plain text or a table based on the response format specified in MaxCompute. You can log on to the MaxCompute client and set odps.sql.select.output.format to specify the format. If you set this parameter to HumanReadable, the result is displayed in plain text. Otherwise, the result is displayed in the CSV format.

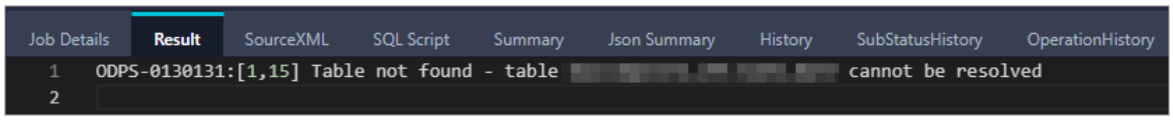
- Table



You can perform the following operations on this tab:

- Select a row to export its data.
- Select the table heading to export data from the current page.
- In the table heading, click the icon and select **Select All Data** or **Deselect All** as needed.
- Click **export** to export data in the CSV format. CSV is a file name extension. You can use a text file or Sublime Text to open CSV files.

- Plain text



- **SourceXML**

  This tab shows the source XML information about the job that is submitted to MaxCompute.

- **SQL Script**

  This tab shows the SQL scripts of the job that is submitted to MaxCompute.

- **Summary**

  This tab shows the overall running information about the job that is submitted to MaxCompute.

- **Json Summary**

  This tab shows the overall running information about the job in the JSON format.

- **History**

  This tab shows the historical information about a Fuxi instance if the instance is executed multiple times.

- **SubStatusHistory**

  This tab shows the detailed information about a job, including the status code, status description, start time, duration, and end time.

Fuxi Sensor

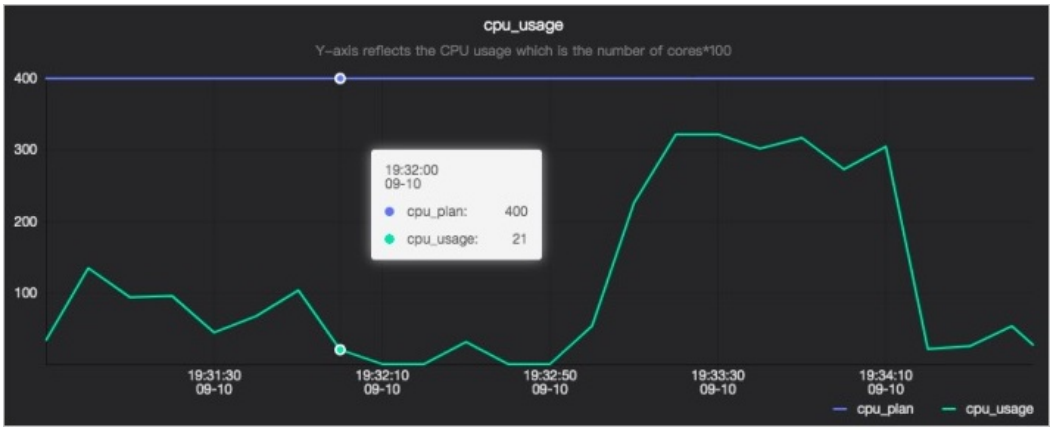
Fuxi Sensor is a resource view that shows a MaxCompute job from all dimensions. You can use Fuxi Sensor to view the memory usage and CPU utilization of a Fuxi instance. You can also use Fuxi Sensor to locate job issues and analyze running performance. For example, you can use Fuxi Sensor in the following scenarios:

- If out-of-memory (OOM) occurs, analyze the amount of memory used.
- Compare the numbers of requested and used resources to optimize the resource request process.

For example, you can use Fuxi Sensor to view the resource usage of a Fuxi instance.

- CPU utilization

The cpu_usage chart has two lines. One indicates the number of requested CPUs (cpu_plan), and the other one indicates the number of used CPUs (cpu_usage). In the y-axis, 400 indicates four processors.

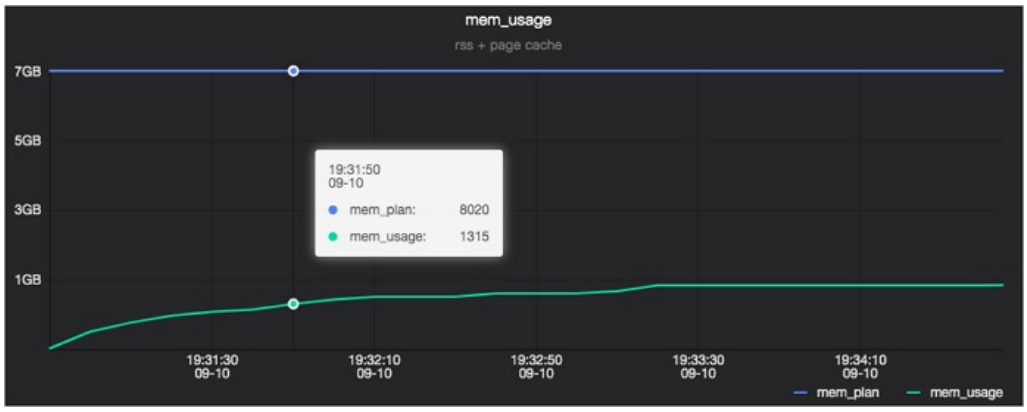


- Memory usage

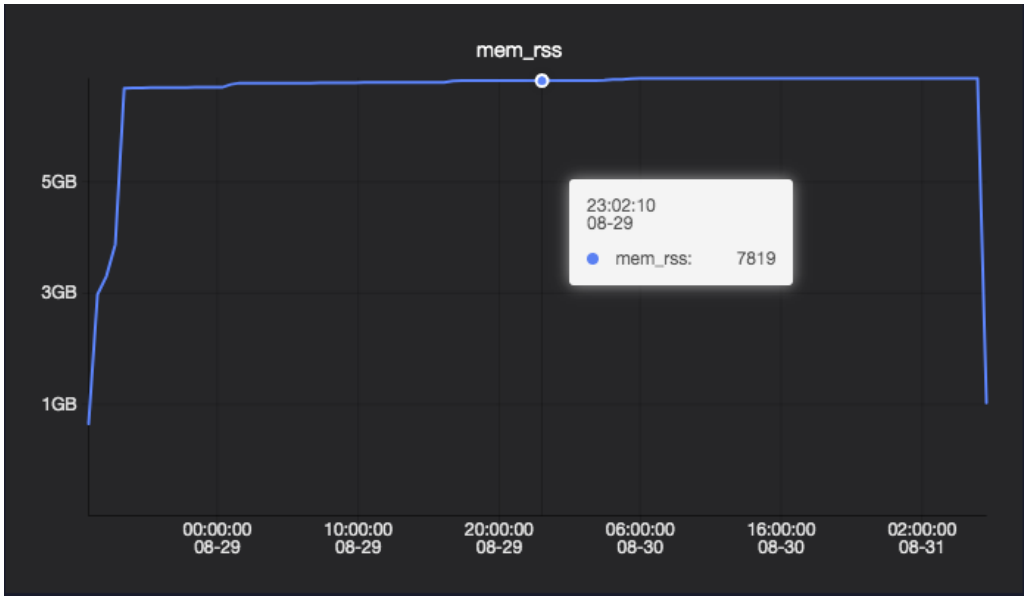
  The mem_usage chart has two lines. One indicates the number of requested memory resources (mem_plan), and the other one indicates the number of used memory resources (mem_usage).

  mem_usage contains Resident Set Size (RSS) and PageCache. RSS indicates the memory that is allocated after kernel page faults occur. This applies when you call Malloc to request memory by using non-file mappings. If the memory is insufficient, RSS cannot be reclaimed. PageCache is the memory occupied by the kernel to cache files that are required by the read and write requests, such as log files. If the memory is insufficient, PageCache can be reclaimed.

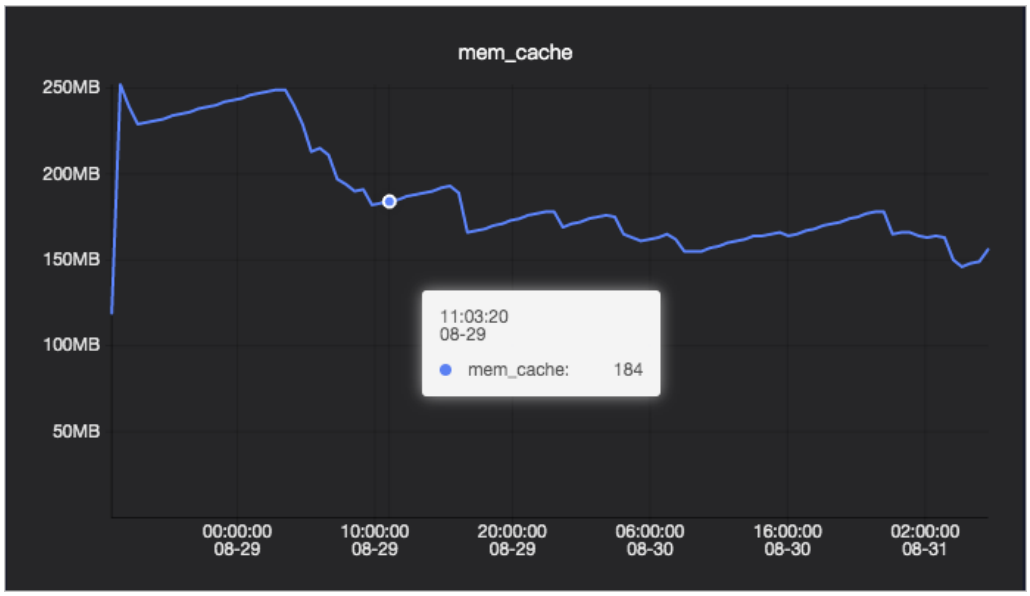
  - Memory details



- RSS usage



- PageCache usage



1.5. Basic concepts and common commands

1.5.1. Terms

This topic introduces the basic terms of MaxCompute.

project

A basic organizational unit of MaxCompute. Similar to a database or schema in a traditional database system, a project is used to isolate multiple users and control access requests. A user can have permissions on multiple projects. After 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 al_project
-- Access a project named al_project.
```

Note After you run the preceding command, you access a project named al_project and manage objects that belong to this project, regardless of which project you are in.

table

A data storage unit of MaxCompute. Logically, a table is a two-dimensional structure that consists of rows and columns, in which each row represents a record and each column represents the values 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. Table columns support all data types in MaxCompute. Tables are the input and output objects of all MaxCompute computing tasks. You can create a table, delete a table, and import data into a table.

You can define partitions for a table to process data more efficiently. You can specify some fields in the table as partition key columns. In most cases, partitions within a table are analogous to directories within a file system. Each value of a partition key 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 analogous to multi-level directories. If you specify the name of the partition that you want to access, MaxCompute scans only the specified partition. This makes data processing more efficient and reduces costs.

data type

The values in a column of a MaxCompute table must be of a specific data type. The following table describes the basic data types supported by MaxCompute and their valid values.

Basic data types

Data type	New in MaxCompute V2.0	Constant	Description	Valid value
TINYINT	Yes	1Y,-127Y	The 8-bit signed integer type.	-128~ 127
SMALLINT	Yes	32767S,-100S	The 16-bit signed integer type.	-32768 ~ 32767
INT	Yes	1000,-15645787	The 32-bit signed integer type.	$-2^{31} \sim 2^{31}-1$
BIGINT	No	100000000000L,-1L	The 64-bit signed integer type.	$-2^{63}+1 \sim 2^{63}-1$
STRING	No	"abc",'bcd',"alibaba",'inc'	The string, which supports UTF-8 encoding. The character behavior encoded in other formats is not defined.	The size of all values in a column of the STRING type cannot exceed 8 MB.
FLOAT	Yes	None	The 32-bit binary floating point type.	N/A
BOOLEAN	No	True,False	The Boolean type.	True or False

Data type	New in MaxCompute V2.0	Constant	Description	Valid value
DOUBLE	No	3.1415926 1E+7	The 64-bit binary floating point type.	-1.0 10^{308} ~ 1.0 10^{308}
DATETIME	No	Datetime '2017-11-11 00:00:00'	The DATETIME type. The standard system time is UTC+8.	0001-01-01 00:00:00 000 ~ 9999- 12-31 23:59:59 999
DECIMAL	No	3.5BD, 99999999999.9999999BD	The exact numeric type based on the decimal system.	Integer: -10^{36} + 1 to 10^{36} - 1 Decimal places: round to 10^{-18}
VARCHAR	Yes	None	The variable-length type. n specifies the length.	1 ~ 65535
BINARY	Yes	None	The binary data type.	The size of all values in a column of the BINARY type cannot exceed 8 MB.
TIMESTAMP	Yes	Timestamp '2017-11-11 00:00:00.123456789'	The TIMESTAMP type. The timestamp is independent of time zones.	0001-01-01 00:00:00 000000000 ~ 9999- 12-31 23:59:59 999999999

If you want to use the new data types in MaxCompute V2.0, you must first add one of the following flags to enable the new data types: `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 contain NULL values.

> **Note** Only lowercase letters can be used in the preceding flags.

Take note of the following points when you use the new data types in MaxCompute V2.0:

- If you add the `set odps.sql.type.system.odps2=true;` flag, the following impacts occur:
  - The semantics of the INT keyword changes. INT in an SQL statement indicates a 32-bit integer.
  - The semantics of an integer constant changes. `Select 1 + a;` is used in this example.
    - If the new data types are disabled, the integer constant is processed as the BIGINT type. If the length of the constant exceeds the range for the BIGINT type, the integer constant is processed as the DOUBLE type.
    - If the new data types are enabled, the integer constant is processed as the INT type. In this example, the constant 1 is a 32-bit integer.
    - If the integer constant is processed as the INT type, inconsistency may occur in function prototypes during subsequent operations. In addition, the actions of peripheral tools and subsequent operations might be changed due to the tables that contain new data types and that are generated after data is written to a disk.

- The rules for implicit conversions of data types change.

  If the new data types are enabled, some implicit type conversions may be disabled. For example, if 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, precision may be reduced or errors may occur. In this case, you can use the CAST function to force the data type conversion.

  Implicit type conversions greatly affect functions and INSERT statements. For example, an INSERT statement can be executed if the new data types are disabled, but returns an error if the types are enabled.

- If the new data types are disabled, some operations and built-in functions that use data of new data types as parameters and return values are ignored. These operations and functions can be used after the new data types are enabled.
  - Some built-in functions can be used only after the new data types are enabled. The built-in functions include the functions that return the values of the INT type, such as YEAR, QUARTER, MONTH, DAY, HOUR, MINUTE, SECOND, MILLISECOND, NANOSECOND, DAYOFMONTH, and WEEKOFYEAR. The parameters of these functions can be of the INT type, and the BIGINT overload is used for these functions. These functions can be implemented by using the DATEPART built-in function.
  - The resolution of user-defined functions (UDFs) changes.
- The parsing of the BIGINT keyword changes.
- The data types supported by partition key columns change.
  - If the new data types are disabled, the data type of a partition key column can only be STRING.
  - If the new data types are enabled, the data type of a partition key column can be STRING, VARCHAR, CHAR, TINYINT, SMALLINT, INT, or BIGINT.
  - If the new data types are disabled, partition fields in INSERT operations are processed as the STRING type.
- The execution rules of LIMIT statements in SET operations change.

  The `Select * from t1 union all select * from t2 limit 10;` statement is used in this example.
  - If the new data types are disabled, the preceding statement is expressed as `Select * from t1 union all select * from ( select * from t2 limit 10) t2;` .
  - If the new data types are enabled, the preceding statement is expressed as `Select * from (select * from t1 union all select * from t2 ) limit 10;` .

  These changes also apply to UNION, INTERSECT, EXCEPT, ORDER BY, DISTRIBUTE BY, SORT BY, and CLUSTER BY.
- The parsing of data types in IN expressions changes.

  The `a in (1, 2, 3)` expression is used in this example.
  - If the new data types are disabled, all the values enclosed in the parentheses () must be of the same type.
  - If the new data types are enabled, all the values enclosed in the parentheses () are implicitly converted to the same type.

- If a constant of the INT type exceeds the range for the INT type, the constant is processed as the BIGINT type. If the constant exceeds the range for the BIGINT type, the constant is processed as the DOUBLE type. If `odps.sql.type.system.odps2` is not set to true, MaxCompute retains the conversion and notifies you that the INT data is processed as the BIGINT type. If odps.sql.type.system.odps2 is not set to true in your scripts, we recommend that you rewrite your scripts and convert these types into BIGINT to prevent confusion.
- VARCHAR constants can be implicitly converted into 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 do not include the millisecond component. You can add `-dfp` to Tunnel commands to display milliseconds in the time values.

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

Complex data types

Data 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 one of the following methods to enable the new data types:

- Execute `o.execute_sql('set odps.sql.type.system.odps2=true;query_sql', hints={"odps.sql.submit.mode" : "script"})` to enable the new data types.
- Use DataFrame to enable the new data types. For example, you can set the hints parameter to specify an immediately executed method, such as persist, execute, or to_pandas. The settings in the following figure are 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'})
```

- If you use DataFrame to enable the new data types and want the settings to take effect globally, set `options.sql.use_odps2_extension` to True.

resource

A unique concept in MaxCompute. To implement UDF and MapReduce features, you must use resources.

- MaxCompute SQL UDFs: After you write a UDF, you must package it into a JAR file and upload the package to MaxCompute as a resource. When you run the UDF, MaxCompute automatically downloads the JAR file and obtains the code to run the UDF. JAR files are a type of MaxCompute resource. When you upload a JAR file, a resource is created in MaxCompute.
- MaxCompute MapReduce: After you write a MapReduce program, you must package it into a JAR file and upload the file to MaxCompute as a resource. When you run a MapReduce job, the MapReduce framework automatically downloads the JAR file and obtains the code to run the MapReduce job.

> **Note**
>
> - Some limits are imposed 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 UDTF description.

MaxCompute supports the following resource types:

- File
- Table: tables in MaxCompute.
- JAR: compiled JAR files.
- Archive: compressed files identified by the resource name extension. Supported file types include .zip, .tgz, .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 complete some computing and counting tasks. If these built-in functions do not meet your requirements, you can use the Java APIs that MaxCompute provides to develop UDFs. UDFs are classified into user-defined scalar-valued functions, user-defined aggregate functions (UDAFs), and user-defined table-valued functions (UDTFs).

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

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

task

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

An execution plan consists of several 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 the stages based on the dependencies in the graph (execution plan). A stage has multiple processes, also known as workers. The workers in each stage work together to complete data computations for the stage. Different workers in a stage process different data, but they all use the same execution logic.

When you run a computing task, the task is converted into an instance. You can perform operations on this instance. For example, you can obtain the status of the instance and terminate the instance.

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

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

instance

Some MaxCompute tasks are converted into 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. 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. After you run the kill command, the status of the instance changes to Canceled.
```

resource quota

A per-process limit on the use of system resources. Two types of quotas are involved: storage and computing. The storage quota is the upper limit of the storage resources 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 for the processes that are run in a project at the same time cannot exceed the computing quota.

1.5.2. Common commands

1.5.2.1. Overview

MaxCompute allows you to perform operations on objects, such as projects, tables, resources, and instances. You can run command and execute statements on the client or use SDKs to perform operations on these objects.

This topic describes how to run the commands and execute the statements on the MaxCompute client.

> **Note**
> - For more information about how to download and configure the client, see Configure the client.
> - For more information about the SDKs, see SDK for Java.

1.5.2.2. Project operations

This topic describes common statements for project operations.

Create or delete a project

MaxCompute does not provide statements for you to create or delete a project. You can create or delete a project in the Apsara Uni-manager Management Console.

- For more information about how to create a project, see Create a project.
- If you want to delete a project, you can log on to the Apsara Uni-manager Management Console and perform the following steps: Choose **Products** > **Big Data** > MaxCompute. In the left-side navigation pane, click **Projects**. Find the project that you want to delete and click **Delete** in the Actions column.

Go to a project

Syntax:

```
use <project_name>;
```

Description: This statement is used to go to a specified project. After you go to the project, you can directly manage all objects in the project.

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

Example:

```
use my_project;
-- my_project is the project that you have permissions to access.
```

> **Note**
> - The preceding statement is executed in the MaxCompute client.
> - All the statement keywords, project names, table names, and column names in MaxCompute are not case-sensitive.

If the test_src table exists in the my_project project, execute the following statement:

```
select * from test_src;
```

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

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

```
select * from my_project2.test_src;
```

Data of the my_project2 project, instead of data of the test_src table in the my_project project, is returned.

View projects

Syntax:

```
list projects;
```

Description: This statement is used to query all projects in MaxCompute.

Clear a project

Syntax of the statement that is used to view objects in the project recycle bin:

```
show recyclebin;
```

Description: This statement is used to query all objects in the project recycle bin.

> **Note** Only the project owner is allowed to execute this statement.

Syntax of the statement that is used to clear the project recycle bin:

```
purge all;
```

Description: This statement is used to clear the project recycle bin to release storage space.

> **Note** Only the project owner is allowed to execute this statement.

Syntax of the statement that is used to clear a table:

```
purge table tblname;
```

Description: This statement is used to clear a specified table from the recycle bin to release storage space.

> **Note**
> - If the specified table exists, the project owner and users who have write permissions on the table are allowed to execute this statement.
> - If the table has been deleted, only the project owner is allowed to execute this statement.

1.5.2.3. Table operations

This topic describes common statements for table operations.

CREATE TABLE

Syntax:

```
CREATE [EXTERNAL] 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], ...)]
[LIFECYCLE days];
CREATE TABLE [IF NOT EXISTS] table_name
[AS select_statement | LIKE existing_table_name];
```

Description: This statement is used to create a table.

Examples:

```
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 if no table with this name exists.
```

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

CHANGEOWNER

Syntax:

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

Description: This statement is used to change the owner of a table.

Examples:

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

DROP TABLE

Syntax:

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

Description: This statement is used to delete a table.

> **Note** If you do not specify IF EXISTS and the table does not exist, an error is returned. If you specify IF EXISTS, a success message is returned, regardless of whether the table exists.

Parameters:

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

Examples:

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

Query table information

Syntax:

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

Description: The statements are used to query information about specific tables. Output fields include Owner, Project, CreateTime, LastDDLTime, LastModifiedTime, InternalTable, Size, Native Columns, Partition Columns, and Extended Info. The Owner field indicates the owner of the table. The Project field indicates the project to which the table belongs. The CreateTime field indicates the time when the table was created. The LastDDLTime field indicates the last time when a DDL statement is executed for the table. The LastModifiedTime field indicates the last time when the table data was modified. The InternalTable field indicates that the queried object is a table. The value of this field is always YES. The Size field indicates the storage occupied by table data, in bytes. The Native Columns field shows the names, data types, and comments of standard columns. The Partition Columns field shows the names, data types, and comments of partition key columns. The Extended Info field shows information about an external table, including the storage handler and location.

Parameters:

table_name: In the first statement, this parameter specifies the name of the table or view. In the second statement, this parameter specifies the name of the external table.

Examples:

Execute the following statement to query information about a partitioned table:

```
DESC sale_detail;
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$odpsuser@aliyun.com | Project: test_project                          |
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:               2014-01-01 17:32:13                                      |
| LastDDLTime:              2014-01-01 17:57:38                                      |
| LastModifiedTime:         1970-01-01 08:00:00                                      |
+------------------------------------------------------------------------------------+
| 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

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

Query partition information

Syntax:

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

Description: This statement is used to query information about a partition.

Examples:

Execute the following statement to query information about a partition:

```
desc meta.m_security_users partition (ds='20151010');
```

The following result is returned:

```
+------------------------------------------------------------------------------------+
| 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 and SHOW TABLES LIKE

Syntax:

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

Description:

- SHOW TABLES: queries all tables in the current project.
- SHOW TABLES like 'chart': queries the tables whose names match chart in the current project. You can use regular expressions in the statement to filter tables.

Examples:

Execute the following statement to query all tables in the current project:

```
show tables;
```

Execute the following statement to query the tables whose names match ods_brand:

```
show tables like 'ods_brand*';
```

The following result is returned:

```
ALIYUN$odps_user@aliyun.com:ods_brand
......
```

> **Note**
>
> - odps_user@aliyun.com: the name of the user who creates the table.
> - table_name: the name of the table.

SHOW PARTITIONS

Syntax:

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

Description: This statement is used to query all partitions of a table.

Parameters:

table_name: the name of the table that you want to query. If the table does not exist or is a non-partitioned table, an error is returned.

Examples:

Execute the following statement to query all partitions of a table:

```
SHOW PARTITIONS table_name;
```

The following result is returned:

```
partition_col1=col1_value1/partition_col2=col2_value1
partition_col1=col1_value2/partition_col2=col2_value2
```

> **Note**
>
> - partition_col1 and partition_col2: the partition key columns of the table.
> - col1_value1, col2_value1, col1_value2, and col2_value2: the values in the partition key columns.

1.5.2.4. Instance operations

This topic describes common statements for instance operations.

Show Instances and 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: These statements are used to query information about instances that you created.

Parameters:

- startdate and enddate: the beginning and end of the time range to query. Information about the instances created within the specified period is returned. The dates must be in the format of yyyy-mm-dd. These parameters are optional. If they are not specified, information about instances that you created in the last three days is returned.
- number: the number of instances to return. 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: indicates that information about the instances executed in the current project is returned. By default, a maximum of 50 records are returned. You can execute this statement only when you have the LIST permission on the current project. To return more records, you must use the -limit number option. For example, you can use `show p -all -limit 100` to return information about 100 instances executed in the current project.
- project name: the name of the project. The account that you use must be a member of the project.

Output fields include StartTime, RunTime, Status, InstanceID, and Query. The values of StartTime and RunTime are in seconds. The Query field indicates the SQL statement that corresponds to the instance. The following code provides an example of the output:

Execute the following statement to query the information about instances created by the current user:

```
show p;
```

The following information is returned:

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

An instance can be in one of the following states:

- Running: The instance is running.
- Success: Instance operations are complete.

- Waiting: The instance is waiting to run.
- Failed: The job failed, but the data in the destination table is not modified.
- Suspended: The instance is suspended.
- Cancelled: The instance is stopped.

Status Instance

Syntax:

```
status instance_id;
```

Description: This statement is used to query the status of a specific instance. Valid values: Success, Failed, Running, and Cancelled.

> **Note** If the instance is not created by the current user, an error is returned.

Parameter: instance_id indicates the ID of the instance whose status you want to query. This parameter uniquely identifies an instance.

Example:

Execute the following statement to query the status of the instance whose ID is 20131225123xxxxxxxxxxxxxx.

```
status 20131225123xxxxxxxxxxxxxx;
```

The following information is returned:

```
Success
```

Top Instance

Syntax:

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

> **Note** Only project owners or administrators can execute the statement.

Description:

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

Kill Instance

Syntax:

```
kill <instance_id>;
```

Description: This statement is used to stop an instance. You can stop an instance only in the Running state. Note that this is an abnormal process. The execution result of this statement only means that the system has received the request. It does not mean that the job has been stopped. Therefore, you must use `status` to query the instance status.

Parameter: instance_id indicates the ID of the instance. This parameter uniquely identifies an instance. It must be the ID of a running instance. Otherwise, an error is returned.

Example:

```
kill 20131225123xxxxxxxxxxxxxx;
-- Stop the instance whose ID is 20131225123xxxxxxxxxxxxxx.
```

Desc Instance

Syntax:

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

Description: This statement is used to query job information based on the instance ID. Output fields include Query, Owner, Startime, Endtime, and Status.

Parameter: instance_id indicates the ID of the instance. This parameter uniquely identifies an instance.

Example:

-- Execute the following statement to query information about the job that corresponds to the instance whose ID is 20150715xxxxxxxxxxxxxx.

```
desc instance 20150715xxxxxxxxxxxxxx;
```

The following information is returned:

```
ID                                  20150715xxxxxxxxxxxxxx
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;
```

Wait Instance

Syntax:

```
wait instance_id;
```

Description: This statement is used to obtain the operational logs of the job based on the instance ID. The returned information includes a Logview URL. You can log on to Logview to view log details.

> **Note** You can obtain the Logview URL of an instance created for more than three days. However, you cannot use the URL to go to Logview because it expires and is cleared.

Parameter: instance_id indicates the ID of the instance. This parameter uniquely identifies an instance.

Sample statements:

Execute the following statement to query the operational logs of the job that corresponds to the instance whose ID is 20170925161122379gxxxxxx.

```
wait 20170925161122379gxxxxxx;
```

The following information is returned:

```
ID = 20170925161122379g3xxxxxx
Log view:
http://logview.odps.aliyun.com/logview/?h=http://service.odps.aliyun.com/api&p=alian&i=2017092516112
23xxxxxxdqp&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)
```

1.5.2.5. Resource operations

This topic describes common statements for resource operations.

Add a resource

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: The statements are used to add resources to MaxCompute.

Parameter	Description
file/archive/table/jar/py	The type of the resource.
local_file	The path of the file that you want to add. The file name is used as the resource name. A resource name is a unique identifier of a resource.
table_name	The name of the MaxCompute table.
[partition (spec)]	If the resource that you want to add is a partitioned table, MaxCompute takes only a partition, not the whole partitioned table, as a resource.
alias	The name of the resource. If this parameter is not specified, the file name is used as the resource name. This feature does support JAR and PY resources.
[comment 'cmt']	The comment of the resource.
[-f]	If a resource with the same name exists, this operation overwrites the existing resource. If this parameter is not specified and a resource with the same name exists, the operation fails.

Examples:

-- Add a table resource whose alias is sale.res to MaxCompute.

```
add table sale_detail partition (ds='20170602') as sale.res comment 'sale detail on 201706 02' -f;
```

The following result is returned:

```
OK: Resource 'sale.res' have been updated.
```

Note The size of each resource file cannot exceed 500 MB. The total size of resource files referenced by a single SQL or MapReduce job cannot exceed 2,048 MB.

Delete a resource

Syntax:

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

Description: This statement is used to delete an existing resource.

Parameters:

resource_name: the name of the resource that you want to delete.

Examples:

Execute the following statement to delete an existing resource:

```
DROP RESOURCE getaddr.jar;
```

The following result is returned:

```
Confirm to "DROP RESOURCE getaddr.jar" (yes/no)? y
-- Enter y to confirm your input.
OK
```

Query resources

Syntax:

```
LIST RESOURCES;
```

Description: This statement is used to query all resources in the current project.

Examples:

Execute the following statement to query all resources in the current project:

```
list resources;
```

The following result is returned:

```
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

Syntax:

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

Description: This statement is used to download a resource from MaxCompute to your computer. You can download file, JAR, archive, and PY resources, but not table resources.

Parameters:

- resource_name: the name of the resource that you want to download.
- path: the path to save the resource.

Examples:

Execute the following statement to download a resource from MaxCompute to your computer:

```
get resource odps-udf-examples.jar d:\;
```

The following result is returned:

```
OK
```

1.5.2.6. Function operations

This topic describes common statements for function operations.

Create a function

Syntax:

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

The following table describes the parameters in this statement.

Parameter	Description
function_name	The name of the user-defined function (UDF). 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 class name for UDF implementation. For a Python UDF, package_to_class indicates the Python script name and class name. The value must be enclosed in a pair of single quotation marks (').
resource_list	The list of resources that the UDF uses. The list must include the resource where the UDF code is located. Make sure that the resources are 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.

Examples:

- 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 test_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 project owners have the permissions to use UDFs to overwrite built-in functions. If you use a UDF that overwrites a built-in function, warning information is included in the Summary parameter after the SQL statement is executed.

Delete a function

Syntax:

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

Parameters:

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

Examples:

```
DROP FUNCTION test_lower;
```

Query functions

Syntax:

```
LIST FUNCTIONS;
-- Query all UDFs in the current project.
LIST FUNCTIONS -p project_name;
-- Query all UDFs in a specific project.
```

1.5.2.7. Time zone configuration operations

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

Time zone configurations

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

- Session level: Run the `set odps.sql.timezone=<timezoneid>;` command along with a computing statement.

  Examples:

  Run the following command to set the time zone to Asia/Tokyo:

  ```
  set odps.sql.timezone=Asia/Tokyo;
  ```

  Run the following command to query the current time zone:

  ```
  select getdate();
  ```

  The following result is returned:

  ```
  +------------+
  | _c0        |
  +------------+
  | 2018-10-30 23:49:50 |
  +------------+
  ```

- Project level: Run the `setProject odps.sql.timezone=<timezoneid>;` command as the project owner.

  > **Notice** After the time zone of a project is configured, it is used for all time computing, and the data of existing jobs is affected. Therefore, exercise caution when you perform this operation. We recommend that you set time zones only for new projects.

Limits and usage notes

- SQL built-in date functions, user-defined functions (UDFs), user-defined types (UDTs), user-defined joins (UDJs), and SELECT TRANSFORM allow you to obtain the timezone attribute of a project to configure the time zone.
- 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 into STRING.

- After you configure the time zone, the output time may be different from the real time after 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.
- MaxCompute, SDK for Java, and the related client are updated to ensure that DATETIME data stored in MaxCompute is accurate across multiple time zones. The updated versions of the client and SDK have the -oversea suffix. The update may affect the display of DATETIME data earlier than January 1, 1928 in MaxCompute.
- If the time zone is not UTC+8, we recommend that you update SDK for Java and the client. This ensures that the SQL-based computing results and data transferred by using Tunnel after January 1, 1900 are accurate and consistent. After the update, for the DATETIME data before January 1, 1900, the SQL-based computing results and data transferred by 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 update, the time in the new version is 352 seconds earlier.
- If you do not update SDK for Java and the client to the version with the -oversea suffix, the SQL-based computing results and data transferred by using Tunnel still 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 modification of time zone configurations for the new client or SDK for Java 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 configure the time zone on the client to ensure time consistency between the client and the server.
- MaxCompute MapReduce, Machine Learning Platform for AI, and MaxCompute Graph support the time zone configuration.
- MaxCompute Spark supports the time zone configuration.
  - If tasks are submitted to a 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 spark.driver.extraJavaOptions -Duser.timezone=America/Los_Angeles. The timezone parameter indicates the time zone that you want to use.

1.5.2.8. Tunnel operations

This topic describes common commands for Tunnel operations.

Use of Tunnel commands

You can run the `tunnel help;` subcommand on the MaxCompute client to query help information. The following result is returned:

```
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.
```

Parameters:

- upload: uploads data to a MaxCompute table. You can upload files or level-1 directories to only one table or partition each time. For a partitioned table, you must specify the partition to which you want to upload data. For a multi-level partitioned table, you must specify the last-level partition.

```
tunnel upload log.txt test_project.test_table/p1="b1",p2="b2";
-- Upload data in the log.txt file to the p1="b1" and p2="b2" partitions of the test_table table t
hat has two levels of partitions in the test_project project.
tunnel upload  log.txt  test_table --scan=true;
-- Upload data in the log.txt file to the test_table table. The scan parameter is used to check wh
ether data in the log.txt file complies with the schema of the test_table table. If it does not, t
he system reports an error and stops the upload.
```

- download: downloads data from a MaxCompute table. You can download data from only one table or partition to a file each time. For a partitioned table, you must specify the partition from which you want to download data. If the table has multiple levels of partitions, you must specify the last-level partition.

```
tunnel download test_project.test_table/p1="b1",p2="b2" test_table.txt;
-- Download data from the test_project.test_table table that has two levels of partitions to the t
est_table.txt file.
```

- resume: resumes the transfer of files or directories. The transfer is interrupted because your network is disconnected or Tunnel is faulty. You can use this command to resume only data uploads. One data download or upload is referred to as a session. You must specify the session ID in the RESUME command before you run this command.

```
tunnel resume;
```

- show: shows historical task information.

```
tunnel show history -n 5;
-- Show the commands used in the last five data uploads or downloads.
tunnel show log;
-- Show the logs of the last data upload or download.
```

- purge: clears a session directory. By default, sessions from the last three days are cleared.

```
tunnel purge 5;
--Clear logs from the last five days.
```

- help: obtains help information.

Upload

Description: This command is used to upload data to a MaxCompute table in append mode.

You can run the `tunnel help upload;` subcommand to query help information. The following result is returned:

```
usage: tunnel upload [options] <path> <[project.]table[/partition]>
              upload data from local file
 -acp,-auto-create-partition <ARG>   auto create target partition if not
                                     exists, default false
 -bs,-block-size <ARG>               block size in MiB, default 100
 -c,-charset <ARG>                   specify file charset, default ignore.
                                     set ignore to download raw data
 -cf,-csv-format <ARG>               use csv format (true|false), default
                                     false. When uploading in csv format,
                                     file splitting not supported.
 -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)
 -ow,-overwrite <true | false>       overwrite specified table or
                                     partition, default: false
 -rd,-record-delimiter <ARG>         specify record delimiter, support
                                     unicode, eg \u0001. default "\r\n"
 -s,-scan <ARG>                      specify scan file
                                     action(true|false|only), default true
 -sd,-session-dir <ARG>              set session dir, default
                                     D:\software\odpscmd_public\plugins\dship
 -ss,-strict-schema <ARG>            specify strict schema mode. If false,
                                     extra data will be abandoned and
                                     insufficient field will be filled
                                     with null. Default true
 -t,-threads <ARG>                   number of threads, default 1
 -te,-tunnel_endpoint <ARG>          tunnel endpoint
 -time,-time <ARG>                   keep track of upload/download elapsed
                                     time or not. Default false
 -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 the parameters in this statement.

Parameter	Description
path	The path and name of the file that you want to upload.
[project.]table[/partition]	The name of the table to which you want to upload data. You must specify the last-level partition for a partitioned table. If the table does not belong to the current project, you must specify the project where the table is located.
-acp	The partition to which you want to upload data. If the specified partition does not exist, a partition is automatically created. Default value: False.

Parameter	Description
-bs	The size of the data block uploaded by Tunnel each time. Default value: 100 MiB. 1 MiB = 1024 × 1024 bytes.
-c	The encoding format of the data file. The default value is UTF-8 without timing. By default, the source data is downloaded.
-cf	Specifies whether the file is a CSV file. Default value: False.
-cp	Specifies whether to compress the file before you upload it to MaxCompute to reduce network traffic. Default value: True.
-dbr	Specifies whether to omit dirty data, such as additional columns, missing columns, or unmatched types of column data. The value True indicates that all data that does not match the schema of the table is omitted. The value False indicates that an error is returned when dirty data is detected. This ensures that raw data in the table to which you want to upload data is not contaminated. Default value: False.
-dfp	The format of DATETIME data. The default format is yyyy-MM-dd HH:mm:ss. If you want to specify the DATETIME data that is accurate to the millisecond, the format `tunnel upload -dfp 'yyyy-MM-dd HH:mm:ss.SSS'` can be used.
-fd	The column delimiter for the data file. The default value is a comma (,).
-h	Specifies whether the data file has table headers. If this parameter is set to True, the data from the second row in the file is uploaded. Default value: False.
-mbr	The maximum number of dirty data records. If the number of dirty data records exceeds the value of this parameter, the upload stops. Default value: 1000.
-ni	The NULL data identifier. The default value is an empty string.
-ow	Specifies whether the uploaded data overwrites the table or partition. Default value: False, which indicates that data is appended.
-rd	The row delimiter for the data file. The default value is \n in a Linux operating system or \r\n in a Windows operating system.
-s	Specifies whether to scan the data file. Default value: True. The value True indicates that the system scans the data and starts to import it only if the data is in the correct format. The value False indicates that the system imports data without scanning. The value Only indicates that the system only scans the data but does not import it.
-sd	The path of the session directory. Default value: *D:/console/plugins/dship/lib/...*
-ss	The strict schema mode. Default value: True. If the parameter is set to False, extra data is discarded and the fields that are not specified are filled with NULL.
-t	The number of threads. Default value: 1.
-te	The endpoint of Tunnel.

Parameter	Description
-time	Specifies whether the upload time is tracked. Default value: False.
-tz	The time zone. By default, the local time zone is used. Example: Asia/Shanghai.

Example:

1. Execute the following statement to create a partitioned table named sale_detail:

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

2. Execute the following statement to add a partition to the sale_detail table:

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

3. Prepare the data.txt file that contains the following data and save the file in the d:\data.txt directory:

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

> **Note** In this example, the third row of the data.txt file does not comply with the schema of the sale_detail table. The sale_detail table defines three columns, but the third row of this file contains only two columns.

4. Run the following command to upload the data.txt file to the sale_detail table.

```
tunnel u d:\data.txt sale_detail/sale_date=201705,region=hangzhou -s false;
```

The data upload fails because the data.txt file contains dirty data. The system returns the session ID and an error message.

```
Upload session: 2017061016392248808 70a002ec60c
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
```

5. Execute the following statement to verify the data:

```
select * from sale_detail where sale_date='201312';
```

The data upload fails because dirty data is detected. As a result, the sale_detail table is empty.

```
ID = 20150610xxxxxxxxxxxvc61z5
+-----------+-------------+-------------+-----------+--------+
| shop_name | customer_id | total_price | sale_date | region |
+-----------+-------------+-------------+-----------+--------+
+-----------+-------------+-------------+-----------+--------+
```

6. Run the following command to query the ID of the failed session.

```
tunnel show history;
```

The following result is returned:

```
20150610xxxxxxxxxxx70a002ec60c  failed  'u --config-file /D:/console/conf/odps_config.ini --proj
ect odpstest_ay52c_ay52 --endpoint http://service.cn-shanghai.maxcompute.aliyun.com/api --id Ulx
xxxxxxxxxxrI1 --key 2m4r3WvTxxxxxxxxxx0InVke7UkvR d:\data.txt sale_detail/sale_date=201312,regio
n=hangzhou -s false'
```

7. Modify the data.txt file to comply with the schema of the sale_detail table.

```
shopx,x_id,100
shopy,y_id,200
```

8. Run the RESUME command to resume the data upload. In the command, 20150610xxxxxxxxxx70a002ec60c is the ID of the failed session.

```
tunnel resume 20150610xxxxxxxxxx70a002ec60c --force;
```

The following information is returned:

```
start resume
20150610xxxxxxxxxx70a002ec60c
Upload session: 20150610xxxxxxxxxx70a002ec60c
Start upload:d:\data.txt
Resume 1 blocks
2015-06-10 16:46:42     upload block: '1'
2015-06-10 16:46:42     upload block complete, blockid=1
upload complete, average speed is 0 KB/s
OK
```

9. Execute the following statement to verify the data:

```
select * from sale_detail where sale_date='201312';
```

If the following result is returned, the upload succeeds.

```
ID = 20150610xxxxxxxxxxa741z5
+-----------+-------------+-------------+-----------+--------+
| shop_name | customer_id | total_price | sale_date | region |
+-----------+-------------+-------------+-----------+--------+
| shopx     | x_id        | 100.0       | 201312    | hangzhou|
| shopy     | y_id        | 200.0       | 201312    | hangzhou|
+-----------+-------------+-------------+-----------+--------+
```

Download

Description: This command is used to download MaxCompute table data or the execution result of a specific instance to your computer.

You can run the `tunnel help download;` subcommand to query help information. The following result is returned:

```
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
 -cf,-csv-format <ARG>            use csv format (true|false), default
                                  false. When uploading in csv format,
                                  file splitting not supported.
 -ci,-columns-index <ARG>         specify the columns index(starts from
                                  0) to download, use comma to split each
                                  index
```

```
  -cn,-columns-name <ARG>           specify the columns name to download,
                                    use comma to split each name
  -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 "\r\n"
  -sd,-session-dir <ARG>            set session dir, default
                                    D:\software\odpscmd_public\plugins\dship
  -t,-threads <ARG>                 number of threads, default 1
  -te,-tunnel_endpoint <ARG>        tunnel endpoint
  -time,-time <ARG>                 keep track of upload/download elapsed
                                    time or not. Default false
  -tz,-time-zone <ARG>              time zone, default local timezone:
                                    Asia/Shanghai
usage: tunnel download [options] instance://<[project/]instance_id> <path>
               download instance result to local file
  -c,-charset <ARG>                 specify file charset, default ignore.
                                    set ignore to download raw data
  -cf,-csv-format <ARG>             use csv format (true|false), default
                                    false. When uploading in csv format,
                                    file splitting not supported.
  -ci,-columns-index <ARG>          specify the columns index(starts from
                                    0) to download, use comma to split each
                                    index
  -cn,-columns-name <ARG>           specify the columns name to download,
                                    use comma to split each name
  -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 "\r\n"
  -sd,-session-dir <ARG>            set session dir, default
                                    D:\software\odpscmd_public\plugins\dshi
  -t,-threads <ARG>                 number of threads, default 1
```

```
 -te,-tunnel_endpoint <ARG>        tunnel endpoint
 -time,-time <ARG>                 keep track of upload/download elapsed
                                   time or not. Default false
 -tz,-time-zone <ARG>              time zone, default local timezone:
                                   Asia/Shanghai
Example:
    tunnel download test_project.test_table/p1="b1",p2="b2" log.txt // Download data from a specific
table.
    tunnel download instance://test_project/test_instance log.txt   // Download the execution result
of a specific instance.
```

The following table describes the parameters in this command.

Parameter	Description
path	The path in which the downloaded data file is saved.
[project.]table[/partition]	The name of the table that you want to download. You must specify the last-level partition for a partitioned table. If the table does not belong to the current project, you must specify the project where the table is located.
[project/]instance_id	The ID of the instance. You must specify this parameter to download the execution result of a specific instance.
-fd	The column delimiter for the data file. The default value is a comma (,).
-rd	The row delimiter for the data file. Default value: \r\n.
-dfp	The format of DATETIME data. The default format is yyyy-MM-dd HH:mm:ss.
-ni	The NULL data identifier. The default value is an empty string.
-c	The encoding format of the data file. Default value: UTF-8.
-cf	Specifies whether the file is a CSV file. Default value: False.
-ci	Specifies that column indexes (starting from 0) are downloaded. Separate column indexes with commas (,).
-cp	The name of the column to download. Separate multiple column names with commas (,).
-e	Specifies whether the data of the DOUBLE type you want to download is represented by an exponential function if required. If the data is not represented by an exponential function, a maximum of 20 bits are retained. Default value: False.
-h	Specifies whether the data file has table headers. Default value: False. If this parameter is set to True, the data from the second row in the file is downloaded. **Note** -h=true and threads>1 cannot be used together. threads>1 indicates multiple threads.
-limit	The number of files you want to download.
-tz	The time zone. By default, the local time zone is used. Example: Asia/Shanghai.
-time	Specifies whether the upload time is tracked. Default value: False.

Parameter	Description
-te	The endpoint of Tunnel.
-t	The number of threads. Default value: 1.
-sd	The path of the session directory. Default value: *D:/console/plugins/dship/lib/...*

Example:

1. Execute the following statement to query the sale_detail table:

   ```
   select * from sale_detail;
   ```

   The following information is returned:

   ```
   ID = 20170724071705393ge3csfb8
   ... ...
   ```

2. Run the following command to download the execution results to a file on the on-premises machine:

   ```
   tunnel download instance://20170724071705393ge3csfb8 result;
   ```

   If the following result is returned, the download succeeds.

   ```
   2017-07-24 15:18:47  -  new session: 2017072415184785b6516400090ca8    total lines: 8
   2017-07-24 15:18:47  -  file [0]: [0, 8), result
   downloading 8 records into 1 file
   2017-07-24 15:18:47  -  file [0] start
   2017-07-24 15:18:48  -  file [0] OK. total: 44 bytes
   download OK
   ```

After you enable use_instance_tunnel on the MaxCompute client, you can use InstanceTunnel to download the query results that are returned by SELECT statements. This helps you download query results of any size at any time. You can use one of the following methods to enable this feature:

- On the client, set use_instance_tunnel to true in the odps_config.ini file and make sure that instance_tunnel_max_record is automatically set to 10000.

  ```
  # download sql results by instance tunnel
  use_instance_tunnel=true
  # the max records when download sql results by instance tunnel
  instance_tunnel_max_record=10000
  ```

  Note The instance_tunnel_max_record parameter specifies the maximum number of SQL result records that can be downloaded by using InstanceTunnel. If you do not specify this parameter, the number of result records that can be downloaded is not limited.

- Set `console.sql.result.instancetunnel` to true.

  ```
  set console.sql.result.instancetunnel=true;
  ```

Resume

Description: This command is used to resume historical operations. Only data uploads can be resumed.

You can run the `tunnel help resume;` subcommand to query help information. The following result is returned:

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

Parameters:

- session_id: the session ID of the failed data upload. You must specify this parameter.
- -f: specifies whether to forcibly resume historical operations. By default, this parameter is not specified.

Example:

Run the RESUME command to resume the session for which the upload failed. In this command, 20150610xxxxxxxxxx70a002ec60c is the ID of the session for which the upload failed.

```
tunnel resume 20150610xxxxxxxxxx70a002ec60c -force;
```

The following information is returned:

```
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
```

Show

Description: This command is used to show historical records.

You can run the `tunnel help show;` subcommand to query help information. The following result is returned:

```
Usage: tunnel show history [options]
              show session information
 -n,-number <ARG>   lines
Example:
       tunnel show history -n 5
       tunnel show log
```

Parameter: -n specifies the number of rows to be displayed.

Purge

Description: This command is used to clear a session directory.

You can run the `tunnel help purge;` subcommand to query help information. The following result is returned:

```
usage: tunnel purge [n]
              force session history to be purged.([n] days before, default
              3 days)
Example:
    tunnel purge 5
```

Parameter: n specifies after how many days historical logs are cleared. Default value: 3.

1.5.2.9. Other operations

This topic describes common statements for other operations.

ALIAS

The ALIAS statement is used to create an alias for a resource. You can create the same alias for different resources to read these resources from the MaxCompute MapReduce or UDF code without code modification.

Syntax:

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

Description: This statement is used to create an alias for a resource.

Parameters:

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

Sample statements:

```
ADD TABLE src_part PARTITION (ds='20171208') AS res_20171208;
ADD TABLE src_part PARTITION (ds='20171209') AS res_20171209;
-- Add the res_20171208 and res_20171209 resources.
ALIAS resName=res_20171208;
jar -resources resName -libjars work.jar -classpath ./work.jar com.company.MainClass args ...;
-- Set the alias of the res_20171208 resource 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 the res_20171209 resource 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: This statement is used to configure built-in or user-defined system variables of MaxCompute to affect MaxCompute execution rules.

Parameters:

- KEY: the name of the attribute that you want to set.
- VALUE: the value of the attribute.

MaxCompute SQL and the latest version of MaxCompute MapReduce support the following SET statements:

```
set odps.stage.mapper.mem=
-- Set the memory size of each mapper. Unit: MB. Default value: 1024.
set odps.stage.reducer.mem=
-- Set the memory size of each reducer. Unit: MB. Default value: 1024.
set odps.stage.joiner.mem=
-- Set the memory size of each joiner. Unit: MB. Default value: 1024.
set odps.stage.mem =
-- Set the total memory size of all workers in a specific MaxCompute job. This statement has a lower
priority than the preceding three statements. The value is in MB. No default value is defined.
set odps.stage.mapper.split.size=
-- Set the input data volume of each mapper to indirectly control the number of workers in each map
stage. The input data volume indicates the size of each slice in the input file. Unit: MB. Default v
alue: 256.
set odps.stage.reducer.num=
-- Set the number of workers in each reduce stage. No default value is defined.
set odps.stage.joiner.num=
-- Set the number of workers in each join stage. No default value is defined.
set odps.stage.num=
-- Modify the parallism of workers at all stages in a specific MaxCompute job. This statement has a
lower priority than the preceding three statements. No default value is defined.
```

The early versions of MaxCompute MapReduce support the following SET statements:

```
set odps.mapred.map.memory=
-- Set the memory size of each mapper. Unit: MB. Default value: 1024.
set odps.mapred.reduce.memory=
-- Set the memory size of each reducer. Unit: MB. Default value: 1024.
setodps.mapred.map.split.size=
-- Set the input data volume of each mapper to indirectly control the number of workers in each map
stage. The input data volume indicates the size of each slice in the input file. Unit: MB. Default v
alue: 256.
set odps.mapred.reduce.tasks=
-- Set the number of workers in each reduce stage. No default value is defined.
```

Examples:

- 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 size of data read by each mapper to 256 MB.

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

SetProject

Syntax

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

Description: This statement is used to configure project attributes. If <KEY>=<VALUE> is not specified, the current attribute configurations of the project are displayed.

The following table describes project attributes.

Attribute	Permission owner	Description	Value Range
odps.table.drop.ignorenonexistent	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 small tasks.	True and False
odps.instance.priority.level	Project owner	Indicates the priority of a task in a project.	1~3 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 by commas (,)
odps.table.lifecycle	Project owner	optional: The LIFECYCLE clause is optional in the statement that is used to create a table. If no lifecycle is set for a table, the table does not expire. mandatory: The LIFECYCLE clause is required. inherit: If no lifecycle is set for a table, the value of odps.table.lifecycle.value is the lifecycle of this table.	optional, mandatory, and inherit
odps.table.lifecycle.value	Project owner	Indicates the default lifecycle.	1~37231 Note The default value is 37231.
odps.instance.remain.days	Project owner	Indicates the retention period for instance information, in days.	3~30
odps.task.sql.outerjoin.ppd	Project owner	Indicates whether the filter conditions in FULL OUTER JOIN are pushed down.	True and False

Attribute	Permission owner	Description	Value Range
odps.function.strictmode	Project owner	Indicates whether to return NULL or report an error when dirty data is involved in the process of using built-in functions. The value False indicates that NULL is returned. The value True indicates that an error is returned.	True and False
odps.task.sql.write.str2null	Project owner	Indicates whether to consider empty strings as NULL. If the value is True, empty strings are considered NULL.	True and False

EXPORT

Syntax:

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

Description: This statement is used to export the metadata of a project to your computer. Metadata is represented by statements acceptable by the MaxCompute client (odpscmd). The exported metadata file can be used to recreate a project.

Show Flags

Syntax:

```
show flags;
```

Description: This statement is used to show the parameters configured by using SET statements.

> **Note** This statement takes effect only at the project level.

COST SQL

Syntax:

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

Description: This statement is used to estimate the costs of an SQL statement based on the size of the input data, the number of UDFs, and the SQL complexity.

Example:

Execute the following statement to estimate the costs of an SQL statement:

```
cost sql select distinct project_name, user_name from meta.m_security_users distribute by project_name sort by project_name;
```

The following information is returned:

```
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 you need to process terabytes of data but do not require real-time processing. It requires a relatively long time to prepare and submit jobs for MaxCompute SQL offline computing. Therefore, MaxCompute SQL is not well-suited for scenarios where thousands or even tens of thousands of transactions need to be processed per second. MaxCompute SQL online computing provides near real-time (NRT) processing capabilities.

The MaxCompute SQL syntax is similar to the SQL syntax. The MaxCompute SQL syntax can be considered a subset of standard SQL. MaxCompute SQL lacks many database features, such as transactions, primary key constraints, and indexes. Therefore, MaxCompute SQL cannot be regarded as a database. The maximum size of each SQL statement supported by MaxCompute is 2 MB.

1.6.1.2. Reserved words

The keywords of SQL statements are reserved words in MaxCompute. Do not use the reserved words when you name tables, columns, or partitions. If you use the reserved words, an error is reported. Reserved words are not case-sensitive.

This section lists the common reserved words. 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

This topic describes the advantages and disadvantages of MaxCompute partitioned tables.

You can specify partition key columns of a partitioned table as filter conditions in WHERE clauses of SELECT statements. This improves SQL query performance and reduces costs. However, some SQL statements for partition operations are less efficient. For example, if more than 10,000 partitions exist in an output table for a job, a large number of parallel operations are performed to write data to partitions. As a result, the SQL statements may fail.

If you use a partitioned table, you must properly evaluate the number and levels of partitions based on the data source to ensure query performance and stability. MaxCompute allows a maximum of six levels of partitions in a partitioned table.

For some statements, the syntax differs between partitioned and non-partitioned tables. For more information, see DDL statements and DML statements.

For more information about the statements that are used to create tables, see Create a table.

1.6.1.4. Type conversion

1.6.1.4.1. Explicit conversions of data types

An explicit conversion uses CAST to convert a value type into another one. This topic describes explicit conversions.

For more information about CAST, see CAST.

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

Explicit conversions

From/To	BIGINT	DOUBLE	STRING	DATETIME	BOOLEAN	DECIMAL
BIGINT	N/A	Y	Y	N	N	Y
DOUBLE	Y	N/A	Y	N	N	Y
STRING	Y	Y	N/A	Y	N	Y
DATETIME	N	N	Y	N/A	N	N
BOOLEAN	N	N	N	N	N/A	N
DECIMAL	Y	Y	Y	N	N	N/A

Y indicates that the conversion is supported. N indicates that the conversion is not supported. N/A indicates that the conversion is not required.

Note

- If the DOUBLE type is converted into the BIGINT type, the fractional part is truncated. Example: cast(1.6 as bigint) = 1.
- If the STRING type that meets the format of the DOUBLE type is converted into the BIGINT type, the STRING type is first converted into the DOUBLE type and then to the BIGINT type. Therefore, the fractional part is truncated. Example: cast("1.6" as bigint) = 1.
- If the STRING type that meets the format of the BIGINT type is converted into the DOUBLE type, one digit is retained after the decimal point. Example: cast("1" as double) = 1.0.
- To convert a constant string into the DECIMAL type, enclose the constant string within a pair of double quotation marks ("). If the value is not enclosed in quotation marks, it is considered a DOUBLE-type value. Example: cast("1.234567890123456789" as decimal).
- An unsupported explicit conversion causes an exception.
- If a conversion fails, the system returns an error and exits.
- The conversion of the DATETIME type uses the default format yyyy-mm-dd hh:mi:ss:ff3. For more information, see Conversion between string and datetime types.
- Some data types cannot be explicitly converted, but can be converted by using SQL built-in functions. For example, you can use the TO_CHAR function to convert the BOOLEAN type into the STRING type. For more information, see TO_CHAR. You can also use the TO_DATE function to convert the STRING type into the DATETIME type.
- If the values of the DECIMAL type are not within 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

An implicit type conversion allows MaxCompute to automatically convert data types based on the context and predefined rules. This topic describes the rules of implicit conversions.

The following table lists the rules of implicit conversions supported by MaxCompute.

Implicit 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
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

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

Implicit 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 conversion is supported. F indicates that the conversion is not supported.

> **Note**
>
> - If an unsupported implicit conversion is performed, an error is returned.
> - If a conversion fails, the system returns an error and exits.
> - MaxCompute automatically performs implicit conversions on data types based on the context. If the types do not match, you can use the CAST function to explicitly convert data types.
> - The rules of implicit conversions apply to specific scopes. In specific scenarios, only some rules take effect.

Implicit conversions with relational operators

Relational operators include =, <>, <, ≤, >, ≥, IS NULL, IS NOT NULL, LIKE, RLIKE, and IN. The rules for implicit conversions that use LIKE, RLIKE, and IN are different from those that use other relational operators. The rules described in this section do not apply to the three operators. The following table lists the rules of implicit conversions when relational operations are used for data of different types.

Implicit conversions with relational operators

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

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

Note

- If an implicit conversion is not supported between two values that you want to compare, the relational operation cannot be performed and an error is returned.
- For more information about relational operators, see Relational operators.

Implicit conversion with special relational operators

Special relational operators include LIKE, RLIKE, and IN.

Syntax of LIKE and RLIKE:

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

Take note of the following points for the two relational operators in implicit conversions:

- The source and pattern parameters of LIKE and RLIKE must be of the STRING type.
- Other types can neither be involved in the operation nor be implicitly converted into the STRING type.
- If the value of source or pattern is NULL, NULL is returned.

Syntax of IN:

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

Rules of implicit conversions for IN:

- The data types of the values on the right of the IN keyword must be consistent.
- MaxCompute automatically compares the values in the column specified by key and values on the right of the IN keyword. If these values are of the BIGINT, DOUBLE, and STRING types, the values are converted into the DOUBLE type for comparison. If these values are of the DATETIME and STRING types, the values are converted into the DATETIME type for comparison. Conversions between other types are not allowed.

Note The memory used by the compiler increases with the number of parameters used in the IN operation. If 5,000 parameters are used in an IN operation, the GCC compiler consumes 17 GB of memory for compilation. We recommend that you limit the number of parameters to 1,024. In this case, the compiler consumes a maximum of 1 GB of memory and requires 39 seconds for compilation.

Implicit conversions with arithmetic operators

Arithmetic operators include addition (+), subtraction (–), multiplication (×), division (/), and percent (%). This section describes the rules for implicit conversions that use these operators.

- Only the STRING, BIGINT, DECIMAL, and DOUBLE types can be used in arithmetic operations.

- Before an arithmetic operation, STRING data is implicitly converted into DOUBLE data.
- If an arithmetic operation involves data of the BIGINT and DOUBLE types, BIGINT data is implicitly converted into DOUBLE data.
- The DATETIME and BOOLEAN types cannot be used in arithmetic operations.

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

Implicit conversions with logical operators

Logical operators include AND, OR, and NOT. This section describes the rules for implicit conversions that use these operators.

- Only the BOOLEAN type can be used in logical operations.
- The other types are not supported by logical operations or implicit 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 built-in functions. These functions can be used to calculate one or more columns of a row and provide data of any type.

The following rules apply to implicit conversions:

- If you call a function and the data type of an input parameter is different from that defined in the function, the data type of the input parameter is converted into the function-defined data type.
- The parameters of each built-in function in MaxCompute SQL have different requirements for implicit conversions. For more information, see Built-in functions.

1.6.1.4.4. CASE WHEN

This topic describes the implicit conversion rules for CASE WHEN.

Implicit conversion rules for CASE WHEN:

- If return values are of the BIGINT and DOUBLE types, all the values are converted into the DOUBLE type.
- If return values include those of the STRING type, all the values are converted into the STRING type. If a conversion, such as conversion from BOOLEAN into STRING, fails, an error is returned.
- Conversions between other types are not allowed.

1.6.1.4.5. Conversions between the STRING and DATETIME types

MaxCompute supports conversions between the STRING and DATETIME types.

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

Valid values of time units

Time unit	String (not case-sensitive)	Valid value
Year	yyyy	0001 to 9999
Month	mm	01 to 12

Time unit	String (not case-sensitive)	Valid value
Day	dd	01 to 31
Hour	hh	00 to 23
Minute	mi	00 to 59
Second	ss	00 to 59
Millisecond	ff3	00 to 999

Note

- If the first digit of a valid value for each time unit is 0, 0 cannot be omitted. For example, 2017-1-9 12:12:12 is an invalid DATETIME format and it cannot be converted from the STRING type into the DATETIME type. It must be written as 2017-01-09 12:12:12.
- Only the STRING type that meets the preceding format requirements can be converted into the DATETIME type. For example, cast("2017-12-31 02:34:34" as datetime) converts 2017-12-31 02:34:34 of the STRING type into the DATETIME type. Similarly, if the DATETIME type is converted into the STRING type, the default conversion format is yyyy-mm-dd hh:mi:ss. Conversions that use the following or similar statements may fail, and errors are returned.

```
cast("2017/12/31 02/34/34" as datetime)
cast("20171231023434" as datetime)
cast("2017-12-31 2:34:34" as datetime)
```

MaxCompute provides the TO_DATE function to convert the STRING type that does not meet the DATETIME format requirements into the 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 unequal 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 than 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.

Operator	Description
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.
A LIKE B	If A or B is NULL, NULL is returned. A is a string and B is the pattern that you want to match. 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 regular 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 regular expression consisting of string constants. If A matches B, TRUE is returned. Otherwise, FALSE is returned. If B is an empty string, 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, namely, 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.

The values of the DOUBLE type in MaxCompute are different in precision. We recommend that you do not use the equal sign (=) for comparison between two values of the DOUBLE type. You can deduct a value of the DOUBLE type from the other value of the DOUBLE type and use the absolute value of the difference to determine whether the two values are equal. If the absolute value is negligible, the two values of the DOUBLE type are considered equal. Example:

```
abs(0.9999999999 - 1.0000000000) < 0.000000001
-- 0.9999999999 and 1.0000000000 have 10 decimal digits, whereas 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.
- In most cases, 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.

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. If both A and B are of the BIGINT type, a value of the DOUBLE type is returned.
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

- You can use only the values of the STRING, BIGINT, DOUBLE, or DECIMAL type to perform arithmetic operations. You cannot use the values of the DATETIME or BOOLEAN type to perform arithmetic operations.
- Values of the STRING type are implicitly converted into those of the DOUBLE type before arithmetic operations.
- If you use the values of the BIGINT and DOUBLE types to perform arithmetic operations, the value of the BIGINT type is implicitly converted into that of the DOUBLE type before arithmetic operations. A value of the DOUBLE type is returned.
- If both A and B are of the BIGINT type, a value of the DOUBLE type is returned after you perform the A/B operation. For other arithmetic operations, a value of the BIGINT type is returned.

1.6.2.3. Bitwise operators

This topic describes bitwise operators in MaxCompute SQL operators.

Bitwise operators

Operator	Description
A & B	The bitwise AND result of A and B is returned. For example, the result of 1&2 is 0, and the result of 1&3 is 1. The bitwise AND result of NULL and a value is NULL. Both A and B must be of the BIGINT type.
A | B	The bitwise OR result of A and B is returned. For example, the result of 1|2 is 3, and the result of 1|3 is 3. The bitwise OR result of NULL and a value is NULL. Both A and B must be of the BIGINT type.

Notice Bitwise operators do not support implicit type conversions. You can use the values only of the BIGINT type in bitwise operations.

1.6.2.4. Logical operators

This topic describes logical operators in MaxCompute SQL.

Logical operators

Operator	Description
	TRUE and TRUE = TRUE
	TRUE and FALSE = FALSE

Operator	Description
A and B	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 A is NULL, NULL is returned.
	If A is TRUE, FALSE is returned.
	If A is FALSE, TRUE is returned.

Note Logical operators do not support implicit type conversions. You can use only the values of the BOOLEAN type in logical operations.

1.6.3. LOAD

This topic describes how to use LOAD statements to import data from external storage, such as Object Storage Service (OSS), to a table or a table partition in MaxCompute.

Usage notes

MaxCompute must be authorized to access data in external storage. The methods used to authorize MaxCompute to access external data by using LOAD statements are the same as those used to authorize MaxCompute external tables. You can use one of the following methods for authorization:

- Specify the AccessKey ID and AccessKey secret in the directory in which data is stored. In this example, a directory on OSS is used.

```
'oss://<yourAccessKeyId>:<yourAccessKeySecret>@oss-cn-hangzhou-zmf.aliyuncs.com/my_bucket_id/my_location/'
```

> **Notice** If you use this method, the AccessKey ID and AccessKey secret are displayed in plaintext. This may pose security risks. We recommend that you do not use this method.

- Use Security Token Service (STS).

Use an extractor or a storage handler to import data

Syntax:

```
LOAD OVERWRITE|INTO TABLE table_name [PARTITION(partcol1=val1, partcol2=val2 ...)]
FROM LOCATION external_location
[STORED BY StorageHandler]
[WITH SERDEPROPERTIES (Options)];
```

Parameters:

- table_name: the name of the destination table into which you want to insert data. You must create a destination table before you insert data into it. The schema of the destination table must be the same as the format of the external data.
- LOAD INTO: inserts data into a table or partition.
- LOAD OVERWRITE: clears the existing data from a table or partition and inserts data into the table or partition.
- SORTED BY: specifies the name of the storage handler. You can use this clause in the same way as you use it for MaxCompute external tables.
- LOCATION: specifies the OSS directory in which the data files you want to read are saved. The system reads all files in the directory by default.
- WITH SERDEPROPERTIES: specifies the properties of the MaxCompute external table. The properties supported by WITH SERDEPROPERTIES are the same as those supported by the MaxCompute external table.

Assume that the vehicle.csv file contains the following data:

```
1,1,51,1,46.81006,-92.08174,9/14/2014 0:00,S
1,2,13,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,3,48,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,4,30,1,46.81006,-92.08174,9/14/2014 0:00,W
1,5,47,1,46.81006,-92.08174,9/14/2014 0:00,S
1,6,9,1,46.81006,-92.08174,9/14/2014 0:00,S
1,7,53,1,46.81006,-92.08174,9/14/2014 0:00,N
1,8,63,1,46.81006,-92.08174,9/14/2014 0:00,SW
1,9,4,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,10,31,1,46.81006,-92.08174,9/14/2014 0:00,N
```

Examples:

```
oss://${accessKeyId}:${accessKeySecret}@oss-cn-hangzhou-zmf.aliyuncs.com/bucket/data_location/
-- Save the vehicle.csv file to a folder in an OSS bucket. Obtain the structure of the directory where the MaxCompute external table is saved.
CREATE TABLE ambulance_data_csv_load (
vehicleId INT,
recordId INT,
patientId INT,
calls INT,
locationLatitute DOUBLE,
locationLongtitue DOUBLE,
recordTime STRING,
direction STRING );
-- Create a destination table named ambulance_data_csv_load.
LOAD OVERWRITE TABLE ambulance_data_csv_load
FROM
LOCATION 'oss://oss-cn-hangzhou-zmf.aliyuncs.com/bucket/data_location/'
STORED BY 'com.aliyun.odps.CsvStorageHandler'
WITH SERDEPROPERTIES (
'odps.properties.rolearn'='acs:ram::xxxxx:role/aliyunodpsdefaultrole',   -- The Alibaba Cloud Resource Name (ARN) of the AliyunODPSDefaultRole role.
'odps.text.option.delimiter'=','
);
-- Import the CSV file from OSS to the destination table.
```

Import data stored in an open source format

Syntax:

```
LOAD OVERWRITE|INTO TABLE table_name [PARTITION(partcol1=val1, partcol2=val2 ...)]
FROM LOCATION external_location
[ROW FORMAT SERDE '<serde class>'
  [WITH SERDEPROPERTIES ('odps.properties.rolearn'='${roleran}' [,'name2'='value2',...])]
]
STORED AS <file format>;
```

Parameters:

- table_name: the name of the destination table into which you want to insert data. You must create a destination table before you insert data into it. The schema of the destination table must be the same as the format of the external data.
- LOAD INTO: inserts data into a table or partition.
- LOAD OVERWRITE: clears the existing data from a table or partition and inserts data into the table or partition.
- STORED AS: specifies the format of the imported data file. Valid values include ORC, PARQUET, RCFILE, SEQUENCEFILE, and TEXTFILE. You can use this clause in the same way as you use it for MaxCompute external tables.
- ROW FORMAT SERDE: This clause is optional. You can use this clause in the same way as you use it for MaxCompute external tables. Mappings between file formats and SerDe classes:
  - SEQUENCEFILE: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
  - TEXTFILE: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
  - RCFILE: org.apache.hadoop.hive.serde2.columnar.LazyBinaryColumnarSerDe
  - ORC: org.apache.hadoop.hive.ql.io.orc.OrcSerde
  - ORCFILE: org.apache.hadoop.hive.ql.io.orc.OrcSerde
  - PARQUET: org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe
  - AVRO: org.apache.hadoop.hive.serde2.avro.AvroSerDe

- SERDEPROPERTIES: If you use STS to grant MaxCompute the required permissions, you must use this parameter to specify the value of the odps.properties.rolearn property. This property indicates the ARN of the AliyunODPSDefaultRole role. If the owner of MaxCompute and that of OSS use the same account, one-click authorization is allowed.

Examples:

```
oss://${accessKeyId}:${accessKeySecret}@oss-cn-hangzhou-zmf.aliyuncs.com/bucket/data_location/ds=202
00910/'
-- Assume that the OSS directory where the MaxCompute external table is saved uses the preceding str
ucture. The MaxCompute external table in the directory does not include partition key columns. The i
nformation of partition key columns is included in the directory information.
CREATE TABLE ambulance_data_csv_load_pt (
vehicleId STRING,
recordId STRING,
patientId STRING,
calls STRING,
locationLatitute STRING,
locationLongtitue STRING,
recordTime STRING,
direction STRING)
PARTITIONED BY (ds STRING);
-- Create a destination table named ambulance_data_csv_load_pt.
LOAD OVERWRITE TABLE ambulance_data_csv_load_pt PARTITION(ds='20200910')
FROM
LOCATION 'oss://<yourAccessKeyId>:<yourAccessKeySecret>@oss-cn-hangzhou-zmf.aliyuncs.com/bucket/data
_location/'
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
STORED AS TEXTFILE;
-- Import data stored in open source formats from OSS to the destination table.
```

Note If you download data to a partitioned table, the schema of the external table in the OSS directory does not include partition key columns. In the preceding example, the external table in the directory contains only the business fields. The table does not include partition key columns.

1.6.4. DDL statements

1.6.4.1. Table operations

1.6.4.1.1. Create a table

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

Syntax:

```
CREATE [EXTERNAL] 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], ...)]
[CLUSTERED BY | RANGE CLUSTERED BY (col_name [, col_name, ...]) [SORTED BY (col_name [ASC | DESC] [,
col_name [ASC | DESC] ...])] INTO number_of_buckets BUCKETS]  -- Specify the shuffle and sort attrib
utes when you create a clustered table.
[STORED BY StorageHandler] -- Used only for external tables.
[WITH SERDEPROPERTIES (Options)] -- Used only for external tables.
[LOCATION OSSLocation] -- Used only for external tables.
[STORED AS AliOrc]-- Specify the storage format of the table. You can specify AliORC only for newly
created internal tables.
[TBLPROPERTIES("compressionstrategy"="normal/high/extreme",-- Specify the compression policy for dat
a storage.
               "transactional"="true")] -- Specify a transactional table. You can then execute UPDAT
E or DELETE statements on the table.
[LIFECYCLE days];
CREATE TABLE [IF NOT EXISTS] table_name
[AS select_statement | LIKE existing_table_name];
```

Parameters:

- table_name and col_name: the table name and column name. The names are not case-sensitive. The names cannot contain special characters. A table or column name can contain only letters, digits, and underscores (_) and can be up to 128 bytes in length. We recommend that you start the table or column name with a letter. A table can contain a maximum of 1,200 columns.
- data_type: the type of data. Valid values: BIGINT, DOUBLE, BOOLEAN, DATETIME, DECIMAL, STRING, ARRAY<T>, and MAP<T1,T2>.

  > **Note** If you want to use new data types, such as TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAMP, and BINARY, you must enable these data types.
  >
  > - To enable new data types at the session level, add `set odps.sql.type.system.odps2=true;` before the SQL statement. Then, commit them for execution.
  > - To enable new data types at the project level, run the following command as the project owner:
  >
  >   ```
  >   setproject odps.sql.type.system.odps2=true;
  >   ```

- DEFAULT value: the default value of the column. If you do not specify the value of the column in an INSERT operation, the default value is used for the column.
- COMMENT table_comment: the comment of the table. The comment must be a valid string that can be up to 1,024 bytes in length.
- PARTITIONED BY (col_name data_type [COMMENT col_comment], ...): the partition field of the table. The following data types are supported: TINYINT, SMALLINT, INT, BIGINT, VARCHAR, and STRING.

  A partition value cannot contain double-byte characters. It must start with a letter, followed by letters, digits, or special characters. The value can be up to 128 bytes in length. The value can contain the following special characters: spaces, colons (:), underscores (_), dollar signs ($), number signs (#), periods (.), exclamation points (!), and at signs (@). Other characters, such as \t, \n, and /, are considered undefined characters. After you use partition fields to partition a table, a full table scan is not triggered when you add partitions, update partition data, or read partition data. This makes data processing more efficient.

  > **Note** A table can contain a maximum of 60,000 partitions at six or lower levels.

- CLUSTERED BY | RANGE CLUSTERED BY (col_name [, col_name, ...]) [SORTED BY (col_name [ASC | DESC] [, col_name

[ASC | DESC] ...])] INTO number_of_buckets BUCKETS: specifies the shuffle and sort attributes when you create a clustered table. Clustered tables are classified into hash-clustered tables and range-clustered tables.

- Hash-clustered tables
  - CLUSTERED BY: the hash key. MaxCompute performs hash operations on the specified columns and distributes data to each bucket based on the hash values. We recommend that you specify a column that has large value ranges and a small number of duplicate key-value pairs in CLUSTERED BY. This prevents data skew and hot spots and makes concurrent execution more efficient. To optimize JOIN operations, 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: specifies how the data in a bucket is sorted. To achieve better performance, we recommend that you set SORTED BY and CLUSTERED BY to the same value. If you specify the SORTED BY clause, MaxCompute automatically generates indexes and uses these indexes to speed up your queries.
  - INTO number_of_buckets BUCKETS: the number of hash buckets. This parameter is required and varies based on your data volume. By default, MaxCompute supports a maximum of 1,111 reducers. Therefore, MaxCompute supports a maximum of 1,111 hash buckets. You can run the `set odps.sql.reducer.instances=xxx;` command to increase the number of hash buckets. However, the maximum number of hash buckets is 4,000. If the value exceeds 4000, computing performance deteriorates.

    Note We recommend that you comply with the following rules when you specify the number of hash buckets:
    - Keep the size of each bucket around 500 MB. For example, if you want to add 1,000 buckets to a partition whose size is 500 GB, the size of each bucket is 500 MB on average. If a table contains large amounts of data, you can increase the size of each bucket from 500 MB to a size in the range of 2 GB to 3 GB. You can also run the `set odps.sql.reducer.instances=xxx;` command to set the number of hash buckets to a value greater than 1111.
    - To optimize JOIN operations, we recommend that you do not specify the shuffle and sort attributes. The number of hash buckets in one table must be the multiple of that in the other table. For example, one table has 256 hash buckets and the other table has 512 hash buckets. We recommend that you set the number of hash buckets to 2^n, such as 512, 1,024, 2,048, or 4,096. This way, MaxCompute can automatically split and merge hash buckets. To make the execution more efficient, we recommend that you do not specify the shuffle and sort attributes.

- Range-clustered tables
  - RANGE CLUSTERED BY: the column for range clustering. MaxCompute performs bucket operations on the specified columns and distributes data to each bucket based on the bucket numbers.
  - SORTED BY: specifies how the data in a bucket is sorted. You can use this clause in the same way you use it for a hash-clustered table.
  - INTO number_of_buckets BUCKETS: the number of hash buckets. Compared with hash-clustered tables, range-clustered tables have no limits on the number of buckets if the data is reasonably distributed. If you do not specify the number of buckets for a range-clustered table, MaxCompute automatically determines the optimal number of buckets based on your data volume.

  Note If join and aggregate operations are performed on range-clustered tables and the join or group key is the range clustering key or the prefix of the range clustering key, you can control flags to remove data shuffling. This makes the execution more efficient. You can use the `set odps.optimizer.enable.range.partial.repartitioning` flag to control data shuffling. Valid values are true and false. The default value is false, which indicates that data shuffling is disabled.

- STORED AS: 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 storage and improves read performance by more than 20%.

- LIFECYCLE: the lifecycle of the table, in days. If you execute the CREATE TABLE LIKE statement to create a table based on the schema of the source table, the lifecycle settings of the source table are not replicated.
- CREATE TABLE [IF NOT EXISTS] table_name [AS select_statement | LIKE existing_table_name]:
  - If you execute the `CREATE TABLE…AS select_statement…` statement to create a table, the data is replicated from the source table. However, the `CREATE TABLE…AS select_statement…` statement does not replicate the partition attributes of the source table. Partition key columns in the source table become standard columns in the destination table.
  - If you execute the `CREATE TABLE…LIKE existing_table_name` statement to create a table, the destination table has the same schema as the source table. However, the statement does not replicate data or lifecycle settings from the source table to the destination table.
- Clauses related to external tables:
  - STORED BY StorageHandler: the storage handler that is specified based on the format of data in the external table.
  - WITH SERDEPROPERTIES (Options): the parameters for authorization, compression, and character parsing in the external table.
  - LOCATION OSSLocation: the Object Storage Service (OSS) directory where the external table is saved.
- TBLPROPERTIE("compressionstrategy"="xxx"): the attribute of the transactional table. You can specify the lifecycle and comment attributes in this clause. compressionstrategy is the compression policy for data storage. The valid values are normal, high, and extreme. The compressionstrategy parameter must be in lowercase. However, the value of this parameter is not case-sensitive. This attribute applies only to the internal tables of MaxCompute.

  The default compression policy is normal for non-archived data and high for archived data.
  - Definitions of the valid values:
    - normal: indicates zstd level 1.
    - high: indicates zstd level 4.
    - extreme: indicates zstd level 9.
  - You can also specify the compression policy for a project. The project owner must use odps.storage.compression.strategy to specify the compression policy in the common attributes for project management. After this attribute is specified, the configuration applies to all generated non-archived data and does not affect the archive policy.
- TBLPROPERTIES("transactional"="true"): indicates that the table to create is a transactional table. You can execute the UPDATE or DELETE statement to update or delete data by row.

> **Note** If you do not specify IF NOT EXISTS in a table creation statement and another table with the same name exists, an error is returned. If you specify IF NOT EXISTS in a table creation statement, a success message is returned. The message is returned regardless of whether a 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 to create. In addition, the metadata of the existing table remains unchanged.

Examples for creating a partitioned table

The following example shows how to create a table named sale_detail to store sales records. The table has two partition key columns: sale_date and region. Syntax:

```
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.
```

CREATE TABLE...AS... example

You can execute the `CREATE TABLE...AS select_statement...` statement to create a table for which data is replicated from the source table. Example:

```
CREATE TABLE sale_detail_ctas1 AS
SELECT * FROM sale_detail;
```

If the sale_detail table contains data, all data in the sale_detail table is replicated to the sale_detail_ctas1 table after you execute the preceding statement.

> **Note** The sale_detail table is a partitioned table. When you execute the `CREATE TABLE...AS select_statement...` statement to create sale_detail_ctas1, partition attributes are not replicated and partition key columns in the sale_detail table are considered standard columns in the sale_detail_ctas1 table. sale_detail_ctas1 is a non-partitioned table that has five columns.

If the SELECT clause in the `CREATE TABLE...AS select_statement...` statement uses constants as column values, we recommend that you specify the names of columns. Example:

```
CREATE TABLE sale_detail_ctas2
AS
SELECT shop_name, customer_id, total_price, '2013' AS sale_date, 'China' AS region
FROM sale_detail;
```

Example without column aliases specified:

```
CREATE TABLE sale_detail_ctas3
AS
SELECT shop_name, customer_id, total_price, '2013', 'China'
FROM sale_detail;
```

CREATE TABLE...LIKE... example

If you want the source and destination tables to have the same schema, you can execute the `CREATE TABLE...LIKE...` statement. Example:

```
CREATE TABLE sale_detail_like LIKE sale_detail;
```

The schema of sale_detail_like is exactly the same as that of sale_detail. The 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.

Create clustered tables

Create hash-clustered tables:

```
CREATE TABLE T1 (a STRING, b STRING, c BIGINT) CLUSTERED BY (c) SORTED by (c) INTO 1024 BUCKETS; --
Create a hash-clustered non-partitioned table.
CREATE TABLE T1 (a STRING, b STRING, c BIGINT) PARTITIONED BY (dt STRING) CLUSTERED BY (c) SORTED by
(c) INTO 1024 BUCKETS; -- Create a hash-clustered partitioned table.
```

Create range-clustered tables:

```
CREATE TABLE T2 (a STRING, b STRING, c BIGINT) RANGE CLUSTERED BY (c) SORTED by (c) INTO 1024 BUCKET
S; -- Create a range-clustered non-partitioned table.
CREATE TABLE T2 (a STRING, b STRING, c BIGINT) PARTITIONED BY (dt STRING) CLUSTERED BY (c) SORTED by
(c) ; -- Create a range-clustered partitioned table. The number of buckets can be empty.
```

Create transactional tables

```
-- Create a non-partitioned transactional table.
create table txn_table(id bigint) tblproperties("transactional"="true");
-- Create a partitioned transactional table.
create table if not exists txn_table_pt(id bigint) partitioned by(ds string) tblproperties ("transac
tional"="true");
```

Query table information

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

```
DESC <table_name>;
-- Query table information.
DESC EXTENDED <table_name>;
-- Query information about an external table.
-- You can also check whether the table is a transactional table. If you want to check whether the t
able is a transactional table, you must upgrade the MaxCompute client to 0.35.0.
```

Query the statements that are used to create tables

MaxCompute allows you to execute the SHOW CREATE TABLE statement to query the statements that are used to create tables.

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

> **Note** You can use this statement to generate SQL DDL statements that are used to create tables. These DDL statements can be used to rebuild the table schema.

1.6.4.1.2. Delete a table

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

Syntax:

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

> **Note** If you do not specify IF EXISTS and the table does not exist, an error is returned. If this option is specified, a success message is returned regardless of whether the table exists.

Examples:

```
CREATE TABLE sale_detail_drop LIKE sale_detail;
DROP TABLE sale_detail_drop;
-- If the table exists, a success message is returned. If the table does not exist, an error is returned.
DROP TABLE IF EXISTS sale_detail_drop2;
-- A success message is returned regardless of whether the sale_detail_drop2 table exists.
```

1.6.4.1.3. Rename a table

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

Syntax:

```
ALTER TABLE table_name RENAME TO new_table_name;
```

> **Note**
>
> - The RENAME operation changes only the name of a table. Data in the table remains unchanged.
> - If a table that has the same name as the table specified by new_table_name exists, an error is returned.
> - If the table specified by table_name does not exist, an error is returned.

Examples:

```
CREATE TABLE sale_detail_rename1 LIKE sale_detail;
ALTER TABLE sale_detail_rename1 RENAME TO sale_detail_rename2;
```

1.6.4.1.4. Change the owner of a table

This topic describes how to use a DDL statement to change the owner of a table.

MaxCompute SQL allows you to execute the `CHANGEOWNER` statement to change the owner of a table.

Syntax:

```
ALTER TABLE table_name CHANGEOWNER TO 'ALIYUN$xxx@aliyun.com';
```

1.6.4.1.5. Modify the comment of a table

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

Syntax:

```
ALTER TABLE table_name SET COMMENT 'tbl comment';
```

> **Note**
>
> - The table specified by table_name must exist.
> - The value of comment can be up to 1,024 bytes.

Examples:

```
ALTER TABLE sale_detail SET COMMENT 'new coments for table sale_detail';
```

You can execute the DESC statement in MaxCompute to view the comment of the table after modification.

1.6.4.1.6. Modify the lifecycle settings of a table

MaxCompute provides the lifecycle management feature to facilitate storage release and simplify data reclamation. This topic describes how to use a DDL statement to modify the lifecycle settings of a table.

Syntax:

```
ALTER TABLE table_name SET lifecycle days;
```

Note

- days: the lifecycle duration, which must be a positive integer in days.
- table_name: the name of the table whose lifecycle settings you want to modify.
- You can specify the lifecycle when you create a table. The lifecycle can be specified only for a table rather than a partition. After a lifecycle is specified for a partitioned table, the lifecycle applies to the partitions of the table.
- If data in a table is modified, the value of LastDataModifiedTime is updated. Therefore, MaxCompute determines whether to reclaim this table based on the value of LastDataModifiedTime and the lifecycle settings.
- For a non-partitioned table, if the data remains unchanged for the duration specified by days after data is last modified, MaxCompute automatically executes a statement such as DROP TABLE to reclaim the table.
- For a partitioned table, MaxCompute determines whether to reclaim a partition based on the value of LastDataModifiedTime for the partition. Unlike non-partitioned tables, a partitioned table is not deleted even if all of the partitions are reclaimed.
- You can only modify the lifecycle settings instead of disabling the lifecycle for non-partitioned tables. You can disable the lifecycle for a specific partition in a partitioned table.

Examples:

```
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.
```

1.6.4.1.7. Disable or enable 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 use a DDL statement to disable or enable the lifecycle feature.

Syntax:

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

Note

- DISABLE LIFECYCLE: disables the lifecycle feature for a table or partition.
  - It prevents the reclamation of a table and its partitions based on the lifecycle feature. This parameter has a higher priority than partition_spec enable lifecycle. If you use table disable lifecycle, partition_spec enable lifecycle is invalid.
  - After the lifecycle feature of a table is disabled, the table lifecycle settings and the ENABLE and DISABLE tags of partitions in the table are retained.
  - After the lifecycle feature of a table is disabled, you can still modify the lifecycle settings of the table and its partitions.
- ENABLE LIFECYCLE: enables the lifecycle feature for a table or partition.
  - A table and its partitions can be reclaimed based on the lifecycle feature again. The lifecycle settings of the current table and its partitions are used by default.
  - Before you enable the lifecycle feature for a table, you can modify the lifecycle settings of the table and its partitions. This prevents data from being mistakenly reclaimed due to the use of previous settings.

Examples:

```
ALTER TABLE trans DISABLE LIFECYCLE;
-- Disable the lifecycle feature for the trans table.
ALTER TABLE trans PARTITION(dt='20141111') DISABLE LIFECYCLE;
-- Disable the lifecycle feature for the dt='20141111' partition in the trans table.
ALTER TABLE trans ENABLE LIFECYCLE;
-- Enable the lifecycle feature for the trans table.
ALTER TABLE trans PARTITION(dt='20141111') ENABLE LIFECYCLE;
-- Enable the lifecycle feature for the dt='20141111' partition in the trans table.
```

1.6.4.1.8. Modify the value of LastDataModifiedTime for a table

MaxCompute SQL allows you to perform the TOUCH operation to modify the value of LastDataModifiedTime for a table. This operation changes the value of LastDataModifiedTime for a table to the current time. This topic describes how to use a DDL statement to modify the value of LastDataModifiedTime for a table.

Syntax:

```
ALTER TABLE table_name TOUCH;
```

Note

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

1.6.4.1.9. Modify the clustering attributes of a table

This topic describes how to use a DDL statement to modify the clustering attributes of a table.

MaxCompute allows you to add or remove the clustering attributes of a partitioned table by using the `ALTER TABLE` statement.

> **Note**
>
> - The `ALTER TABLE` statement can modify only the clustering attributes of a partitioned table. The clustering attributes of a non-partitioned table cannot be modified after these attributes are added. The `ALTER TABLE` statement is applicable to the tables that already exist. After new clustering attributes are added, new partitions are stored based on the modified clustering attributes.
> - The `ALTER TABLE` statement affects only the new partitions of partitioned tables, including the partitions generated by using the INSERT OVERWRITE statement. New partitions are stored based on the new clustering attributes, while the clustering attributes and storage of existing partitions remain unchanged. After you specify the clustering attributes for a table, you can remove the clustering attributes and add clustering attributes for the table again. You can specify different clustering columns, sort columns, and numbers of buckets for new partitions.
> - The `ALTER TABLE` statement takes effect only on the new partitions of a table. Therefore, you cannot specify a partition in this statement.

Add hash clustering attributes to a table

Syntax:

```
ALTER TABLE table_name
[CLUSTERED BY (col_name [, col_name, ...]) [SORTED BY (col_name [ASC | DESC] [, col_name [ASC | DESC
] ...])] INTO number_of_buckets BUCKETS]
```

Remove hash clustering attributes from a table

Syntax:

```
ALTER TABLE table_name NOT CLUSTERED;
```

Add range clustering attributes to a table

If you do not specify the number of buckets, MaxCompute automatically determines the optimal number based on the data volume.

Syntax:

```
ALTER TABLE table_name
[RANGE CLUSTERED BY (col_name [, col_name, ...]) [SORTED BY (col_name [ASC | DESC] [, col_name [ASC
| DESC] ...])] INTO number_of_buckets BUCKETS]
```

Remove range clustering attributes from a table or partition

Syntax:

```
ALTER TABLE table_name NOT CLUSTERED;
ALTER TABLE table_name partition_spec NOT CLUSTERED;
```

1.6.4.1.10. Delete data from a non-partitioned table

This topic describes how to use a DDL statement to delete data from a non-partitioned table.

Syntax:

```
TRUNCATE TABLE table_name;
```

Note This statement is used to delete data from a specific non-partitioned table. To delete data from a partitioned table, execute the `ALTER TABLE table_name DROP PARTITION(partition_spec)` statement.

1.6.4.1.11. Archive table data

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

If a project does not have enough space and data needs to be compressed or deleted, you can use table archiving of MaxCompute to compress data by about 50%. Table archiving uses a compression algorithm with a high 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 compression ratio from 1:3 to 1:1.5. Table archiving allows data to consume only half of the physical space.

However, this feature comes at a price. If a data block or machine is faulty, it requires a long time to restore the data block, and read performance deteriorates. Therefore, this feature is suitable for the compression of cold data. For example, you can use this feature to store a large number of logs that are less used as RAID files for a long time.

Syntax:

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

Note

- If the table specified by table_name does not exist, an error is returned.
- If the partition specified by partition_name does not exist, an error is returned.

Examples:

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

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 the logical size and physical size after the data is archived. In the archiving process, multiple small files are automatically merged. After the data is archived, you can execute the `DESC EXTENDED` statement to check whether the data in the partition has been archived and view the usage of physical space:

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

1.6.4.1.12. Forcibly delete data from a table or partition

This topic describes how to use a DDL statement to forcibly delete data from a table or partition.

If you want to forcibly and irrecoverably delete data from a table or partition to immediately release storage, you can execute a DROP statement with the PURGE option.

Syntax:

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

Examples:

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

1.6.4.2. View-based operation

1.6.4.2.1. Create a view

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

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.
> - A view can contain only one valid SELECT statement.
> - A view can reference other views but cannot reference itself. Circular references are not supported.
> - You cannot write data to a view. For example, the `INSERT INTO` and `INSERT OVERWRITE` statements 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.
> - If the `CREATE VIEW` statement is executed without the IF NOT EXISTS option and the view already exists, an exception is returned. In this case, you can execute `CREATE VIEW or REPLACE VIEW` to recreate the view. The permissions on the view remain unchanged after the view is recreated.

Examples:

```
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;
-- Create a view named sale_detail_view.
```

1.6.4.2.2. Delete a view

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

Syntax:

```
DROP VIEW [IF EXISTS] view_name;
```

> **Note** If the partition does not exist and IF EXISTS is not specified, an error is returned.

Examples:

```
DROP VIEW IF EXISTS sale_detail_view;
```

1.6.4.2.3. Rename a view

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

Syntax:

```
ALTER VIEW view_name RENAME TO new_view_name;
```

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

Examples:

```
alter view sale_detail_view rename to market;
```

1.6.4.3. Column and partition operations

1.6.4.3.1. Add partitions

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

Syntax:

```
alter table table_name add [if not exists] partition partition_spec;
-- Add one partition at a time.
alter table table_name add [if not exists] partition partition_spec [PARTITION partition_spec PARTIT
ION partition_spec...] ;
-- Add multiple partitions at a time.
```

> **Note**
>
> - If you do not specify IF NOT EXISTS 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 partition values.

Examples:

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

1.6.4.3.2. Delete a partition

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

MaxCompute allows you to delete partitions that meet a specific filter condition. If you want to delete one or more partitions that meet a filter condition at a time, you can use an expression to specify the condition, use the condition to filter partitions, and then delete the partitions.

If you do not specify a filter condition, execute the following statements:

```
alter table table_name drop [if exists] PARTITION partition_spec;
-- Delete one partition at a time.
alter table table_name drop [if exists] PARTITION partition_spec,PARTITION partition_spec,[PARTITION
partition_spec....] ;
-- Delete multiple partitions at a time.
```

Note Syntax for partition_spec:

```
partition_spec:(partition_col1 = partition_col_value1, partition_col2 = partiton_col_value2, ...
)
```

If you specify a filter condition, execute the following statement:

```
ALTER TABLE table_name DROP [IF EXISTS] PARTITION partition_filtercondition;
-- Delete a partition based on the specified filter condition.
```

Note Syntax for PARTITION partition_filtercondition:

```
PARTITION partition_filtercondition
    : PARTITION (partition_col relational_operators partition_col_value)
    | PARTITION (scalar(partition_col) relational_operators partition_col_value)
    | PARTITION (partition_filtercondition1 AND|OR partition_filtercondition2)
    | PARTITION (NOT partition_filtercondition)
    | PARTITION (partition_filtercondition1)[,PARTITION (partition_filtercondition2), ...]
```

Parameters:

- table_name: the name of the partitioned table from which you want to delete a partition.
- IF EXISTS: If IF EXISTS is not specified and the partition does not exist, an error is returned.
- partition_spec: the name of the partition that you want to delete. The value is not case-sensitive. partition_col indicates the name of the partition key column, and partition_col_value indicates the value of the partition key column.
- PARTITION partition_filtercondition: the partition that you want to delete. The value is not case-sensitive.
  - partition_col: the name of the partition key column.
  - relational_operators: the relational operator.
  - partition_col_value: a value in the partition key column, which is a comparison value or regular expression. The data type of this value must be the same as that of the partition key column.
  - scalar(): the scalar function. This function generates a scalar based on the input value, processes the value in partition_col, and uses the relational operator specified by relational_operators to compare the processed value with partition_col_value.
  - The filter condition that is used to delete partitions supports the following logical operators: NOT, AND, and OR. You can use PARTITION (NOT partition_filtercondition) to obtain the complementary set of the condition. You can use PARTITION (partition_filtercondition1 AND|OR partition_filtercondition2) to obtain the condition that is used to match the partitions you want to delete.
  - Multiple PARTITION partition_filtercondition clauses are supported. If these clauses are separated by commas (,), OR is used for each clause to obtain the condition that is used to match the partitions you want to delete.

Limits:

- Each PARTITION partition_filtercondition clause can be used for only one partition key column.
- If you use an expression to specify PARTITION partition_filtercondition, only the built-in scalar function can be used for the expression.

Examples:

The filter condition is not specified.

```
-- Delete a partition from the sale_detail table. The partition stores the sales records of the China (Hangzhou) region in December 2017.
ALTER TABLE sale_detail DROP IF EXISTS PARTITION(sale_date='201712',region='hangzhou');
-- Delete two partitions from the sale_detail table at a time. The partitions store the sales records of the China (Hangzhou) and China (Shanghai) regions in December 2017.
ALTER TABLE sale_detail DROP IF EXISTS PARTITION(sale_date='201712',region='hangzhou'),PARTITION(sale_date='201712',region='shanghai');
```

The filter condition is specified.

```
-- Create a partitioned table.
CREATE TABLE IF NOT EXISTS sale_detail(
shop_name     STRING,
customer_id   STRING,
total_price   DOUBLE)
PARTITIONED BY (sale_date STRING);
-- Add partitions.
ALTER TABLE sale_detail ADD IF NOT EXISTS
PARTITION (sale_date= '201910')
PARTITION (sale_date= '201911')
PARTITION (sale_date= '201912')
PARTITION (sale_date= '202001')
PARTITION (sale_date= '202002')
PARTITION (sale_date= '202003')
PARTITION (sale_date= '202004')
PARTITION (sale_date= '202005')
PARTITION (sale_date= '202006')
PARTITION (sale_date= '202007');
-- Delete multiple partitions from the table at a time.
ALTER TABLE sale_detail DROP IF EXISTS PARTITION(sale_date < '201911');
ALTER TABLE sale_detail DROP IF EXISTS PARTITION(sale_date >= '202007');
ALTER TABLE sale_detail DROP IF EXISTS PARTITION(sale_date LIKE '20191%');
ALTER TABLE sale_detail DROP IF EXISTS PARTITION(sale_date IN ('202002','202004','202006'));
ALTER TABLE sale_detail DROP IF EXISTS PARTITION(sale_date BETWEEN '202001' AND '202007');
ALTER TABLE sale_detail DROP IF EXISTS PARTITION(substr(sale_date, 1, 4) = '2020');
ALTER TABLE sale_detail DROP IF EXISTS PARTITION(sale_date < '201912' OR sale_date >= '202006');
ALTER TABLE sale_detail DROP IF EXISTS PARTITION(sale_date > '201912' AND sale_date <= '202004');
ALTER TABLE sale_detail DROP IF EXISTS PARTITION(NOT sale_date > '202004');
-- Delete partitions by using a condition. The condition is specified by expressions that have the OR relationship.
ALTER TABLE sale_detail DROP IF EXISTS PARTITION(sale_date < '201911'), PARTITION(sale_date >= '202007');
-- Add partitions in other formats.
ALTER TABLE sale_detail ADD IF NOT EXISTS
PARTITION (sale_date= '2019-10-05');
PARTITION (sale_date= '2019-10-06')
PARTITION (sale_date= '2019-10-07');
-- Delete multiple partitions at a time and use regular expressions to match the partitions you want to delete.
ALTER TABLE sale_detail DROP IF EXISTS PARTITION(sale_date RLIKE '2019-\\d+-\\d+');
-- Create a table that has multi-level partitions.
CREATE TABLE IF NOT EXISTS region_sale_detail(
shop_name     STRING,
customer_id   STRING,
total_price   DOUBLE)
PARTITIONED BY (sale_date STRING , region STRING );
-- Add partitions.
ALTER TABLE region_sale_detail ADD IF NOT EXISTS
```

```
PARTITION (sale_date= '201910',region = 'shanghai')
PARTITION (sale_date= '201911',region = 'shanghai')
PARTITION (sale_date= '201912',region = 'shanghai')
PARTITION (sale_date= '202001',region = 'shanghai')
PARTITION (sale_date= '202002',region = 'shanghai')
PARTITION (sale_date= '201910',region = 'beijing')
PARTITION (sale_date= '201911',region = 'beijing')
PARTITION (sale_date= '201912',region = 'beijing')
PARTITION (sale_date= '202001',region = 'beijing')
PARTITION (sale_date= '202002',region = 'beijing');
-- Delete the partitions that meet the sale_date < '201911' and region = 'beijing' conditions from t
he table at a time.
ALTER TABLE region_sale_detail DROP IF EXISTS PARTITION(sale_date < '201911'),PARTITION(region = 'be
ijing');
```

If you execute the following statement, the following error is returned: `FAILED: ODPS-0130071:[1,82] Semantic analysis exception - invalid column reference region, partition expression must have one and only one column reference` .

```
-- An error is returned because PARTITION partition_filtercondition can be specified only for one pa
rtition key column.
ALTER TABLE region_sale_detail DROP IF EXISTS PARTITION(sale_date < '201911' AND region = 'beijing')
;
```

1.6.4.3.3. Add columns or comments

This topic describes how to use DDL statements to add columns or comments.

Syntax:

```
ALTER TABLE table_name ADD COLUMNS (col_name1 type1,col_name2 type2...) ;
-- Add columns.
ALTER TABLE table_name ADD COLUMNS (col_name1 type1 comment 'XXX',col_name2 type2 comment 'XXX');
-- Add both columns and comments.
```

Examples:

```
ALTER TABLE sale_detail ADD COLUMNS (customer_name STRING, education BIGINT);
-- Add two columns to the sale_detail table.
ALTER TABLE sale_detail ADD COLUMNS (customer_name STRING comment 'Customer', education BIGINT comme
nt 'Education' );
-- Add two columns and their comments to the sale_detail table.
```

1.6.4.3.4. Change the name of a column

This topic describes how to use a DDL statement to change the name of a column.

Syntax:

```
ALTER TABLE table_name CHANGE COLUMN old_col_name RENAME TO new_col_name;
```

> **Note**
> - The column specified by old_col_name must exist in the table.
> - The table cannot contain the column specified by new_col_name.

Examples:

```
ALTER TABLE sale_detail CHANGE COLUMN customer_name RENAME TO customer;
-- Change the name of a column in the sale_detail table.
```

1.6.4.3.5. Modify the comment of a column

This topic describes how to use a DDL statement to modify the comment of a column.

Syntax:

```
ALTER TABLE table_name CHANGE COLUMN col_name COMMENT 'comment_string';
```

> **Note**
> - The value of comment_string cannot exceed 1,024 bytes.
> - The data type and position of a column cannot be changed.
> - The column specified by col_name must exist.

Examples:

```
ALTER TABLE sale_detail CHANGE COLUMN customer COMMENT 'customer';
-- Modify the comment of a column in the sale_detail table.
```

1.6.4.3.6. Modify the name and comment of a column at the same time

This topic describes how to use a DDL statement to modify the name and comment of a column at the same time.

Syntax:

```
ALTER TABLE table_name CHANGE COLUMN old_col_name new_col_name column_type COMMENT 'column_comment';
```

> **Note**
> - The column specified by old_col_name must exist.
> - The table cannot contain the column specified by new_col_name.
> - The data type and position of a column cannot be changed.
> - The value of column_comment cannot exceed 1,024 bytes.

Examples:

```
ALTER TABLE sale_detail CHANGE COLUMN customer customer_name string COMMENT 'Customer';
-- Modify the name and comment of a column in the sale_detail table.
```

1.6.4.3.7. Change LastDataModifiedTime of a non-partitioned table or a partition in a partitioned table.

MaxCompute SQL allows you to execute the TOUCH operation to modify the value of LastDataModifiedTime for a non-partitioned table or a partition in a partitioned table. This operation changes the value of LastDataModifiedTime to the current time. This topic describes how to use a DDL statement to change the value of LastDataModifiedTime for a non-partitioned table or a partition in a partitioned table.

Syntax:

```
ALTER TABLE table_name TOUCH;
-- Change the value of LastDataModifiedTime for a non-partitioned table.
ALTER TABLE table_name TOUCH PARTITION(partition_col='partition_col_value', ...);
-- Change the value of LastDataModifiedTime for a partition in a partitioned table.
```

Note

- If the table specified by table_name does not exist, an error is returned.
- If the partition specified by partition_col='partition_col_value' does not exist, an error is returned.
- This operation changes the value of LastDataModifiedTime for a non-partitioned table or a partition in a partitioned table. In this case, MaxCompute considers a change to the table data and recalculates the lifecycle.

Sample statements:

```
ALTER TABLE result_table TOUCH;
-- Change the value of LastDataModifiedTime for a non-partitioned table.
ALTER TABLE sale_detail TOUCH PARTITION(sale_date='201712');
-- Change the value of LastDataModifiedTime for a partition in a partitioned table.
```

1.6.4.3.8. Modify partition values

This topic describes how to use a DDL statement to modify partition values.

Syntax:

```
ALTER TABLE table_name PARTITION (partition_col1 = 'partition_col_value1', partition_col2 = 'partito
n_col_value2', ...)
RENAME TO PARTITION (partition_col1 = 'partition_col_newvalue1', partition_col2 = 'partiton_col_newv
alue2', ...) ;
```

Note

- This statement can modify only the values rather than the names of partition key columns.
- To modify the values of one or more partitions in a table that has multi-level partitions, you must specify the partition values at each level.
- If the table specified by table_name does not exist, an error is returned.

Examples:

```
ALTER TABLE sale_detail PARTITION (sale_date = '201712', region = 'hangzhou') RENAME TO PARTITION (sale_date = '201710', region = 'beijing');
-- Modify the partition values in the sale_detail table.
```

1.6.4.3.9. Merge partitions

MaxCompute SQL allows you to execute the `MERGE PARTITION` statement to merge multiple partitions of a partitioned table into one partition. This operation deletes the dimension information about the merged partitions and transfers data to a specific partition. This topic describes how to use a DDL statement to merge partitions.

Syntax:

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

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

Limits and troubleshooting

- Extreme storage is not supported.
- Tables that depend on the file order are not supported, such as tables in Xlib or ALGO.
- External tables and table shards are not supported. After the partitions of a clustered table are merged, the merged partitions do not have the clustering attributes.
- Hash operations are performed by a catalog server on tables to merge partitions. A capacity limit is imposed on merged partitions. A hard link in Apsara Distributed File System can have a maximum of seven copies.
- You can merge a maximum of 4,000 partitions at a time.
- The number of partitions that can wait on a catalog server for merging is 10 million.
- If an error that indicates the catalog server is busy is returned, try again later.
- If a hard link in Apsara Distributed File System is faulty, clear the recycle bin and try again.

Examples:

```
odps@ jet_zwz>SHOW PARTITIONS intpstringstringstring;
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>DESC intpstringstringstring;
+------------+------------+------------+------------+
| value      | ds         | hh         | mm         |
+------------+------------+------------+------------+
| 1          | 20181101   | 00         | 00         |
| 1          | 20181101   | 00         | 10         |
| 1          | 20181101   | 10         | 00         |
| 1          | 20181101   | 10         | 10         |
+------------+------------+------------+------------+
-- Partitions and data of a partitioned table:
odps@ jet_zwz>ALTER TABLE intpstringstringstring MERGE PARTITION(hh='00') OVERWRITE PARTITION(ds='20
181101', hh='00', mm='00');
ID = 20190404025755844g80qwa7a
OK
-- Merge all partitions that meet the hh='00' condition into the ds=20181101/hh=00/mm=00 partition.
odps@ jet_zwz>SHOW PARTITIONS intpstringstringstring;
ds=20181101/hh=00/mm=00
ds=20181101/hh=10/mm=00
ds=20181101/hh=10/mm=10
OK
-- Query the partitions of the table after partition merge.
odps@ jet_zwz>DESC intpstringstringstring;
+------------+------------+------------+------------+
| value      | ds         | hh         | mm         |
+------------+------------+------------+------------+
| 1          | 20181101   | 00         | 00         |
| 1          | 20181101   | 00         | 00         |
| 1          | 20181101   | 10         | 00         |
| 1          | 20181101   | 10         | 10         |
+------------+------------+------------+------------+
-- Data in two partitions that meet the specified condition is merged into the specified partition.
```

1.6.5. DML statements

1.6.5.1. INSERT

1.6.5.1.1. Update data in a table

This topic describes how to use an INSERT statement to update data in a table.

The `INSERT OVERWRITE` and `INSERT INTO` statements are used to write the data processing results of MaxCompute SQL to a destination table.

The `INSERT INTO` statement inserts data into a table or partition. The `INSERT INTO` statement cannot be used to insert data into clustered tables. If you want to insert a small amount of test data, you can use this statement with VALUES.

The `INSERT OVERWRITE` statement clears the original data in a table or partition and inserts data into the table or partition. The `INSERT OVERWRITE` statement cannot be used to insert data into specific columns.

Syntax:

```
INSERT OVERWRITE|INTO TABLE table_name [PARTITION (partcol1=val1, partcol2=val2 ...)] [(col1,col2 ..
.)]
select_statement
FROM from_statement
[ZORDER BY zcol1 [, zcol2 ...]] ;
```

> **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, instead of directly followed by the table_name parameter.
> - To ensure data consistency during concurrent writes, MaxCompute uses the atomicity, consistency, isolation, durability (ACID) properties.
> - The mappings between the source and destination tables are based on the column sequence in the SELECT clause. The mappings between the column names of the tables are not considered.
> - If the destination table consists of static partitions and you want to insert data into a static partition, partition key columns cannot be included in the SELECT clause.

Parameters:

- table_name: the name of the table into which you want to insert data.
- PARTITION (partcol1=val1, partcol2=val2 ...): the name of the partition into which you want to insert data. The value must be a constant. It cannot be an expression, such as a function.
- [(col1,col2 ...)]: the names of the columns into which you want to insert data. This parameter is not supported in the `INSERT OVERWRITE` statement.
- select_statement: the SELECT clause used to query the data that you want to insert from the source table.
- from_statement: the FROM clause used to indicate the data source. For example, the value can be the name of the source table.
- [ZORDER BY zcol1 [, zcol2 ...]]: the columns that are used to sort data. If you write data to a table or partition, you can use this clause to place rows with similar data records in the columns specified in select_statement in adjacent positions. This improves filtering performance for queries and reduces storage costs.

  The `ORDER BY x, y` clause sorts data records based on the ordering of x coming before y. The `ZORDER BY x, y` clause places rows that have similar x values in adjacent positions and rows that have similar y values in adjacent positions. If the filter condition of an SQL query statement includes sort columns, the ORDER BY clause filters and sorts data based on x, whereas the ZORDER BY clause filters and sorts data based on x or on both x and y. ZORDER BY increases the column store ratio.

> **Note**
>
> - The mappings between the source and destination tables are based on the column sequence in the SELECT clause. The mappings between the column names of the tables are not considered.
> - If the destination table consists of static partitions and you want to insert data into a partition, partition key columns cannot be included in the SELECT clause.
> - The ZORDER BY clause occupies a large number of resources to write data. As a result, data writes based on ZORDER BY require a longer time than data writes without ordering.
> - If the destination table is a clustered table, ZORDER BY is not supported.
> - ZORDER BY can be used with DISTRIBUTE BY. However, ZORDER BY cannot be used with ORDER BY, CLUSTER BY, or SORT BY.

Examples:

Calculate the sales of different regions listed in the sale_detail table. Then, insert the obtained data into the sale_detail_insert table.

```
-- Create a partitioned table named sale_detail, add partitions, and insert data into the table. You
do not need to define partition key columns as standard columns in the statement that is used to cre
ate the source 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 partitions to the source table.
ALTER TABLE sale_detail ADD PARTITION (sale_date='2013', region='china');
-- Insert data into the source table. INSERT INTO TABLE table_name can be abbreviated as INSERT INTO
table_name. However, INSERT OVERWRITE TABLE table_name has no abbreviation.
INSERT INTO sale_detail PARTITION (sale_date='2013', region='china') VALUES ('s1','c1',100.1),('s2',
'c2',100.2),('s3','c3',100.3);
-- Create a destination table named sale_detail_insert that has the same schema as the source table.
CREATE TABLE sale_detail_insert LIKE sale_detail;
-- Add partitions to the destination table.
ALTER TABLE sale_detail_insert ADD PARTITION (sale_date='2013', region='china');
-- Extract data from the sale_detail table and insert the data into the sale_detail_insert table.
-- Fields in the destination table do not need to be declared or rearranged.
-- If the destination table consists of static partitions and partition fields have been declared in
PARTITION(), you do not need to include these fields in the SELECT clause. You only need to search f
or the fields based on the sequence of standard columns in the destination table and sequentially ma
p these fields to those in the destination table. If the destination table consists of dynamic parti
tions, partition fields must be included in the SELECT clause. For more information, see Insert data
into dynamic partitions.
INSERT OVERWRITE TABLE sale_detail_insert PARTITION (sale_date='2013', region='china')
  SELECT
  shop_name,
  customer_id,
  total_price
  FROM sale_detail
  ZORDER BY customer_id, total_price;
```

Take note of the following points:

- The mappings between the source and destination tables are based on the column sequence in the SELECT clause. The mappings between the column names of the tables are not considered. Example:

  ```
  INSERT OVERWRITE TABLE sale_detail_insert PARTITION (sale_date='2013', region='china')
      SELECT customer_id, shop_name, total_price FROM sale_detail;
  ```

  In the created sale_detail_insert table, the columns shop_name STRING, customer_id STRING, and total_price BIGINT are listed in sequence. If you insert data from the sale_detail table to the sale_detail_insert table, the data is inserted from the customer_id, shop_name, and total_price columns in sequence. In this case, data in sale_detail.customer_id is inserted into sale_detail_insert.shop_name, and data in sale_detail.shop_name is inserted into sale_detail_insert.customer_id.

- If you insert data into a partition, partition key columns cannot be included in the SELECT clause. If you execute the following statement, an error is returned because sale_date and region are partition key columns. These columns cannot be included in the INSERT statement that is used to insert table data into a static partition.

  ```
  INSERT OVERWRITE TABLE sale_detail_insert PARTITION (sale_date='2013', region='china')
     SELECT shop_name, customer_id, total_price, sale_date, region FROM sale_detail;
  ```

- The value of PARTITION must be a constant and cannot be an expression. The following example shows invalid settings:

```
INSERT OVERWRITE TABLE sale_detail_insert PARTITION (sale_date=datepart('2016-09-18 01:10:00', 'yyyy') , region='china')
   SELECT shop_name, customer_id, total_price FROM sale_detail;
```

Precautions

If you want to update table data to a dynamic partition, take note of the following points:

- If you execute the `INSERT INTO PARTITION` statement and the specified partition does not exist, the system automatically creates this partition.
- If multiple jobs that use the `INSERT INTO PARTITION` statement concurrently run and the specified partitions do not exist, the system attempts to create these partitions. However, only one partition can be created.
- If you cannot control the concurrency of the jobs that use the `INSERT INTO PARTITION` statement, we recommend that you use the `ALTER TABLE` statement to create partitions in advance.

1.6.5.1.2. Insert data into multiple objects

This topic describes how to use an INSERT statement to insert data to multiple objects.

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

Syntax:

```
FROM from_statement
INSERT OVERWRITE | INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)]
select_statement1 [FROM from_statement]
[INSERT OVERWRITE | INTO TABLE tablename2 [PARTITION (partcol1=val3, partcol2=val4 ...)]
select_statement2 [FROM from_statement]];
```

> **Note**
>
> - In most cases, an SQL statement supports up to 256 outputs. If the number of outputs exceeds 256, a syntax error is returned.
> - In a MULTI INSERT statement, non-partitioned tables and destination partitions in a partitioned table must be unique.
> - The `INSERT OVERWRITE` and `INSERT INTO` operations cannot be simultaneously performed on different partitions of a table. If these operations are simultaneously performed on the partitions, an error is returned.
> - PARTITION (partcol1=val1, partcol2=val2 ...): The parameter values can be only constants rather than expressions, such as functions.

Examples:

```
-- Create a destination table named sale_detail_multi.
create table sale_detail_multi like sale_detail;
-- Insert data from sale_detail into sale_detail_multi.
set odps.sql.allow.fullscan=true; -- Enable the full table scan, which is valid only for this sessio
n.
from sale_detail
insert overwrite table sale_detail_multi partition (sale_date='2010', region='china' )
select shop_name, customer_id, total_price
insert overwrite table sale_detail_multi partition (sale_date='2011', region='china' )
select shop_name, customer_id, total_price ;
-- An error is returned because the same partition appears more than once.
from sale_detail
insert overwrite table sale_detail_multi partition (sale_date='2010', region='china' )
select shop_name, customer_id, total_price
insert overwrite table sale_detail_multi partition (sale_date='2010', region='china' )
select shop_name, customer_id, total_price;
-- An error is returned because the INSERT OVERWRITE and INSERT INTO operations are simultaneously p
erformed on different partitions of a partitioned table.
from sale_detail
insert overwrite table sale_detail_multi partition (sale_date='2010', region='china' )
select shop_name, customer_id, total_price
insert into table sale_detail_multi partition (sale_date='2011', region='china' )
select shop_name, customer_id, total_price;
```

1.6.5.1.3. Insert data into dynamic partitions

This topic describes how to use an INSERT statement to insert data into dynamic partitions.

You can use the INSERT OVERWRITE statement to insert data into the following partitions of a partitioned table:

- Static partitions: You must specify partition values in the INSERT statement to insert data into the specified partitions.
- Dynamic partitions: You need only to specify the names of partition key columns in the INSERT statement. The values of the partition key columns are provided in the SELECT clause. The system automatically inserts data into the required partitions based on the values.

Syntax:

```
INSERT OVERWRITE|INTO TABLE tablename PARTITION (partcol1, partcol2 ...)
select_statement FROM from_statement;
```

Note

- A maximum of 10,000 dynamic partitions can be generated by using the INSERT INTO statement. A maximum of 60,000 dynamic partitions can be generated by using the INSERT OVERWRITE statement.
- An SQL statement that is used to insert data into dynamic partitions in a distributed environment can generate a maximum of 512 dynamic partitions in a single process. If the number of dynamic partitions exceeds this limit, an exception occurs.
- The dynamically generated partition values cannot be NULL or contain special characters. If a value is NULL or contains special characters, an exception occurs.

  ```
  FAILED: ODPS-0123031:Partition exception - invalid dynamic partition value: province=xxx
  ```

- If the destination 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.
- If the destination table is a hash-clustered table, dynamic partitions are not supported.
- If you insert data into dynamic partitions, the mappings between the columns in select_statement and dynamic partitions in the destination table are determined by the column sequence, not by the column names. If the sequence of columns in the source table is different from that in the destination table, we recommend that you specify the columns in select_statement based on the column sequence in the destination table.

Examples:

Insert data from a source table into a destination table. You can obtain the partitions generated based on the region field only after the required statement is executed.

```
-- Create a destination table named total_revenues.
create table total_revenues (revenue bigint) partitioned by (region string);
-- Insert the data from sale_detail into total_revenues.
insert overwrite table total_revenues partition(region)
select total_price as revenue, region from sale_detail;
```

Insert data from a source table into a destination table. If the destination table has multi-level partitions, level-1 partition sale_date must be specified.

```
insert overwrite table sale_detail_dypart partition (sale_date='2013', region)
select shop_name,customer_id,total_price,region from sale_detail;
```

To insert data into dynamic partitions, the dynamic partition key columns must be specified in the SELECT clause. Otherwise, the execution fails.

```
insert overwrite table sale_detail_dypart partition (sale_date='2013', region)
select shop_name,customer_id,total_price from sale_detail;
```

If you specify only low-level subpartitions when you insert data into dynamic partitions, you may fail to insert data into high-level partitions.

```
insert overwrite table sales partition (region='china', sale_date)
select shop_name,customer_id,total_price,sale_date from sale_detail;
```

If the data types of values in partition key columns are inconsistent with those in the SELECT clause, an error is returned when data is inserted into dynamic partitions in MaxCompute V1.0. MaxCompute V2.0 supports the implicit conversions of data types.

```
-- Create a destination table named parttable.
create table parttable(a int, b double) partitioned by (p string);
-- Insert the data from source table src into destination table parttable.
insert into parttable partition(p) select key, value, current_timestmap() from src;
-- Query data in parttable.
select * from parttable;
```

1.6.5.1.4. VALUES

This topic describes how to use the INSERT...VALUES statement to insert data into a table that contains small amounts of data.

Syntax:

```
INSERT INTO TABLE tablename
[PARTITION (partcol1=val1, partcol2=val2,...)][(co1name1,colname2,...)]
[VALUES (col1_value,col2_value,...),(col1_value,col2_value,...),...]
```

Parameters:

- tablename: the name of the table into which you want to insert data. The table must be an existing table.
- [PARTITION (partcol1=val1, partcol2=val2,...)]: the information about partitions. If you want to insert data into a partitioned table, you must specify this parameter.
- [(co1name1,colname2,...)]: the names of the columns in the destination table.
- col_value: the value in the specified column in the destination table. Separate multiple values with commas (,). Each value in the column must be a constant. If no values are specified, the default value NULL is used.

> **Note**
>
> - You can use only the `INSERT INTO` statement rather than the `INSERT OVERWRITE` statement to insert data into specific columns.
> - Functions cannot be used to construct constants for some complex data types, such as ARRAY, in VALUES. However, you can use the following statement to pass the values of the ARRAY type to VALUES:
>
>   ```
>   INSERT INTO TABLE srcp (p='abc') SELECT 'a', ARRAY('1', '2', '3');
>   ```
>
> - To pass the values of the DATETIME or TIMESTAMP type, you must specify the data type in VALUES.
>
>   ```
>   INSERT INTO TABLE srcp (p='abc') VALUES (datetime'2017-11-11 00:00:00',timestamp'2017-11-1
>   1 00:00:00.123456789');
>   ```

Example:

Insert data into a specific partition.

```
-- Delete the srcp table that already exists in the system.
DROP TABLE IF EXISTS srcp;
-- Create a partitioned table named srcp.
CREATE TABLE IF NOT EXISTS srcp (key string,value bigint) PARTITIONED BY (p string);
-- Insert data into the abc partition of the srcp table.
INSERT INTO TABLE srcp PARTITION (p='abc') VALUES ('a',1),('b',2),('c',3);
-- Query data from the srcp table.
SELECT * FROM srcp WHERE p='abc';
```

The following information is returned:

```
+------------+------------+------------+
| key        | value      | p          |
+------------+------------+------------+
| a          | 1          | abc        |
| b          | 2          | abc        |
| c          | 3          | abc        |
+------------+------------+------------+
```

Insert data into any partition.

```
-- Delete the srcp table that already exists in the system.
DROP TABLE IF EXISTS srcp;
-- Create a partitioned table named srcp.
CREATE TABLE IF NOT EXISTS srcp (key string,value bigint) PARTITIONED BY (p string);
-- Insert data into a partition of the srcp table.
INSERT INTO TABLE srcp PARTITION (p)(key,p) VALUES ('d','20170101'),('e','20170101'),('f','20170101'
);
-- Query data from the srcp table.
SELECT * FROM srcp WHERE p='20170101';
```

The following information is returned:

```
+------------+------------+------------+
| key        | value      | p          |
+------------+------------+------------+
| d          | NULL       | 20170101   |
| e          | NULL       | 20170101   |
| f          | NULL       | 20170101   |
+------------+------------+------------+
```

Use complex data types to construct constants and execute an `INSERT` statement to import data.

```
-- Create a partitioned table named srcp.
create table if not exists srcp (key string,value array<int>) partitioned by (p string);
-- Add a partition to the srcp table.
alter table srcp add if not exists partition (p='abc');
-- Insert data into the abc partition of the srcp table.
insert into table srcp partition (p='abc') select 'a', array(1, 2, 3);
-- Query data from the srcp table.
select * from srcp where p='abc';
```

The following information is returned:

```
+------------+------------+------------+
| key        | value      | p          |
+------------+------------+------------+
| a          | [1,2,3]    | abc        |
+------------+------------+------------+
```

Use the `INSERT...VALUES` statement to write data of the DATETIME or TIMESTAMP type.

```
-- Create a partitioned table named srcp.
create table if not exists srcp (key string, value timestamp) partitioned by (p string);
-- Add a partition to the srcp table.
alter table srcp add if not exists partition (p='abc');
-- Insert data into the abc partition of the srcp table.
insert into table srcp partition (p='abc') values (datetime'2017-11-11 00:00:00',timestamp'2017-11-1
1 00:00:00.123456789');
-- Query data from the srcp table.
select * from srcp where p='abc';
```

The following information is returned:

```
+------------+------------+------------+
| key        | value      | p          |
+------------+------------+------------+
| 2017-11-11 00:00:00 | 2017-11-11 00:00:00.123 | abc        |
+------------+------------+------------+
```

Features of VALUES TABLE

When you use INSERT...VALUES, the values after VALUES must be constants. If you want to perform simple computing operations on the inserted data, we recommended that you use VALUES TABLE of MaxCompute.

Syntax:

```
values (<col1_value>,<col2_value>,...),(<col1_value>,<col2_value>,...),<table_name> (<col1_name> ,<c
ol2_name>,...)...
```

You can use VALUES TABLE in the following scenarios:

- If no physical tables are available, create a table that contains multiple rows of data and perform computing operations on the table.

  For example, `VALUES (…), (…) t(a, b)` defines that a table named t contains the a and b columns. The data type of the a column is STRING and that of the b column is BIGINT. The data type of a column must be derived from the VALUES list.

  ```
  -- Delete the srcp table that already exists in the system.
  DROP TABLE IF EXISTS srcp;
  -- Create a partitioned table named srcp.
  CREATE TABLE IF NOT EXISTS srcp (key string,value bigint) PARTITIONED BY (p string);
  -- Insert data into the srcp table.
  INSERT INTO TABLE srcp PARTITION (p) SELECT concat(a,b), length(a)+length(b),'20170102' FROM VALUE
  S ('d',4),('e',5),('f',6) t(a,b);
  -- Query data from the srcp table.
  SELECT * FROM srcp WHERE p='20170102';
  ```

  The following information is returned:

  ```
  +------------+------------+------------+
  | key        | value      | p          |
  +------------+------------+------------+
  | d4         | 2          | 20170102   |
  | e5         | 2          | 20170102   |
  | f6         | 2          | 20170102   |
  +------------+------------+------------+
  ```

- Use VALUES TABLE rather than the combination of `SELECT * FROM` and `UNION ALL` to create a constant-

type table.

```
SELECT 1 c UNION ALL SELECT 2 c;
-- The preceding statement is equivalent to the following statement:
SELECT * FROM VALUES (1), (2) t(c);
```

The following information is returned:

```
+------------+
| c          |
+------------+
| 1          |
| 2          |
+------------+
```

- Special forms of VALUES TABLE. If the expressions of the SELECT statement do not contain upstream table data, you can execute the SELECT statement without the FROM clause. In this case, the underlying implementation is to select data from an anonymous VALUES table that contains only one row and no columns. This way, you can test user-defined functions (UDFs) or other functions without the need to manually create DUAL tables.

```
-- Create a partitioned table named srcp.
create table if not exists srcp (key string,value bigint) partitioned by (p string);
-- Insert data into the srcp table.
insert into table srcp partition (p) select abs(-1), length('abc'), getdate();
-- Query data from the srcp table.
select * from srcp;
```

The following information is returned:

```
+------------+------------+------------+
| key        | value      | p          |
+------------+------------+------------+
| 1          | 3          | 2020-11-25 18:39:48 |
+------------+------------+------------+
```

Support for expressions that do not contain constants

Constant expressions are supported in VALUES expressions. Expressions that do not contain constants are also supported in VALUES expressions. Syntax:

```
select * from values (udf(1)),(to_date('20190101', 'yyyyMMdd')),(getdate()) t(d);
```

1.6.5.2. SELECT

1.6.5.2.1. SELECT operations

This topic describes how to use SELECT statements for SELECT operations.

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];
```

Limits

- A SELECT statement can return a maximum of 10,000 rows of results. However, no limits are imposed when the SELECT statement is used as a clause. If the SELECT statement is used as a clause, the clause returns all results in response to the query from the upper layer.
- You are not allowed to perform full table scans on partitioned tables by using SELECT statements. To perform a full table scan on a partitioned table, add `set odps.sql.allow.fullscan=true;` before the SQL statement that is used for the full table scan. Then, commit and run the added command with the SQL statement. To enable the full table scan for the entire project, run the following command:

```
setproject odps.sql.allow.fullscan=true;
```

Column expression (select_expr)

Each select_expr expression indicates a column from which you want to read data. You can use the following select_expr expressions to read data from a table.

- Specify the names of the columns from which you want to read data.

```
select shop_name from sale_detail;
```

- Use an asterisk (*) to represent all columns.

```
select * from sale_detail;
-- You can also use a WHERE clause to specify filter conditions.
select * from sale_detail where shop_name like 'hang%';
```

- Use a regular expression.
- Use DISTINCT before the name of a column to filter out duplicate values from the column and return only distinct values. If you use ALL before the name of a column, all values including the duplicate values in the column are returned. If DISTINCT is not used, the default value ALL is used.

```
-- Query data from the region column in the sale_detail table and return only distinct values.
select distinct region from sale_detail;
```

The following result is returned:

```
+------------+
| region     |
+------------+
| shanghai   |
+------------+
```

```
-- If you use DISTINCT in a SELECT statement where multiple columns are specified, DISTINCT applie
s to all the columns you specified, instead of a single column.
select distinct region, sale_date from sale_detail;
```

The following result is returned:

```
+------------+------------+
| region     | sale_date  |
+------------+------------+
| shanghai   | 20191110   |
+------------+------------+
```

Information about a queried destination table (table_reference)

table_reference specifies the table that you want to query. Nested subqueries are also supported in table_reference. Example:

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

WHERE

The following table describes the operators that are used as filter conditions supported by the WHERE clause.

Operator	Description
>, <, =, ≥, ≤, and <>	The relational operators.
LIKE and RLIKE	The Source and Pattern parameters of LIKE and RLIKE must be of the STRING type.
IN and NOT IN	If a subquery is added after the IN or NOT IN operator, the values in only one column can be returned for the subquery and the number of return values cannot exceed 1,000.
BETWEEN...AND	The operator that specifies the query range.

In the WHERE clause of a SELECT statement, you can specify the partitions that you want to scan in a table to avoid a full table scan.

```
SELECT sale_detail. *
FROM sale_detail
WHERE sale_detail.sale_date >= '2008'
AND sale_detail.sale_date <= '2014';
```

User-defined functions (UDFs) support partition pruning. UDFs are executed as small jobs and then replaced with the execution results. You can use one of the following methods to implement UDF-based partition pruning:

- 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 backfills partitions with execution results. A maximum of 1,000 partitions can be backfilled with execution results. If you add an annotation to a UDF class, an error may be returned, which indicates that more than 1,000 partitions are backfilled with execution results. If you want to ignore the error, add the `set odps.sql.udf.ppr.to.subquery = false;` flag to globally disable this feature. After this feature is disabled, UDF-based partition pruning no longer takes effect.

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

```
SELECT sale_detail. *
FROM sale_detail
WHERE sale_detail.sale_date BETWEEN '2008' AND '2014';
```

GROUP BY

In most cases, the GROUP BY clause is used with aggregate functions. If a SELECT statement contains aggregate functions, the following rules apply:

- During the parsing of SQL statements, the GROUP BY clause precedes a SELECT operation. Therefore, the value of the GROUP BY clause must be the names of the columns in the input table for the SELECT operation. The value can also be an expression composed of the columns in the input table. The value cannot be the aliases of the output columns from the SELECT operation.
- The value of the GROUP BY clause may be both a column name or an expression composed of input table columns and the alias of an output column from the SELECT operation. In this case, the column name or expression is used as the value of the GROUP BY clause.
- If the `set hive.groupby.position.alias=true;` flag is added before a SELECT statement, the integer constants in GROUP BY are considered column numbers in the SELECT operation.

```
set hive.groupby.position.alias=true;
-- Run this command with the next SELECT statement.
select region, sum(total_price) from sale_detail group by 1;
-- 1 indicates the region column, which is the first column read by the SELECT statement. This sta
tement groups the table data based on the values in the region column and returns distinct region
values and total sales of each group.
```

Examples:

```
-- The statement uses the name of the region column in the input table as the value of the GROUP BY
clause and groups the table data based on the values in the region column.
select region from sale_detail group by region;
-- The statement groups the table data based on the values in the region column and returns the tota
l sales of each group.
select sum(total_price) from sale_detail group by region;
-- The statement groups the table data based on the values in the region column and returns distinct
values and total sales of each group.
select region, sum(total_price) from sale_detail group by region;
-- An error is returned because the alias of the column in the SELECT operation is used as the value
of the GROUP BY clause.
select region as r from sale_detail group by r;
-- A complete expression of the column is required.
select 2 + total_price as r from sale_detail group by 2 + total_price;
-- An error is returned because all the columns that do not use aggregate functions in the SELECT op
eration must be included in the GROUP BY clause.
select region, total_price from sale_detail group by region;
-- The statement can be successfully executed because all the columns that do not use aggregate func
tions are included in the GROUP BY clause.
select region, total_price from sale_detail group by region, total_price;
```

ORDER BY|DISTRIBUTE BY|SORT BY

- ORDER BY: This clause is used to sort all data records based on the specified columns.
  - To sort data records in descending order, use the desc keyword. By default, data records are sorted in ascending order.
  - When you use ORDER BY to sort data records, NULL is considered the smallest value. This rule is consistent with MySQL, but is different from Oracle.
  - 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.

- 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.

```
-- Set the flag.
set hive.orderby.position.alias=true;
-- Create a table named src.
create table src(key BIGINT,value BIGINT);
-- Query the src table and sort the records that are returned in ascending order by value.
SELECT * FROM src ORDER BY 2 limit 100;
The preceding statement is equivalent to the following statement:
SELECT * FROM src ORDER BY value limit 100;
```

- You can use the OFFSET clause with the ORDER BY LIMIT clause to skip the number of rows specified by OFFSET.

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

Examples:

```
-- Query the sale_detail table and sort the first 100 records that are returned in ascending order by region.
SELECT * FROM sale_detail ORDER BY region LIMIT 100;
-- By default, the LIMIT clause is used with the ORDER BY clause. If only the ORDER BY clause is used, an error is returned.
SELECT * FROM sale_detail ORDER BY region;
-- ORDER BY is followed by a column alias.
SELECT region AS r FROM sale_detail ORDER BY region LIMIT 100;
SELECT region AS r FROM sale_detail ORDER BY r LIMIT 100;
```

- DISTRIBUTE BY: This clause is used to shard data based on hash values of specific columns. You must specify the alias of an output column from the SELECT operation as the value of the DISTRIBUTE BY clause.

  Examples:

```
-- The statement queries values in the region column of the sale_detail table and shards data based on the hash values of the region column.
SELECT region FROM sale_detail DISTRIBUTE BY region;
-- The statement can be successfully executed because the column name is used as the alias.
SELECT region AS r FROM sale_detail DISTRIBUTE BY region;
The preceding statement is equivalent to the following statement:
SELECT region AS r FROM sale_detail DISTRIBUTE BY r;
```

- SORT BY: This clause is used to sort specific data records. You can add a keyword, such as ASC or DESC, to sort specific data records. If ASC is used, data records are sorted in ascending order. If DESC is used, data records are sorted in descending order. If SORT BY is not followed by a keyword, data records are sorted in ascending order by default.

- If the SORT BY clause is preceded by the DISTRIBUTE BY clause, the SORT BY clause sorts specific results of the DISTRIBUTE BY clause.

  The DISTRIBUTE BY clause determines how to distribute the output values of the map stages to reducers. To prevent duplicate output values from being distributed to different reducers or to process the same group of data together, you can use the DISTRIBUTE BY clause. This ensures that the same group of data is distributed to the same reducer. Then, use SORT BY to sort the group of data. Example:

  ```
  -- The statement queries values in the region column of the sale_detail table, shards data based
  on the hash values of the region column, and then sorts the sharded data.
  SELECT region FROM sale_detail DISTRIBUTE BY region SORT BY region;
  ```

- If the SORT BY clause is not preceded by the DISTRIBUTE BY clause, the SORT BY clause sorts the data of each reducer. This process sorts specific data records. This ensures that the output values of each reducer are sorted and increases the data compression ratio. If data is filtered during data reading, the amount of data read from disks is reduced, which makes subsequent global sorting more efficient. Example:

  ```
  SELECT region FROM sale_detail SORT BY region;
  ```

> **Note**
> - The value of ORDER BY, DISTRIBUTE BY, or SORT BY must be the alias of an output column from the SELECT operation. The column alias can be Chinese.
> - During the parsing of MaxCompute SQL statements, the ORDER BY, DISTRIBUTE BY, or SORT BY clause is executed after the SELECT operation. Therefore, the value of ORDER BY, DISTRIBUTE BY, or SORT BY must be the alias of an output column from the SELECT operation.
> - The ORDER BY clause cannot be used with the DISTRIBUTE BY or SORT BY clause. The GROUP BY clause cannot be used with the DISTRIBUTE BY or SORT BY clause.

LIMIT

The number in the LIMIT clause is a constant that limits the number of rows to return.

- Remove the limit on the simultaneous execution of the ORDER BY and LIMIT clauses

  The ORDER BY clause needs to sort all data of a single node. By default, the ORDER BY clause is used with the LIMIT clause to prevent a single node from processing large amounts of data. You can remove the limit on the simultaneous execution of the ORDER BY and LIMIT clauses for a project or session.

  - To remove the limit for a project, run the `setproject odps.sql.validate.orderby.limit=false;` command.
  - To remove the limit for a session, run the `setproject odps.sql.validate.orderby.limit=false;` command with the SQL statement.

  > **Note** If a single node has large amounts of data to sort after the limit is removed, more resources and time are required.

- Limits on the number of rows to return

  If you use a SELECT statement without the LIMIT clause or the number in the LIMIT clause exceeds the specified upper limit, you can view only the rows within the upper limit.

  The upper limit on the number of rows to return varies based on projects. You can use one of the following methods to configure the upper limit:

- If project protection is disabled, use_instance_tunnel is set to true, and instance_tunnel_max_record is not specified, no limits are imposed on the number of rows to return. For more information about project protection, see Data protection. For more information about the configurations of use_instance_tunnel=true and instance_tunnel_max_record, see Tunnel operations.
- If project protection is enabled, the number of rows to return is limited by READ_TABLE_MAX_ROW. The maximum value of this parameter is 10000.

> **Note** You can run the `show SecurityConfiguration;` command to view the configuration of ProjectProtection. If ProjectProtection is set to true, you can determine whether to disable project protection. You can run the `set ProjectProtection=false;` command to disable project protection. ProjectProtection is disabled by default. For more information about project protection, see Data protection.

1.6.5.2.2. Subquery

This topic describes how to use a SELECT statement to perform subqueries.

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

Syntax:

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

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

Examples:

```
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.

1.6.5.2.3. UNION ALL

This topic describes how to use a SELECT statement to perform UNION ALL operations.

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 UNION ALL between two top-level query statements. To combine the query results of the two statements, you must rewrite the UNION ALL clause as a subquery.

Syntax before rewriting:

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

Syntax 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.

Examples:

```
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
the src table and the result of the first UNION ALL operation.
```

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.

1.6.5.2.4. JOIN

This topic describes how to use SELECT statements to perform JOIN operations.

MaxCompute supports multiple JOIN operations in an SQL statement. MaxCompute does not support CROSS JOIN. A CROSS JOIN operation joins two tables without the need to specify conditions in the ON clause.

Syntax:

```
join_table:
        table_reference JOIN table_factor [join_condition]
        | table_reference {LEFT OUTER|RIGHT OUTER|FULL OUTER|INNER|NATURAL} JOIN table_reference joi
n_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.

Types of JOIN operations:

- LEFT OUTER JOIN: shortened as LEFT JOIN. It returns all rows in the left table, including the rows that do not match any rows in the right table.

```
SELECT a.shop_name AS ashop, b.shop_name AS bshop FROM shop a
LEFT 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 distinguish the columns in the SELECT operation.
```

> **Note** If the values in some rows of the right table are duplicate, we recommend that you do not consecutively use LEFT JOIN. If you consecutively use LEFT JOIN, data bloat may occur and interrupt your jobs.

- RIGHT OUTER JOIN: shortened as RIGHT JOIN. It returns all rows in the right table, including rows that do not match any rows in the left table.

```
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;
```

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

```
SELECT a.shop_name AS ashop, b.shop_name AS bshop FROM shop a
FULL OUTER JOIN sale_detail b ON a.shop_name=b.shop_name;
```

- INNER JOIN: The INNER keyword can be omitted. INNER JOIN only returns rows in which a match between the two tables exists.

```
SELECT a.shop_name FROM shop a INNER JOIN sale_detail b ON a.shop_name=b.shop_name;
SELECT a.shop_name FROM shop a JOIN sale_detail b ON a.shop_name=b.shop_name;
```

- 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.

```
SELECT * FROM src NATURAL JOIN src2;
-- Both the src and src2 tables include the key1 and key2 fields. In this case, the preceding statement 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;
```

- Implicit JOIN operation: A JOIN operation that is performed without the need to specify the JOIN keyword.

```
SELECT * FROM table1, table2 WHERE table1.id = table2.id;
-- The preceding statement is equivalent to the following statement:
SELECT * FROM table1 JOIN table2 ON table1.id = table2.id;
```

Join conditions: You must use equi-joins and combine conditions by using AND. You can use non-equi joins or combine multiple conditions by using OR in a MAPJOIN operation.

```
-- MaxCompute supports multiple JOIN operations in an SQL statement.
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;
-- MaxCompute does not support non-equi joins and returns an error.
SELECT a. * FROM shop a JOIN sale_detail b ON a.shop_name ! = b.shop_name;
```

You can use parentheses () to specify the priority of JOIN operations. The JOIN operation enclosed in parentheses () has a high priority.

```
SELECT * FROM src JOIN (src2 JOIN src3 on xxx) ON yyy;
-- In the preceding statement, src2 JOIN src3 is first executed. Then, the JOIN operation is performed for the src table and the result of the first JOIN operation.
```

Examples:

The test_table_a and test_table_b tables are available. You want to query the two tables for the rows whose values in the origin column are equal to those in the id column and whose data timestamp is greater than 20180101. If you use LEFT JOIN, all rows in the left table test_table_a are returned.

```
SELECT s.id
        ,s.name
        ,s.origin
        ,d.value
FROM  (SELECT * FROM  test_table_a WHERE ds > "20180101" ) s
LEFT JOIN (SELECT * FROM  test_table_b WHERE ds > "20180101") d
ON s.origin = d.id;
```

1.6.5.2.5. MAPJOIN hint

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

Background information

You can explicitly specify MAPJOIN in a SELECT statement to join a large table with one or more small tables. MAPJOIN speeds up your query.

A JOIN operation contains three stages: map, shuffle, and reduce. In most cases, tables are joined in the reduce stage.

MAPJOIN joins tables in the map stage instead of the reduce stage. This reduces data transmission time and system resource consumption and optimizes jobs.

In the map stage, MAPJOIN loads all data in the specified tables into the memory of the program that executes the JOIN operation. The tables specified for MAPJOIN must be small tables, and the total memory occupied by the table data cannot exceed 512 MB.

When a large table is joined with one or more small tables, MAPJOIN loads all data in the specified small tables into the memory of the program that executes the JOIN operation. This way, tables are connected in the map stage to accelerate the execution of the JOIN operation.

Usage

To use MAPJOIN, you must specify the hint `/*+ MAPJOIN(table) */` in a SELECT statement.

For example, sale_detail is a large table whose alias is b, and shop is a small table whose alias is a.

- The following statement is used to perform a common JOIN operation:

```
SELECT  a.shop_name,
        b.customer_id,
        b.total_price
FROM shop a JOIN sale_detail b
ON a.shop_name = b.shop_name;
```

- The following statement is used to perform a JOIN operation with MAPJOIN specified:

```
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;
```

Limits

- When you use MAPJOIN and reference a small table or a subquery, you must reference the alias of the table or subquery.
- MAPJOIN supports small tables in subqueries.
- The left table in a LEFT OUTER JOIN operation must be a large table.
- The right table in a RIGHT OUTER JOIN operation must be a large table.
- Either the left or right table in an INNER JOIN operation can be a large table.
- MAPJOIN cannot be used in a FULL OUTER JOIN operation.
- For MAPJOIN, you can use non-equi joins or combine conditions by using OR. You can leave the ON condition unspecified or use `MAPJOIN ON 1 = 1`, such as `SELECT /* + MAPJOIN(a) */ a.id FROM shop a JOIN table_name b ON 1=1`, to calculate the Cartesian product. However, this calculation method may increase the data amount.
- MaxCompute allows you to specify a maximum of 128 small tables for MAPJOIN. If you specify more than 128 small tables, a syntax error is returned. Separate small tables with commas (,), such as `/*+MAPJOIN(a,b,c)*/`.
- The total memory occupied by small tables cannot exceed 512 MB. MaxCompute compresses your data before storage. As a result, the data amount of small tables sharply increases after they are loaded into the memory. 512 MB indicates the maximum data amount after small tables are loaded into the memory.

Examples

In a MaxCompute SQL statement, you cannot use complex join conditions, such as non-equi joins or the OR logical operator, in the ON condition for a common JOIN operation. However, you can use them for a JOIN operation with MAPJOIN specified.

```
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;
```

1.6.5.2.6. SELECT TRANSFORM

1.6.5.2.6.1. Overview

This topic describes the syntax of the SELECT TRANSFORM statement in MaxCompute.

The SELECT TRANSFORM statement allows you to start a specific child process and insert data in the required format into the child process by using standard input. Then, the SELECT TRANSFORM statement parses the standard output of the child process to obtain the output data. The SELECT TRANSFORM statement allows you to run scripts in other programming languages without the need to write user-defined functions (UDFs).

Syntax:

```
SELECT TRANSFORM(arg1, arg2 ...)
(ROW FORMAT DELIMITED (FIELDS TERMINATED BY field_delimiter (ESCAPED BY character_escape))
(LINES 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))
(LINES SEPARATED BY line_separator) (NULL DEFINED AS null_value))
```

Parameters:

- SELECT TRANSFORM: the keyword, which can be replaced with the MAP or REDUCE keyword. These keywords have the same syntax. We recommend that you use SELECT TRANSFORM to make the syntax clearer.
- (arg1, arg2 ...): the input data. The data format is similar to that of the SELECT statement. In the default format, the results of expressions for each parameter are implicitly converted into a value of the STRING type. Then, the parameters are combined by \t and passed to the specified child process.

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

- USING: the commands that are used to start a child process. Take note of the following points:
  - In most MaxCompute SQL statements, the USING clause specifies resources. However, in the SELECT TRANSFORM statement, the USING clause specifies the commands to start a child process. The USING clause is used to ensure compatibility with the Hive syntax.
  - The syntax of the USING clause is similar to that of a shell script. However, instead of running the shell script, the USING clause creates a child process based on the input commands. Therefore, some shell features, such as input and output redirection, pipe, and loop, cannot be used. A shell script can be used as the commands to start a child process if necessary.
- RESOURCES: 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;` flag before SQL statements to specify resources.

    > **Note** This is a global configuration. It allows all SELECT TRANSFORM statements to access the specified resources. Separate multiple resource files with commas (,).

- ROW FORMAT: the input and output formats. The syntax includes two ROW FORMAT clauses. The first clause specifies the format of the input data, and the second clause specifies the format of the output data. By default, \t is used as a column delimiter, \n is used as a row delimiter, and \N is used to represent NULL values.

  > **Note**
  > - For field_delimiter, character_escape, and line_separator, only one character is accepted. If you specify a string, the first character in the string takes priority over the others.
  > - MaxCompute supports syntax in the formats that are specified by Hive, such as inputRecordReader, outputRecordReader, and SerDe. You must enable the Hive-compatible mode to use the syntax. To enable the Hive-compatible mode, add `set odps.sql.hive.compatible=true;` before SQL statements.
  > - If you specify the syntax that is supported by Hive, such as inputRecordReader and outputRecordReader, the performance of SQL queries may deteriorate.

- AS: the output columns.

> **Note**
> - If you do not specify the data types for output columns, the STRING type is used.
> - The output data is obtained after the standard output of the child process is parsed. If the data is not of the STRING type, the system implicitly calls the CAST function to convert the data into the STRING type. Exceptions may occur during the conversion process.
> - You must specify data types for all output columns.
> - If you omit the AS keyword, the field before the first \t in the standard output data is the key and all the following parts are values. This is equivalent to AS(key, value).

1.6.5.2.6.2. SELECT TRANSFORM examples

Call shell scripts

This topic describes how to use SELECT TRANSFORM to call shell scripts and provides examples.

Assume that you use a shell script to generate 50 rows of data. The values are from 1 to 50. The following code shows the output of the data field:

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

You can use shell commands as the input for SELECT TRANSFORM.

> **Note** SELECT TRANSFORM allows you to use the scripts of multiple programming languages, such as AWK, Python, Perl, and shell, as the input to implement simple features. You do not need to compile script files or upload resources, which simplifies development.

You can upload a script to implement complex features. For more information, see Call Python scripts.

Call Python scripts

This topic describes how to use SELECT TRANSFORM to call Python scripts and provides examples.

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 str(token[0]) +'\t' + str(token[1])
       line = sys.stdin.readline()
   ```

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

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

3. Execute SELECT TRANSFORM to call the resource.

```
-- Create a test table.
CREATE TABLE testdata(c1 bigint,c2 bigint);
-- Insert test data into the test table.
INSERT INTO TABLE testdata VALUES (1,4),(2,5),(3,6);
-- Execute the following statement:
SELECT
TRANSFORM (testdata.c1, testdata.c2)
USING 'python myplus.py'resources 'myplus.py'
AS (result1,result2)
FROM testdata;
-- The preceding statement is equivalent to the following statement:
set odps.sql.session.resources=myplus.py;
SELECT TRANSFORM (testdata.c1, testdata.c2)
USING 'python myplus.py'
AS (result1,result2)
FROM testdata;
```

Note In MaxCompute, you can use Python scripts as the input for SELECT TRANSFORM. For example, you can run Python commands to call shell scripts.

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

The following result is returned:

```
+------------+------------+
| result1    | result2    |
+------------+------------+
| 1          | 4          |
|            | NULL       |
| 2          | 5          |
|            | NULL       |
| 3          | 6          |
|            | NULL       |
+------------+------------+
```

Call Java scripts

This topic describes how to use SELECT TRANSFORM to call Java scripts and provides examples.

Java scripts are called in a similar way to Python scripts. In this example, you must prepare a Java script file, export it as a JAR package, and then execute the `ADD FILE` statement to add the JAR package as a resource to MaxCompute. Then, the resource can be called by using SELECT TRANSFORM.

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

```
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 a test table.
CREATE TABLE testdata(c1 bigint,c2 bigint);
-- Insert test data into the test table.
INSERT INTO TABLE testdata VALUES (1,4),(2,5),(3,6);
-- Execute the SELECT TRANSFORM statement.
SELECT TRANSFORM(testdata.c1, testdata.c2) USING 'java -cp Sum.jar com.aliyun.odps.test.Sum' res
ources 'Sum.jar' FROM testdata;
-- The preceding statement is equivalent to the following statement:
set odps.sql.session.resources=Sum.jar;
SELECT TRANSFORM(testdata.c1, testdata.c2) USING 'java -cp Sum.jar com.aliyun.odps.test.Sum' FRO
M testdata;
```

The following result is returned:

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

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

Although UDTF frameworks are provided for Java and Python, SELECT TRANSFORM makes it easier to compile code. SELECT TRANSFORM is a simpler process because it is not subject to dependencies or 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.

Call the scripts of other programming languages

This topic describes how to use SELECT TRANSFORM to call the scripts of other programming languages and provides examples.

SELECT TRANSFORM supports the programming languages described in the preceding topics. It also supports commonly used UNIX commands and script interpreters, such as AWK and Perl interpreters.

Example on how to call an AWK script to display the second column:

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

Example on how to call a Perl script:

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

> **Note** PHP and Ruby are not deployed in MaxCompute compute clusters. Therefore, you cannot call PHP or Ruby scripts in MaxCompute.

Call scripts in series

This topic describes how to use SELECT TRANSFORM to call scripts in series and provides examples.

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;
```

You can also use the MAP or REDUCE keyword to obtain 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;
```

1.6.5.2.6.3. Performance

This topic describes the advantages of SELECT TRANSFORM and user-defined table-valued functions (UDTFs).

The performance of SELECT TRANSFORM and UDTFs varies based on specific scenarios. In most cases, SELECT TRANSFORM performs better. However, UDTFs perform better as the data amount increases. SELECT TRANSFORM is more suitable for ad hoc data analytics because the development of SELECT TRANSFORM is easier.

The following sections describe the advantages of UDTFs and SELECT TRANSFORM.

Advantages of UDTFs

- For UDTFs, the input parameters and output results support multiple data types. For SELECT TRANSFORM, the child process transfers data based on the standard input and output and processes all data as the STRING type. Therefore, SELECT TRANSFORM requires one more step of data type conversions than UDTFs.
- For SELECT TRANSFORM, data transfer depends on the operating system pipe. The pipe has only a 4 KB cache that cannot be specified. If the operating system pipe is empty or fully occupied, SELECT TRANSFORM cannot respond.
- For UDTFs, the constant parameters do not need to be transferred. However, SELECT TRANSFORM does not support this feature.

Advantages of SELECT TRANSFORM

- SELECT TRANSFORM supports two processes: parent and child processes. UDTFs support only a single process. If the usage of computing resources is high and the data throughput is low, SELECT TRANSFORM can take advantage of the multi-core feature of the server.
- SELECT TRANSFORM calls underlying systems to read and write data during data transfer. This allows SELECT TRANSFORM to provide better performance in data transfer than Java programs.
- SELECT TRANSFORM supports the native code of tools such as AWK. This allows SELECT TRANSFORM to deliver more performance benefits than Java programs.

1.6.5.2.7. GROUPING SETS

1.6.5.2.7.1. Overview

This topic describes GROUPING SETS in MaxCompute.

In some cases, you must execute the UNION ALL clause multiple times to aggregate and analyze data from multiple dimensions. For example, you want to aggregate Column A, aggregate Column B, and then aggregate Column A and Column B. The GROUPING SETS clause is a better choice in these cases.

GROUPING SETS is an extension to the GROUP BY clause in a SELECT statement. GROUPING SETS allows you to group results by using various methods without the need to execute multiple SELECT statements. This allows the MaxCompute engine to generate more efficient execution plans with high performance.

> **Note** Most examples in this topic are demonstrated by using MaxCompute Studio. We recommend that you install MaxCompute Studio before you proceed with subsequent operations. For more information, see MaxCompute Studio.

1.6.5.2.7.2. Example

This topic provides an example on how to execute the GROUPING SETS statement.

Example:

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. Execute the GROUPING SETS statement.

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

The following result is returned:

```
+----+--------+------+------------+
| os | device | city | cnt        |
+----+--------+------+------------+
| NULL | NULL   | NULL | 7          |
| NULL | NULL   | Beijing | 4          |
| NULL | NULL   | Shijiazhuang | 3          |
| ios | Phone  | NULL | 1          |
| linux | PC     | NULL | 1          |
| linux | Phone  | NULL | 1          |
| windows | PC     | NULL | 3          |
| windows | Phone  | NULL | 1          |
+----+--------+------+------------+
```

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 execution of GROUPING SETS is simpler and more efficient.

Note NULL is used as placeholders for the expressions that are not used in GROUPING SETS. This way, you can perform UNION operations on the result sets.

1.6.5.2.7.3. CUBE and ROLLUP

This topic describes the CUBE and ROLLUP functions for GROUPING SETS.

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

Examples:

```
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), ()
 )
```

1.6.5.2.7.4. GROUPING and GROUPING_ID

This topic describes the GROUPING and GROUPING_ID functions in GROUPING SETS.

NULL is used as placeholders in the results of GROUPING SETS. As a result, the NULL placeholders cannot be distinguished from the NULL values. To address this issue, MaxCompute provides the GROUPING function.

GROUPING allows you to specify the name of a column as a parameter. If specific rows are aggregated based on the column specified in the GROUPING function, 0 is returned, which indicates that NULL is an input value. Otherwise, 1 is returned, which indicates that NULL is a placeholder in GROUPING SETS.

GROUPING_ID allows you to specify the names of one or more columns as parameters. The GROUPING results of these columns are formed into integers by using bitmap.

Examples:

```
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);
```

The following result is returned:

```
+------------+------------+------------+------------+------------+------------+------------+--------
----+
| a          | b          | c          | _c3        | ga         | gb         | gc         | groupin
gid |
+------------+------------+------------+------------+------------+------------+------------+--------
----+
| NULL       | NULL       | NULL       | 1          | 1          | 1          | 1          | 7
|
| NULL       | NULL       | 3          | 1          | 1          | 1          | 0          | 6
|
| NULL       | 2          | NULL       | 1          | 1          | 0          | 1          | 5
|
| NULL       | 2          | 3          | 1          | 1          | 0          | 0          | 4
|
| 1          | NULL       | NULL       | 1          | 0          | 1          | 1          | 3
|
| 1          | NULL       | 3          | 1          | 0          | 1          | 0          | 2
|
| 1          | 2          | NULL       | 1          | 0          | 0          | 1          | 1
|
| 1          | 2          | 3          | 1          | 0          | 0          | 0          | 0
|
+------------+------------+------------+------------+------------+------------+------------+--------
----+
```

By default, the columns that are not used in GROUP BY are filled with NULL. You can use the GROUPING function to obtain more useful data.

```
SELECT
  IF(GROUPING(os) == 0, os, 'ALL') as os,
  IF(GROUPING(device) == 0, device, 'ALL') as device,
  IF(GROUPING(city) == 0, city, 'ALL') as city ,
  COUNT(*) as count
FROM requests
GROUP BY os, device, city GROUPING SETS((os, device), (city), ());
```

The following figure shows the returned result.

SELECTIF(GROUPING(os) == 0, os, 'ALL') as os,IF(GROUPING(device) == 0, device, 'ALL') as device,IF(GROUPING(city) == 0, city, 'ALL') as ...

	os	device	city	count
1	ALL	ALL	ALL	7
2	ALL	ALL	Beijing	4
3	ALL	ALL	Shijiazhuang	3
4	ios	Phone	ALL	1
5	linux	PC	ALL	1
6	linux	Phone	ALL	1
7	windows	PC	ALL	3
8	windows	Phone	ALL	1

1.6.5.2.8. UNION, INTERSECT, and EXCEPT

This topic describes the SQL statements that use the UNION, INTERSECT, and EXCEPT set operators, such as UNION, UNION ALL, UNION DISTINCT, INTERSECT, INTERSECT ALL, INTERSECT DISTINCT, EXCEPT, 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;
```

Parameters:

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

Examples:

UNION:

- If UNION ALL is used, all records of the two datasets are returned.

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

  The following result is returned:

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

- If multiple UNION ALL clauses are used, parentheses can be used to prioritize the clauses.

  ```
  SELECT * FROM src UNION ALL (SELECT * FROM src2 UNION ALL SELECT * FROM src3);
  ```

- If UNION is used, duplicate records are not included in the returned records. UNION is equivalent to UNION DISTINCT.

  ```
  SELECT * FROM VALUES (1, 2), (1, 2), (3, 4) t(a, b)
  UNION
  SELECT * FROM VALUES (1, 2), (1, 4) t(a, b);
  -- The preceding statement is equivalent to the following statement:
  SELECT DISTINCT * FROM (<Result of UNION ALL>)t;
  ```

  The following result is returned:

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

- Assume that UNION is followed by a CLUSTER BY, DISTRIBUTE BY, SORT BY, ORDER BY, or LIMIT clause. If `set odps.sql.type.system.odps2=false;` is used, the clause works on the result of the last select_statement after the UNION operator. If `odps.sql.type.system.odps2=true;` is used, the clause works on the result of all the UNION operations.

  ```
  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 LIMI
  T 3;
  ```

  The following result is returned:

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

INTERSECT:

- INTERSECT ALL:

  ```
  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);
  ```

  The following result is returned:

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

- INTERSECT DISTINCT:

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

  The following result is returned. The preceding statement is equivalent to the `SELECT DISTINCT * FROM (< Result of INTERSECT ALL >) t;` statement.

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

EXCEPT:

- EXCEPT ALL:

```
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);
```

The following result is returned:

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

- EXCEPT DISTINCT:

```
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);
```

The following result is returned. The preceding statement is equivalent to the `SELECT DISTINCT * FROM left_branch EXCEPT ALL SELECT DISTINCT * FROM right_branch;` statement.

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

Note

- Set operations do not necessarily return sorted results.
- The two branches for a set operation must have the same number of output columns with consistent data types. If data types are inconsistent, implicit conversion of the data types may be performed. To ensure compatibility for set operations, MaxCompute forbids the implicit conversion between STRING and other data types.
- MaxCompute supports a maximum of 256 branches for a set operation. If the limit is exceeded, an error is returned.

1.6.5.3. EXPLAIN

This topic describes the EXPLAIN statement in MaxCompute SQL.

This statement shows the structure of the execution plan of a DML statement. An execution plan is a program that executes SQL statements.

Syntax:

```
EXPLAIN <DML query>;
```

> **Note**
>
> The execution results of an EXPLAIN statement include:
>
> - Dependencies among all the jobs of this DML statement
> - Dependencies among all the tasks of each job
> - Dependencies among all operators in a task

Examples:

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

The output of the EXPLAIN statement consists of the following three parts:

- Job dependencies: `job0 is root job`. This query requires only one job (job0). Therefore, only one row of information is displayed.
- Task dependencies. Example:

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

  > **Note**
  >
  > - Job0 contains three tasks, M1_Stg1, M2_Stg1, and J3_1_2_Stg1. J3_1_2_Stg1 is executed after the M1_Stg1 and M2_Stg1 tasks are complete.
  > - Naming conventions for a task: MaxCompute provides the task types of map, reduce, join, and local work. The first letter in a task name indicates the type of the task. For example, M2Stg1 is a map task. The number following the first letter indicates the task ID. This ID is unique among all tasks that correspond to the current query. Digits that are combined by underscores (_) indicate the task dependencies. For example, J3_1_2_Stg1 indicates that the current task with the ID of 3 depends on two tasks whose IDs are 1 and 2.

- Operator structure, where each operator string describes the execution semantics of a task.

```
In Task M1_Stg1:
  Data source: yudi_2.src                          # Data source describes the input of the task.
  TS: alias: 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                         # FileSinkOperator
In Task M2_Stg1:
  Data source: yudi_2.src1
  TS: alias: b
      RS: order: +
          keys:
               b.value
          values:
               b.value
          partitions:
               b.value
```

The following table describes the operators.

Operator	Description
TableScanOperator	Describes the logic of FROM statement blocks in a query statement. The alias of the input table is displayed in the EXPLAIN results.
SelectOperator	Describes the logic of SELECT statement blocks in a query statement. The columns that are transferred to the next operator are displayed in the EXPLAIN results. Separate multiple columns with commas (,). ◦ If columns are referenced, the value is in the < alias >.< column_name > format. ◦ If the result of an expression is transferred, the value is displayed as a function, such as func1(arg1_1, arg1_2, func2(arg2_1, arg2_2)). ◦ If constants are transferred, the value is immediately displayed.
FilterOperator	Describes the logic of WHERE statement blocks in a query statement. The EXPLAIN results include a WHERE clause, with the display rules that are similar to those of SelectOperator.
JoinOperator	Describes the logic of JOIN statement blocks in a query statement. The EXPLAIN results show which tables are joined in which way.
GroupByOperator	Describes the logic of aggregate operations. This operator 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 data distribution between tasks. If the result of the current task is transferred to another task, ReduceSinkOperator must be used to distribute data at the last stage of the current task. The sorting method of output data records, the distributed keys and values, and the columns used to calculate the hash value are displayed in the EXPLAIN results.

Operator	Description
FileSinkOperator	Describes the logic of storage operations on final data records. If an INSERT statement block exists in a query, the name of the required table is displayed in the EXPLAIN results.
LimitOperator	Describes the logic of LIMIT statement blocks in a query statement. The number of records specified by LIMIT is displayed in the EXPLAIN results.
MapjoinOperator	Describes JOIN operations on large tables. This operator is similar to JoinOperator.

> **Note**
> - If a query statement is complex, the EXPLAIN statement returns excessive results. As a result, API limits are reached and complete results cannot be obtained. In this case, you can split a query and execute the EXPLAIN statement on each subquery to understand the structure of the job.
> - If you specify more than 10,000 partitions in a query, large amounts of source data needs to be read. To circumvent this limit, you can add filter conditions in the query to filter out most partitions.

1.6.5.4. IF statement

This topic describes the IF statement that MaxCompute SQL supports.

MaxCompute SQL supports the IF-ELSE statement. You can use the IF-ELSE statement to execute SQL scripts with specific conditions. A 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 if 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;
```

1.6.5.5. UPDATE and DELETE

This topic describes the UPDATE and DELETE statements in MaxCompute SQL.

The UPDATE and DELETE statements are used to manage data of specific rows in tables or partitions.

Background information

MaxCompute data manipulation language (DML) statements have limits on table operations. It allows you to use only the INSERT OVERWRITE and INSERT INTO statements to delete and update some rows in a table. For example, if you use an INSERT statement to modify small amounts of data in a table or partition, you must use a SELECT statement to read all data and modify the data. Then, execute the INSERT statement to insert the modified data into the table or partition. This is a low-efficiency process. However, if you use the UPDATE and DELETE statements, the amount of data read or write in a query significantly decreases.

History tables were considered the best to perform multiple UPDATE operations at a time. If you use a history table, you must add auxiliary columns, such as start_date and end_date, in the table. These columns indicate the lifecycle of the data records in a row. To query the current status of a table, you must find the latest status from large amounts of data based on the timestamp. This is a time-consuming process. However, the UPDATE and DELETE statements directly read data from the current table.

Description

When you execute a DELETE statement, the system automatically generates a Delta file. txnid(bigint) and rowid(bigint) in the Delta file indicate which rows in which transactional table are deleted from the Base file.

> **Note** The Delta file stores the row numbers that are used to indicate the deleted rows in the Base file.

If you execute a DELETE statement again, another Delta file named f2.del is generated. The f2.del file also stores the row numbers based on the Base file. During data reading, all Delta files are queried and data that is not deleted is returned.

Similarly, the UPDATE statement is converted into the combination of the DELETE and INSERT INTO statements.

If jobs are run in parallel and the tables on which you want to perform operations are the same, conflicts may occur. For more information, see ACID semantics of MaxCompute concurrent write jobs.

Usage notes

- The UPDATE and DELETE operations must be performed based on transactional tables. For more information about how to create a transactional table, see Create a table.
- When you execute an UPDATE statement, you must add `set odps.service.mode=off;` before the statement and commit them at the same time.
- If the ratio of the rows that you want to delete or update to all rows is small, the performance of the data reads from a table is slightly affected. In this case, we recommend that you use the UPDATE or DELETE statement. If the ratio is less than 5%, we recommend that you use the UPDATE or DELETE statement. The maximum ratio varies based on your business scenarios.
- If you perform the UPDATE or DELETE operation multiple times to update or delete a small number of rows in a table, we recommend that you execute the COMPACT statement after the UPDATE and DELETE statements. This reduces the storage usage of the table.
- If you perform less UPDATE or DELETE operations to update or delete a large number of rows in a table and will frequently read data from the table, we recommend that you use the INSERT OVERWRITE and INSERT INTO statements. For example, a total of 10% of the data in a table needs to be updated or deleted ten times a day. In this case, we recommend that you evaluate the sum of the costs of the UPDATE or DELETE statement and the read performance consumption based on the generated Delta file. Then, compare it with that of the INSERT OVERWRITE and INSERT INTO statements to select an efficient method.
- After you delete data, a Delta file is generated. As a result, the storage resources may not be released. If you want to use the DELETE statement to delete data and release the storage resources, you can use the ALTER TABLE COMPACT statement to merge the Base files with Delta files.

Limits

- If you use the ALTER TABLE COMPACT statement in MaxCompute SQL mode of DataWorks, a compatibility issue occurs. To handle this issue, we recommend that you use the MaxCompute client whose version must be 0.35.0.
- You can specify the transactional attribute for a table only when you create the table. However, you are not allowed to modify the transactional attribute of an existing table by using the ALTER TABLE COMPACT statement. If you modify the transactional attribute of an existing table, an error is returned.

```
alter table not_txn_tbl set tblproperties("transactional"="true");
-- FAILED is returned. Catalog Service Failed, ErrorCode: 151, Error Message: Set transactional is
not supported.
```

- You cannot specify clustered tables or external tables as transactional tables.
- Existing internal tables or external tables cannot be converted into transactional tables, and transactional tables cannot be converted into standard tables.
- Tasks from peripheral systems, such as Graph and Machine Learning Platform for AI (PAI), are not supported. If these tasks access a transactional table, an error is returned.
- CLONE TABLE, MERGE PARTITION, and incremental replication operations cannot be performed on transactional tables.
- You cannot use Kunlunjing to back up data in transactional tables. If you want to perform the UPDATE, DELETE, or INSERT OVERWRITE operation on important data in transactional tables, you must back up the data in advance.
- You must have the SELECT and UPDATE permissions on a table to perform the UPDATE and DELETE operations.

DELETE

Syntax:

```
delete from <table_name> [WHERE where_condition];
```

Description: This statement is used to delete some rows from a table.

The following examples are provided:

Example 1: Delete some rows from a table.

```
-- Create a transactional table.
create table if not exists acid_delete(id bigint) tblproperties ("transactional"="true");
-- Insert data into the table.
insert overwrite table acid_delete values(1),(2),(3);
-- Check the inserted data.
select * from acid_delete;
```

The following information is returned:

```
+------------+
| id         |
+------------+
| 1          |
| 2          |
| 3          |
+------------+
```

```
-- Delete the row whose id value is 2. You must enter Yes or No to confirm the operation in the MaxCompute client.
delete from acid_delete where id = 2;
-- Read data from the table. Only the rows whose id values are 1 and 3 are retained in the table.
select * from acid_delete;
```

The following information is returned:

```
+------------+
| id         |
+------------+
| 1          |
| 3          |
+------------+
```

Example 2: Delete some rows from a partitioned table.

```
-- Create a transactional table.
create table if not exists acid_delete_pt(id bigint) partitioned by(ds string) tblproperties ("transactional"="true");
-- Add partitions to the table.
alter table acid_delete_pt add if not exists partition (ds= '2019');
-- Insert data into the partitions.
insert overwrite table acid_delete_pt partition (ds='2019') values(1),(2),(3);
-- Query the table to check whether data is inserted.
select * from acid_delete_pt where ds = '2019';
```

The following information is returned:

```
+-------+--------+
| id    | ds     |
+-------+--------+
| 1     | 2019   |
| 2     | 2019   |
| 3     | 2019   |
+-------+--------+
```

```
-- Delete data from the partitions. You must enter Yes or No to confirm the operation in the MaxCompute client.
delete from acid_delete_pt where ds='2019' and id= 2;
-- Read data from the table. Only the rows whose id values are 1 and 3 are retained in the table.
select * from acid_delete_pt where ds = '2019';
```

The following information is returned:

```
+-------+--------+
| id    | ds     |
+-------+--------+
| 1     | 2019   |
| 3     | 2019   |
+-------+--------+
```

UPDATE

Syntax:

```
update <table_name> set col1 = value1 [, col2 = value2 ...] [WHERE where_condition];
update <table_name> set (col1 [, col2 ...]) = (value1 [, value2 ...]);
```

Description: This statement is used to update some rows in a table.

> **Note** The UPDATE statement is converted into the combination of the DELETE and INSERT INTO statements to update some rows.

The following examples are provided:

Example 1: Update some rows in a table.

```
-- Create a transactional table.
create table if not exists acid_update(id bigint) tblproperties ("transactional"="true");
-- Insert data into the table.
insert overwrite table acid_update values(1),(2),(3);
-- Query the table to check whether data is inserted.
select * from acid_update;
```

The following information is returned:

```
+------------+
| id         |
+------------+
| 1          |
| 2          |
| 3          |
+------------+
```

```
-- Update the value 2 in the id column to 4.
set odps.service.mode=off;
update acid_update set id = 4 where id = 2;
-- Query the table to check whether the value 2 is updated to 4 in the id column.
select * from acid_update;
```

The following information is returned:

```
+------------+
| id         |
+------------+
| 4          |
| 1          |
| 3          |
+------------+
```

Example 2: Update some rows in a partitioned table.

```
-- Create a transactional table.
create table if not exists acid_update_pt(id bigint) partitioned by(ds string) tblproperties ("trans
actional"="true");-- Add a partition.
-- Add a partition to the table.
alter table acid_update_pt add if not exists partition (ds= '2019');
-- Insert data into the table.
insert overwrite table acid_update_pt partition (ds='2019') values(1),(2),(3);
-- Query the table to check whether data is inserted.
select * from acid_update_pt where ds = '2019';
```

The following information is returned:

```
+-------+--------+
| id    | ds     |
+-------+--------+
| 1     | 2019   |
| 2     | 2019   |
| 3     | 2019   |
+-------+--------+
```

```
-- Update the id values of rows in partition 2019 to 4. The rows must have the id values of 2.
set odps.service.mode=off;
update acid_update_pt set id = 4 where ds = '2019' and id = 2;
-- Query the table to check whether the value 2 is updated to 4 in the id column of the rows that me
et the specified conditions.
select * from acid_update_pt where ds = '2019';
```

The following information is returned:

```
+-------+--------+
| id    | ds     |
+-------+--------+
| 4     | 2019   |
| 1     | 2019   |
| 3     | 2019   |
+-------+--------+
```

Example 3: Update several columns and update a destination table by using the data in a source table.

```
-- Create a destination table named acid_update_t and create a table named acid_update_s for joins.
create table if not exists acid_update_t(id bigint,value1 bigint,value2 bigint) tblproperties ("tran
sactional"="true");
create table if not exists acid_update_s(id bigint,value1 bigint,value2 bigint);
-- Insert data into these tables.
insert overwrite table acid_update_t values(2,20,21),(3,30,31),(4,40,41);
insert overwrite table acid_update_s values(1,100,101),(2,200,201),(3,300,301);
--1.Use constants to update the acid_update_t table.
set odps.service.mode=off;
update acid_update_t set (value1, value2) = (3, 4);
--2.Use the data in the acid_update_s table to update the acid_update_t table. The acid_update_t tab
le is left joined with the acid_update_s table. If you want to update only the intersection of the t
wo tables, you must add the WHERE clause in update <table_name>.
set odps.service.mode=off;
update acid_update_t set (value1, value2) = (select value1, value2 from acid_update_s where acid_upd
ate_t.id = acid_update_s.id);
--3.Use the data in the acid_update_s table to update the acid_update_t table. Add filter conditions
for the acid_update_t table to update only the intersection of the two tables.
set odps.service.mode=off;
update acid_update_t set (value1, value2) = (select value1, value2 from acid_update_s where acid_upd
ate_t.id = acid_update_s.id) where acid_update_t.id in (select id from acid_update_s);
--4.Use the aggregate results of the acid_update_t and acid_update_s tables to update the acid_updat
e_t table.
set odps.service.mode=off;
update acid_update_t set (id, value1, value2) = (select id, max(value1),max(value2) from acid_update
_s where acid_update_t.id = acid_update_s.id group by acid_update_s.id) where acid_update_t.id in (s
elect id from acid_update_s);
```

MERGE INTO

The MERGE INTO statement encapsulates the INSERT, UPDATE, and DELETE operations. You can use this statement to perform multiple operations on a destination table after the table is joined with a source table.

MaxCompute DML provides the UPDATE and DELETE statements to enrich its expressions. However, if you want to use several INSERT, UPDATE, and DELETE statements to perform multiple UPDATE operations on a destination table at a time, you must execute several statements, which triggers multiple full table scans. The MERGE INTO statement requires only one full table scan for you to complete the operations. The MERGE INTO statement is more efficient than the INSERT, UPDATE, and DELETE statements.

If you use the INSERT, UPDATE, and DELETE statements in a job to update a table, the update succeeds only when all the statements are successfully executed. If only some of these statements are successfully executed, the execution cannot be rolled back. However, the MERGE INTO statement is atomic, which can be used to prevent this issue.

Syntax:

```
MERGE INTO <target table> AS T USING <source expression/table> AS S
ON <boolean expression1>
WHEN MATCHED [AND <boolean expression2>] THEN UPDATE SET <set clause list>
WHEN MATCHED [AND <boolean expression3>] THEN DELETE
WHEN NOT MATCHED [AND <boolean expression4>] THEN INSERT VALUES <value list>
```

Description: This statement is used to update or insert data from the source expression or table into the destination table. This statement is also used to delete data from the source expression or table from the destination table.

Parameter:

- source expression/table: the source table. The value of this parameter can be a table, view, or subquery.
- target table: the destination table, which must be an existing physical table.
- ON: the clause that is used to determine whether the source table is joined with the destination table. boolean expression specifies the join condition.
- WHEN MATCHED...THEN: specifies the action when the ON clause determines that the two tables are joined. Data obtained from multiple WHEN MATCHED [AND <boolean expression>] clauses must have no intersections. The MERGE INTO statement has clear objectives. You are not allowed to perform the INSERT or UPDATE operation on the same row in a single MERGE INTO statement. When you execute the MERGE INTO statement, make sure that the data in the source tables that meet the ON joint condition is unique. Otherwise, an error is returned.

  > **Note**
  > - If a MERGE INTO statement has three WHEN clauses, the UPDATE, DELETE, and INSERT operations can appear at most once in each clause.
  > - WHEN NOT MATCHED must be the last WHEN clause and can be used only for an INSERT operation.
  > - If both the UPDATE and DELETE statements are included in a MERGE INTO statement, the statement that appears first must include [AND <boolean expression>].

The following examples are provided:

Prepare the test data.

```
-- Create a destination table.
create table if not exists acid_address_book_base1
(id bigint,first_name string,last_name string,phone string)
partitioned by(year string, month string, day string, hour string)
tblproperties ("transactional"="true");
-- Create a source table.
create table if not exists tmp_table1
(id bigint, first_name string, last_name string, phone string, _event_type_ string);
-- Insert the test data into the destination table.
insert overwrite table acid_address_book_base1
partition(year='2020', month='08', day='20', hour='16')
values (4, 'nihaho', 'li', '222'), (5, 'tahao', 'ha', '333'),
(7, 'djh', 'hahh', '555');
-- Insert the test data into the source table.
insert overwrite table tmp_table1 values
(1, 'hh', 'liu', '999', 'I'), (2, 'cc', 'zhang', '888', 'I'),
(3, 'cy', 'zhang', '666', 'I'),(4, 'hh', 'liu', '999', 'U'),
(5, 'cc', 'zhang', '888', 'U'),(6, 'cy', 'zhang', '666', 'U');
```

If data meets the ON joint condition and is from the source table, you can use the data to perform the MERGE INTO operation on the destination table. If data does not meet the ON joint condition but is of the I type specified by event_type in the source table, you can insert the data into the destination table.

```
merge into acid_address_book_base1 as t using tmp_table1 as s
on s.id = t.id and t.year='2020' and t.month='08' and t.day='20' and t.hour='16'
when matched
then update set t.first_name = s.first_name, t.last_name = s.last_name, t.phone = s.phone
when not matched and (s._event_type_='I')
then insert values(s.id, s.first_name, s.last_name,s.phone,'2020','08','20','16');
-- Query the destination table after the preceding operations are complete.
select * from acid_address_book_base1;
```

The following information is returned:

```
+------------+------------+-----------+-------+------+-------+-----+------+
| id         | first_name | last_name | phone | year | month | day | hour |
+------------+------------+-----------+-------+------+-------+-----+------+
| 4          | hh         | liu       | 999   | 2020 | 08    | 20  | 16   |
| 5          | cc         | zhang     | 888   | 2020 | 08    | 20  | 16   |
| 7          | djh        | hahh      | 555   | 2020 | 08    | 20  | 16   |
| 1          | hh         | liu       | 999   | 2020 | 08    | 20  | 16   |
| 2          | cc         | zhang     | 888   | 2020 | 08    | 20  | 16   |
| 3          | cy         | zhang     | 666   | 2020 | 08    | 20  | 16   |
+------------+------------+-----------+-------+------+-------+-----+------+
```

ALTER TABLE COMPACT

If you perform DELETE operations on a transactional table, the Base file that you want to delete is not modified. A Delta file is generated every time you perform a DELETE operation. As a result, if you perform multiple DELETE operations, the generated Delta files occupy more storage resources.

If you perform several DELETE operations on the same table or partition, a large number of Delta files may be generated based on the amount of the data that you delete. Similarly, if you perform an UPDATE operation which is converted into the combination of the DELETE and INSERT INTO operations, Delta files are also generated. If the system reads data from the table, it must load the generated Delta files to find the deleted rows. A large number of Delta files reduce data reading efficiency.

In this case, you can use the ALTER TABLE COMPACT statement to merge the Base files with Delta files to reduce storage usage and improve data reading efficiency.

Syntax:

```
alter table <table_name> [partition (partition_key = 'partition_value' [, ...])] compact [major|mino
r];
```

Description: This statement is used to merge the Base files with Delta files to reduce storage usage and improve data reading efficiency.

- Minor compaction: merges all Delta files with their Base files. The Delta files must be generated based on the same Base file. Delta files that are generated based on different Base files cannot be merged.
- Major compaction: merges all Delta files generated based on the same Base file with the Base file, deletes the Delta files, and merges small files in multiple Base files that correspond to tables. If the size of the Base file is less than 32 MB or a Delta file is generated, the INSERT OVERWRITE operation is performed on the table. However, if the size of the Base file is greater than or equal to 32 MB or no Delta files are generated, the INSERT OVERWRITE operation is not performed on the table.

Sample statements:

```
alter table acid_delete compact minor;
alter table acid_update_pt partition (ds = '2019') compact major;
```

The following information is returned:

```
Summary:
table name: acid_update_pt /ds=2019  instance count: 2  run time: 16
  before merge, file count:        8  file size: 2423  file physical size: 7269
   after merge, file count:        2  file size: 642  file physical size: 1926
```

1.6.6. Built-in functions

1.6.6.1. Mathematical functions

1.6.6.1.1. ABS

This topic describes the ABS function in mathematical functions and provides examples.

Syntax:

```
double abs(double number)
bigint abs(bigint number)
decimal abs(decimal number)
```

Description: This function calculates the absolute value of number.

Parameters: The number parameter supports a value of the BIGINT, DOUBLE, or DECIMAL type. If the input value is of the BIGINT, DOUBLE, or DECIMAL type, a value of the same type is returned. If the input value is of the STRING type, it is implicitly converted to a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: The type of the return value depends on that of the input value, which can be DOUBLE, BIGINT, or DECIMAL. If the input value is NULL, NULL is returned.

> **Note** If the input value is of the BIGINT type and is greater than the maximum value of the BIGINT type, a value of the DOUBLE type is returned. However, the precision may be lost.

Examples:

```
abs(null)=null
abs(-1)=1
abs(-1.2)=1.2
abs("-2")=2.0
abs(12232083745629837659238745692374 8)=1.2232083745629837e32
```

The following example shows the usage of an ABS function in SQL statements. Other built-in functions, except window functions and aggregate functions, are used in a similar way.

```
select abs(id) from tbl1;
-- Calculate the absolute value of the id field in tbl1.
```

1.6.6.1.2. ACOS

This topic describes the ACOS function in mathematical functions and provides examples.

Syntax:

```
double acos(double number)
decimal acos(decimal number)
```

Description: This function calculates the arccosine of number.

Parameters: The number parameter supports a value of the DOUBLE or DECIMAL type. The value ranges from -1 to 1. If the input value is of the STRING or BIGINT type, it is implicitly converted to a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned. The value ranges from 0 to π. If the input value is NULL, NULL is returned.

Examples:

```
acos("0.87") = 0.5155940062460905
acos(0) = 1.5707963267948966
```

1.6.6.1.3. ASIN

This topic describes the ASIN function in mathematical functions and provides examples.

Syntax:

```
double asin(double number)
decimal asin(decimal number)
```

Description: This function calculates the arcsine of number.

Parameters: The number parameter supports a value of the DOUBLE or DECIMAL type. The value ranges from -1 to 1. If the input value is of the STRING or BIGINT type, it is implicitly converted to a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned. The value ranges from -π/2 to π/2. If the input value is NULL, NULL is returned.

Examples:

```
asin(1) = 1.5707963267948966
asin(-1) = -1.5707963267948966
```

1.6.6.1.4. ATAN

This topic describes the ATAN function in mathematical functions and provides examples.

Syntax:

```
double atan(double number)
```

Description: This function calculates the arctangent of number.

Parameters: The number parameter supports a value of the DOUBLE type. If the input value is of the STRING or BIGINT type, it is implicitly converted to a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the DOUBLE type is returned. The value ranges from -π/2 to π/2. If the input value is NULL, NULL is returned.

Examples:

```
atan(1) = 0.7853981633974483;
atan(-1) = -0.7853981633974483
```

1.6.6.1.5. CEIL

This topic describes the CEIL function in mathematical functions and provides examples.

Syntax:

```
bigint ceil(double value)
bigint ceil(decimal value)
```

Description: This function rounds up value and returns the nearest integer.

Parameters: The value parameter supports a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted to a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the BIGINT type is returned. If the input value is NULL, NULL is returned.

Examples:

```
ceil(1.1) = 2
ceil(-1.1) = -1
```

1.6.6.1.6. CONV

This topic describes the CONV function in mathematical functions and provides examples.

Syntax:

```
string conv(string input, bigint from_base, bigint to_base)
```

Description: This function converts a number from one number system to another.

Parameters:

- input: the integer you want to convert, which is of the STRING type. If the input value is of the BIGINT or DOUBLE type, it is implicitly converted to a value of the STRING type before calculation.
- from_base, to_base: decimal numbers. The values can be 2, 8, 10, or 16. If the input value is of the STRING or DOUBLE type, it is implicitly converted to a value of the BIGINT type before calculation.

Return value: A value of the STRING type is returned. If an input value is NULL, NULL is returned. The conversion process runs at 64-bit precision. If an overflow occurs, an error is returned. If the input value is a negative value that begins with an en dash (-), an error is returned. If the input value is a decimal, it is converted to an integer before the conversion of number systems. The decimal part is left out.

Examples:

```
conv('1100', 2, 10) = '12'
conv('1100', 2, 16) = 'c'
conv('ab', 16, 10) = '171'
conv('ab', 16, 16) = 'ab'
```

1.6.6.1.7. COS

This topic describes the COS function in mathematical functions and provides examples.

Syntax:

```
double cos(double number)
decimal cos(decimal number)
```

Description: This function calculates the cosine of number, which is a radian value.

Parameters: number, a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned. If the input value is NULL, NULL is returned.

Examples:

```
cos(3.1415926/2) = 2.6794896585028633e-8
cos(3.1415926) = -0.9999999999999986
```

1.6.6.1.8. COSH

This topic describes the COSH function in mathematical functions.

Syntax:

```
double cosh(double number)
decimal cosh(decimal number)
```

Description: This function calculates the hyperbolic cosine of number.

Parameters: The number parameter supports a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted to a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned. If the input value is NULL, NULL is returned.

1.6.6.1.9. COT

This topic describes the COT function in mathematical functions.

Syntax:

```
double cot(double number)
decimal cot(decimal number)
```

Description: This function calculates the cotangent of number, which is a radian value.

Parameters: number, a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned. If the input value is NULL, NULL is returned.

1.6.6.1.10. EXP

This topic describes the EXP function in mathematical functions.

Syntax:

```
double exp(double number)
decimal exp(decimal number)
```

Description: This function calculates the exponential value of number.

Parameters: The number parameter supports a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted to a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: The exponential value of number is returned. The value is of the DOUBLE or DECIMAL type. If the input value is NULL, NULL is returned.

1.6.6.1.11. FLOOR

This topic describes the FLOOR function in mathematical functions and provides examples.

Syntax:

```
bigint floor(double number)
bigint floor(decimal number)
```

Description: This function rounds down number and returns the nearest integer that is no greater than the value of number.

Parameters: The number parameter supports a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted to a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the BIGINT type is returned. If the input value is NULL, NULL is returned.

Examples:

```
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
```

1.6.6.1.12. LN

This topic describes the LN function in mathematical functions.

Syntax:

```
double ln(double number)
decimal ln(decimal number)
```

Description: This function calculates the natural logarithm of number.

Parameters: The number parameter supports a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted to a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned. If the input value is NULL, NULL is returned. If the input value is a negative value or 0, an error is returned.

Examples:

```
select ln(1000);
```

The following result is returned:

```
+------------+
| _c0        |
+------------+
| 6.907755278982137 |
+------------+
```

1.6.6.1.13. LOG

This topic describes the LOG function in mathematical functions.

Syntax:

```
double log(double base, double x)
decimal log(decimal base, decimal x)
```

Description: This function calculates the logarithm of x whose base number is base.

Parameters:

- base: a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.
- x: a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: The logarithm value of the DOUBLE or DECIMAL type is returned. If an input value is NULL, NULL is returned. If an input value is a negative value or 0, an error is returned. If the value of base is 1, an error is returned. The value 1 causes division by zero.

Examples:

```
select log(10,100);
```

The following result is returned:

```
+------------+
| _c0        |
+------------+
| 2.0        |
+------------+
```

1.6.6.1.14. POW

This topic describes the POW function in mathematical functions.

Syntax:

```
double pow(double x, double y)
decimal pow(decimal x, decimal y)
```

Description: This function calculates the yth power of x, namely, x^y.

Parameters:

- x: a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation. If the input value is of another data type, an

error is returned.

- y: a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned. If an input value is NULL, NULL is returned.

Examples:

```
select pow(10,2);
```

The following result is returned:

```
+------------+
| _c0        |
+------------+
| 100.0      |
+------------+
```

1.6.6.1.15. RAND

This topic describes the RAND function in mathematical functions and provides examples.

Syntax:

```
double rand(bigint seed)
```

Description: This function returns a random number of the DOUBLE type based on seed. The value ranges from 0 to 1.

Parameters: The seed parameter supports a value of the BIGINT type. It specifies the starting point in generating random numbers. This parameter is optional.

Return value: A value of the DOUBLE type is returned.

Examples:

```
select rand();
select rand(1);
```

1.6.6.1.16. ROUND

This topic describes the ROUND function in mathematical functions and provides examples.

Syntax:

```
double round(double number, [bigint decimal_places])
decimal round(decimal number, [bigint decimal_places])
```

Description: This function returns a number rounded to the specified decimal place.

Parameters:

- number: a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted to a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.
- decimal_places: a constant of the BIGINT type. It specifies the decimal place to which the number is rounded. If the input value is of another data type, an error is returned. If this parameter is not specified, the number is

rounded to the ones place. The default value is 0.

> **Note** decimal_places supports negative values. A negative value indicates counting from the decimal point to the left, and the decimal part is excluded. If decimal_places exceeds the length of the integer part, 0 is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned. If an input value is NULL, NULL is returned.

Examples:

```
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
```

1.6.6.1.17. SIN

This topic describes the SIN function in mathematical functions.

Syntax:

```
double sin(double number)
decimal sin(decimal number)
```

Description: This function calculates the sine of number, which is a radian value.

Parameters: The number parameter supports a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted to a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned. If the input value is NULL, NULL is returned.

1.6.6.1.18. SINH

This topic describes the SINH function in mathematical functions.

Syntax:

```
double sinh(double number)
decimal sinh(decimal number)
```

Description: This function calculates the hyperbolic sine of number.

Parameters: The number parameter supports a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted to a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned. If the input value is NULL, NULL is returned.

1.6.6.1.19. SQRT

This topic describes the SQRT function in mathematical functions.

Syntax:

```
double sqrt(double number)
decimal sqrt(decimal number)
```

Description: This function calculates the square root of number.

Parameters: The number parameter supports a value of the DOUBLE or DECIMAL type. It must be greater than 0. If it is less than 0, an error is returned. If the input value is of the STRING or BIGINT type, it is implicitly converted to a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned. If the input value is NULL, NULL is returned.

1.6.6.1.20. TAN

This topic describes the TAN function in mathematical functions.

Syntax:

```
double tan(double number)
decimal tan(decimal number)
```

Description: This function calculates the tangent of number, which is a radian value.

Parameters: number, a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned. If the input value is NULL, NULL is returned.

1.6.6.1.21. TANH

This topic describes the TANH function in mathematical functions.

Syntax:

```
double tanh(double number)
decimal tanh(decimal number)
```

Description: This function calculates the hyperbolic tangent of number.

Parameters: The number parameter supports a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted to a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned. If the input value is NULL, NULL is returned.

1.6.6.1.22. TRUNC

This topic describes the TRUNC function in mathematical functions and provides examples.

Syntax:

```
double trunc(double number[, bigint decimal_places])
decimal trunc(decimal number[, bigint decimal_places])
```

Description: This function truncates the input value of number to a specified number of decimal places.

Parameters:

- number: a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into a value of the DOUBLE type before calculation. If the input value is of another data type, an error is returned.
- decimal_places: the decimal place, which is a constant of the BIGINT type. This parameter indicates the position where the number is truncated. If it is of another data type, it is implicitly converted into the BIGINT type. If you do not specify this parameter, the number is truncated to the ones place.

> **Note** decimal_places can be a negative value, which indicates that the number is truncated from the decimal point to the left and the decimal part is left out. If decimal_places exceeds the length of the integer part, 0 is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned. If an input value is NULL, NULL is returned.

> **Note**
> - If a value of the DOUBLE type is returned, the return value may not be properly displayed. This issue exists in all systems. For more information, see trunc(125.815,1) in the following example.
> - The number is filled with zeros from the specified position.

Examples:

```
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
```

1.6.6.1.23. Additional mathematical functions

1.6.6.1.23.1. Usage notes

This topic provides notes for you to use additional mathematical functions.

MaxCompute 2.0 provides additional mathematical functions. If you want to call additional mathematical functions, you must add the following SET statement before the SQL statement that contains the additional mathematical functions:

```
set odps.sql.type.system.odps2=true;
```

> **Note** You must simultaneously submit and execute the SET statement and the SQL statement that contains the additional mathematical functions.

1.6.6.1.23.2. LOG2

This topic describes the LOG2 function in mathematical functions and provides examples.

Syntax:

```
double log2(double number)
double log2(decimal number)
```

Description: This function calculates the logarithm of number with the base number of 2.

Parameters: number, a value of the DOUBLE or DECIMAL type.

Return value: A value of the DOUBLE type is returned. If the input value is 0 or NULL, NULL is returned.

Examples:

```
log2(null) = null
log2(0) = null
log2(8) = 3.0
```

1.6.6.1.23.3. LOG10

This topic describes the LOG10 function in mathematical functions and provides examples.

Syntax:

```
double log10(double number)
double log10(decimal number)
```

Description: This function calculates the logarithm of number with the base number of 10.

Parameters: The number parameter supports a value of the DOUBLE or DECIMAL type.

Return value: A value of the DOUBLE type is returned. If the input value is 0 or NULL, NULL is returned.

Examples:

```
log10(null)=null
log10(0)=null
log10(8)=0.9030899869919435
```

1.6.6.1.23.4. BIN

This topic describes the BIN function in mathematical functions and provides examples.

Syntax:

```
string bin(bigint number)
```

Description: This function calculates the binary code of number.

Parameters: The number parameter supports a value of the BIGINT type.

Return value: A value of the STRING type is returned. If the input value is 0, 0 is returned. If the input value is NULL, NULL is returned.

Examples:

```
bin(0) = '0'
bin(null) = 'null'
bin(12) = '1100'
```

1.6.6.1.23.5. HEX

This topic describes the HEX function in mathematical functions and provides examples.

Syntax:

```
string hex(bigint number)
string hex(string number)
string hex(binary number)
```

Description: This function converts an integer or a string into a hexadecimal number.

Parameters: If number is of the BIGINT type, a hexadecimal number is returned. If number is of the STRING type, a string in hexadecimal format is returned.

Return value: A value of the STRING type is returned. If the input value is 0, 0 is returned. If the input value is NULL, NULL is returned.

Examples:

```
hex(0) = '0'
hex('abc') = '616263'
hex(17) = '11'
hex('17') = '3137'
hex(null)
-- An exception occurs and the execution fails.
```

1.6.6.1.23.6. UNHEX

This topic describes the UNHEX function in mathematical functions and provides examples.

Syntax:

```
binary unhex(string number)
```

Description: This function converts a hexadecimal string into a string.

Parameters: The number parameter specifies a hexadecimal string.

Return value: A value of the BINARY type is returned. If the input value is 0, an error is returned. If the input value is NULL, NULL is returned.

Examples:

```
unhex('616263') = 'abc'
unhex(616263) = 'abc'
```

1.6.6.1.23.7. RADIANS

This topic describes the RADIANS function in mathematical functions and provides examples.

Syntax:

```
double radians(double number)
```

Description: This function converts a degree into a radian value.

Parameters: number, a value of the DOUBLE type.

Return value: A value of the DOUBLE type is returned. If the input value is NULL, NULL is returned.

Examples:

```
radians(90) = 1.5707963267948966
radians(0) = 0.0
radians(null) = null
```

1.6.6.1.23.8. DEGREES

This topic describes the DEGREES function in mathematical functions and provides examples.

Syntax:

```
double degrees(double number)
double degrees(decimal number)
```

Description: This function converts a radian value to a degree.

Parameters: The number parameter supports a value of the DOUBLE or DECIMAL type.

Return value: A value of the DOUBLE type is returned. If the input value is NULL, NULL is returned.

Examples:

```
degrees(1.5707963267948966) = 90.0
degrees(0) = 0.0
degrees(null) = null
```

1.6.6.1.23.9. SIGN

This topic describes the SIGN function in mathematical functions and provides examples.

Syntax:

```
double sign(double number)
double sign(decimal number)
```

Description: This function returns the sign of an input value.

Parameters: The number parameter supports a value of the DOUBLE or DECIMAL type.

Return value: A value of the DOUBLE type is returned. If the input value is a positive value, 1.0 is returned. If the input value is a negative value, -1.0 is returned. If the input value is 0, 0.0 is returned. If the input value is NULL, NULL is returned.

Examples:

```
sign(-2.5) = -1.0
sign(2.5) = 1.0
sign(0) = 0.0
sign(null) = null
```

1.6.6.1.23.10. E

This topic describes the E function in mathematical functions and provides examples.

Syntax:

```
double e()
```

Description: This function returns the value of e.

Return value: A value of the DOUBLE type is returned.

Examples:

```
e() = 2.718281828459045
```

1.6.6.1.23.11. PI

This topic describes the PI function in mathematical functions and provides examples.

Syntax:

```
double pi()
```

Description: This function calculates the value of π.

Return value: A value of the DOUBLE type is returned.

Examples:

```
pi() = 3.141592653589793
```

1.6.6.1.23.12. FACTORIAL

This topic describes the FACTORIAL function in mathematical functions and provides examples.

Syntax:

```
bigint factorial(int number)
```

Description: This function calculates the factorial of number.

Parameters: number indicates a value of the INT type. The value ranges from 0 to 20.

Return value: A value of the BIGINT type is returned. If the input value is 0, 1 is returned. If the input value is NULL or a value that does not fall into the range from 0 to 20, NULL is returned.

Examples:

```
factorial(5) = 120 --5! = 5*4*3*2*1 = 120
```

1.6.6.1.23.13. CBRT

This topic describes the CBRT function in mathematical functions and provides examples.

Syntax:

```
double cbrt(double number)
```

Description: This function calculates the cube root of number.

Parameters: The number parameter supports a value of the DOUBLE type.

Return value: A value of the DOUBLE type is returned. If the input value is NULL, NULL is returned.

Examples:

```
cbrt(8) = 2
cbrt(null) = null
```

1.6.6.1.23.14. SHIFTLEFT

This topic describes the SHIFTLEFT function in mathematical functions and provides examples.

Syntax:

```
int shiftleft(tinyint|smallint|int number1, int number2)
bigint shiftleft(bigint number1, int number2)
```

Description: This function shifts a value left by a specific number of places (<<).

Parameters:

- number1: an integer of the TINYINT, SMALLINT, INT, or BIGINT type.
- number2: an integer of the INT type.

Return value: A value of the INT or BIGINT type is returned.

Examples:

```
shiftleft(1,2)=4
-- Shift the binary value of 1 two places to the left (1<<2, 0001 shifted to 0100).
shiftleft(4,3)=32
-- Shift the binary value of 4 three places to the left (4<<3, 0100 shifted to 100000).
```

1.6.6.1.23.15. SHIFTRIGHT

This topic describes the SHIFTRIGHT function in mathematical functions and provides examples.

Syntax:

```
int shiftright(tinyint|smallint|int number1, int number2)
bigint shiftright(bigint number1, int number2)
```

Description: This function shifts a value right by a specific number of places (>>).

Parameters:

- number1: an integer of the TINYINT, SMALLINT, INT, or BIGINT type.
- number2: an integer of the INT type.

Return value: A value of the INT or BIGINT type is returned.

Examples:

```
shiftright(4,2) = 1
-- Shift the binary value of 4 two places to the right (4>>2, 0100 shifted to 0001).
shiftright(32,3) = 4
-- Shift the binary value of 32 three places to the right (32>>3, 100000 shifted to 0100).
```

1.6.6.1.23.16. SHIFTRIGHTUNSIGNED

This topic describes the SHIFTRIGHTUNSIGNED function in mathematical functions and provides examples.

Syntax:

```
int shiftrightunsigned(tinyint|smallint|int number1, int number2)
bigint shiftrightunsigned(bigint number1, int number2)
```

Description: This function shifts an unsigned value right by a specific number of places (>>>).

Parameters:

- number1: an integer of the TINYINT, SMALLINT, INT, or BIGINT type.
- number2: an integer of the INT type.

Return value: A value of the INT or BIGINT type is returned.

Examples:

```
shiftrightunsigned(8,2) = 2
-- Shift the binary unsigned value of 8 two places to the right (8>>>2,1000 shifted to be 0010).
shiftrightunsigned(-14,2) = 1073741820
-- Shift the binary value of -14 two places to the right (-14>>>2, 11111111 11111111 11111111 111100
10 shifted to be 00111111 11111111 11111111 11111100).
```

1.6.6.1.23.17. FORMAT_NUMBER

This topic describes the FORMAT_NUMBER function in mathematical functions and provides examples.

Syntax:

```
string format_number(float|double|decimal expr1, expr2)
```

Description: This function converts a number to a string in the specified format.

Parameters:

- expr1: a numeric expression that you want to format.
- expr2: the number of decimal places. It can be of the INT type. It can also be expressed in the format of #,###,###.##.

  > **Note**
  > - If expr2 is greater than 0, the value is rounded to the specified place after the decimal point.
  > - If expr2 is equal to 0, the value has no decimal point or fractional part.
  > - If expr2 is less than 0 or greater than 340, an error is returned.

Return value: A value of the STRING type is returned.

Examples:

```
select format_number(5.230134523424545456,3);
-- The value 5.230 is returned.
select format_number(12332.123456, '#,###,###,###. ###');
-- The value 12,332.123 is returned.
```

1.6.6.1.23.18. WIDTH_BUCKET

This topic describes the WIDTH_BUCKET function in mathematical functions and provides examples.

Syntax:

```
WIDTH_BUCKET(NUMERIC expr, NUMERIC min_value, NUMERIC max_value, INT num_buckets)
```

Description: This function specifies the number of buckets and the minimum and maximum values of the acceptable range for a bucket. It allows you to construct equi-width buckets, in which the bucket range is divided into intervals that have an identical size. It returns the ID of the bucket into which the value of a specific expression falls. This function supports the following data types: DECIMAL(precision,scale) in the MaxCompute V2.0 data type edition, BIGINT, INT, FLOAT, DOUBLE, and DECIMAL.

Parameters:

- expr: the expression for which you want to identify the matching bucket ID.
- min_value: the minimum value of the acceptable range for the bucket.
- max_value: the maximum value of the acceptable range for the bucket. The value must be greater than min_value.
- num_buckets: the number of buckets. The value must be greater than 0.

Return value: A value of the BIGINT type is returned. The value ranges from 0 to num_buckets plus 1. If the value of expr is less than min_value, 0 is returned. If the value of expr is greater than max_value, the value of num_buckets plus 1 is returned. If the value of expr is NULL, NULL is returned. In other cases, the ID of the bucket into which the value falls is returned. The bucket ID is named based on the following formula: `Bucket ID = FLOOR[num_buckets × (expr - min_value)/(max_value - min_value) + 1]`.

Examples:

```
SELECT key,value,WIDTH_BUCKET(value,100,500,5) as value_group
FROM VALUES
    (1,99),
    (2,100),
    (3,199),
    (4,200),
    (5,499),
    (6,500),
    (7,501),
    (8,NULL)
AS t(key,value);
```

The following result is returned:

```
+-------+--------+-------------+
| key   | value  | value_group |
+-------+--------+-------------+
| 1     | 99     | 0           |
| 2     | 100    | 1           |
| 3     | 199    | 2           |
| 4     | 200    | 2           |
| 5     | 499    | 5           |
| 6     | 500    | 6           |
| 7     | 501    | 6           |
| 8     | \N     | \N          |
+-------+--------+-------------+
```

1.6.6.2. String functions

1.6.6.2.1. CHAR_MATCHCOUNT

This topic describes the CHAR_MATCHCOUNT function in string functions and provides examples.

Syntax:

```
bigint char_matchcount(string str1, string str2)
```

Description: This function returns the number of characters that belong to str1 and appear in str2.

Parameters: The str1 and str2 parameters support a value of the STRING type. The values must be valid UTF-8 strings. If invalid characters are found during the comparison of the two strings, a negative value is returned.

Return value: A value of the BIGINT type is returned. If an input value is NULL, NULL is returned.

Examples:

```
char_matchcount('abd', 'aabc') = 2
-- The characters a and b of str1 appear in str2.
```

1.6.6.2.2. CHR

This topic describes the CHR function in string functions.

Syntax:

```
string chr(bigint ascii)
```

Description: This function converts specified an ASCII code into the required character.

Parameters: ascii indicates the ASCII code in the BIGINT type. If the input value is of the STRING, DOUBLE, or DECIMAL type, it is implicitly converted into a value of the BIGINT type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the STRING type is returned. The input value ranges from 0 to 255. If the input value does not fall into this range, an error is returned. If an input value is NULL, NULL is returned.

1.6.6.2.3. CONCAT

This topic describes the CONCAT function in string functions and provides examples.

Syntax:

```
string concat(string a, string b...)
```

Description: This function concatenates all specified strings.

Parameters: a and b, values of the STRING type. If an input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, it is implicitly converted into a value of the STRING type before calculation. If an input value is of another data type, an error is returned.

Return value: A value of the STRING type is returned. If the parameters are not specified or an input value is NULL, NULL is returned.

Examples:

```
concat('ab', 'c') = 'abc'
concat() = null
concat('a', null, 'b') = null
```

1.6.6.2.4. INSTR

This topic describes the INSTR function in string functions and provides examples.

Syntax:

```
bigint instr(string str1, string str2[, bigint start_position[, bigint nth_appearance]])
```

Description: This function determines the position of substring str2 in string str1.

Parameters:

- str1: a value of the STRING type, which indicates the string in which you search for the substring. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, it is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type, an error is returned.
- str2: a value of the STRING type, which indicates the substring you want to search for. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, it is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type, an error is returned.
- start_position: a value of the BIGINT type. If the input value is of another data type, an error is returned. It indicates the character in str1 from which the search starts. The default value is 1, which indicates that the search starts from the first character.
- nth_appearance: a value of the BIGINT type. If the value is greater than 0, it indicates the position where the substring matches the string for the nth_appearance time. If the value is of another data type or less than or equal to 0, an error is returned.

Return value: A value of the BIGINT type is returned. If str2 is not matched in str1, 0 is returned. If an input value is NULL, NULL is returned. If str2 is an empty string, the substring always matches the string.

Examples:

```
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
```

1.6.6.2.5. IS_ENCODING

This topic describes the IS_ENCODING function in string functions and provides examples.

Syntax:

```
boolean is_encoding(string str, string from_encoding, string to_encoding)
```

Description: This function determines whether the input string str can be converted from the character set specified by from_encoding to the character set specified by to_encoding. It can be used to determine whether the input string is garbled. Generally, from_encoding is set to utf-8, and to_encoding is set to gbk.

Parameters:

- str: a value of the STRING type. An empty string can belong to any character set.
- from_encoding and to_encoding: a value of the STRING type. The two parameters specify the source and destination character sets. If an input value is NULL, NULL is returned.

Return value: A value of the BOOLEAN type is returned. If str can be converted, true is returned. Otherwise, false is returned.

Examples:

```
is_encoding('test', 'utf-8', 'gbk') = true
is_encoding('test', 'utf-8', 'gb2312') = true
```

1.6.6.2.6. KEYVALUE

This topic describes the KEYVALUE function in string functions and provides examples for reference only.

Syntax:

```
KEYVALUE(STRING srcStr,STRING split1,STRING split2, STRING key)
KEYVALUE(STRING srcStr,STRING key) //split1 = ";", split2 = ":"
```

Description: This function splits the source string srcStr into key-value pairs by split1, separates key-value pairs by split2, and then returns the value of the specified key.

Parameters:

- srcStr: the source string that you want to split.
- key: a value of the STRING type. After you split the source string by split1 and split2, the value of the specified key is returned.
- split1 and split2: the delimiters that are used to split strings. You must split the source string by using the specified delimiters. If you do not specify these two parameters, the default value of split1 is a semicolon (;) and that of split2 is a colon (:). If multiple key-value pairs are obtained after the source string is split by split1 and multiple delimiters specified by split2 are used to split the pairs, the returned result is undefined.

Return value: A value of the STRING type is returned. If the value of split1 or split2 is NULL, NULL is returned. If the value of srcStr or key is NULL, or key is not matched, NULL is returned. If multiple key-value pairs match the key, the value that corresponds to the first matched key is returned.

The following examples are provided:

Example 1:

```
KEYVALUE('0:1\;1:2', 1) = '2'
```

The source string is "0:1\;1:2". split1 and split2 use their default values. The default value of split1 is a semicolon (;) and that of split2 is a colon (:). After the source string is split by split1, the key-value pairs of 0:1\,1:2 are generated. After the key-value pairs are split by split2, the key-value pairs change to the following form:

```
0 1/
1 2
-- The value 2 that corresponds to key 1 is returned.
```

Example 2:

```
KEYVALUE("\;decreaseStore:1\;xcard:1\;isB2C:1\;tf:21910\;cart:1\;shipping:2\;pf:0\;market:shoes\;ins
tPayAmount:0\;","\;",":","tf") = "21910" value:21910
```

The source string is
“\;decreaseStore:1\;xcard:1\;isB2C:1\;tf:21910\;cart:1\;shipping:2\;pf:0\;market:shoes\;instPayAmount:0\;". After the source string is split by \; specified by split1, the following key-value pairs are generated:

```
decreaseStore:1, xcard:1, isB2C:1, tf:21910, cart:1, shipping:2, pf:0, market:shoes, instPayAmount:0
```

After the key-value pairs are split by a colon (:) specified by split2, the key-value pairs change to the following form:

```
decreaseStore 1
xcard 1
isB2C 1
tf 21910
cart 1
shipping 2
pf 0
market shoes
instPayAmount 0
-- The value 21910 that corresponds to key tf is returned.
```

1.6.6.2.7. LENGTH

This topic describes the LENGTH function in string functions and provides examples.

Syntax:

```
bigint length(string str)
```

Description: This function returns the length of the string str.

Parameters: The str parameter supports a value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, it is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the BIGINT type is returned. If the input value is NULL, NULL is returned. If the input value is not encoded in UTF-8, -1 is returned.

Examples:

```
length('china') = 5
```

1.6.6.2.8. LENGTHB

This topic describes the LENGTHB function in string functions and provides examples.

Syntax:

```
bigint lengthb(string str)
```

Description: This function calculates the length of str in bytes.

Parameters: str, a value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, it is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the BIGINT type is returned. If the input value is NULL, NULL is returned.

Examples:

```
lengthb('hi! China') = 10
```

1.6.6.2.9. MD5

This topic describes the MD5 function in string functions.

Syntax:

```
string md5(string value)
```

Description: This function calculates the MD5 value of value.

Parameters: value, a value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, it is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the STRING type is returned. If the input value is NULL, NULL is returned.

1.6.6.2.10. PARSE_URL

This topic describes the PARSE_URL function in string functions and provides examples.

Syntax:

```
string parse_url(string url, string part[,string key])
```

Description: This function parses url and extracts information by key.

Parameters:

- url: the URL to parse. If url is NULL, NULL is returned. If url is an invalid URL, an error is returned.
- part: a value of the STRING type. Valid values: HOST, PATH, QUERY, REF, PROTOCOL, AUTHORITY, FILE, and USERINFO. The input value is not case-sensitive. If the input value is invalid, an error is returned.
- key: If part is QUERY, the value of key is exacted from the query string in URL and is returned. Otherwise, key is omitted.

Return value: A value of the STRING type is returned.

Examples:

```
url = file://username:password@example.com:8042/over/there/index.dtb? type=animal&name=narwhal#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"
```

1.6.6.2.11. REGEXP_EXTRACT

This topic describes the REGEXP_EXTRACT function in string functions and provides examples.

Syntax:

```
string regexp_extract(string source, string pattern[, bigint occurrence])
```

Description: This function splits source based on the regular expression specified by pattern and returns the characters in group at the nth occurrence, where n is specified by occurrence.

Parameters:

- source: a value of the STRING type. This parameter indicates the string that you want to split.
- pattern: a constant of the STRING type. If pattern is an empty string or group is not specified in pattern, an

error is returned.

- occurrence: a constant of the BIGINT type, which must be greater than or equal to 0. If the input value is of another data type or is smaller than 0, an error is returned. If you do not specify this parameter, the default value 1 is used. The value 1 indicates that characters in the first group are returned. If you set occurrence to 0, all substrings that match the regular expression specified by pattern are returned.

> **Note** Data is encoded by using UTF-8.

Return value: A value of the STRING type is returned. If an input value is NULL, NULL is returned.

Examples:

```
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 on the MaxCompute client, use two backslashes (\\) as the
escape characters.
regexp_extract('foothebar', 'foothebar')
-- An error is returned because group is not specified in pattern.
```

1.6.6.2.12. REGEXP_INSTR

This topic describes the REGEXP_INSTR function in string functions and provides examples.

Syntax:

```
bigint regexp_instr(string source, string pattern[,bigint start_position[, bigint nth_occurrence[, b
igint return_option]])
```

Description: This function returns the start or end position of the substring that matches pattern at the nth occurrence, where n is specified by nth_occurrence, in the source string from the start position specified by start_position.

Parameters:

- source: a value of the STRING type. This parameter indicates the string that you want to search.
- pattern: a constant of the STRING type. If pattern is an empty string, an error is returned.
- start_position: a constant of the BIGINT type. This parameter indicates the start position for the search. If you do not specify this parameter, the default value 1 is used. If the value is of another data type or less than or equal to 0, an error is returned.
- nth_occurrence: a constant of the BIGINT type. If you do not specify this parameter, the default value 1 is used, indicating the position where a substring matches pattern in the search for the first time. If the value is of another data type or less than or equal to 0, an error is returned.
- return_option: a constant of the BIGINT type. Valid values are 0 and 1. If the value is of another data type or invalid, an error is returned. The value 0 indicates that the start position of the matched substring is returned. The value 1 indicates that the end position of the matched substring is returned.

Return value: A value of the BIGINT type is returned. This parameter indicates the start or end position of the matched substring returned in the source string based on the type specified by return_option. If an input value is NULL, NULL is returned.

Examples:

```
regexp_instr("i love www.taobao.com", "o[[:alpha:]]{1}", 3, 2) = 14
```

1.6.6.2.13. REGEXP_SUBSTR

This topic describes the REGEXP_SUBSTR function in string functions and provides examples.

Syntax:

```
string regexp_substr(string source, string pattern[, bigint start_position[, bigint nth_occurrence]]
)
```

Description: This function returns the substring that matches a given pattern at the nth occurrence, where n is specified by nth_occurrence, in the source string from the start position specified by start_position.

Parameters:

- source: a value of the STRING type. It specifies the string that contains the substring to return.
- pattern: a constant of the STRING type. It specifies the pattern used to match the substring. If the pattern value is NULL, an error is returned.
- start_position: a constant of the BIGINT type. It must be greater than 0. If it is of another data type or is less than or equal to 0, the function returns an error. If start_position is not specified, the default value 1 is used. In this case, the function starts matching from the first character of the source string.
- nth_occurrence: a constant of the BIGINT type. It must be greater than 0. If it is of another data type or is less than or equal to 0, an error is returned. If it is not specified, the default value 1 is used. In this case, the function returns the substring that first matches the specified pattern.

Return value: A value of the STRING type is returned. If an input value is NULL, NULL is returned. If no substrings match the specified pattern, NULL is returned.

Examples:

```
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"
```

1.6.6.2.14. REGEXP_COUNT

This topic describes the REGEXP_COUNT function in string functions and provides examples.

Syntax:

```
bigint regexp_count(string source, string pattern[, bigint start_position])
```

Description: This function returns the number of times a substring matches a given pattern in the source string from the start position specified by start_position.

Parameters:

- source: a value of the STRING type. It specifies the string that contains the substring to match. If the input value is of another data type, an error is returned.
- pattern: a constant of the STRING type. It specifies the pattern used to match the substring. If the pattern value is an empty string or is of another data type, an error is returned.
- start_position: a constant of the BIGINT type. It must be greater than 0. If it is of another data type or is less than or equal to 0, the function returns an error. If start_position is not specified, the default value 1 is used. In this case, the function starts matching from the first character of the source string.

Return value: A value of the BIGINT type is returned. If an input value is NULL, NULL is returned. If no substrings match the specified pattern, 0 is returned.

Examples:

```
regexp_count('abababc', 'a.c') = 1
regexp_count('abcde', '[[:alpha:]]{2}', 3) = 1
```

1.6.6.2.15. REGEXP_REPLACE

This topic describes the REGEXP_REPLACE function in string functions and provides examples.

Syntax:

```
string regexp_replace(string source, string pattern, string replace_string[, bigint occurrence])
```

Description: This function substitutes the string specified by replace_string for the substring that matches a given pattern at the nth occurrence, where n is specified by occurrence, in the source string, and returns the result.

Parameters:

- source: a value of the STRING type. It specifies the string that contains the substring to substitute.
- pattern: a constant of the STRING type. It specifies the pattern used to match the substring. If the value of pattern is NULL, an error is returned.
- replace_string: a value of the STRING type. It specifies the string to substitute for the substring that matches pattern.
- occurrence: a constant of the BIGINT type. It specifies the number of times at which the substring matches the pattern and is substituted with replace_string. It must be greater than or equal to 0. If the input value is 0, all matched substrings are substituted. If the input value is less than 0 or is not of the BIGINT type, an error is returned. Default value: 0.

> **Note** The action of referencing a character class that does not exist is undefined.

Return value: A value of the STRING type is returned. If the function references a character class that does not exist, substitution is not performed. If replace_string is NULL and substrings match pattern, NULL is returned. If replace_string is NULL and no substrings match pattern, the source string is returned. If an input value is NULL, NULL is returned.

Examples:

```
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 one character class is defined in pattern. The second referenced character class does not ex
ist.
-- Try to avoid this action. The result of referencing a non-existent group is undefined.
regexp_replace("abcd", "(. *)(.)$", "\\2", 0) = "d"
regexp_replace("abcd", "a", "\\1", 0) = "bcd"
-- No character class is defined in pattern. \1 references a non-existent character class.
-- Try to avoid this action. The result of referencing a non-existent group is undefined.
```

1.6.6.2.16. SPLIT_PART

This topic describes the SPLIT_PART function in string functions and provides examples.

Syntax:

```
string split_part(string str, string separator, bigint start[, bigint end])
```

Description: This function uses a delimiter specified by separator to split str, and returns a substring that starts from the character specified by start and ends with the character specified by end.

Parameters:

- str: a value of the STRING type, which indicates the string that you want to split. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, it is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type, an error is returned.
- separator: a constant of the STRING type, which indicates the delimiter used to split a string. It can be a character or string. If it is of another data type, an error is returned.
- start: a constant of the BIGINT type, which must be greater than 0. If the value is not a constant or is of another data type, an error is returned. It indicates the start number of the segment to be returned. The number starts from 1. 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 the value is not a constant or is of another data type, an error is returned. If this parameter is not specified, the last segment is returned.

Return value: A value of the STRING type is returned.

- If you set start to a value greater than the number of segments, an empty string is returned.
- If separator is absent in str and start is set to 1, the entire str is returned.
- If str is an empty string, an empty string is returned.
- If separator is an empty string, str is returned.
- If you set end to a value greater than the number of segments, all segments are returned.
- If an input value is NULL, NULL is returned.

Examples:

```
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) = ''
```

1.6.6.2.17. SUBSTR

This topic describes the SUBSTR function in string functions and provides examples.

Syntax:

```
string substr(string str, bigint start_position[, bigint length])
```

Description: This function returns a substring that starts from start_position in str and has a length specified by length.

Parameters:

- str: a value of the STRING type. If the input value is of the BIGINT, DECIMAL, DOUBLE, or DATETIME type, it is implicitly converted to a value of the STRING type before calculation. If the input value is of another data type, an error is returned.
- start_position: a value of the BIGINT type. The function starts counting from the first character. If start_position is a negative value, the start position is counted backward from the last character of the string. The value -1 indicates the last character. If the input value is of another data type, an error is returned.
- length: a value of the BIGINT type. It specifies the length of the substring. The input value must be greater than 0. If the input value is of another data type or is less than or equal to 0, an error is returned.

Return value: A value of the STRING type is returned. If an input value is NULL, NULL is returned.

> **Note** If length is not specified, this function returns a substring that starts from start_position in str to the end of str.

Examples:

```
substr("abc", 2) = "bc"
substr("abc", 2, 1) = "b"
substr("abc",-2,2) = "bc"
substr("abc",-3) = "abc"
```

1.6.6.2.18. TOLOWER

This topic describes the TOLOWER function in string functions and provides examples.

Syntax:

```
string tolower(string source)
```

Description: This function converts the string source to lowercase letters.

Parameters: The source parameter supports a value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, it is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the STRING type is returned. If the input value is NULL, NULL is returned.

Examples:

```
tolower("aBcd") = "abcd"
tolower("HAHACd") = "hahacd"
```

1.6.6.2.19. TOUPPER

This topic describes the TOUPPER function in string functions and provides examples.

Syntax:

```
string toupper(string source)
```

Description: This function converts source into uppercase letters.

Parameters: source, a value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, it is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the STRING type is returned. If the input value is NULL, NULL is returned.

Examples:

```
toupper("aBcd") = "ABCD"
toupper("HahaCd") = "HAHACD"
```

1.6.6.2.20. TO_CHAR

This topic describes the TO_CHAR function in string functions and provides examples.

Syntax:

```
string to_char(boolean value)
string to_char(bigint value)
string to_char(double value)
string to_char(decimal value)
```

Description: This function converts data of the BOOLEAN, BIGINT, DECIMAL, or DOUBLE type into the STRING type.

Parameters: value, a value of the BOOLEAN, BIGINT, DECIMAL, or DOUBLE type. If the input value is of another data type, an error is returned.

Return value: A value of the STRING type is returned. If the input value is NULL, NULL is returned.

Examples:

```
to_char(123) = '123'
to_char(true) = 'TRUE'
to_char(1.23) = '1.23'
to_char(null) = 'null'
```

1.6.6.2.21. TRIM

This topic describes the TRIM function in string functions.

Syntax:

```
string trim(string str)
```

Description: This function eliminates the spaces on the left and right sides of the string str.

Parameters: The str parameter supports a value of the STRING type. If the input value is of the BIGINT, DECIMAL, DOUBLE, or DATETIME type, it is implicitly converted to a value of the STRING type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the STRING type is returned. If the input value is NULL, an error is returned.

1.6.6.2.22. LTRIM

This topic describes the LTRIM function in string functions and provides examples.

Syntax:

```
string ltrim(string str)
```

Description: This function removes the spaces on the left side of str.

Parameters: str, a value of the STRING type. If the input value is of the BIGINT, DECIMAL, DOUBLE, or DATETIME type, it is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the STRING type is returned. If the input value is NULL, NULL is returned.

Examples:

```
select ltrim(' abc ');
-- The following result is returned:
+-----+
| _c0 |
+-----+
| abc  |
+-----+
```

1.6.6.2.23. RTRIM

This topic describes the RTRIM function in string functions and provides examples.

Syntax:

```
string rtrim(string str)
```

Description: This function eliminates the spaces on the right side of the string str.

Parameters: The str parameter supports a value of the STRING type. If the input value is of the BIGINT, DECIMAL, DOUBLE, or DATETIME type, it is implicitly converted to a value of the STRING type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the STRING type is returned. If the input value is NULL, NULL is returned.

Examples:

```
select rtrim('a abc ');
-- The following result is returned:
+-----+
| _c0 |
+-----+
| a abc |
+-----+
```

1.6.6.2.24. REVERSE

This topic describes the REVERSE function in string functions and provides examples.

Syntax:

```
string reverse(string str)
```

Description: This function returns a string in reverse order.

Parameters: The str parameter supports a value of the STRING type. If the input value is of the BIGINT, DECIMAL, DOUBLE, or DATETIME type, it is implicitly converted to a value of the STRING type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the STRING type is returned. If the input value is NULL, an error is returned.

Examples:

```
select reverse('abcedfg');
-- The following result is returned:
+-----+
| _c0 |
+-----+
| gfdecba |
+-----+
```

1.6.6.2.25. SPACE

This topic describes the SPACE function in string functions and provides examples.

Syntax:

```
string space(bigint n)
```

Description: This function returns a string of a length specified by n.

Parameters: The n parameter supports a value of the BIGINT type. The value cannot exceed 2 MB. If this parameter is left empty, an error is returned.

Return value: A value of the STRING type is returned.

Examples:

```
select length(space(10));
-- The value 10 is returned.
select space(400000000000);
-- An error is returned because the length exceeds 2 MB.
```

1.6.6.2.26. REPEAT

This topic describes the REPEAT function in string functions and provides examples.

Syntax:

```
string repeat(string str, bigint n)
```

Description: This function returns n duplicates of the string str.

Parameters:

- str: a value of the STRING type. If the input value is of the BIGINT, DECIMAL, DOUBLE, or DATETIME type, it is implicitly converted to a value of the STRING type before calculation. If the input value is of another data type, an error is returned.
- n: a value of the BIGINT type. The returned string cannot exceed 2 MB in length. If this parameter is left empty, an error is returned.

Return value: A value of the STRING type is returned.

Examples:

```
select repeat('abc',5);
-- abcabcabcabcabc is returned.
```

1.6.6.2.27. ASCII

This topic describes the ASCII function in string functions and provides examples.

Syntax:

```
bigint ascii(string str)
```

Description: This function returns the ASCII code of the first character in the string str.

Parameters: The str parameter supports a value of the STRING type. If the input value is of the BIGINT, DECIMAL, DOUBLE, or DATETIME type, it is implicitly converted to a value of the STRING type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the BIGINT type is returned.

Examples:

```
select ascii('abcde');
-- The value 97 is returned.
```

1.6.6.2.28. Additional string functions

1.6.6.2.28.1. Usage notes

This topic provides notes for you to use additional string functions.

MaxCompute V2.0 provides additional string functions. You must add the following flag before the SQL statement that contains the additional string functions:

```
set odps.sql.type.system.odps2=true;
```

> **Note** You must submit the flag along with the SQL statement that contains the additional string functions for execution.

1.6.6.2.28.2. CONCAT_WS

This topic describes the CONCAT_WS function in string functions and provides examples.

Syntax:

```
string concat_ws(string SEP, string a, string b...)
string concat_ws(string SEP, array)
```

Description: This function concatenates all input strings in an array by using a specified delimiter.

Parameters:

- SEP: the delimiter of the STRING type. If this parameter is not specified, an error is returned.
- a, b...: a value of the STRING type. If the input value is of the BIGINT, DECIMAL, DOUBLE, or DATETIME type, it is implicitly converted to a value of the STRING type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the STRING type is returned. If no input values are specified or an input value is NULL, NULL is returned.

Examples:

```
concat_ws(':','name','hanmeimei')='name:hanmeimei'
concat_ws(':','avg',null,'34')=null
```

1.6.6.2.28.3. LPAD

This topic describes the LPAD function in string functions and provides examples.

Syntax:

```
string lpad(string a, int len, string b)
```

Description: This function pads the left side of a with b based on the length specified by len.

Parameters:

- len: a value of the INT type.
- a: a value of the STRING type.
- b: a value of the STRING type.

Return value: A value of the STRING type is returned. 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, no value is returned.

Examples:

```
lpad('abcdefgh',10,'12')='12abcdefgh'
lpad('abcdefgh',5,'12')='abcde'
lpad('abcdefgh',0,'12')
-- No value is returned.
```

1.6.6.2.28.4. RPAD

This topic describes the RPAD function in string functions and provides examples.

Syntax:

```
string rpad(string a, int len, string b)
```

Description: This function pads the right side of string a with string b until the new padded string has len characters.

Parameters:

- len: a value of the INT type.
- a: a value of the STRING type.
- b: a value of the STRING type.

Return value: A value of the STRING type is returned. If len is smaller than the number of characters in a, a is truncated from the left to obtain a string with len characters, and the obtained string is returned. If len is 0, NULL is returned.

Examples:

```
rpad('abcdefgh',10,'12')='abcdefgh12'
rpad('abcdefgh',5,'12')='abcde'
rpad('abcdefgh',0,'12')
-- NULL is returned.
```

1.6.6.2.28.5. REPLACE

This topic describes the REPLACE function in string functions and provides examples.

Syntax:

```
string replace(string a, string OLD, string NEW)
```

Description: This function substitutes the string NEW for the part of string a that is exactly the same as the string OLD, and returns the string a.

Parameters: All the parameters support a value of the STRING type.

Return value: A value of the STRING type is returned. If an input value is NULL, NULL is returned.

Examples:

```
replace('ababab','abab','12')='12ab'
replace('ababab','cdf','123')='ababab'
replace('123abab456ab',null,'abab')=null
```

1.6.6.2.28.6. SOUNDEX

This topic describes the SOUNDEX function in string functions and provides examples.

Syntax:

```
string soundex(string a)
```

Description: This function converts a normal string to a string of the soundex type.

Parameters: The a parameter supports a value of the STRING type.

Return value: A value of the STRING type is returned. If the input value is NULL, NULL is returned.

Examples:

```
soundex('hello') = 'H400'
```

1.6.6.2.28.7. SUBSTRING_INDEX

This topic describes the SUBSTRING_INDEX function in string functions and provides examples.

Syntax:

```
string substring_index(string a, string SEP, int count))
```

Description: This function truncates the string a to a substring from the first character to the nth delimiter, where n is specified by count. If count is a positive value, the string is truncated from left to right. Otherwise, the string is truncated from right to left.

Parameters:

- a: a value of the STRING type.
- SEP: a value of the STRING type.
- count: a value of the INT type.

Return value: A value of the STRING type is returned. If an input value is NULL, NULL is returned.

Examples:

```
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
```

1.6.6.2.28.8. TRANSLATE

This topic describes the TRANSLATE function in string functions and provides examples.

Syntax:

```
string translate(string|varchar str1, string|varchar str2, string|varchar str3)
```

Description: This function replaces the common substring of str1 and str2 with str3.

Parameters: The str parameter supports a value of the STRING or VARCHAR type. If the input value is of the BIGINT, DECIMAL, DOUBLE, or DATETIME type, it is implicitly converted to a value of the STRING type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the STRING type is returned. If an input value is NULL, NULL is returned.

Examples:

```
translate('MaxComputer','puter','pute')='MaxCompute'
translate('aaa','b','c')='aaa'
translate('MaxComputer','puter',null)=null
```

1.6.6.2.28.9. URL_ENCODE

This topic describes the URL_ENCODE function in string functions and provides examples.

Syntax:

```
string url_encode(string input[, string encoding])
```

Description: This function encodes the input string in the application/x-www-form-urlencoded MIME format and returns the encoded string.

- All letters remain unchanged.
- Periods (.), hyphens (-), asterisks (*), and underscores (_) remain unchanged.
- Spaces are converted to plus signs (+).
- Other characters are converted to byte values based on encoding. Each byte value is then represented in the format of %xy, where xy is the hexadecimal representation of the character value.

Parameters:

- input: the string to encode.
- encoding: the encoding format. GBK and UTF-8 are supported. If you do not specify this parameter, the default value UTF-8 is used.

Return value: A value of the STRING type is returned. If an input value is NULL, NULL is returned.

Examples:

```
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"
```

1.6.6.2.28.10. URL_DECODE

This topic describes the URL_DECODE function in string functions and provides examples.

Syntax:

```
string url_decode(string input[, string encoding])
```

Description: This function converts an input string from the application/x-www-form-urlencoded MIME format into a normal string. This is the inverse function of URL_ENCODE.

- All letters remain unchanged.
- Periods (.), hyphens (-), asterisks (*), and underscores (_) remain unchanged.
- Each plus sign (+) is converted into a space.
- The function first performs decoding based on the percent sign (%). The sequences that are in the %xy format are converted into byte values. Then, the function performs decoding based on the encoding format specified by the second parameter. Consecutive byte values are decoded to the required strings based on the format specified by encoding.
- Other characters remain unchanged.
- The return value of the function is a string encoded in UTF-8.

Parameters:

- input: the string that you want to decode.
- encoding: the specified encoding format. Valid values include GBK and UTF-8. If you do not specify this parameter, the default value UTF-8 is used.

Return value: A value of the STRING type is returned. If an input value is NULL, NULL is returned.

Examples:

```
URL_DECODE('%E7%A4%BA%E4%BE%8Bfor+URL_DECODE%3A%2F%2F+%28fdsf%29')= "Example for URL_DECODE:// (fdsf)"
URL_DECODE('Example+for+URL_DECODE+%3A%2F%2F+dsf%28fasfs%29', 'GBK') = "Example for URL_DECODE:// dsf(fasfs)"
```

1.6.6.2.28.11. JSON_TUPLE

This topic describes the JSON_TUPLE function in string functions and provides examples.

Syntax:

```
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. It specifies a standard JSON string.
- key: a value of the STRING type. It describes 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.

- 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 value does not exist in the JSON string, it is considered invalid.
- If the JSON parameter is valid and the key value exists in the JSON string, the required string is returned.

> **Note**
>
> - This function can parse JSON data that contains Chinese characters.
> - This function can parse nested JSON data.
> - This function can parse JSON data that contains nested arrays.
> - This function parses a JSON string in the same way as the GET_JSON_OBJECT function for which `set o dps.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, you can use JSON_TUPLE with LATERAL VIEW.

Examples:

Prepare a school table that contains the following data:

```
Table: school
+------------+------------+
| Id         | json       |
+------------+------------+
| 1          | {
                "School name": "Hupan college",
                "Location":"Hangzhou",
                "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;
-- The preceding and following statements are equivalent.
select get_json_object(school.json,"$.SchoolRank") item0,get_json_object(school.json,"$.Class1") ite
m1 from school;
```

The following result is returned:

```
+-------+-------+
| item0 | item1 |
+-------+-------+
| 00    | {"Student":[{"studentId":1,"scoreRankIn3Year":[1,2,[3,2,6]]},{"studentId":2,"scoreRankIn3Y
ear":[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 |
+-------+-------+
| Hupan college | Hangzhou |
+-------+-------+
```

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] .
scoreRankIn3Year") 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.Stude
nt[0].scoreRankIn3Year[2][1]") q as item0,item1;
```

The following result is returned:

```
+------------+-------+-------+
| id         | item0 | item1 |
+------------+-------+-------+
| 1          | [3,2,6] | 2     |
+------------+-------+-------+
```

1.6.6.2.28.12. FROM_JSON

This topic describes the FROM_JSON function in string functions and provides examples.

Syntax:

```
FROM_JSON(jsonStr, schema)
```

Description: This function processes a JSON string and returns a value of the ARRAY, MAP, or STRUCT type with the specified schema.

Parameters:

- jsonStr: a value of the STRING type. This parameter specifies the JSON string to process.
- schema: the data structure. The syntax of schema is the same as that of table creation statements. Examples:
  - `array<bigint>`
  - `map<string, array<string>>`
  - `struct<a:int, b:double, c:map<string,string>>`

  For the STRUCT type, you can also specify a value in the `'a INT, b DOUBLE'` format, which is equivalent to `struct<a:int, b:double>`.

The following table describes the mapping relationships between JSON and MaxCompute data types.

JSON	MaxCompute
Object	STRUCT/MAP/STRING
Array	ARRAY/STRING
Number	TINYINT/SMALLINT/IBT/BIGINT/FLOAT/DOUBLE/DECIMAL/STRING
true/false	BOOLEAN/STRING
String	STRING
null	All types

Note For the OBJECT and ARRAY types, this function preferentially parses the data. For the types that are not supported by MaxCompute, this function ignores the data.

Return value: A value of the ARRAY, MAP, or STRUCT type is returned.

Note If the schema parameter is left empty, an error is returned.

Examples:

```
SELECT from_json('{"a":1, "b":0.8}', 'a INT, b DOUBLE');
-- {"a":1, "b":0.8} is returned.
SELECT from_json('{"time":"26/08/2015"}', 'time string');
-- {"time":"26/08/2015"} is returned.
SELECT from_json('{"a":1, "b":0.8}', 'a INT, b DOUBLE, c STRING');
-- {"a":1, "b":0.8, c: NULL} is returned.
SELECT from_json('[1, 2, 3, "a"]', 'array<BIGINT>');
-- [1, 2, 3] is returned.
```

1.6.6.2.28.13. TO_JSON

This topic describes the TO_JSON function in string functions and provides examples.

Syntax:

```
TO_JSON(expr)
```

Description: This function returns a JSON string for a value of a given type.

Parameters: The expr parameter supports a value of the ARRAY, MAP, or STRUCT type.

Return value: A value of the STRING type is returned.

Examples:

```
SELECT to_json(named_struct('a', 1, 'b', 2));
-- {"a":1,"b":2} is returned.
SELECT to_json(named_struct('time', "26/08/2015"));
-- {"time":"26/08/2015"} is returned.
SELECT to_json(array(named_struct('a', 1, 'b', 2)));
-- [{"a":1,"b":2}] is returned.
SELECT to_json(map('a', named_struct('b', 1)));
-- {"a":{"b":1}} is returned.
SELECT to_json(map('a', 1));
-- {"a":1} is returned.
SELECT to_json(array((map('a', 1))));
-- [{"a":1}] is returned.
```

1.6.6.3. Date functions

1.6.6.3.1. DATEADD

This topic describes the date function DATEADD and provides examples.

Syntax:

```
datetime dateadd(datetime date, bigint delta, string datepart)
```

Description: This function modifies a date value based on datepart and delta that you specified.

Parameters:

- date: a date value of the DATE, DATETIME, or TIMESTAMP type. If the input value is of the STRING type, it is implicitly converted into a value of the DATETIME type before the calculation. If the input value is of another data type, an error is returned.
- delta: a value of the BIGINT type, which indicates the interval to add to the specified part of the date value. If the input value is of the STRING or DOUBLE type, it is implicitly converted into a value of the BIGINT type before the calculation. If the input value is of another data type, an error is returned. If the value of delta is greater than 0, this function adds the interval to the date value. In other cases, this function subtracts the interval from the date value.

  > **Note**
  > - If you add or subtract the interval specified by delta at a date part, a carry or return at more significant date parts may occur. The year, month, hour, minute, and second parts are computed by using different numeral systems. The year part uses the base-10 numeral system. The month part uses the base-12 numeral system. The hour part uses the base-24 numeral system. The minute and second parts use the base-60 numeral system.
  > - If the DATEADD function adds an interval specified by delta to the month part of a date value of the DATETIME type and this operation does not cause an overflow of day, keep day unchanged. If the operation causes an overflow of day, set day to the last day of the specified month.

- datepart: the part you want to modify in the date value. The value is a constant of the STRING type. If the value is in an invalid format or is not a constant of the STRING type, an error is returned. The value of this parameter is specified in compliance with the rules of conversions between the STRING and DATETIME types.

The value yyyy indicates that the DATEADD function adds an interval to the year part of the date value. The value mm indicates that the DATEADD function adds an interval to the month part of the date value. This parameter also supports extended date formats, such as -year, -month, -mon, -day, and -hour. For more information about the rules of data type conversions, see Conversion between string and datetime types.

Return value: A value of the DATETIME type is returned. If an input value is NULL, NULL is returned.

Examples:

- Example 1: common usage

```
select dateadd(datetime '2005-02-28 00:00:00', 1, 'dd') ;
-- The return value is 2005-03-01 00:00:00. After one day is added, the result is beyond the last
day of February. The actual date value is the first day of March.
select dateadd(datetime '2005-02-28 00:00:00', -1, 'dd');
-- The return value is 2005-02-27 00:00:00. One day is subtracted.
select dateadd(datetime '2005-02-28 00:00:00', 20, 'mm');
-- The return value is 2006-10-28 00:00:00. After 20 months are added, the month overflows, and th
e year increases by 1.
select dateadd(datetime '2005-02-28 00:00:00', 1, 'mm');
-- The return value is 2005-03-28 00:00:00. One month is added.
select dateadd(datetime '2005-01-29 00:00:00', 1, 'mm');
-- The return value is 2005-02-28 00:00:00. February in 2005 has only 28 days. Therefore, the last
day of February is returned.
select dateadd(datetime '2005-03-30 00:00:00', -1, 'mm');
-- The return value is 2005-02-28 00:00:00. One month is subtracted. February in 2005 has only 28
days. Therefore, the last day of February is returned.
```

- Example 2: usage of DATEADD in which a value of the DATETIME type is expressed as a constant

```
select dateadd(2005-03-30 00:00:00, -1, 'mm');
-- In MaxCompute SQL statements, a value of the DATETIME type cannot be directly expressed as a co
nstant. This statement uses an invalid expression for a value of the DATETIME type.
```

```
select dateadd(cast("2005-03-30 00:00:00" as datetime), -1, 'mm');
-- This statement uses the DATEADD function in which a constant of the STRING type is explicitly c
onverted to a value of the DATETIME type.
```

1.6.6.3.2. DATEDIFF

This topic describes the date function DATEDIFF and provides examples.

Syntax:

```
bigint datediff(datetime date1, datetime date2, string datepart)
```

Description: This function calculates the difference between date1 and date2. The difference is measured in the time unit specified by datepart.

Parameters:

- date1 and date2: the minuend and subtrahend. The values are of the DATE, DATETIME, or TIMESTAMP type. If an input value is of the STRING type, it is implicitly converted into a value of the DATETIME type before the calculation. If the input value is of another data type, an error is returned.
- datepart: the time unit, which is a constant of the STRING type. This parameter supports extended date formats. If datepart is not in the specified format or is of another data type, an error is returned.

Return value: A value of the BIGINT type is returned. If an input value is NULL, NULL is returned. If date1 is earlier than date2, a negative value is returned.

> **Note**
>
> - If the difference between two dates is more precise than the unit specified by datepart, the excessive parts are discarded in the return value. For example, if the unit specified by datepart is day, the hour, minute, and second parts are discarded in the return value.
> - This function omits the parts with smaller units based on the unit specified by datepart and calculates the result. The parts with smaller units refer to the excessive parts previously described.

Examples:

```
-- The start time is 2005-12-31 23:59:59 and the end time is 2006-01-01 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(datetime'2013-05-31 13:00:00', '2013-05-31 12:30:00', 'ss') = 1800
    datediff(datetime'2013-05-31 13:00:00', '2013-05-31 12:30:00', 'mi') = 30
-- The start time is 2018-06-04 19:33:23.234 and the end time is 2018-06-04 19:33:23.250. Date value
s with milliseconds do not adopt the standard DATETIME type and therefore cannot be implicitly conve
rted into the DATETIME type. In this case, an explicit conversion is required.
    datediff(to_date('2018-06-04 19:33:23.250', 'yyyy-MM-dd hh:mi:ss.ff3'),to_date('2018-06-04 19:33
:23.234', 'yyyy-MM-dd hh:mi:ss.ff3') , 'ff3') = 16
```

1.6.6.3.3. DATEPART

This topic describes the date function DATEPART and provides examples.

Syntax:

```
bigint datepart(datetime date, string datepart)
```

Description: This function returns a specific part of a date value. The part is specified by datepart.

Parameters:

- date: a date value of the DATE, DATETIME, or TIMESTAMP type. If the input value is of the STRING type, it is implicitly converted into a value of the DATETIME type before the calculation. If the input value is of another data type, an error is returned.
- datepart: a constant of the STRING type. This parameter supports extended date formats. If datepart is not in the specified format or is of another data type, an error is returned.

Return value: A value of the BIGINT type is returned. If an input value is NULL, NULL is returned.

Examples:

```
datepart('2017-06-08 01:10:00', 'yyyy') = 2017
datepart('2017-06-08 01:10:00', 'mm') = 6
```

1.6.6.3.4. DATETRUNC

This topic describes the date function DATETRUNC and provides examples.

Syntax:

```
datetime datetrunc (datetime date,string datepart)
```

Description: This function truncates a date value to the accuracy specified by datepart and returns a new date.

Parameters:

- date: a date value of the DATE, DATETIME, or TIMESTAMP type. If the input value is of the STRING type, it is implicitly converted into a value of the DATETIME type before the calculation. If the input value is of another data type, an error is returned.
- datepart: a constant of the STRING type. This parameter supports extended date formats. If datepart is not in the specified format or is of another data type, an error is returned.

Return value: A value of the DATETIME type is returned. If an input value is NULL, NULL is returned.

Examples:

```
datetrunc('2017-12-07 16:28:46', 'yyyy') = 2017-01-01 00:00:00
-- The function truncates the date value to the accuracy of the year 2017 and returns the date value
that is accurate to the year. In this example, the date value that is accurate to the year 2017 is r
eturned.
datetrunc('2017-12-07 16:28:46', 'month') = 2017-12-01 00:00:00
-- The function truncates the date value to the accuracy of the month December and returns the date
value that is accurate to the month. In this example, the date value that is accurate to the month "
2017-12" is returned.
datetrunc('2017-12-07 16:28:46', 'DD') = 2017-12-07 00:00:00
-- The function truncates the date value to the accuracy of the day 07 and returns the date value th
at is accurate to the day. In this example, the date value that is accurate to the day "2017-12-07"
is returned.
```

1.6.6.3.5. GETDATE

This topic describes the GETDATE function in date functions.

Syntax:

```
datetime getdate()
```

Description: This function returns the current system time as a date value. MaxCompute uses UTC+8 as the standard time zone.

Return value: A value of the DATETIME type is returned. The value indicates the current date and time.

> **Note** In a MaxCompute SQL task that is executed in distributed mode, the GETDATE function always returns a fixed value. The return value is an arbitrary time during the execution of the MaxCompute SQL task. The time is accurate to seconds. In MaxCompute V2.0 that supports more data types, the time is accurate to milliseconds.

1.6.6.3.6. ISDATE

This topic describes the ISDATE function in date functions.

Syntax:

```
boolean isdate(string date, string  format)
```

Description: This function determines whether a date string can be converted into a date value in a specified format. If the date string can be converted into a date value in the specified format, True is returned. Otherwise, False is returned.

Parameters:

- date: a value of the STRING type. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, it is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type, an error is returned.
- format: a constant of the STRING type. This parameter does not support extended date formats. If the input value is of another data type, an error is returned. If redundant format strings exist in format, this function converts the date string that corresponds to the first format string into a date value. The rest strings are considered delimiters. For example, isdate("1234-yyyy", "yyyy-yyyy") returns True.

Return value: A value of the BOOLEAN type is returned. If an input value is NULL, NULL is returned.

1.6.6.3.7. LASTDAY

This topic describes the LASTDAY function in date functions.

Syntax:

```
datetime lastday(datetime date)
```

Description: This function returns the last day of the month in which the date value falls. The value is accurate to days. The hour, minute, and second parts are expressed as 00:00:00.

Parameters: The date parameter supports a value of the DATETIME type. If the input value is of the STRING type, it is implicitly converted into a value of the DATETIME type before calculation. If the input value is of another data type, an error is returned.

Return value: A value of the DATETIME type is returned. If the input value is NULL, NULL is returned.

1.6.6.3.8. TO_DATE

This topic describes the TO_DATE function and provides examples of using this function.

Syntax:

```
datetime to_date(string date, string format)
```

Description: This function converts a date string in a specified format into a date value.

Parameters:

- date: a date value of the STRING type, which indicates the date string you want to convert. If the input value is of the BIGINT, DOUBLE, DECIMAL, or DATETIME type, it is implicitly converted into a value of the STRING type before calculation. If the input value is of another data type or an empty string, an error is returned.
- format: a constant of the STRING type, which indicates a date format. If the input value is not a constant or is not of the STRING type, an error is returned. The Extended Date/Time Format (EDTF) is not supported. In this format, the characters are parsed as invalid characters and omitted. The value of this parameter must contain yyyy. Otherwise, an error is returned. If redundant format strings are included in the value of this parameter, only the date value that corresponds to the first format string is used, and the date values that correspond to the other format strings are parsed as delimiters. For example, to_date('1234-2234 ', 'yyyy-yyyy ') is parsed as 1234-01-01 00:00:00.

Return value: A value of the DATETIME type is returned and the value is in the format of yyyy-mm-dd hh:mi:ss:ff3. If the value of any input parameter is NULL, NULL is returned.

> **Note** In the format, yyyy indicates a 4-digit year, mm indicates a 2-digit month, dd indicates a 2-digit day, hh indicates an hour in 24-hour display, mi indicates a 2-digit minute, ss indicates a 2-digit second, and ff3 indicates a 3-digit millisecond.

Examples:

```
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.
to_date('20181030 15-13-12.345','yyyymmdd hh-mi-ss.ff3')=2018-10-30 15:13:12
```

1.6.6.3.9. TO_CHAR

This topic describes the TO_CHAR function and provides examples of using this function.

Syntax:

```
string to_char(datetime date, string format)
```

Description: This function converts a date value of the DATETIME type into a string in a specified format.

Parameters:

- date: a date value of the DATETIME type, which indicates the date value you want to convert. If the input value is of the STRING type, it is implicitly converted into the DATETIME type before calculation. If the input value is not of the STRING type, an error is returned.
- format: a constant of the STRING type. If it is not a constant or is not of the STRING type, an error is returned. In the format parameter, the date format part is replaced by the related data and other characters remain unchanged in the output.

Return value: A value of the STRING type is returned. If the value of an input parameter is NULL, NULL is returned.

Examples:

```
to_char('2017-12-03 00:00:00', 'Alibaba Cloud Financial Servicesyyyy-mm*dd') = 'Alibaba Cloud Financ
ial Services2017-12*03'
to_char('2017-07-18 00:00:00', 'yyyymmdd') = '20170718'
to_char('Alibaba2017-12*3', 'Alibabayyyy-mm*dd')
-- NULL is returned.
to_char('2017-24-01', 'yyyy')
-- NULL is returned.
to_char('2017718', 'yyyymmdd')
-- NULL is returned.
```

1.6.6.3.10. UNIX_TIMESTAMP

This topic describes the UNIX_TIMESTAMP function and provides examples of using this function.

Syntax:

```
bigint unix_timestamp(datetime date)
```

Description: This function converts a date value to a UNIX timestamp that is an integer.

Parameters: date: a date value of the DATETIME type. If the input value is of the STRING type, it is implicitly converted into a value of the DATETIME type before calculation. If the input value is not of the STRING type, an error is returned. If you enable new data types, the implicit conversion fails. In this case, you must use the CAST function for conversion, for example, unix_timestamp(cast(... as datetime)).

Return value: A UNIX timestamp of the BIGINT type is returned. If the value of the input parameter is NULL, NULL is returned.

Examples:

```
select unix_timestamp(datetime'2009-03-20 11:11:00');
-- 1237518660 is returned.
```

1.6.6.3.11. FROM_UNIXTIME

This topic describes the FROM_UNIXTIME function and provides examples of using this function.

Syntax:

```
datetime from_unixtime(bigint unixtime)
```

Description: This function converts unixtime of the BIGINT type to a date value of the DATETIME type.

Parameters: unixtime: a date value of the BIGINT type in the UNIX format. Its value is accurate to seconds. If the input value is of the STRING, DOUBLE, or DECIMAL type, it is implicitly converted into a value of the BIGINT type before calculation.

Return value: A value of the DATETIME type is returned. If an input value is NULL, NULL is returned.

> **Note** In the Hive-compatible mode where `set odps.sql.hive.compatible=true;` has been run, if the value of the input parameter is of the STRING type, a date value of the STRING type is returned.

Examples:

```
from_unixtime(123456789) = 1973-11-30 05:33:09;
```

1.6.6.3.12. WEEKDAY

This topic describes the WEEKDAY function.

Syntax:

```
bigint weekday (datetime date)
```

Description: This function returns the day of the week for a specified date.

Parameters: date: a date value of the DATETIME type. If the input value is of the STRING type, it is implicitly converted into a value of the DATETIME type before calculation. If the input value is not of the STRING type, an error is returned.

Return value: A value of the BIGINT type is returned. If the value of the input parameter is NULL, NULL is returned. Monday is treated as the first day of a week and its return value is 0. Days are numbered in ascending order starting from 0. The return value for Sunday is 6.

1.6.6.3.13. WEEKOFYEAR

This topic describes the WEEKOFYEAR function and provides examples of using this function.

Syntax:

```
bigint weekofyear(datetime date)
```

Description: This function returns the calendar week of the year that the specified date falls in. Monday is treated as the first day of a week.

> **Note** If a week spans two years, whether this week belongs to the previous year or the next year is based on which year contains more than four days. If more days fall in the previous year, the week is considered as the last week of the previous year. If more days fall in the next year, the week is considered as the first week of the next year.

Parameters: date: a date value of the DATETIME type. If the value of the input parameter is of the STRING type, it is implicitly converted into a value of the DATETIME type before calculation. If the value of the input parameter is not of the STRING type, an error is returned.

Return value: A value of the BIGINT type is returned. If the value of the input parameter is NULL, NULL is returned.

Examples:

```
select weekofyear(to_date("20141229", "yyyymmdd"));
-- The following result is returned:
+------------+
| _c0        |
+------------+
| 1          |
+------------+
-- 20141229 is in year 2014, but most days of the week fall in year 2015. Therefore, the return valu
e 1 indicates the first week of year 2015.
select weekofyear(to_date("20141231", "yyyymmdd"));
-- 1 is returned.
select weekofyear(to_date("20151229", "yyyymmdd"));
-- 53 is returned.
```

1.6.6.3.14. Additional date functions

1.6.6.3.14.1. Usage notes

This topic describes the configuration operation that you must perform before you use the date functions that are added to MaxCompute V2.0.

MaxCompute V2.0 provides more date functions. You must add the following SET statement before the SQL statements for these new functions:

```
set odps.sql.type.system.odps2=true;
```

> **Note** You must submit and execute the SET statement and the SQL statements for the new functions at the same time.

1.6.6.3.14.2. YEAR

This topic describes the YEAR function and provides examples of using this function.

Syntax:

```
INT YEAR(DATETIME/STRING date)
```

Description: This function returns the year in which the specified date value falls.

Parameters: date: a date value of the DATETIME or STRING type. The format of the date value must include yyyy-mm-dd and exclude redundant strings. Otherwise, NULL is returned.

Return value: A value of the INT type is returned.

Examples:

```
SELECT YEAR('1970-01-01 12:30:00');
-- 1970 is returned.
SELECT YEAR('1970-01-01');
-- 1970 is returned.
SELECT YEAR('70-01-01');
-- 70 is returned.
SELECT YEAR('1970-01-01');
-- 1970 is returned.
SELECT YEAR('1970/03/09');
-- NULL is returned.
SELECT YEAR(null);
-- An error is returned.
```

1.6.6.3.14.3. QUARTER

This topic describes the QUARTER function and provides examples of using this function.

Syntax:

```
INT QUARTER (DATETIME/TIMESTAMP/STRING date)
```

Description: This function returns the quarter of the year for a date value. The quarter is an integer from 1 to 4.

Parameters: date: a date value of the DATETIME, TIMESTAMP, or STRING type. The format of the date value must include yyyy-mm-dd. If the date value is not of the DATETIME, TIMESTAMP, or STRING type, NULL is returned.

Return value: A value of the INT type is returned. If the value of an input parameter is NULL, NULL is returned.

Examples:

```
SELECT QUARTER('1970-11-12 10:00:00');
-- 4 is returned.
SELECT QUARTER('1970-11-12');
-- 4 is returned.
```

1.6.6.3.14.4. MONTH

This topic describes the MONTH function and provides examples of using this function.

Syntax:

```
INT MONTH(DATETIME/STRING date)
```

Description: This function returns the month part of a date value.

Parameters: date: a value of the DATETIME or STRING type. The date format must include yyyy-mm-dd. If it is not of the DATETIME or STRING type, an error is returned.

Return value: A value of the INT type is returned.

Examples:

```
SELECT MONTH('2014-09-01');
-- 9 is returned.
SELECT MONTH('20140901');
-- NULL is returned.
```

1.6.6.3.14.5. DAY

This topic describes the DAY function and provides examples of using this function.

Syntax:

```
INT DAY(DATETIME/STRING date)
```

Description: This function returns the day of a specified date value.

Parameters: date: a date value of the DATETIME or STRING type. The format of the date value can be yyyy-mm-dd or yyyy-mm-dd hh:mi:ss. If the date value is not of the DATETIME or STRING type, an error is returned.

Return value: A value of the INT type is returned.

Examples:

```
SELECT DAY('2014-09-01');
-- 1 is returned.
SELECT DAY('20140901');
-- NULL is returned.
```

1.6.6.3.14.6. DAYOFMONTH

This topic describes the DAYOFMONTH function and provides examples of using this function.

Syntax:

```
int dayofmonth(date)
```

Description: This function returns the value of the day part of a date value. For example, if the date is October 13, 2019, 13 is returned after you run the `int dayofmonth(2019-10-13)` command.

Parameters: date: a date value of the STRING type. The date format must include yyyy-mm-dd. If it is not of the STRING type, an error is returned.

Return value: A value of the INT type is returned.

Examples:

```
dayofmonth('2019-09-01');
-- 1 is returned.
dayofmonth('20190901');
-- NULL is returned.
```

1.6.6.3.14.7. HOUR

This topic describes the HOUR function and provides examples of using this function.

Syntax:

```
INT HOUR(DATETIME/STRING date)
```

Description: This function returns the value of the hour part of a date.

Parameters: date: a value of the DATETIME or STRING type. If the value is not of the DATETIME or STRING type, an error is returned.

Return value: A value of the INT type is returned.

Examples:

```
SELECT HOUR('2014-09-01 12:00:00');
-- 12 is returned.
SELECT HOUR('12:00:00');
-- 12 is returned.
SELECT HOUR('20140901120000');
-- NULL is returned.
```

1.6.6.3.14.8. MINUTE

This topic describes the MINUTE function and provides examples of using this function.

Syntax:

```
INT MINUTE(DATETIME/STRING date)
```

Description: This function returns the value of the minute part of a date.

Parameters: date: a value of the DATETIME or STRING type. If the value is not of the DATETIME or STRING type, an error is returned.

Return value: A value of the INT type is returned.

Examples:

```
SELECT MINUTE('2014-09-01 12:30:00');
-- 30 is returned.
SELECT MINUTE('12:30:00');
-- 30 is returned.
SELECT MINUTE('20140901120000');
-- NULL is returned.
```

1.6.6.3.14.9. SECOND

This topic describes the SECOND function and provides examples of using this function.

Syntax:

```
INT SECOND(DATETIME/STRING date)
```

Description: This function returns the value of the second part of a date value.

Parameters: date: a date value of the DATETIME or STRING type. If it is not of the DATETIME or STRING type, an error is returned.

Return value: A value of the INT type is returned.

Examples:

```
SELECT SECOND('2014-09-01 12:30:45');
-- 45 is returned.
SELECT SECOND('12:30:45');
-- 45 is returned.
SELECT SECOND('20140901123045');
-- NULL is returned.
```

1.6.6.3.14.10. FROM_UTC_TIMESTAMP

This topic describes the FROM_UTC_TIMESTAMP function and provides examples of using this function.

Syntax:

```
timestamp from_utc_timestamp({any primitive type}*, string timezone)
```

Description: This function converts a UTC timestamp to a timestamp for a specified time zone.

Parameters:

- {any primitive type}*: the timestamp, which is a value of the TIMESTAMP, DATETIME, TINYINT, SMALLINT, INT, or BIGINT type. If the value is of the TINYINT, SMALLINT, INT, or BIGINT type, the unit is milliseconds.
- timezone: the time zone to which you want to convert the timestamp, such as Pacific Standard Time (PST).

Return value: A value of the TIMESTAMP type is returned.

Examples:

```
SELECT from_utc_timestamp(1501557840000, 'PST');
-- The unit of the input value is milliseconds and 2017-08-01 04:24:00 is returned.
SELECT from_utc_timestamp('1970-01-30 16:00:00','PST');
-- 1970-01-30 08:00:00.0 is returned.
SELECT from_utc_timestamp('1970-01-30','PST');
-- 1970-01-29 16:00:00.0 is returned.
```

1.6.6.3.14.11. CURRENT_TIMESTAMP

This topic describes the CURRENT_TIMESTAMP function and provides examples of using this function.

Syntax:

```
timestamp current_timestamp()
```

Description: This function returns the current timestamp. The return value is not fixed.

Return value: A value of the TIMESTAMP type is returned.

Examples:

```
select current_timestamp();
-- '2017-08-03 11:50:30.661' is returned.
```

1.6.6.3.14.12. ADD_MONTHS

This topic describes the ADD_MONTHS function and provides examples of using this function.

Syntax:

```
string add_months(string startdate, int nummonths)
```

Description: This function adds a specified number of months to a date specified by startdate. The number of months is specified by nummonths.

Parameters:

- startdate: a date value of the STRING type. The date format must contain yyyy-mm-dd. Otherwise, NULL is returned.
- num_months: a value of the INT type.

Return value: A value of the STRING type is returned and the value is in the format of yyyy-mm-dd.

Examples:

```
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
```

1.6.6.3.14.13. LAST_DAY

This topic describes the LAST_DAY function and provides examples of using this function.

Syntax:

```
string last_day(string date)
```

Description: This function returns the last date of the month part of a date value.

Parameters: date: a date value of the STRING type and in the format of yyyy-mm-dd hh:mi:ss or yyyy-mm-dd.

Return value: A value of the STRING type is returned and the value is in the format of yyyy-mm-dd.

Examples:

```
last_day('2017-03-04') = '2017-03-31'
last_day('2017-07-04 11:40:00') = '2017-07-31'
last_day('20170304') = null
```

1.6.6.3.14.14. NEXT_DAY

This topic describes the NEXT_DAY function and provides examples of using this function.

Syntax:

```
string next_day(string startdate, string week)
```

Description: This function returns the date of the first day that is later than startdate and matches the week value. The return value indicates the date of the specified day in the next week.

Parameters:

- startdate: a date value of the STRING type and in the format of yyyy-mm-dd hh:mi:ss or yyyy-mm-dd.
- week: a date value of the STRING type. The full name of a day in a week, or the first two or three letters of

the day, for example, MO, TUE, or FRIDAY.

Return value: A value of the STRING type is returned and the value is in the format of yyyy-mm-dd.

Examples:

```
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
```

1.6.6.3.14.15. MONTHS_BETWEEN

This topic describes the MONTHS_BETWEEN function and provides examples of using this function.

Syntax:

```
double months_between(datetime/timestamp/string date1, datetime/timestamp/string date2)
```

Description: This function returns the number of months between date1 and date2.

Parameters:

- date1: a value of the DATETIME, TIMESTAMP, or STRING type. The value is in the format of yyyy-mm-dd hh:mi:ss or yyyy-mm-dd.
- date2: a value of the DATETIME, TIMESTAMP, or STRING type. The value is in the format of yyyy-mm-dd hh:mi:ss or yyyy-mm-dd.

Return value: A value of the DOUBLE type is returned.

- If date1 is later than date2, a positive value is returned. If date2 is later than date1, a negative value is returned.
- If both date1 and date2 correspond to the last days of two months, the return value is an integer that represents the number of months. Otherwise, the return value is computed by using the following formula: date1 - date2)/31

Examples:

```
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
```

1.6.6.3.14.16. EXTRACT

This topic describes the EXTRACT function and provides examples of using this function.

Syntax:

```
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 YEAR, MONTH, DAY, HOUR, or MINUTE.
- timestamp: a date value of the TIMESTAMP type.

Return value: A value of the INT type is returned.

Examples:

```
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.

Examples:

```
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 the hour part is 24, and the specified time value is 31. The return value is
7 (31/24).
-- The maximum value of the minute part is 60, and the specified time value is 61. The return value
is 1 (61/60).
```

1.6.6.4. Window functions

1.6.6.4.1. Overview

In MaxCompute SQL, window functions can be used for flexible data analysis and processing. This topic describes the precautions for using window functions and the syntax of window functions.

Notice

- Window functions can only be included in the SELECT clause.
- A window function cannot contain nested window functions or aggregate functions.
- Window functions cannot be used with aggregate functions of the same level.
- A maximum of five window functions can be used in a MaxCompute SQL statement.

Syntax:

```
window_func() over (partition by [col1,col2…]
[order by [col1[asc|desc], col2[asc|desc]…]] windowing_clause)
```

Parameters:

- PARTITION BY: the columns in the window that is used for calculation. Rows with the same values of the partition key columns are considered in the same window. A window can contain a maximum of 100 million

rows of data. We recommend that the number of rows in a window not exceed 5 million. If the number of rows exceeds 5 million, an error is returned.

- ORDER BY: specifies how to sort data in a window.
- windowing_clause: You can use ROWS to specify how to define the window for the function. Valid values:
  - rows between x preceding|following and y preceding|following: indicates a window that ranges 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 that ranges 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 10000. 0 indicates the current row.
> - You must specify ORDER BY before you use ROWS to specify a window range.
> - Window functions that allow you to use rows to define the window include AVG, COUNT, MAX, MIN, STDDEV, and SUM.

1.6.6.4.2. COUNT

This topic describes the COUNT function and provides examples of using this function.

Syntax:

```
Bigint count([distinct] expr) over(partition by [col1, col2…]
[order by [col1[asc|desc], col2[asc|desc]…]] [windowing_clause])
```

Description: This function returns the counted values of specified rows.

Parameters:

- expr: a value of any data type. If the value for a row is NULL, this row is not used for calculation. If the distinct keyword is specified, only the distinct values are counted.
- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by [col1[asc|desc], col2[asc|desc]: If ORDER BY is not specified, the counted value of expr in the current window is returned. If ORDER BY is specified, the returned results are sorted in the specified order and the value accumulated from the starting row to the current row in the current window is returned.

> **Note** If the distinct keyword is specified, ORDER BY cannot be used.

Return value: A value of the BIGINT type is returned.

Examples:

The test_src table contains the user_id column of the BIGINT type.

```
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 counted value of the user_id column in the current window is re
turned.
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          |      -- This row is the starting row of this window.
| 1       | 2          |      -- Two rows are found from the starting row to the current row. 2 is r
eturned.
| 1       | 3          |
| 2       | 1          |
| 3       | 1          |
+---------+------------+
-- If ORDER BY is specified, the counted value from the starting row to the current row in the curre
nt window is returned.
```

If duplicate values are specified for ORDER BY, the processing method is based on the compatibility between MaxCompute and Hive.

- If MaxCompute is not compatible with Hive, the value of COUNT for each row is returned.

```
set odps.sql.hive.compatible=false;
select user_id, price, count(price) over
  (partition by user_id order by price) as count from test_src;
+------------+------------+------------+
| user_id    | price      | count      |
+------------+------------+------------+
| 1          | 4.5        | 1          |  -- This row is the starting row of this window.
| 1          | 5.5        | 2          |  -- The value of COUNT for the second row is 2.
| 1          | 5.5        | 3          |  -- The value of COUNT for the third row is 3.
| 1          | 6.5        | 4          |
| 2          | NULL       | 0          |
| 2          | 3.0        | 1          |
| 3          | NULL       | 0          |
| 3          | 4.0        | 1          |
+------------+------------+------------+
```

- If MaxCompute is compatible with Hive, the return value is the value of COUNT for the last row of the rows with the same value.

```
set odps.sql.hive.compatible=true;
select user_id, price, count(price) over
  (partition by user_id order by price) as count from test_src;
+------------+------------+------------+
| user_id    | price      | count      |
+------------+------------+------------+
| 1          | 4.5        | 1          | -- This row is the starting row of this window.
| 1          | 5.5        | 3          | -- The value of COUNT for the second row is the same as t
he value of COUNT for the third row because the prices of the two rows are the same.
| 1          | 5.5        | 3          | -- The value of COUNT for the third row is 3.
| 1          | 6.5        | 4          |
| 2          | NULL       | 0          |
| 2          | 3.0        | 1          |
| 3          | NULL       | 0          |
| 3          | 4.0        | 1          |
+------------+------------+------------+
```

1.6.6.4.3. AVG

This topic describes the AVG function and provides examples of using this function.

Syntax:

```
avg([distinct] expr) over(partition by [col1, col2…]
[order by [col1[asc|desc], col2[asc|desc]…]] [windowing_clause])
```

Description: This function returns the average value of input values in specified rows.

Parameters:

- expr: a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into the DOUBLE type before calculation. If the input value is not of the STRING or BIGINT type, an error is returned. If the input value for a row is NULL, the row is not used for calculation. Values of the BOOLEAN type are not used for calculation. If the distinct keyword is specified, the average value of distinct values is calculated.
- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by [col1[asc|desc], col2[asc|desc]: If ORDER BY is not specified, the average value of all values in the current window is returned. If ORDER BY is specified, the returned results are sorted in a specified order and the accumulated average value of the values from the starting row to the current row is returned.

> **Note** If the distinct keyword is specified, ORDER BY cannot be used.

Return value: A value of the DOUBLE type is returned.

If duplicate values are specified for ORDER BY, the processing method is based on the compatibility between MaxCompute and Hive.

- If MaxCompute is not compatible with Hive, the return value is the average value for each row.

```
set odps.sql.hive.compatible=false;
select user_id, price, avg(price) over
  (partition by user_id order by price) from test_src;
+------------+------------+-------------------+
| user_id    | price      | _c2               |
+------------+------------+-------------------+
| 1          | 4.5        | 4.5               |    -- This row is the starting row of this window.
| 1          | 5.5        | 5.0               |    -- The return value is the average value of the
values for the first and second rows.
| 1          | 5.5        | 5.166666666666667 |    -- The return value is the average value of the
values from the first row to the third row.
| 1          | 6.5        | 5.5               |
| 2          | NULL       | NULL              |
| 2          | 3.0        | 3.0               |
| 3          | NULL       | NULL              |
| 3          | 4.0        | 4.0               |
+------------+------------+-------------------+
```

- If MaxCompute is compatible with Hive, the return value is the average value of values for the last row of the rows with the same value.

```
set odps.sql.hive.compatible=true;
select user_id, price, avg(price) over
  (partition by user_id order by price) from test_src;
+------------+------------+-------------------+
| user_id    | price      | _c2               |
+------------+------------+-------------------+
| 1          | 4.5        | 4.5               |  -- This row is the starting row of this window.
| 1          | 5.5        | 5.166666666666667 |  -- The return value is the average value of value
s from the first row to the third row.
| 1          | 5.5        | 5.166666666666667 |  -- The return value is the average value of value
s from the first row to the third row.
| 1          | 6.5        | 5.5               |
| 2          | NULL       | NULL              |
| 2          | 3.0        | 3.0               |
| 3          | NULL       | NULL              |
| 3          | 4.0        | 4.0               |
+------------+------------+-------------------+
```

1.6.6.4.4. MAX

This topic describes the MAX function.

Syntax:

```
max([distinct] expr) over(partition by [col1, col2…]
[order by [col1[asc|desc], col2[asc|desc]…]] [windowing_clause])
```

Description: This function calculates the maximum value of specified rows in the current window.

Parameters:

- expr: a value of any data type except BOOLEAN. If the value for a row is NULL, this row is not used for calculation. If the distinct keyword is specified, the maximum value of distinct values is calculated. The calculation result is not affected regardless of whether the parameter is specified.
- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by [col1[asc|desc], col2[asc|desc]: If ORDER BY is not specified, the maximum value of all values in the current window is returned. If ORDER BY is specified, the returned results are sorted in the specified order and

the maximum value of values from the starting row to the current row in the current window is returned.

> **Note** If the distinct keyword is specified, ORDER BY cannot be used.

Return value: A value of the same data type as expr is returned.

1.6.6.4.5. MIN

This topic describes the MIN function.

Syntax:

```
min([distinct] expr) over(partition by [col1, col2…]
[order by [col1[asc|desc], col2[asc|desc]…]] [windowing_clause])
```

Description: This function returns the minimum value of values in specified rows.

Parameters:

- expr: a value of any data type except BOOLEAN. If the value for a row is NULL, the row is not used for calculation. If the distinct keyword is specified, the minimum value of distinct values is calculated. The calculation result is not affected regardless of whether the parameter is specified.
- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by [col1[asc|desc], col2[asc|desc]: If ORDER BY is not specified, the minimum value in the current window is returned. If ORDER BY is specified, the returned results are sorted in the specified order and the minimum value of values from the starting row to the current row in the current window is returned.

> **Note** If the distinct keyword is specified, ORDER BY cannot be used.

Return value: A value of the same data type as expr is returned.

1.6.6.4.6. MEDIAN

This topic describes the MEDIAN function.

Syntax:

```
double median(double number1,number2...) over(partition by [col1, col2…])
decimal median(decimal number1,number2...) over(partition by [col1, col2…])
```

Description: This function calculates the median.

Parameters:

- number1,number2...: numbers of the DOUBLE or DECIMAL type. You can enter 1 to 255 numbers. At least one number is required. If the input value is of the DOUBLE type, it is automatically converted into an array of the DOUBLE type before calculation. If the input value is of the STRING or BIGINT type, it is implicitly converted into the DOUBLE type before calculation. If it is not of the STRING or BIGINT type, an error is returned. If the input value is NULL, NULL is returned.
- partition by [col1, col2...]: the columns in the window that is used for calculation.

Return value: A value of the DOUBLE or DECIMAL type is returned.

1.6.6.4.7. STDDEV

This topic describes the STDDEV function and provides examples of using this function.

Syntax:

```
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]…]] [windowing_clause])
```

Description: This function calculates the population standard deviation of values in specified rows.

Parameters:

- expr: a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into the DOUBLE type before calculation. If the input value is not of the STRING or BIGINT type, an error is returned. If the value for a row is NULL, this row is not used for calculation. If the distinct keyword is specified, the population standard deviation of distinct values is calculated.
- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by [col1[asc|desc], col2[asc|desc]: If ORDER BY is not specified, the population standard deviation of rows in the current window is returned. If ORDER BY is specified, the returned results are sorted in the specified order and the population standard deviation from the starting row to the current row in the current window is returned.

> **Note** If the distinct keyword is specified, ORDER BY cannot be used.

Return value: If the input value is of the DECIMAL type, a value of the DECIMAL type is returned. Otherwise, a value of the DOUBLE type is returned.

Examples:

```
select window, seq, stddev_pop('1\01') over (partition by window order by seq) from dual;
```

If duplicate values are specified for ORDER BY, the processing method is based on the compatibility between MaxCompute and Hive.

- If MaxCompute is not compatible with Hive, the return value is the value of STDDEV for each row.

```
set odps.sql.hive.compatible=false;
select user_id, price, stddev(price) over
  (partition by user_id order by price) from test_src;
+------------+------------+--------------------+
| user_id    | price      | _c2                |
+------------+------------+--------------------+
| 1          | 4.5        | 0.0                |  -- This row is the starting row of this window.
| 1          | 5.5        | 0.5                |  -- The return value is the value of STDDEV for t
he first and second rows.
| 1          | 5.5        | 0.4714045207910316 |  -- The return value is the value of STDDEV from
the first row to the third row.
| 1          | 6.5        | 0.7071067811865475 |
| 2          | NULL       | NULL               |
| 2          | 3.0        | 0.0                |
| 3          | NULL       | NULL               |
| 3          | 4.0        | 0.0                |
+------------+------------+--------------------+
```

- If MaxCompute is compatible with Hive, the return value is the value of STDDEV for the last row of the rows with the same value.

```
set odps.sql.hive.compatible=true;
select user_id, price, stddev(price) over
  (partition by user_id order by price) from test_src;
+------------+------------+--------------------+
| user_id    | price      | _c2                |
+------------+------------+--------------------+
| 1          | 4.5        | 0.0                | -- This row is the starting row of this window.
| 1          | 5.5        | 0.4714045207910316 | -- The return value is the value of STDDEV from t
he first row to the third row.
| 1          | 5.5        | 0.4714045207910316 | -- The return value is the value of STDDEV from t
he first row to the third row.
| 1          | 6.5        | 0.7071067811865475 |
| 2          | NULL       | NULL               |
| 2          | 3.0        | 0.0                |
| 3          | NULL       | NULL               |
| 3          | 4.0        | 0.0                |
+------------+------------+--------------------+
```

1.6.6.4.8. STDDEV_SAMP

This topic describes the STDDEV_SAMP function.

Syntax:

```
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])
```

Description: This function calculates the sample standard deviation.

Parameters:

- expr: a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into the DOUBLE type before calculation. If the input value is not of the STRING or BIGINT type, an error is returned. If the value for a row is NULL, the row is not used for calculation. If the distinct keyword is specified, the sample standard deviation of distinct values is calculated.
- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by [col1[asc|desc], col2[asc|desc]: If ORDER BY is not specified, the sample standard deviation in the current window is returned. If ORDER BY is specified, the returned results are sorted in the specified order and the sample standard deviation from the starting row to the current row in the current window is returned.

> **Note** If the distinct keyword is specified, ORDER BY cannot be used.

Return value: If the input value is of the DECIMAL type, a value of the DECIMAL type is returned. Otherwise, a value of the DOUBLE type is returned.

1.6.6.4.9. SUM

This topic describes the SUM function and provides examples of using this function.

Syntax:

```
sum([distinct] expr) over(partition by [col1, col2…]
[order by [col1[asc|desc], col2[asc|desc]…]] [windowing_clause])
```

Description: This function returns the sum of values in specified rows.

Parameters:

- expr: a value of the DOUBLE, DECIMAL, or BIGINT type. If the input value is of the STRING type, it is implicitly converted to the DOUBLE type before calculation. If it is not of the STRING type, an error is returned. If the value for a row is NULL, the row is not used for calculation. If the distinct keyword is specified, the sum of distinct values is calculated.
- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by [col1[asc|desc], col2[asc|desc]: If ORDER BY is not specified, the sum of the expr values in the current window is returned. If ORDER BY is specified, the returned results are sorted in the specified order and the sum of the values from the starting row to the current row in the current window is returned.

> **Note** If the distinct keyword is specified, ORDER BY cannot be used.

Return value: If the input value is of the BIGINT type, a value of the BIGINT type is returned. If the input value is of the DECIMAL type, a value of the DECIMAL type is returned. If the input value is of the DOUBLE or STRING type, a value of the DOUBLE type is returned.

If duplicate values are specified for ORDER BY, the processing method is based on the compatibility between MaxCompute and Hive.

- If MaxCompute is not compatible with Hive, the return value is the sum of values for each row.

```
set odps.sql.hive.compatible=false;
select user_id, price, sum(price) over
  (partition by user_id order by price) from test_src;
+------------+------------+------------+
| user_id    | price      | _c2        |
+------------+------------+------------+
| 1          | 4.5        | 4.5        |  -- This row is the starting row of this window.
| 1          | 5.5        | 10.0       |  -- The return value is the sum of values for the first a
nd second rows.
| 1          | 5.5        | 15.5       |  -- The return value is the sum of values from the first
row to the third row.
| 1          | 6.5        | 22.0       |
| 2          | NULL       | NULL       |
| 2          | 3.0        | 3.0        |
| 3          | NULL       | NULL       |
| 3          | 4.0        | 4.0        |
+------------+------------+------------+
```

- If MaxCompute is compatible with Hive, the return value is the sum of values for the last row of the rows with the same value.

```
set odps.sql.hive.compatible=true;
select user_id, price, sum(price) over
  (partition by user_id order by price) from test_src;
+------------+------------+------------+
| user_id    | price      | _c2        |
+------------+------------+------------+
| 1          | 4.5        | 4.5        |  -- This row is the starting row of this window.
| 1          | 5.5        | 15.5       |  -- The return value is the sum of values from the first
row to the third row.
| 1          | 5.5        | 15.5       |  -- The return value is the sum of values from the first
row to the third row.
| 1          | 6.5        | 22.0       |
| 2          | NULL       | NULL       |
| 2          | 3.0        | 3.0        |
| 3          | NULL       | NULL       |
| 3          | 4.0        | 4.0        |
+------------+------------+------------+
```

1.6.6.4.10. DENSE_RANK

This topic describes the DENSE_RANK function and provides examples of using this function.

Syntax:

```
Bigint dense_rank() over(partition by [col1, col2…]
order by [col1[asc|desc], col2[asc|desc]…])
```

Description: This function returns the ranking of values. The data in rows with the same column value in ORDER BY has the same ranking.

Parameters:

- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by [col1[asc|desc], col2[asc|desc]: the value on which the ranking values are based.

Return value: A value of the BIGINT type is returned.

Examples:

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
```

Group employees by department, sort the employees in each group in descending order of sal, and then obtain the rankings of employees in each group.

```
select deptno
        , ename
        , sal
        , dense_rank() over (partition by deptno order by sal desc) as nums
-- DEPTNO (department) is the column in the window that is used for calculation. Values in the sal c
olumn are sorted to generate the ranking of each employee.
    from emp;
-- The following result is returned:
+------------+-------+------------+------------+
| 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          |
+------------+-------+------------+------------+
```

1.6.6.4.11. RANK

This topic describes the RANK function and provides examples of using this function.

Syntax:

```
Bigint rank() over(partition by [col1, col2…]
order by [col1[asc|desc], col2[asc|desc]…])
```

Description: This function determines the ranking of values in rows with the same column value, after the values are sorted in descending order by using the ORDER BY clause.

Parameters:

- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by [col1[asc|desc], col2[asc|desc]: the field based on which values are ranked.

Return value: A value of the BIGINT type is returned.

Examples:

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
```

Group employees by department, sort the employees in each group in descending order of sal, and then obtain the ranking values of employees in each group.

```
select deptno
        , ename
        , sal
        , rank() over (pratition by deptno order by sal desc) as nums
-- DEPTNO (department) is the column in the window that is used for calculation. Values in the sal c
olumn are sorted to generate the ranking value for each employee.
    from emp;
-- The following result is returned:
+------------+-------+------------+------------+
| 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          |
+------------+-------+------------+------------+
```

1.6.6.4.12. LAG

This topic describes the LAG function and provides examples of using this function.

Syntax:

```
lag(expr, Bigint offset, default) over(partition by [col1, col2…]
[order by [col1[asc|desc], col2[asc|desc]…]])
```

Description: This function returns the values for a row at a given offset before the current row. For example, if the current row is rn, the values for the rn - offsetth row are returned.

Parameters:

- expr: a value of any data type.
- offset: the offset, which is a constant of the BIGINT type. The value of the offset must be greater than 0. If the input value is of the STRING or DOUBLE type, it is implicitly converted into the BIGINT type before calculation.
- default: The default value when the offset is out of the valid range. The value of this parameter must be a constant. The default value of this parameter is NULL.
- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by col1[asc|desc], col2[asc|desc]: specifies how values in the rows are sorted.

Return value: A value of the same data type as expr is returned.

Examples:

```
select seq, lag(seq+100, 1) over (partition by window order by seq) as r from sliding_window;
-- The following result is returned:
+------------+------------+
| seq        | r          |
+------------+------------+
| 0          | NULL       |
| 1          | 100        |
| 2          | 101        |
| 3          | 102        |
| 4          | 103        |
| 5          | 104        |
| 6          | 105        |
| 7          | 106        |
| 8          | 107        |
| 9          | 108        |
+------------+------------+
```

1.6.6.4.13. LEAD

This topic describes the LEAD function and provides examples of using this function.

Syntax:

```
lead(expr, Bigint offset, default) over(partition by [col1, col2…]
[order by [col1[asc|desc], col2[asc|desc]…]])
```

Description: This function returns the values for a row at a given offset after the current row. For example, if the current row is rn, the values for the rn + offsetth row are returned.

Parameters:

- expr: a value of any data type.
- offset: the offset, which is a constant of the BIGINT type. The value of this parameter must be greater than 0. If the input value is of the STRING or DOUBLE type, it is implicitly converted into the BIGINT type.
- default: the default value when the value of offset is out of the valid range. The value of this parameter is a constant. The default value of this parameter is NULL.

- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by [col1[asc|desc], col2[asc|desc]: specifies how return values are sorted.

Return value: A value of the same data type as expr is returned.

Examples:

Column information in the test_lead table:
c_int_a,c_Double_a,c_String_a,c_String_b,c_time_a,c_time_b,c_String_in_fact_num.

```
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) from test_lead;
select c_string_a,c_time_b,c_double_a,lead(c_double_a,1) over(partition by c_string_a order by c_tim
e_b) from test_lead;
select c_string_in_fact_num,c_string_a,c_int_a,lead(c_int_a) over(partition by c_string_in_fact_num
order by c_string_a) from test_lead;
```

1.6.6.4.14. PERCENT_RANK

This topic describes the PERCENT_RANK function.

Syntax:

```
percent_rank() over(partition by [col1, col2…]
order by [col1[asc|desc], col2[asc|desc]…])
```

Description: This function returns the relative percent rank of a row in a group of data.

Parameters:

- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by col1[asc|desc], col2[asc|desc]: the value that is used to calculate the percent rank.

Return value: A value of the DOUBLE type is returned. Valid values: [0,1]. The relative percent rank is calculated by using the following formula: (Rank - 1)/(Number of rows - 1)

> **Note** The number of rows in a single window cannot exceed 10 million.

1.6.6.4.15. ROW_NUMBER

This topic describes the ROW_NUMBER function and provides examples of using this function.

Syntax:

```
row_number() over(partition by [col1, col2…]
order by [col1[asc|desc], col2[asc|desc]…])
```

Description: This function returns the ordinal number of the current row within a group of rows, counting from 1.

Parameters:

- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by [col1[asc|desc], col2[asc|desc]: the values that need to be sorted to return the ordinal number of the current row.

Return value: A value of the BIGINT type is returned.

Examples:

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
```

Group employees by department, sort the employees in each group in descending order of sal, and then obtain the ordinal numbers of employees in each group.

```
select deptno
        , ename
        , sal
        , row_number() over (partition by deptno order by sal desc) as nums
-- DEPTNO (department) is the column in the window that is used for calculation. Values in the sal c
olumn are sorted to return the ordinal number of the current row.
    from emp;
-- The following result is returned:
+------------+-------+------------+------------+
| 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          |
+------------+-------+------------+------------+
```

1.6.6.4.16. CLUSTER_SAMPLE

This topic describes the CLUSTER_SAMPLE function and provides examples of using this function.

Syntax:

```
boolean cluster_sample([Bigint x, Bigint y])
over(partition by [col1, col2..])
```

Description: This function performs cluster sampling.

Parameters:

- x: a constant of the BIGINT type. The value of this parameter must be greater than or equal to 1. If y is specified, x indicates that a window is divided into x portions. Otherwise, x indicates that the records of x rows in a window are extracted. In this case, True is returned for the x rows. If the value of x is NULL, NULL is returned.
- y: a constant of the BIGINT type. The value of y must be greater than or equal to 1 and less than or equal to x. y indicates that y records of the x portions in a window are extracted. In this case, True is returned for the y records. If the value of y is NULL, NULL is returned.
- partition by [col1, col2...]: the columns in the window that is used for calculation.

Return value: A value of the BOOLEAN type is returned.

Examples:

The test_tbl table contains two columns: key and value. key specifies a group, which can be groupa or groupb. value indicates the value of key. Data in the test_tbl table:

```
+------------+--------------------+
| 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  |
+------------+--------------------+
```

If you want to extract a sample of 10% of the values from each group, execute the following MaxCompute SQL statement:

```
select key, value from (select key, value, cluster_sample(10, 1) over(partition by key) as flag from
tbl) sub where flag = true;
-- The following result is returned:
+--------+--------------------+
| key    | value              |
+--------+--------------------+
| groupa | -0.273930924365012  |
| groupb | -1.11828960785008 |
+-----+----------------------
```

1.6.6.4.17. NTILE

This topic describes the NTILE function and provides examples of using this function.

Syntax:

```
BIGINT ntile(BIGINT n) over(partition by [col1, col2…]
[order by [col1[asc|desc], col2[asc|desc]…]] [windowing_clause]))
```

Description: This function divides rows into n ranking groups of as equal size as possible and returns the ranking group that a given row falls into. If rows are not evenly divided into ranking groups, more rows are included in the first ranking group.

Parameters: n: a value of the BIGINT type.

Return value: A value of the BIGINT type is returned.

Examples:

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
```

Classify all employees into three groups by department, sort employees in each group in descending order of sal, and then obtain the ranking groups of employees in each group.

```
select  deptno,ename,sal,ntile(3) over(partition by deptno order by sal desc) as nt3 from emp;
-- The following result is returned:
+------------+-------+------------+------------+
| 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          |
+------------+-------+------------+------------+
```

1.6.6.4.18. NTH_VALUE

This topic describes the NTH_VALUE function and provides examples of using this function.

Syntax:

```
nth_value(expr, number [, skipNull]) over(partition by [col1, col2…]
[order by [col1[asc|desc], col2[asc|desc]…]] [windowing_clause])
```

Description: This function calculates the nth value in the window. If the value of n exceeds the total number of rows in the window, NULL is returned.

Parameters:

- expr: a value of any basic data type.
- number: an integer that is greater than or equal to 1.
- skipNull: specifies whether to ignore the rows with NULL values when you calculate the nth value. The value of this parameter is of the BOOLEAN type. The default value is False.
- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by [col1[asc|desc], col2[asc|desc]: If ORDER BY is not specified, the value of expr of the nth row in the current window is returned. If ORDER BY is specified, the returned results are sorted in the specified order and the value of expr of the nth row from the starting row to the current row in the current window is returned.

Return value: A value of the same data type as expr is returned.

Examples:

```
select user_id, price, nth_value(price, 2) over
  (partition by user_id) as nth_value from test_src;
+------------+------------+------------+
| user_id    | price      | nth_value  |
+------------+------------+------------+
| 1          | 5.5        | 4.5        |
| 1          | 4.5        | 4.5        | -- This row is the second row in the current window.
```

```
| 1          | 6.5        | 4.5        |
| 1          | 5.5        | 4.5        |
| 2          | NULL       | 3.0        |
| 2          | 3.0        | 3.0        | -- This row is the second row in the current window.
| 3          | 4.0        | NULL       |
| 3          | NULL       | NULL       | -- This row is the second row in the current window.
+------------+------------+------------+
-- If ORDER BY is not specified, rows from the first row to the last row are in the current window.
The value of the second row is returned.
select user_id, price, nth_value(price, 3) over
  (partition by user_id) as nth_value from test_src;
+------------+------------+------------+
| user_id    | price      | nth_value  |
+------------+------------+------------+
| 1          | 5.5        | 6.5        |
| 1          | 4.5        | 6.5        |
| 1          | 6.5        | 6.5        | -- This row is the third row in the current window.
| 1          | 5.5        | 6.5        |
| 2          | NULL       | NULL       |
| 2          | 3.0        | NULL       |
| 3          | 4.0        | NULL       |
| 3          | NULL       | NULL       |
+------------+------------+------------+
-- If ORDER BY is not specified, rows from the first row to the last row are in the current window.
The value of the third row is returned.
-- The second and third windows have only two rows.
select user_id, price, nth_value(price, 2) over
  (partition by user_id order by price) as nth_value from test_src;
+------------+------------+------------+
| user_id    | price      | nth_value  |
+------------+------------+------------+
| 1          | 4.5        | NULL       | -- The current window has only one row. The second row exce
eds the window length.
| 1          | 5.5        | 5.5        |
| 1          | 5.5        | 5.5        |
| 1          | 6.5        | 5.5        |
| 2          | NULL       | NULL       |
| 2          | 3.0        | 3.0        |
| 3          | NULL       | NULL       |
| 3          | 4.0        | 4.0        |
+------------+------------+------------+
-- If ORDER BY is specified, rows from the first row to the current row belong to the current window
. The value of the second row is returned.
select user_id, price, nth_value(price, 1, true) over
  (partition by user_id) as nth_value from test_src;
+------------+------------+------------+
| user_id    | price      | nth_value  |
+------------+------------+------------+
| 1          | 5.5        | 5.5        |
| 1          | 4.5        | 5.5        |
| 1          | 6.5        | 5.5        |
| 1          | 5.5        | 5.5        |
| 2          | NULL       | 3.0        | -- The value of the first row is NULL, and therefore this r
ow is skipped.
| 2          | 3.0        | 3.0        |
| 3          | 4.0        | 4.0        |
| 3          | NULL       | 4.0        |
+------------+------------+------------+
-- If ORDER BY is not specified, rows from the first row to the last row belong to the current windo
```

```
w.
-- The value of the first row is returned. skipNull is set to True.
```

1.6.6.4.19. CUME_DIST

This topic describes the CUME_DIST function and provides examples of using this function.

Syntax:

```
cume_dist() over(partition by [col1, col2…]
order by [col1[asc|desc], col2[asc|desc]…]
```

Description: This function calculates the cumulative distribution. The cumulative distribution is the ratio of rows whose values are greater than or equal to the current value to all rows in a group.

Parameters: ORDER BY: the value used for comparison.

Return value: The ratio of the number of rows whose values are greater than or equal to the current value in the group to the total number of rows in the group is returned.

Examples:

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
```

Group all employees by department and obtain the cumulative distribution of sal 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;
-- The following result is returned:
+------------+-------+------------+-----------+
| 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%    |
| 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%    |
+------------+-------+------------+-----------+
```

1.6.6.4.20. FIRST_VALUE

This topic describes the FIRST_VALUE function and provides examples of using this function.

Syntax:

```
first_value(expr) over(partition by [col1, col2…]
[order by [col1[asc|desc], col2[asc|desc]…]] [windowing_clause]
```

Description: This function sorts rows and returns the first value in the range from the starting row to the current row.

Parameters:

- expr: a value of any basic data type.
- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by [col1[asc|desc], col2[asc|desc]: If ORDER BY is not specified, the value of expr of the starting row in the current window is returned. If ORDER BY is specified, the returned results are sorted in the specified order and the value of expr of the starting row in the current window is returned.

Return value: A value of the same data type as expr is returned.

Examples:

```
select user_id, price, first_value(price) over
  (partition by user_id) as first_value from test_src;
+------------+------------+-------------+
| user_id    | price      | first_value |
+------------+------------+-------------+
| 1          | 5.5        | 5.5         | -- This row is the starting row of this window.
| 1          | 4.5        | 5.5         |
| 1          | 6.5        | 5.5         |
| 1          | 5.5        | 5.5         |
| 2          | NULL       | NULL        | -- This row is the starting row of this window.
| 2          | 3.0        | NULL        |
| 3          | 4.0        | 4.0         | -- This row is the starting row of this window.
| 3          | NULL       | 4.0         |
+------------+------------+-------------+
-- If ORDER BY is not specified, rows from the first row to the last row belong to the current windo
w. The value of the starting row in the current window is returned.
select user_id, price, first_value(price) over
  (partition by user_id order by price) as first_value from test_src;
+------------+------------+-------------+
| user_id    | price      | first_value |
+------------+------------+-------------+
| 1          | 4.5        | 4.5         | -- This row is the starting row of this window.
| 1          | 5.5        | 4.5         |
| 1          | 5.5        | 4.5         |
| 1          | 6.5        | 4.5         |
| 2          | NULL       | NULL        | -- This row is the starting row of this window.
| 2          | 3.0        | NULL        |
| 3          | NULL       | NULL        | -- This row is the starting row of this window.
| 3          | 4.0        | NULL        |
+------------+------------+-------------+
-- If ORDER BY is specified, rows from the first row to the current row belong to the current window
. The value of the starting row in the current window is returned.
```

1.6.6.4.21. LAST_VALUE

This topic describes the LAST_VALUE function and provides examples of using this function.

Syntax:

```
last_value(expr) over(partition by [col1, col2…]
[order by [col1[asc|desc], col2[asc|desc]…]] [windowing_clause])
```

Description: This function sorts rows and returns the last value in the range from the starting row to the current row.

Parameters:

- expr: a value of any basic data type.
- partition by [col1, col2...]: the columns in the window that is used for calculation.
- order by [col1[asc|desc], col2[asc|desc]: If ORDER BY is not specified, the value of expr of the last row in the current window is returned. If ORDER BY is specified, the returned results are sorted in the specified order and the value of expr of the current row in the current window is returned.

Return value: A value of the same data type as expr is returned.

Examples:

```
select user_id, price, last_value(price) over
  (partition by user_id) as last_value from test_src;
+------------+------------+------------+
| user_id    | price      | last_value |
+------------+------------+------------+
| 1          | 5.5        | 5.5        |
| 1          | 4.5        | 5.5        |
| 1          | 6.5        | 5.5        |
| 1          | 5.5        | 5.5        | -- This row is the last row in this window.
| 2          | NULL       | 3.0        |
| 2          | 3.0        | 3.0        | -- This row is the last row in this window.
| 3          | 4.0        | NULL       |
| 3          | NULL       | NULL       | -- This row is the last row in this window.
+------------+------------+------------+
-- If ORDER BY is not specified, the rows from the first row to the last row belong to the current w
indow, and the value of the last row in the current window is returned.
select user_id, price, last_value(price) over
  (partition by user_id order by price) as last_value from test_src;
+------------+------------+------------+
| user_id    | price      | last_value |
+------------+------------+------------+
| 1          | 4.5        | 4.5        | -- This row is the current row in the current window.
| 1          | 5.5        | 5.5        | -- This row is the current row in the current window.
| 1          | 5.5        | 5.5        | -- This row is the current row in the current window.
| 1          | 6.5        | 6.5        | -- This row is the current row in the current window.
| 2          | NULL       | NULL       | -- This row is the current row in the current window.
| 2          | 3.0        | 3.0        | -- This row is the current row in the current window.
| 3          | NULL       | NULL       | -- This row is the current row in the current window.
| 3          | 4.0        | 4.0        | -- This row is the current row in the current window.
+------------+------------+------------+
-- If ORDER BY is specified, rows from the first row to the current row belong to the current window
. The value of the current row in the current window is returned.
```

1.6.6.5. Aggregate functions

1.6.6.5.1. Expressions of filter conditions in aggregate functions

This topic describes the expressions of filter conditions in aggregate functions and provides examples.

Filter conditions can be added for all aggregate functions. If FILTER is specified, only the rows for which the value of where_condition is TRUE are processed by the required aggregate functions.

Syntax:

```
aggregate_name(expression[,...]) [FILTER (WHERE where_condition)]
```

Examples:

```
SELECT
      SUM(x),
      SUM(x) FILTER (WHERE y > 1),
      SUM(x) FILTER (WHERE y > 2)
      FROM VALUES(NULL, 1),(1,2),(2,3),(3,NULL) AS t(x,y);
```

The following result is returned:

```
+------------+------------+------------+
| _c0        | _c1        | _c2        |
+------------+------------+------------+
| 6          | 3          | 2          |
+------------+------------+------------+
```

> **Note**
>
> - Only built-in aggregate functions support FILTER (WHERE where_condition). User-defined aggregate functions (UDAFs) do not support FILTER (WHERE where_condition).
> - COUNT(*) cannot be used with FILTER (WHERE where_condition). To implement this feature, use the COUNT_IF function.

1.6.6.5.2. COUNT

This topic describes the COUNT function and provides examples of using this function.

Syntax:

```
bigint count([distict|all] value)
```

Description: This function returns the number of records.

Parameters:

- distinct|all: specifies whether to deduplicate records during counting. The default value is all, which indicates that all records are counted. If this parameter is set to distinct, only records with distinct values are counted.
- value: a value of any data type. If the input value for a row is NULL, this row is not used for calculation. You can set value to an asterisk (*). This means that you can run count(*) to count all records.

Return value: A value of the BIGINT type is returned.

Examples:

The tbla table contains the col1 column of the BIGINT type. Data in the tbla table:

```
+------+
| col1 |
+------+
| 1 |
+------+
| 2 |
+------+
| NULL |
+------+
```

Execute the following SQL statements:

```
select count(*) from tbla;
-- 3 is returned.
select count(col1) from tbla;
-- 2 is returned.
```

The COUNT function can be used with the GROUP BY clause. For example, the test_src table contains two columns: key and value. key is of the STRING type and value is of the DOUBLE type.

Data in the test_src table:

```
+-----+-------+
| key | value |
+-----+-------+
| a   | 2.0   |
+-----+-------+
| a   | 4.0   |
+-----+-------+
| b   | 1.0   |
+-----+-------+
| b   | 3.0   |
+-----+-------+
```

Execute the following statement:

```
select key, count(value) as count from test_src group by key;
-- The following result is returned:
+-----+-------+
| key | count |
+-----+-------+
| a   | 2     |
+-----+-------+
| b   | 2     |
+-----+-------+
```

The COUNT function aggregates the values with the same key. The usage of other aggregate functions is the same as that of this function and is not described in detail in this document.

1.6.6.5.3. COUNT_IF

This topic describes the aggregate function COUNT_IF and provides examples.

Syntax:

```
bigint count_if(BOOLEAN expr)
```

Description: This function calculates the number of records whose value of expr is TRUE.

Parameters: expr, which is of the BOOLEAN type. If the value for a row is FALSE or NULL, the row is not counted.

Return value: A value of the BIGINT type is returned.

Examples:

```
SELECT COUNT_IF(x > 1), COUNT_IF(x <=1) FROM VALUES(NULL),(0),(1),(2) AS t(x);
```

The following result is returned:

```
+------------+------------+
| _c0        | _c1        |
+------------+------------+
| 1          | 2          |
+------------+------------+
```

1.6.6.5.4. AVG

This topic describes the AVG function and provides examples of using this function.

Syntax:

```
double avg(double value)
decimal avg(decimal value)
```

Description: This function returns the average value of a column.

Parameters: value: a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into the DOUBLE type before calculation. If the input value is not of the STRING or BIGINT type, an error is returned. If the input value for a row is NULL, this row is not used for calculation. Values of the BOOLEAN type cannot be used for calculation.

Return value: If the input value is of the DECIMAL type, a value of the DECIMAL type is returned. If the input value is not of the DECIMAL type, a value of the DOUBLE type is returned.

Examples:

The tbla table contains the value column of the BIGINT type. Data in the tbla table:

```
+-------+
| value |
+-------+
| 1     |
| 2     |
| NULL  |
+-------+
```

Execute the following statement:

```
select avg(value) as avg from tbla;
-- The average value of the value column is 1.5, which is calculated by using the following formula:
(1 + 2)/2
+------+
| avg  |
+------+
| 1.5  |
+------+
```

1.6.6.5.5. MAX

This topic describes the MAX function and provides examples of using this function.

Syntax:

```
max(value)
```

Description: This function calculates the maximum value of values in a column.

Parameters: value: a value of any data type. If the value for a row is NULL, this row that corresponds to the column is not used for calculation. Values of the BOOLEAN type are not used for calculation.

Return value: A value of the same data type as value is returned.

Examples:

The tbla table contains the col1 column of the BIGINT type. Data in the tbla table:

```
+------+
| col1 |
+------+
| 1    |
+------+
| 2    |
+------+
| NULL |
+------+
```

Execute the following statement:

```
select max(value) from tbla;
-- 2 is returned.
```

1.6.6.5.6. MIN

This topic describes the MIN function and provides examples of using this function.

Syntax:

```
min(value)
```

Description: This function calculates the minimum value of values in a column.

Parameters: value: a value of any data type. If the value for a row is NULL, this row that corresponds to the column is not used for calculation. Values of the BOOLEAN type are not used for calculation.

Return value: A value of the same data type as value is returned.

Examples:

The tbla table contains the value column of the BIGINT type. Data in the tbla table:

```
+------+
| value|
+------+
| 1 |
+------+
| 2 |
+------+
| NULL |
+------+
```

Execute the following statement:

```
select min(value) from tbla;
-- 1 is returned.
```

1.6.6.5.7. MEDIAN

This topic describes the MEDIAN function and provides examples of using this function.

Syntax:

```
double median(double number)
decimal median(decimal number)
```

Description: This function calculates the median value for a group of values.

Parameters: value: a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into the DOUBLE type before calculation. If the input value is not of the STRING or BIGINT type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned.

Examples:

The tbla table contains the value column of the BIGINT type. Data in the tbla table:

```
+------+
| value|
+------+
| 1 |
+------+
| 2 |
+------+
| 3 |
+------+
| 4 |
+------+
| 5 |
+------+
```

Execute the following statement:

```
select median(value) from tbla;
-- 3.0 is returned.
```

1.6.6.5.8. STDDEV

This topic describes the STDDEV function and provides examples of using this function.

Syntax:

```
double stddev(double number)
decimal stddev(decimal number)
```

Description: This function calculates the population standard deviation.

Parameters: value: a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into the DOUBLE type before calculation. If the input value is not of the STRING or BIGINT type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned.

Examples:

The tbla table contains the value column of the BIGINT type. Data in the tbla table:

```
+------+
| value|
+------+
| 1 |
+------+
| 2 |
+------+
| 3 |
+------+
| 4 |
+------+
| 5 |
+------+
```

Execute the following statement:

```
select stddev(value) from tbla;
-- 1.4142135623730951 is returned.
```

1.6.6.5.9. STDDEV_SAMP

This topic describes the STDDEV_SAMP function and provides examples of using this function.

Syntax:

```
double stddev_samp(double number)
decimal stddev_samp(decimal number)
```

Description: This function returns the sample standard deviation of a group of values.

Parameters: value: a value of the DOUBLE or DECIMAL type. If the input value is of the STRING or BIGINT type, it is implicitly converted into the DOUBLE type before calculation. If the input value is not of the STRING or BIGINT type, an error is returned.

Return value: A value of the DOUBLE or DECIMAL type is returned.

Examples:

The tbla table contains the value column of the BIGINT type. Data in the tbla table:

```
+------+
| value|
+------+
| 1 |
+------+
| 2 |
+------+
| 3 |
+------+
| 4 |
+------+
| 5 |
+------+
```

Execute the following statement:

```
select stddev_samp(value) from tbla;
-- 1.58113883008418981 is returned.
```

1.6.6.5.10. SUM

This topic describes the SUM function and provides examples of using this function.

Syntax:

```
sum(value)
```

Description: This function returns the sum of values in specified rows.

Parameters: value: a value of the DOUBLE, DECIMAL, or BIGINT type. If the input value is of the STRING type, it is implicitly converted into the DOUBLE type before calculation. If the input value for a row is NULL, this row is not used for calculation. Values of the BOOLEAN type are not used for calculation.

Return value: If the input value is of the BIGINT type, a value of the BIGINT type is returned. If the input value is of the DOUBLE or STRING type, a value of the DOUBLE type is returned.

Examples:

The tbla table contains the value column of the BIGINT type. Data in the tbla table:

```
+------+
| value|
+------+
| 1 |
+------+
| 2 |
+------+
| NULL |
+------+
```

Execute the following statement:

```
select sum(value) from tbla;
-- 3 is returned.
```

1.6.6.5.11. WM_CONCAT

This topic describes the WM_CONCAT function and provides examples of using this function.

Syntax:

```
string wm_concat(string separator, string str)
```

Description: This function returns a string of values that are separated by using a given delimiter. The delimiter is specified by separator.

Parameters:

- separator: the delimiter, which is a constant of the STRING type. If it is not of the STRING type or is not a constant, an error is returned.
- str: a value of the STRING type. If the input value is of the BIGINT, DOUBLE, or DATETIME type, it is implicitly converted into the STRING type before calculation. If the input value is not of the BIGINT, DOUBLE, or DATETIME type, an error is returned.

Return value: A value of the STRING type is returned.

> **Note** If test_src in select wm_concat(',', name) from test_src; is an empty set, NULL is returned.

Examples:

Group and sort values in the test table and use a string to list the values in the same group.

```
-- Create the test table.
create table test(id int , alphabet string);
-- Insert data into the test table.
insert into test values (1,'a'),(1,'b'),(1,'c'),(2,'D'),(2,'E'),(2,'F');
-- Group and sort values in the test table based on the id column. Then, use a string to list values
in the same group.
select id,wm_concat('',alphabet) from test group by id order by id limit 100;
+------------+------------+
| id         | _c1        |
+------------+------------+
| 1          | abc        |
| 2          | DEF        |
+------------+------------+
```

1.6.6.5.12. PERCENTILE

This topic describes the PERCENTILE function and provides examples of using this function.

Syntax:

```
double percentile(bigint col, p)
array<double> percentile(bigint col, array(p1 [, p2]...))
```

Description: This function returns the pth percentile of the specified column. p must be between 0 and 1.

> **Notice** You can calculate the percentile only for integer values.

Parameters:

- col: the name of the table column of the BIGINT type.
- p: the percentile that must be between 0 and 1.

Examples:

The var_test table contains the c1 column. Data in the var_test table:

```
+------------+
| c1         |
+------------+
| 8          |
| 9          |
| 10         |
| 11         |
+------------+
```

Execute the following statement to calculate the pth percentile of the c1 column.

```
select percentile(c1,0),percentile(c1,0.3),percentile(c1,0.5),percentile(c1,1) from var_test;
-- The following result is returned:
+------------+------------+------------+------------+
| _c0 | _c1 | _c2 | _c3 |
+------------+------------+------------+------------+
| 8.0 | 8.9 | 9.5 | 11.0 |
+------------+------------+------------+------------+
select percentile(c1,array(0,0.3,0.5,1))from var_test;
-- The following result is returned:
+------+
| _c0 |
+------+
| [8, 8.9, 9.5, 11] |
+------+
```

1.6.6.5.13. Additional aggregate functions

1.6.6.5.13.1. Usage notes

This topic describes the configuration operation that you must perform before you use additional aggregate functions.

MaxCompute V2.0 provides more aggregate functions. You must add the following SET statement before the SQL statements for these new functions:

```
set odps.sql.type.system.odps2=true;
```

Note You must submit and execute the SET statement and the SQL statements for the new functions at the same time.

1.6.6.5.13.2. COLLECT_LIST

This topic describes the COLLECT_LIST function and provides examples of using this function.

Syntax:

```
ARRAY collect_list(col)
```

Description: This function aggregates the expressions that are specified by col in a given group into an array.

Parameters: col: a column of the table, which is of any data type.

Return value: A value of the ARRAY type is returned.

1.6.6.5.13.3. COLLECT_SET

This topic describes the COLLECT_SET function.

Syntax:

```
array collect_set(col)
```

Description: This function aggregates the expressions that are specified by col in a given group into an array without duplicate elements.

Parameters: col: a table column, which can be of any data type.

Return value: A value of the ARRAY type is returned.

1.6.6.5.13.4. VARIANCE or VAR_POP

This topic describes the VARIANCE or VAR_POP function and provides examples of using this function.

Syntax:

```
DOUBLE variance(col)
DOUBLE var_pop(col)
```

Description: This function calculates the variance of a specified column of a numeric data type.

Parameters: col: the column of a numeric data type. If it is not of a numeric data type, NULL is returned.

Return value: A value of the DOUBLE type is returned.

Examples:

The test table contains thec1 column. Data in the test table:

```
+------------+
| c1         |
+------------+
| 8          |
| 9          |
| 10         |
| 11         |
+------------+
```

Execute the following statement to calculate the variance of the c1 column:

```
select variance(c1) from test;
-- You can also execute the following statement:
select var_pop(c1) from test;
-- The following result is returned:
+------------+
| _c0 |
+------------+
| 1.25 |
+------------+
```

1.6.6.5.13.5. VAR_SAMP

This topic describes the VAR_SAMP function and provides examples of using this function.

Syntax:

```
DOUBLE var_samp(col)
```

Description: This function calculates the sample variance of a specified column of a numeric data type.

Parameters: col: a column of a numeric data type. If it is not of a numeric data type, NULL is returned.

Return value: A value of the DOUBLE type is returned

Examples:

The test table contains the c1 column. Data in the test table:

```
+------------+
| c1         |
+------------+
| 8          |
| 9          |
| 10         |
| 11         |
+------------+
```

Execute the following statement to calculate the sample variance of the c1 column:

```
select var_samp(c1) from test;
-- The following result is returned:
+------------+
| _c0 |
+------------+
| 1.6666666666666667 |
+------------+
```

1.6.6.5.13.6. COVAR_POP

This topic describes the COVAR_POP function and provides examples of using this function.

Syntax:

```
DOUBLE covar_pop(col1, col2)
```

Description: This function calculates the population covariance of two columns of a numeric data type.

Parameters: col1 and col2: the columns of a numeric data type. If they are not of a numeric data type, NULL is returned.

Return value: A value of the DOUBLE type is returned.

Examples:

The test table contains the c1 and c2 columns. Data in the test table:

```
+------------+------------+
| c1         | c2         |
+------------+------------+
| 3          | 2          |
| 14         | 5          |
| 50         | 14         |
| 26         | 75         |
+------------+------------+
```

Execute the following statement to calculate the population covariance of the c1 and c2 columns:

```
select covar_pop(c1,c2) from test;
-- The following result is returned:
+------------+
| _c0        |
+------------+
| 123.49999999999997|
+------------+
```

1.6.6.5.13.7. COVAR_SAMP

This topic describes the COVAR_SAMP function and provides examples of using this function.

Syntax:

```
DOUBLE covar_samp(col1, col2)
```

Description: This function calculates the sample covariance of two columns of a numeric data type.

Parameters: col1 and col2: the columns of a numeric data type. If they are not of a numeric data type, NULL is returned.

Return value: A value of the DOUBLE type is returned

Examples:

The test table contains the c1 and c2 columns. Data in the test table:

```
+------------+------------+
| c1         | c2         |
+------------+------------+
| 3          | 2          |
| 14         | 5          |
| 50         | 14         |
| 26         | 75         |
+------------+------------+
```

Execute the following statement to calculate the sample covariance of the c1 and c2 columns:

```
select covar_samp(c1,c2) from test;
-- The following result is returned:
+------------+
| _c0        |
+------------+
| 164.66666666666663|
+------------+
```

1.6.6.5.13.8. ANY_VALUE

This topic describes the ANY_VALUE function and provides examples of using this function.

Syntax:

```
ANY_VALUE(value)
```

Description: This function returns a non-deterministic value from a specific column.

Parameters: value: a value of any data type. If the input value for a row is NULL, this row is not used for calculation.

Return value: A value of the same data type as the input value is returned.

Examples:

```
select key, ANY_VALUE(value)
from values (1, 'value1'),(1, 'value2'),(2, 'value3'),(2, NULL),(3, NULL) as t(key, value)
group by key;
-- The following result is returned:
+------------+------------+
| key        | _c1        |
+------------+------------+
| 1          | value1     |
| 2          | value3     |
| 3          | NULL       |
+------------+------------+
```

Note The value returned by ANY_VALUE is not deterministic and different values may be returned for the same input value.

1.6.6.5.13.9. NUMERIC_HISTOGRAM

This topic describes the NUMERIC_HISTOGRAM function.

Syntax:

```
map<double, double> numeric_histogram(bigint buckets , double value)
```

Description: This function returns the approximate histogram of a given column.

Parameters:

- buckets: the maximum number of buckets in the column whose approximate histogram is returned. The value must be of the BIGINT type.
- value: the column whose approximate histogram you want to obtain. The column must be of the DOUBLE type.

Return value: A value of the MAP<DOUBLE,DOUBLE> type is returned. In the return value, a key indicates an x-coordinate, and its value indicates the height on the approximate histogram of the column.

1.6.6.5.13.10. PERCENTILE_APPROX

This topic describes the PERCENTILE_APPROX function and provides examples of using this function.

Syntax:

```
double percentile_approx(double col, p [, B]))
array<double>  percentile_approx(double col, array(p1 [, p2]...) [, B])
```

Description: This function returns the approximate percentile value of the col column at the given percentage p.

Parameters:

- p: the given percentile value. Valid values: [0.0,1.0]. You can specify one percentage to return an approximate percentile value or multiple percentages to return an array that consists of percentile values.
- B: the accuracy of the return value. A higher accuracy indicates a more accurate value. If you do not specify this parameter, 10000 is used. If the number of values in the col column is less than B, an exact percentile value is returned.

Return value: percentile_approx(double col, p [, B])) returns a single approximate percentile value. percentile_approx(double col, array(p1 [, p2]...) [, B]) returns an array that consists of multiple percentile values.

Examples:

```
select percentile_approx(10.0, 0.5, 100);
-- 10.0 is returned.
select percentile_approx(10.0, array(0.5, 0.4, 0.1), 100);
-- [10.0,10.0,10.0] is returned.
```

1.6.6.5.13.11. APPROX_DISTINCT

This topic describes the APPROX_DISTINCT function and provides examples of using this function.

Syntax:

```
bigint approx_distinct(value)
```

Description: This function returns the approximate number of distinct input values.

Parameters: value: the input data that is used for deduplication.

Return value: A value of the BIGINT type is returned. If all input values are NULL, 0 is returned. This function produces a standard error of 5%.

1.6.6.6. Other functions

1.6.6.6.1. ARRAY

This topic describes the ARRAY function and provides examples of using this function.

Syntax:

```
ARRAY array(value1,value2, ...)
```

Description: This function creates an array of a given data type that is specified by value.

Parameters: value: any data type. All the values must be of the same data type.

Return value: A value of the ARRAY type is returned.

Examples:

```
select array(123,456,789);
-- The following result is returned:
[123, 456, 789]
```

1.6.6.6.2. ARRAY_CONTAINS

This topic describes the ARRAY_CONTAINS function and provides examples of using this function.

Syntax:

```
boolean array_contains(ARRAY<T> a, value v)
```

Description: This function checks whether array a contains value v.

Parameters:

- a: a value of the ARRAY type.
- v: a value of the same data type as other values in the array.

Return value: A value of the BOOLEAN type is returned.

Examples:

```
select array_contains(array('a','b'), 'a');
-- True is returned.
select array_contains(array(456,789),123);
-- False is returned.
```

1.6.6.6.3. CAST

This topic describes the CAST function.

Syntax:

```
cast(expr as <type>)
```

Description: This function converts the result of an expression into another data type. For example, `cast('1' as bigint)` converts '1' of the STRING type into 1 of the INT type. If the conversion fails, an error is returned.

Notes:

- `cast(double as bigint)`: converts a value of the DOUBLE type to the BIGINT type.
- `cast(string as bigint)`: converts a value of the STRING type to the BIGINT type. If the string consists of numerals expressed as integers, it is directly converted into the BIGINT type. If the string consists of numerals expressed in the FLOAT or EXPONENTIAL form, it is converted into the DOUBLE type and then into the BIGINT type.
- The date format for `cast(string as datetime)` and `cast(datetime as > string)` is yyyy-mm-dd hh:mi:ss.

1.6.6.6.4. COALESCE

This topic describes the COALESCE function.

Syntax:

```
coalesce(expr1, expr2, ...)
```

Description: This function returns the first non-NULL value in a list. If all values in the list are NULL, NULL is returned.

Parameters: expr: the values you want to test. All these values must be of the same data type or are all NULL. Otherwise, an error is returned.

> **Note** The list must include at least one parameter. Otherwise, an error is returned.

Return value: A value of the same data type as the expr parameter is returned.

1.6.6.6.5. DECODE

This topic describes the DECODE function and provides examples of using this function.

Syntax:

```
decode(expression, search, result[, search, result]...[, default])
```

Description: This function implements the IF-THEN-ELSE conditional branching feature.

Parameters:

- expression: the expression that you want to compare.
- search: the search item that you want to compare with expression.
- result: the value returned if the values of search and expression match.
- default: the value returned if no search item matches the expression. If default is not specified, NULL is returned. This parameter is optional.

Return value: The matched search value is returned. If no matched item exists, default is returned. If default is not specified, NULL is returned.

> **Note**
>
> - Three or more parameters must be specified.
> - All results must be of the same data type or be NULL. Inconsistent data types will cause an error. All values of search and expression must be of the same data type. Otherwise, an error is returned.
> - If search in the DECODE function has duplicate values that match the expression, the first mapping result is returned.

Examples:

```
select
decode(customer_id,
1, 'Taobao',
2, 'Alipay',
3, 'Aliyun',
NULL, 'N/A',
'Others') as result
FROM VALUES (1,10),
        (1,10),
        (2,null),
        (3,13),
        (4,0),
        (null,6),
        (5,18)
        as t(customer_id,value);
```

In this example, the DECODE function implements the 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;
```

> **Note**
>
> - In most cases, if "NULL = NULL" appears during data computation, the result returned by the MaxCompute SQL engine is NULL, whereas the DECODE function considers that the two NULL values are the same.
> - In this example, if the value of customer_id is NULL, the DECODE function returns N/A.

1.6.6.6.6. EXPLODE

This topic describes the EXPLODE function and provides examples of using this function.

Syntax:

```
explode (var)
```

Description: This function is a user-defined table-valued function (UDTF) that transposes one row into multiple rows.

- If var is of the ARRAY <T> type, the data of the ARRAY type stored in a column is converted into multi-row data.
- If var is of the MAP<K, V> type, each key-value pair of the MAP type stored in a column is transposed to a row with two columns. One column is used for storing keys and the other column is used for storing values.

Parameters: var: a value of the ARRAY<T> or MAP<K, V> type.

Return value: transposed rows.

> **Note**
>
> Limits on the EXPLODE function:
>
> - A SELECT statement can specify only the column that is used for this function. Other columns cannot be specified.
> - This function cannot be used with the GROUP BY, CLUSTER BY, DISTRIBUTE BY, or SORT BY clause.

Examples:

```
select explode(array(null, 'a', 'b', 'c'));
-- The following result is returned:
+------------+
| col        |
+------------+
| NULL       |
| a          |
| b          |
| c          |
+------------+
```

1.6.6.6.7. GET_IDCARD_AGE

This topic describes the GET_IDCARD_AGE function.

Syntax:

```
get_idcard_age(idcardno)
```

Description: This function returns the current age based on the ID card number. The current age is the current year minus the birth year in the ID card number.

Parameters: idcardno: the ID card number of the STRING type. The value is a 15-digit or 18-digit number. During data 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.

Return value: A value of the BIGINT type is returned. If the input value is NULL, NULL is returned. If the difference between the current year and the birth year is greater than 100, NULL is returned.

1.6.6.6.8. GET_IDCARD_BIRTHDAY

This topic describes the GET_IDCARD_BIRTHDAY function.

Syntax:

```
get_idcard_birthday(idcardno)
```

Description: This function returns the date of birth based on the ID card number.

Parameters: idcardno: the ID card number of the STRING type. The value is a 15-digit or 18-digit number. During data 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.

Return value: A value of the DATETIME type is returned. If the input value is NULL, NULL is returned.

1.6.6.6.9. GET_IDCARD_SEX

This topic describes the GET_IDCARD_SEX function.

Syntax:

```
get_idcard_sex(idcardno)
```

Description: This function returns the gender based on the ID card number. The return value is M or F. M indicates male and F indicates female.

Parameters: idcardno: the ID card number of the STRING type. The value is a 15-digit or 18-digit number. During data 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.

Return value: A value of the STRING type is returned. If the input value is NULL, NULL is returned.

1.6.6.6.10. GREATEST

This topic describes the GREATEST function.

Syntax:

```
greatest(var1, var2, ...)
```

Description: This function returns the maximum value of the input parameters.

Parameters: var1 and var2: values of the BIGINT, DOUBLE, DECIMAL, DATETIME, or STRING type. If all input values are NULL, NULL is returned.

Return value:

- The maximum value in the values of all input parameters. If implicit conversion is not required, the data type of the return value is the same as the data types of input parameters.
- NULL is interpreted as the minimum value.

- If a data type conversion is performed among the DOUBLE, BIGINT, and STRING types, a value of the DOUBLE type is returned. If a data type conversion is performed between the STRING and DATETIME types, a value of the DATETIME type is returned. If a data type conversion is performed among the DECIMAL, DOUBLE, BIGINT, and STRING types, a value of the DECIMAL type is returned. Implicit conversions of other data types are not allowed.
- If `odps.sql.hive.compatible` is set to true and the value of an input parameter is NULL, NULL is returned.

1.6.6.6.11. INDEX

This topic describes the INDEX function and provides examples of using this function.

Syntax:

```
index(var1[var2])
```

Description: This function returns the var2th value if var1 is of the ARRAY<T> type and returns the value whose key is var2 in var1 if var1 is of the MAP<K, V> type.

Parameters:

- var: a value of the ARRAY<T> or MAP<K, V> type.
- var2: If the value of var1 is of the ARRAY<T> type, the value of var2 is of the BIGINT type and greater than or equal to 0. If the value of var1 is of the MAP<K, V> type, the value of var2 is of the K type.

Return value:

- If the value of var1 is of the ARRAY<T> type, a value of the T type is returned. If the value of var2 is beyond the range of elements in the value of the ARRAY<T> type, NULL is returned.
- If the value of var1 is of the MAP<K, V> type, a value of the V type is returned. If the value whose key is var2 does not exist in values of the MAP<K, V> type, NULL is returned.

Examples:

If the value of var1 is an array, execute the following SQL statement:

```
select array('a','b','c')[2] from dual;
-- The following result is returned:
+-----+
| _c0 |
+-----+
| c   |
+-----+
```

If the value of var1 is of the MAP<K, V> type, execute the following SQL statement:

```
select str_to_map("test1=1,test2=2")["test1"] from dual;
-- The following result is returned:
+-----+
| _c0 |
+-----+
| 1   |
+-----+
```

> **Note**
> - To execute the SQL statement, remove the index, and execute `var1[var2]` directly. Otherwise, a syntax error is returned.
> - If the value of var1 is NULL, NULL is returned.

1.6.6.6.12. MAX_PT

This topic describes the MAX_PT function and provides examples of using this function.

Syntax:

```
max_pt(table_full_name)
```

Description: This function returns the maximum value of each first-level partition of a partitioned table in alphabetic order. This function also reads data from the data file in the first-level partitions.

Parameters: table_full_name: the table name that has a project name, such as prj.src. The value of this parameter is of the STRING type. You must have the permissions to read the table.

Return value: The maximum value in each first-level partition is returned.

Examples:

The tbl table is a partitioned table and contains the following partitions:

```
pt ='20120901'
pt ='20120902'
```

The return value of the following statement is '20120902'. The MaxCompute SQL statement reads data from the 20120902 partition.

```
select * from tbl where pt=max_pt('myproject.tbl');
```

> **Note** If a partition is added by using the ALTER TABLE statement and the partition contains no data file, the maximum value in this partition is not returned.

1.6.6.6.13. ORDINAL

This topic describes the ORDINAL function and provides examples of using this function.

Syntax:

```
ordinal(bigint nth, var1, var2, ...)
```

Description: This function sorts the input variables in ascending order, and returns the value in the nth bit.

Parameters:

- nth: a value of the BIGINT type. This parameter specifies the position in which the value is to be returned. If the value of this parameter is NULL, NULL is returned.
- var: the value of the BIGINT, DOUBLE, DATETIME, or STRING type.

Return value:

- The value in the nth bit is returned. If no implicit conversion is required, the return value is of the same data type as the input parameter.

- If a data type conversion is performed among the DOUBLE, BIGINT, and STRING types, a value of the DOUBLE type is returned. If a data type conversion is performed between the STRING and DATETIME types, a value of the DATETIME type is returned. Implicit conversions of other data types are not allowed.
- NULL is interpreted as the minimum value.

Examples:

```
ordinal(3, 1, 3, 2, 5, 2, 4, 6) =  2
```

1.6.6.6.14. LEAST

This topic describes the LEAST function.

Syntax:

```
least(var1, var2, ...)
```

Description: This function returns the minimum value of the input parameters.

Parameters: var: a value of the BIGINT, DOUBLE, DECIMAL, DATETIME, or STRING type. If the values of all input parameters are NULL, NULL is returned.

Return value:

- The minimum value of all values of the input parameters is returned. If implicit conversion is not required, the return value must be of the same data type as the input parameters.
- If a data type conversion is performed among the DOUBLE, BIGINT, and STRING types, a value of the DOUBLE type is returned. If a data type conversion is performed between the STRING and DATETIME types, a value of the DATETIME type is returned. If a data type conversion is performed among the DECIMAL, DOUBLE, BIGINT, and STRING types, a value of the DECIMAL type is returned. Implicit conversions of other data types are not allowed.
- NULL is interpreted as the minimum value.

1.6.6.6.15. SIZE

This topic describes the SIZE function and provides examples of using this function.

Syntax:

```
INT size(map)
INT size(array)
```

Description: size(map) returns the number of key-value pairs in the specified map parameter. size(array) returns the number of elements in the specified array parameter.

Parameters:

- map: a value of the MAP type.
- array: a value of the ARRAY type.

Return value: A value of the INT type is returned.

Examples:

```
select size(map('a',123,'b',456));
-- 2 is returned.
select size(map('a',123,'b',456,'c',789));
-- 3 is returned.
select size(array('a','b'));
-- 2 is returned.
select size(array(123,456,789));
-- 3 is returned.
```

1.6.6.6.16. SPLIT

This topic describes the SPLIT function and provides examples of using this function.

Syntax:

```
split(str, pat)
```

Description: This function returns an array after str is split with pat.

Parameters:

- str: the string that you want to split, which is of the STRING type.
- pat: a delimiter of the STRING type, which supports regular expressions.

Return value: The result after str is split with pat is returned. The return value is of the ARRAY<STRING> type.

Examples:

```
select split("a,b,c",",");
-- The following result is returned:
+------+
| _c0  |
+------+
| [a, b, c] |
+------+
```

1.6.6.6.17. STR_TO_MAP

This topic describes the STR_TO_MAP function and provides examples of using this function.

Syntax:

```
str_to_map(text [, delimiter1 [, delimiter2]])
```

Description: This function splits text into key-value pairs by using delimiter1 and then splits the key and value in a key-value pair by using delimiter2.

Parameters:

- text: the string that you want to split. The value of this parameter is of the STRING type.
- delimiter1: the delimiter that splits text into key-value pairs. The value of this parameter is of the STRING type. If this parameter is not specified, a comma (,) is used.
- delimiter2: the delimiter that splits each key-value pair into a key and a value. The value of this parameter is of the STRING type. If this parameter is not specified, a period (.) is used.

Return value: The result after text is split by using delimiter1 and delimiter2 is returned. The return value is of the MAP<STRING, STRING> type.

Examples:

```
select str_to_map("test1=1,test2=2");
-- The following result is returned:
+------------+
| a          |
+------------+
| {test1:1, test2:2} |
```

1.6.6.6.18. UUID

This topic describes the UUID function.

Syntax:

```
string uuid()
```

Description: This function returns a random ID, for example, 29347a88-1e57-41ae-bb68-a9edbdd94212.

> **Note** The return value of this function is a random global ID that has a low probability of duplication.

1.6.6.6.19. UNIQUE_ID

This topic describes the UNIQUE_ID function.

Syntax:

```
string unique_id()
```

Description: This function returns a unique random ID, for example, 29347a88-1e57-41ae-bb68-a9edbdd94212_1.

> **Note** This function is more efficient than the UUID function.

1.6.6.6.20. SAMPLE

This topic describes the SAMPLE function and provides examples of using this function.

Syntax:

```
boolean sample(x, y, column_name1,column_name2,...)
```

Description: This function samples all values that are read from the column specified by column_name based on x and y, and filters out the rows that do not meet the sampling condition.

Parameters:

- x and y: integer constants that are greater than 0. The values of the two parameters are of the BIGINT type. These parameters indicate that the values fall into x portions by calling the hash function and the yth portion is used. If y is not specified, the first portion is used, and you must not specify column_name. If the value of x or y is not of the BIGINT type or is less than or equal to 0, an error is returned. If the value of y is greater than the value of x, an error is returned. If the value of x or y is NULL, NULL is returned.
- column_name: the column from which you want to sample values. This parameter is optional. If this parameter is not specified, random sampling is performed based on the values of x and y. This parameter can be of any data type, and its value can be NULL. No implicit conversion is performed. If the value of column_name is a

constant, an error is returned.

Return value: A value of the BOOLEAN type is returned.

> **Note** To avoid data skew due to NULL values, the system performs uniform hashing on NULL values in the columns specified by column_name. This ensures that data evenly falls into x portions. If column_name is not specified and the amount of data is small, random hashing may be performed. In this case, we recommend that you specify column_name for uniform hashing.

Examples:

The tbla table contains the cola column.

```
select * from tbla where sample (4, 1 , cola) = true;
-- The values in the cola column fall into four portions by calling the hash function, and data in the first portion is extracted.
select * from tbla where sample (4, 2) = true;
-- The values in each row are randomly hashed into four portions, and data in the second portion is extracted.
```

1.6.6.6.21. CASE WHEN expression

This topic describes the CASE WHEN expression and provides examples of using this function.

MaxCompute provides two syntax formats for the CASE WHEN expression.

```
case value
when value1 then result1
when value2 then result2
...
else resultn
end
```

```
case
when (_condition1) then result1
when (_condition2) then result2
when (_condition3) then result3
...
else resultn
end
```

The CASE WHEN expression can return different values based on the calculation result of the value expression. The following example shows that different regions are obtained based on the value of shop_name.

```
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 all result values are of the BIGINT or DOUBLE type, the data types are converted into the DOUBLE type before the values are returned.
> - If some result values are of the STRING type, the data types of all values are converted into the STRING type before the values are returned. If the conversion cannot be performed, for example, the BOOLEAN type cannot be converted into the STRING type, an error is returned.
> - Conversions between other data types are not allowed.

1.6.6.6.22. IF

This topic describes the IF function and provides examples of using this function.

Syntax:

```
if(testCondition, valueTrue, valueFalseOrNull)
```

Description: This function determines whether testCondition evaluates to true. If testCondition evaluates to true, valueTrue is returned. Otherwise, valueFalse or NULL is returned.

Parameters:

- testCondition: the expression that you want to evaluate. The value of this parameter is of the BOOLEAN type.
- valueTrue: the value returned if testCondition evaluates to true.
- valueFalseOrNull: the value returned if testCondition evaluates to false. It can be set to NULL.

Return value: The data type of the return value is the same as that of valueTrue or valueFalseOrNull.

Examples:

```
select if(1=2,100,200);
-- The following result is returned:
+------------+
| _c0        |
+------------+
| 200        |
+------------+
```

1.6.6.6.23. Additional functions

1.6.6.6.23.1. Usage notes

This topic describes the configuration operation that you must perform before you use other functions that are added to MaxCompute V2.0.

MaxCompute V2.0 provides more other functions. You must add the following SET statement before the SQL statements for these new functions:

```
set odps.sql.type.system.odps2=true;
```

> **Note** You must submit and execute the SET statement and the SQL statements for the new functions at the same time.

1.6.6.6.23.2. MAP

This topic describes the MAP function and provides examples of using this function.

Syntax:

```
map map(K key1, V value1, K key2, V value2, ...)
```

Description: This function creates mappings by using the given key-value pairs.

Parameters:

- key: All keys must be of the same data types, such as those after an implicit conversion. The data types must be basic types.
- value: All value types, such as those after an implicit conversion, must be the same and can be of any data types.

Return value: A value of the MAP type is returned.

Examples:

The t_table table contains the c1, c2, c3, c4, and c5 columns. The c1, c4, and C5 columns are of the BIGINT type, and the c2 and c3 columns are of the STRING type. Data in the t_table table:

```
+------------+----+----+------------+------------+
| c1         | c2 | c3 | c4         | c5         |
+------------+----+----+------------+------------+
| 1000       | k11 | k21 | 86         | 15         |
| 1001       | k12 | k22 | 97         | 2          |
| 1002       | k13 | k23 | 99         | 1          |
+------------+----+----+------------+------------+
```

Execute the following statement:

```
select  map(c2,c4,c3,c5) from t_table;
-- The following result is returned:
+------+
| _c0  |
+------+
| {k11:86, k21:15} |
| {k12:97, k22:2} |
| {k13:99, k23:1} |
+------+
```

1.6.6.6.23.3. MAP_KEYS

This topic describes the MAP_KEYS function and provides examples of using this function.

Syntax:

```
ARRAY map_keys(map<K, V>)
```

Description: This function returns all keys in the MAP parameter as an array.

Parameters: map: a value of the MAP type.

Return value: A value of the ARRAY type is returned. If the input value is NULL, NULL is returned.

Examples:

The t_table_map table contains two columns: c1 and t_map. c1 is of the BIGINT type and t_map is of the MAP<STRING,BIGINT> type.

```
+------------+-------+
| c1         | t_map |
+------------+-------+
| 1000       | {k11:86, k21:15} |
| 1001       | {k12:97, k22:2} |
| 1002       | {k13:99, k23:1} |
+------------+-------+
```

Execute the following statement:

```
select  c1,map_keys(t_map) from t_table_map;
-- The following result is returned:
+------------+------+
| c1         | _c1  |
+------------+------+
| 1000       | [k11, k21] |
| 1001       | [k12, k22] |
| 1002       | [k13, k23] |
+------------+------+
```

1.6.6.6.23.4. MAP_VALUES

This topic describes the MAP_VALUES function and provides examples of using this function.

Syntax:

```
ARRAY map_values(map<K, V>)
```

Description: This function returns all values in the map parameter as an array.

Parameters: map: a value of the MAP type.

Return value: A value of the ARRAY type is returned. If the input value is NULL, NULL is returned.

Examples:

```
select map_values(map('a',123,'b',456));
-- The following result is returned:
[123, 456]
```

1.6.6.6.23.5. SORT_ARRAY

This topic describes the SORT_ARRAY function and provides examples of using this function.

Syntax:

```
ARRAY sort_array(ARRAY<T>)
```

Description: This function sorts data in a given array.

Parameters: ARRAY<T>: data of the ARRAY type. Data in the array can be of any data type.

Return value: A value of the ARRAY type is returned.

Examples:

The t_array table contains three columns: c1, c2, and c3. The c1 and c3 columns are of the ARRAY<STRING> type, and the c2 column is of the ARRAY<INT> type.

```
+------------+---------+--------------+
| c1         | c2      | c3           |
+------------+---------+--------------+
| [a, c, f, b]  | [4, 5, 7, 2, 5, 8]  |  [You, Me, Him] |
+------------+---------+--------------+
```

Execute the following statement:

```
select sort_array(c1),sort_array(c2),sort_array(c3) from t_array;
-- The following result is returned:
[a, b, c, f] [2, 4, 5, 5, 7, 8] [Him, You, Me]
```

1.6.6.6.23.6. POSEXPLODE

This topic describes the POSEXPLODE function and provides examples of using this function.

Syntax:

```
posexplode(ARRAY<T>)
```

Description: This function converts a given array into a table that has two columns. The first column lists subscripts of each value in the array, starting from 0. The second column lists array elements.

Parameters: ARRAY<T>: a value of the ARRAY type. Data in the array can be of any data type.

Return value: The generated table is returned.

Examples:

```
select posexplode(array('a','c','f','b'));
-- The following result is returned:
+------------+-----+
| pos        | val |
+------------+-----+
| 0          | a   |
| 1          | c   |
| 2          | f   |
| 3          | b   |
+------------+-----+
```

1.6.6.6.23.7. STRUCT

This topic describes the STRUCT function and provides examples of using this function.

Syntax:

```
STRUCT struct(value1,value2, ...)
```

Description: This function creates a value of the STRUCT type based on the given value list.

Parameters: value: a value of any data type.

Return value: A value of the STRUCT type is returned. The fields in returned results are sequentially named as col1, col2, ...

Examples:

```
select struct('a',123,'ture',56.90);
-- The following result is returned:
{col1:a, col2:123, col3:ture, col4:56.9}
```

1.6.6.6.23.8. NAMED_STRUCT

This topic describes the NAMED_STRUCT function and provides examples of using this function.

Syntax:

```
STRUCT named_struct(string name1, T1 value1, string name2, T2 value2, ...)
```

Description: This function creates a value of the STRUCT type based on the given name/value list.

Parameters:

- value: a value of any data type.
- name: the name of the field of the STRING type. This parameter is a constant.

Return value: A value of the STRUCT type is returned. Fields in the return value are sequentially named as name1, name2, ...

Examples:

```
select named_struct('user_id',10001,'user_name','bob','married','F','weight',63.50);
-- The following result is returned:
{user_id:10001, user_name:bob, married:F, weight:63.5}
```

1.6.6.6.23.9. INLINE

This topic describes the INLINE function and provides examples of using this function.

Syntax:

```
inline(array<STRUCT<f1:T1, f2:T2, ... >>)
```

Description: This function expands a given STRUCT array. Each array element is given one row and each STRUCT element corresponds to one column in each row.

Parameters: STRUCT: the values in the array, which are of any data type.

Return value: The function generated by the table is returned.

Examples:

Column information of the t_table table is t_struct struct<user_id:bigint,user_name:string,married:string,weight:double>. Data in the t_table table:

```
+-----------+
| t_struct |
+-----------+
| {user_id:10001, user_name:LiLei, married:N, weight:63.5} |
| {user_id:10002, user_name:HanMeiMei, married:Y, weight:43.5} |
+-----------+
```

Execute the following statement:

```
select inline(array(t_struct)) from t_table;
-- The following result is returned:
+------------+-----------+---------+------------+
| user_id    | user_name | married | weight     |
+------------+-----------+---------+------------+
| 10001      | LiLei     | N       | 63.5       |
| 10002      | HanMeiMei | Y       | 43.5       |
+------------+-----------+---------+------------+
```

1.6.6.6.23.10. BETWEEN AND expression

This topic describes the BETWEEN AND expression function and provides examples of using this function.

Syntax:

```
A [NOT] BETWEEN B AND C
```

If the value of A, B, or C is NULL, NULL is returned. If the value of A is greater than or equal to the value of B and less than or equal to the value of C, True is returned. Otherwise, False is returned.

Examples:

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
```

Query the data with sal greater than or equal to 1000 and less than or equal to 1500.

```
select * from emp where sal BETWEEN 1000 and 1500;
-- The following result is returned:
+-------+-------+-----+------------+------------+------------+------------+------------+
| 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  |
+-------+-------+-----+------------+------------+------------+------------+------------+
```

1.6.6.6.23.11. NVL

This topic describes the NVL function and provides an example of using this function.

Function declaration:

```
nvl(T value, T default_value)
```

Description: This function returns default_value if value is NULL. Otherwise, value is returned.

Example:

Assume that the t_data table has three columns: c1 of the STRING type, c2 of the BIGINT type, and c3 of the DATETIME type. The following data is contained in the table:

```
+----+------------+------------+
| 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 |
+----+------------+------------+
```

After the NVL function is called, the NULL value in c1 is returned as 00000, the NULL value in c2 is returned as 0, and the NULL value in c3 is returned as a hyphen (-).

```
-- 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 |
+-----+------------+-----+
```

1.6.6.6.23.12. TABLE_EXISTS

This topic describes the TABLE_EXISTS function and provides an example of using this function.

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:

```
-- Use this function in a SELECT statement.
select if(table_exists('abd'), col1, col2) from src;
```

1.6.6.6.23.13. PARTITION_EXISTS

This topic describes the PARTITION_EXISTS function and provides an example of using this function.

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. In this parameter, you must specify the values of partition key columns in a table based on the sequence of these columns. The number of values must be the same as the number 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 with ds='20190101' and hr='1' exist.
```

1.6.7. UDFs

1.6.7.1. Overview

This topic describes user-defined functions (UDFs) in MaxCompute.

MaxCompute offers a variety of built-in functions to meet your computing requirements. You can also create UDFs.

UDFs work in a way similar to built-in functions. For the mapping between Java and MaxCompute data types, see Types of parameters and returned values.

> **Note** If a UDF has the same name as a built-in function, MaxCompute executes the UDF by default. To call the built-in function, you must execute the `select ::Function name (expression) ;` statement. For example, if a UDF in MaxCompute is named CONCAT, MaxCompute calls the UDF rather than the CONCAT built-in function. To call the CONCAT built-in function, execute the `select ::concat('ab', 'c') ;` statement.

The following table describes three types of UDFs supported by MaxCompute.

UDF type	Description

UDF type	Description
UDF	User-defined scalar function. The input and output of a user-defined scalar function have a one-to-one mapping relationship. Every time a user-defined scalar function reads one row of data, one value is returned.
UDTF	User-defined table-valued function. A UDTF is used in scenarios in which multiple rows of data are returned after you call the function. Only this type of function can return multiple fields. A UDTF is not equivalent to a user-defined type (UDT).
UDAF	User-defined aggregate function. The input and output of a user-defined aggregate function have a many-to-one mapping relationship. Multiple input records are aggregated to generate one output value. This type of function can be used with the GROUP BY clause of SQL.

Note When you use a UDF in an SQL statement, the system may prompt insufficient memory. The reason is that the memory size required for the computing task exceeds the default memory size. In this case, you can execute the `set odps.sql.udf.joiner.jvm.memory=xxxx;` statement to adjust the memory size.

MaxCompute UDFs can be shared by multiple projects. UDFs in one project can be used in another project. SQL statement for cross-project UDF sharing:

```
select other_project:udf_in_other_project(arg0, arg1) as res from table_t;
```

If you use Maven, you can obtain the dependency from the Maven repository.

```
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>odps-sdk-udf</artifactId>
   <version>0.29.10-public</version>
</dependency>
```

1.6.7.2. Types of parameters and return values

This topic describes the types of parameters and return values of user-defined functions (UDFs) in MaxCompute.

Java UDFs in MaxCompute V2.0 support more basic data types in addition to BIGINT, STRING, DOUBLE, and BOOLEAN. These UDFs also support complex data types, such as ARRAY, MAP, and STRUCT, as well as Writable types.

To use a new basic data type, UDFs specify their function signatures in the following ways:

- A user-defined aggregate function (UDAF) or user-defined table-valued function (UDTF) uses the @Resolve annotation, for example, `@Resolve("smallint->varchar(10)")`.
- A user-defined scalar function uses the evaluate method. In this case, the built-in function types of MaxCompute and Java function types have a one-to-one mapping relationship.

> **Notice**
>
> - You can use `type,*` to pass any number of parameters, for example, @resolve("string,*->array<string>"). You must add Subtype after ARRAY.
> - The field name and field type of com.aliyun.odps.data.Struct cannot be reflected. Therefore, you must use the @Resolve annotation to obtain the field name and field type. In other words, to use STRUCT in a UDF, you must add the @Resolve annotation to the UDF class. This annotation affects only the overloads of parameters or return values that contain com.aliyun.odps.data.Struct.
> - Only one @Resolve annotation can be provided for the class. Therefore, only one overload with a STRUCT parameter or return value exists in a UDF.

To use a complex data type, UDFs specify their function signatures in the following ways:

- A UDTF uses the @Resolve annotation, for example, `@Resolve("array<string>,struct<a1:bigint,b1:string>,string->map<string,bigint>,struct<b1:bigint>")`.
- A user-defined scalar function uses the signature of the evaluate method to match input and output types. This involves the following mapping between MaxCompute and Java data types:
  - ARRAY in MaxCompute maps to java.util.List.
  - MAP in MaxCompute maps to java.util.Map.
  - STRUCT in MaxCompute maps to com.aliyun.odps.data.Struct.
- A UDAF uses the @Resolve annotation, for example, `@Resolve("smallint->varchar(10)")`.

The following table describes the mapping between MaxCompute and Java data types.

MaxCompute data type	Java data 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

MaxCompute data type	Java data type
STRUCT	com.aliyun.odps.data.Struct

Note

- Make sure that the input and output parameters in a UDF are of a Java type. Otherwise, error ODPS-0130071 is returned.
- Java data types and the data types of return values are objects. Java data types must start with an uppercase letter.
- The NULL value in SQL statements is represented by a NULL reference in Java. Java Primitive Type cannot have NULL values and must not be used.
- The ARRAY type in the preceding table maps java.util.List.

MaxCompute V2.0 allows you to use Writable types as parameters and return values when you define Java UDFs. The following table describes the mapping between MaxCompute data types and Java Writable types.

MaxCompute data type	Java Writable type
TINYINT	ByteWritable
SMALLINT	ShortWritable
INT	IntWritable
BIGINT	LongWritable
FLOAT	FloatWritable
DOUBLE	DoubleWritable
DECIMAL	BigDecimalWritable
BOOLEAN	BooleanWritable
STRING	Text
VARCHAR	VarcharWritable
BINARY	BytesWritable
DATETIME	DatetimeWritable
TIMESTAMP	TimestampWritable
INTERVAL_YEAR_MONTH	IntervalYearMonthWritable
INTERVAL_DAY_TIME	IntervalDayTimeWritable
ARRAY	N/A
MAP	N/A
STRUCT	N/A

1.6.7.3. UDF

This topic describes user-defined scalar functions (UDFs) and provides examples of using UDFs.

UDFs must inherit the class com.aliyun.odps.udf.UDF and implement the evaluate method of the class. The evaluate method must be a non-static public method, and the types of parameters and return values are used as the UDF signature in SQL statements. You can implement multiple evaluate methods in a UDF. To call a UDF, the framework matches the correct evaluate method based on the parameter type called by the UDF.

> **Note**
>
> - We recommend that you do not use the classes that have the same name but different function logic in different JAR packages. For example, in UDF(UDAF/UDTF): udf1, udf2, the JAR package of udf1 is udf1.jar and the JAR package of udf2 is udf2.jar. Assume that the two JAR packages contain the com.aliyun.UserFunction.class class. If you use the two UDFs in the same SQL statement, the system randomly loads the class contained in one of the two JAR packages. This may cause the UDFs to be executed in different ways or even cause a compilation failure.
> - If you execute an SQL statement that has two UDFs, the UDFs are not isolated from each other. This is because the UDFs share the same classpath. If the resources referenced by the UDFs contain the same class, the class that the classloader attempts to load is uncertain. To avoid this issue, make sure that the resources referenced by the two UDFs do not contain the same class.

You can use `void setup(ExecutionContext ctx)` to initialize a UDF and use `void close()` to terminate a UDF.

The following example shows how to implement a UDF:

```
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();
  }
}
```

The following example shows how to implement Concat by using a Writable type:

```
package com.aliyun.odps.udf.example;
import com.aliyun.odps.io.Text;
import com.aliyun.odps.udf.UDF;
public class MyConcat extends UDF {
  private Text ret = new Text();
  public Text evaluate(Text a, Text b) {
    if (a == null || b == null) {
      return null;
    }
    ret.clear();
    ret.append(a.getBytes(), 0, a.getLength());
    ret.append(b.getBytes(), 0, b.getLength());
    return ret;
  }
}
```

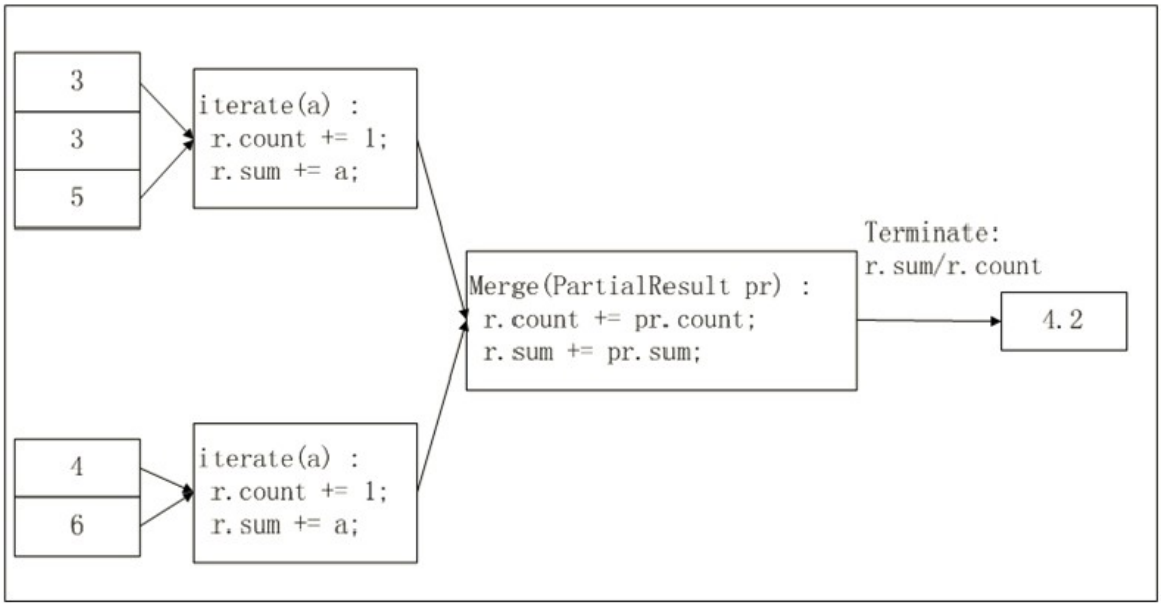
1.6.7.4. UDAF

This topic describes user-defined aggregate functions (UDAFs) and provides examples of using UDAFs.

UDAFs must inherit the com.aliyun.odps.udf.Aggregator class and implement the following methods:

```
import com.aliyun.odps.udf.ContextFunction;
import com.aliyun.odps.udf.ExecutionContext;
import com.aliyun.odps.udf.UDFException;
public abstract class Aggregator implements ContextFunction {
    @Override
    public void setup(ExecutionContext ctx) throws UDFException {
    }
    @Override
    public void close() throws UDFException {
    }
    /**
     * Create an aggregation buffer * @return Writable aggregation buffer
     */
    abstract public Writable newBuffer();
    /**
     * @param buffer: aggregation buffer * @param args: a parameter specified to call a UDAF in SQL.
It cannot be NULL, but the values in args can be NULL, which indicates that the input data is NULL *
@throws UDFException.
     */
    abstract public void iterate(Writable buffer, Writable[] args) throws UDFException;
    /**
     * Generate the final result * @param buffer * @return Final result of Object UDAF * @throws UDF
Exception.
     */
    abstract public Writable terminate(Writable buffer) throws UDFException;
    abstract public void merge(Writable buffer, Writable partial) throws UDFException;
}
```

The most important methods are iterate, merge, and terminate because the main logic of UDAFs relies on these methods. In addition, you must implement the user-defined Writable buffer. The following figure illustrates the implementation logic and procedure to calculate the avg value by using a MaxCompute UDAF.



In the preceding figure, the input data is sliced based on the specified size. The size of each slice is suitable for a worker to complete the calculation in a specified time. The slice size must be manually configured.

The calculation procedure of a UDAF involves two steps:

- Step 1: Each worker counts the data quantity and total sum in a slice. You can consider the data quantity and total sum in each slice as an intermediate result.
- Step 2: Each worker gathers the information about each slice generated in Step 1. In the final output, r.sum/r.count is the average value of all input data.

Sample code for a UDAF that is used to calculate the average value:

```
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 = (DoubleWritable) 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;
  }
}
```

Sample code for implementing Concat by using Writable types:

```
package com.aliyun.odps.udf.example;
import com.aliyun.odps.io.Text;
import com.aliyun.odps.udf.UDF;
public class MyConcat extends UDF {
  private Text ret = new Text();
  public Text evaluate(Text a, Text b) {
    if (a == null || b == null) {
      return null;
    }
    ret.clear();
    ret.append(a.getBytes(), 0, a.getLength());
    ret.append(b.getBytes(), 0, b.getLength());
    return ret;
  }
}
```

1.6.7.5. UDTFs

1.6.7.5.1. Overview

This topic describes user-defined table-valued functions (UDTFs) and provides examples of using UDTFs.

UDTFs must inherit the com.aliyun.odps.udf.UDTF class and implement four methods. The following table describes the methods.

Method	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 for 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 input parameters of the UDTF specified in the SQL statement. The input parameters are passed in as Object[], and the results are returned by using the forward function. You must call forward in the process function to determine the output data.
public void close() throws UDFException	The method for terminating a UDTF. The framework calls this method only once. The method is called after the last record is processed.
public void forward(Object ...o) throws UDFException	You can call the forward method to return data. One record is returned each time forward is called. The record corresponds to the column specified by the AS clause in the SQL statement of the 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);
       }
     }
   }
```

Assume that you create a UDTF in MaxCompute with the registered function name of user_udtf. Execute the following statement to use the UDTF:

```
select user_udtf(col0, col1) as (c0, c1) from my_table;
```

The following example shows the values of col0 and col1 in my_table.

```
+------+------+
| col0 | col1 |
+------+------+
| A B  | 1    |
| C D  | 2    |
+------+------+
```

The following example shows the execution result of the SELECT statement.

```
+----+----+
| c0 | c1 |
+----+----+
| A  | 1  |
| B  | 1  |
| C  | 2  |
| D  | 2  |
+----+----+
```

1.6.7.5.2. Examples of using UDTFs

This topic provides examples of using UDTFs.

Use a UDTF to read resources in MaxCompute

You can use a UDTF to read resources in MaxCompute. Example:

1. Compile a UDTF program. After the compilation is successful, export the JAR package. In this example, the JAR package is named udtfexample1.jar.

   ```
   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 {
  ExecutionContext 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 + "|tableResource1RecordCoun
t="
     + 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 the my_udtf 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. Run this UDTF.

```
select mp_udtf("10","20") as (a, b, fileResourceLineCount) from table_resource1;
```

Returned result:

```
+-------+------------+-------+
| a | b      | fileResourceLineCount |
+-------+------------+-------+
| 10    | 2          | fileResourceLineCount=3|tableResource1RecordCount=0|tableResource2RecordC
ount=0 |
| 10    | 2          | fileResourceLineCount=3|tableResource1RecordCount=0|tableResource2RecordC
ount=0 |
+-------+------------+-------+
```

Use complex data types in a UDF

The following example defines a UDF with three overloads. The first overload uses the ARRAY type, the second overload uses the MAP type, and the third overload uses the STRUCT type. The third overload uses STRUCT as the data type of parameters or returned values. Therefore, the @Resolve annotation must be added to the UDF class to specify the STRUCT type.

```
import com.aliyun.odps.udf.UDF;
import com.aliyun.odps.udf.annotation.Resolve;
@Resolve("struct,string->string")
public class UdfArray extends UDF {
    public String evaluate(List vals, Long len) {
        return vals.get(len.intValue());
    }
    public String evaluate(Map map, String key) {
        return map.get(key);
    }
    public String evaluate(Struct struct, String key) {
        return struct.getFieldValue("a") + key;
    }
}
```

You can pass complex data types into a UDF.

```
create function my_index as 'UdfArray' using 'myjar.jar';
select id, my_index(array('red', 'yellow', 'green'), colorOrdinal) as color_name from co
```

Use Hive UDFs

MaxCompute V2.0 supports Hive UDFs. Some Hive UDFs and UDTFs can be directly used in MaxCompute.

> **Note** MaxCompute V2.0 is compatible with Hive 2.1.0 and Hadoop 2.7.2. If you develop a UDF by using other versions of Hive or Hadoop, you may need to recompile the UDF on the compatible Hive or Hadoop version.

The following example shows how to use a Hive UDF in MaxCompute:

```
package com.aliyun.odps.compiler.hive;
import org.apache.hadoop.hive.ql.exec.UDFArgumentException;
import org.apache.hadoop.hive.ql.metadata.HiveException;
import org.apache.hadoop.hive.ql.udf.generic.GenericUDF;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspectorFactory;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
public class Collect extends GenericUDF {
  @Override
  public ObjectInspector initialize(ObjectInspector[] objectInspectors) throws UDFArgumentException
{
    if (objectInspectors.length == 0) {
      throw new UDFArgumentException("Collect: input args should >= 1");
    }
    for (int i = 1; i < objectInspectors.length; i++) {
      if (objectInspectors[i] ! = objectInspectors[0]) {
        throw new UDFArgumentException("Collect: input oi should be the same for all args");
      }
    }
    return ObjectInspectorFactory.getStandardListObjectInspector(objectInspectors[0]);
  }
  @Override
  public Object evaluate(DeferredObject[] deferredObjects) throws HiveException {
    List<Object> objectList = new ArrayList<>(deferredObjects.length);
    for (DeferredObject deferredObject : deferredObjects) {
      objectList.add(deferredObject.get());
    }
    return objectList;
  }
  @Override
  public String getDisplayString(String[] strings) {
    return "Collect";
  }
}
```

The Hive UDF in the example packages any type and number of parameters into arrays. In this example, the output JAR package is named test.jar.

```
-- Add a resource.
Add jar test.jar;
-- Create a Hive UDF.
CREATE FUNCTION hive_collect as 'com.aliyun.odps.compiler.hive.Collect' using 'test.jar';
-- Use the Hive UDF.
set odps.sql.hive.compatible=true;
select hive_collect(4y,5y,6y) from dual;
```

Returned result:

```
+------+
| _c0  |
+------+
| [4, 5, 6] |
+------+
```

Note Take note of the following items when you use Hive UDFs that are compatible with MaxCompute:

- Specify the JAR package when you create a Hive UDF. MaxCompute cannot automatically add all JAR packages to the classpath. The `add jar` command in MaxCompute creates a permanent resource in the project.
- Insert `set odps.sql.hive.compatible=true;` before an SQL statement, and commit and execute the SET statement with the SQL statement.
- Pay attention to the limits imposed by the Java sandbox on MaxCompute.

1.6.7.6. Python UDFs

1.6.7.6.1. Limits

This topic describes limits on Python user-defined functions (UDFs).

MaxCompute uses Python 2.7. User code runs in a sandbox, which is a limited environment. In this environment, the following operations are prohibited:

- Read and write local files.
- Start subprocesses.
- Start threads.
- Initiate socket-based communication.
- Call other systems.

Due to these limits, the code that you upload must be implemented by standard Python, and C extension modules are not allowed in your code.

In addition, the modules that involve the preceding operations are disabled in the Python standard library. The following modules are available in the standard library:

- The modules that are implemented by standard Python.
- The following C extension modules:
  - 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

- Some modules with limited functionality. For example, the size of data that your code can write into standard output sys.stdout and standard error output sys.stderr in a sandbox is limited to 20 KB. Characters that exceed this limit are ignored.

1.6.7.6.2. Third-party libraries

This topic describes third-party libraries for Python user-defined functions (UDFs).

Common third-party libraries such as NumPy are installed in the runtime environment to supplement the standard library.

> **Warning** The use of third-party libraries is also subject to limits. For example, when you use a third-party library, you are not allowed to access local data and you can use only limited network I/O resources. The related APIs in the third-party libraries are disabled.

1.6.7.6.3. Types of parameters and return values

This topic describes the types of parameters and return values of Python user-defined functions (UDFs).

The types of parameters and return values are specified by using the following method:

```
@odps.udf.annotate(signature)
```

Python UDFs support MaxCompute SQL data types, such as BIGINT, STRING, DOUBLE, BOOLEAN, DATETIME, DECIMAL, complex data types, and nested complex data types. Complex data types include ARRAY, MAP, and STRUCT. Before you execute an SQL statement, the types of parameters and return values of all functions must be determined. Python is a dynamically typed language. You must add decorators to the UDF class to specify the function signature.

The function signature is specified by a string. The following syntax is supported:

```
arg_type_list '->' type_list
arg_type_list: type_list | '*' | '' | 'char(n)' | 'varchar(n)'
type_list: [type_list ','] type
type: 'bigint' | 'string' | 'double' | 'boolean' | 'datetime' | 'float' | 'binary' | 'date' | 'decim
al' | 'decimal(precision,scale)'
```

Note

- The part to the left of the arrow indicates the types of parameters. The part to the right of the arrow indicates the types of return values.
- The return value of a UDTF can contain multiple columns, while the return value of a UDF or UDAF contains only one column.
- An asterisk (*) represents variable-length parameters. If variable-length parameters are used, UDFs, UDTFs, and UDAFs can match input parameters of any type.
- In a project that uses the MaxCompute V2.0 data type edition, you can set the decimal parameter to precision or scale.
- The return value cannot be of the CHAR or VARCHAR type.

The following example shows a valid signature:

```
'bigint,double->string'            # Parameters are of the BIGINT and DOUBLE types, and return value
s are of the STRING type.
'bigint,boolean->string,datetime'  # UDTF parameters are of the BIGINT and BOOLEAN types, and return
values are of the STRING and DATETIME types.
'*->string'                        # Input parameters are variable-length parameters and can be of a
ny type, and return values are of the STRING type.
'->double'                         # Parameters are left empty, and return values are of the DOUBLE
type.
'array<bigint>->struct<x:string, y:int>' # Parameters are of the ARRAY<BIGINT> type, and return valu
es are of the STRUCT<x:STRING, y:BIGINT> type.
'->map<bigint, string>'            # Parameters are left empty, and return values are of the MAP<BIG
INT, STRING> type.
```

If an invalid signature is found during query parsing, an error is reported, and the execution of the function is prohibited. During the execution of the function, the UDF parameters of the type specified by the function signature are passed. The return values must be of the same type as those specified by the function signature. Otherwise, an error is reported. The following table describes the mappings between Python data types and MaxCompute SQL data types.

MaxCompute SQL data type	Python data type
BIGINT	INT
STRING	STR
DOUBLE	FLOAT

MaxCompute SQL data type	Python data type
BOOLEAN	BOOL
DATETIME	INT
FLOAT	FLOAT
CHAR	STR
VARCHAR	STR
BINARY	BYTEARRAY
DATE	INT
DECIMAL	DECIMAL.DECIMAL
ARRAY	LIST
MAP	DICT
STRUCT	COLLECTIONS.NAMEDTUPLE

Note

- A value of the DATETIME type is converted to a value of the INT type. The value of the INT type follows the UNIX format, which is the number of milliseconds that have elapsed since 00:00:00 Thursday, January 1, 1970. You can process data of the DATETIME type by using the DATETIME module in the Python standard library.
- silent is added to `odps.udf.int(value,[silent=True])`. If silent is set to True and value cannot be converted to the INT type, None is returned, and no error is reported.
- The NULL value corresponds to None in Python.

1.6.7.6.4. UDF

This topic describes user-defined scalar functions (UDFs) and provides an example of using UDFs.

To implement Python UDFs, you must define a new-style class and implement the evaluate method.

```
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
```

Note A Python UDF must have its signature specified by using annotate.

1.6.7.6.5. UDAF

This topic describes user-defined aggregate functions (UDAFs) and provides an example of using UDAFs.

- class odps.udf.BaseUDAF: is inherited to implement Python UDAFs.
- BaseUDAF.new_buffer(): returns buffer of the intermediate value of a UDAF. buffer must be a marshallable object (such as LIST or DICT), and the buffer size cannot increase with the data volume. Under extreme circumstances, the buffer size cannot exceed 2 MB after the marshal operation.
- BaseUDAF.iterate(buffer[, args, ...]): aggregates args to buffer of the intermediate value.
- BaseUDAF.merge(buffer, pbuffer): aggregates buffer of two intermediate values. It merges pbuffer to buffer.
- BaseUDAF.terminate(buffer): converts buffer of the intermediate value to a value of a basic data type in MaxCompute SQL.

Example of using a UDAF to obtain the average value:

```
@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]
```

1.6.7.6.6. UDTF

This topic describes user-defined table-valued functions (UDTFs) and provides an example of using UDTFs.

- class odps.udf.BaseUDTF: the base class of Python UDTFs. This class can be inherited to implement the PROCESS and CLOSE methods.
- BaseUDTF.__init__(): the initialization method. To implement this method for a derived class, you must call the super(BaseUDTF, self).__init__() initialization method of the base class at the beginning of code execution. Throughout the lifecycle of a UDTF, the INIT method is called only once. It is called only before the first record is processed. If a UDTF needs to save internal states, all states can be initialized by using this method.
- BaseUDTF.process([args, ...]): is called by the MaxCompute SQL framework. The PROCESS method is called for each record in SQL. The parameters in the PROCESS method are the UDTF input parameters that are specified in SQL statements.
- BaseUDTF.forward([args, ...]): the UDTF output method. This method is called by user code. One record is generated each time the FORWARD method is called. The parameters in the FORWARD method are the UDTF output parameters that are specified in SQL statements.
- BaseUDTF.close(): the UDTF termination method. This method is called only once by the MaxCompute SQL framework. It is called only 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):
   # Use commas (,) to separate the string into multiple records.
   def process(self, arg):
       props = arg.split(',')
       for p in props:
           self.forward(p)
```

Note The types of parameters and the return value of a Python UDTF can be specified without the need to use annotate to specify the UDTF signature. This way, the function can match input parameters of any type in SQL. However, the type of the return value cannot be deduced. All output parameters are considered to be of the STRING type. Therefore, when FORWARD is called, all output values must be converted to the STRING type.

1.6.7.6.7. Resource reference

This topic describes how to reference resources in Python user-defined functions (UDFs) and provides an example for reference.

You can reference file and table resources in Python UDFs by using the odps.distcache module.

The following sample code shows the syntax that is used to reference file resources:

```
odps.distcache.get_cache_file(resource_name)
-- Return the content of a specified resource.
```

The resource_name parameter is a string that corresponds to the name of an existing file resource in the current project. If the resource name is invalid or the file resource does not exist, an error is returned.

The return value is a file-like object. If this object is no longer used, the caller must call the `close` method to release the open file resource.

Example:

```
@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)
```

The following sample code shows the syntax that is used to reference table resources:

```
odps.distcache.get_cache_table(resource_name)
```

The resource_name parameter is a string that corresponds to the name of an existing table resource in the current project. If the resource name is invalid or the table resource does not exist, an error is returned.

The return value is of the GENERATOR type. The caller traverses the table to obtain the content. A record of the ARRAY type is obtained each time the caller traverses the table.

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)
```

1.6.7.7. Python 3 UDF

This topic describes the precautions when you use Python 3 user-defined functions (UDFs).

Python Software Foundation announces the End of Life (EOL) for Python 2. Due to this reason, MaxCompute supports Python 3 and uses Python 3.7.3.

Limits

Python 3 is not compatible with Python 2. You can select Python 2 or Python 3 to execute an SQL statement. However, you cannot use Python 2 code and Python 3 code in a single SQL statement at the same time.

Enable Python 3

You can enable Python 3 only for jobs. To enable Python 3 for a job, add the following command before the required SQL statement and execute them together:

```
set odps.sql.python.version=cp37;
```

Use third-party libraries

MaxCompute Python UDFs support third-party libraries. The third-party library NumPy is installed in the Python 2 runtime environment to supplement the standard library.

NumPy is not installed in the Python 3 runtime environment in MaxCompute. To use a NumPy UDF, you must manually upload a NumPy wheel package. If you download a NumPy wheel package from Python Package Index (PyPI), the package name is numpy-<Version>-cp37-cp37m-manylinux1_x86_64.whl.

Port Python 2 UDFs

Python Software Foundation announces the EOL for Python 2. Therefore, we recommend that you port Python 2 UDFs. The method used to port Python 2 UDFs varies based on projects.

- In a new project or an existing project where no Python UDFs exist, we recommend that you use Python 3 to

write all Python UDFs.

- Exercise caution when you enable Python 3 in an existing project where a large number of Python 2 UDFs exist. This is because you cannot use Python 2 code and Python 3 code in a single SQL statement at the same time. If you plan to gradually replace Python 2 UDFs with Python 3 UDFs, use the following methods:
  - Use Python 3 to write new UDFs and enable Python 3 for new jobs.
  - Rewrite Python 2 UDFs to make them compatible with both Python 2 and Python 3. For more information about how to rewrite UDFs, see Porting Python 2 code to Python 3.

> **Note** If you write a public UDF that needs to be shared among multiple projects, we recommend that you write the UDF to be compatible with both Python 2 and Python 3.

1.6.7.8. SQL Function

This topic describes how to use SQL functions. SQL functions allow you to reference SQL user-defined functions (UDFs) in SQL scripts.

Background information

You can use SQL functions of MaxCompute to resolve the following issues:

- Generally, a large amount of similar code exists, which is inconvenient to maintain and prone to errors. To use UDFs, you must develop code, compile the code in Java, and then create resources and functions. The process is complicated. In addition, UDFs cannot catch up with built-in functions in terms of performance. Example:

```
SELECT
    NVL(STR_TO_MAP(GET_JSON_OBJECT(col, '$.key1')), 'default') AS key1,
    NVL(STR_TO_MAP(GET_JSON_OBJECT(col, '$.key2')), 'default') AS key2,
    ...
    NVL(STR_TO_MAP(GET_JSON_OBJECT(col, '$.keyN')), 'default') AS keyN
FROM t;
```

- One function can be transferred as a parameter to another only by using code similar to lambda expressions in Java.

Feature description

As a type of UDFs, SQL functions allow you to create UDFs in SQL and use function type parameters and anonymous functions. This way, you can define business logic more flexibly. You can use SQL functions to simplify feature implementation and improve code reuse. SQL functions provide the following features:

- SQL functions allow you to reference and call SQL UDFs in SQL scripts.
- You can specify built-in functions, UDFs, or SQL functions in function type parameters when you call SQL functions.
- You can specify anonymous functions in function type parameters when you call SQL functions.

Create a permanent SQL function

After you create a permanent SQL function and store it in the metadata system, all query statements can reference this function. To reference SQL functions, you must have function-level permissions. The following example shows the syntax:

```
CREATE SQL FUNCTION function_name(@parameter_in1 datatype[, @parameter_in2 datatype...])
[RETURNS @parameter_out datatype]
AS [BEGIN]
function_expression
[END];
```

Parameters:

- function_name: the name of the SQL function that you want to create. Each function name must be unique and can be registered only once. The SQL function names cannot be the same as those of built-in functions.
- parameter_in: the input parameters of the SQL function.
- RETURNS: the variable to be returned by the SQL function. If you do not specify RETURNS, the value of function_name is returned.
- parameter_out: the output parameters of the SQL function.
- function_expression: the expression of the SQL function.

Example:

```
CREATE SQL FUNCTION MY_ADD(@a BIGINT) AS @a + 1;
```

In the preceding example, @a + 1 indicates the implementation logic of the SQL function. You can write it as an expression to specify a built-in operator, built-in function, or UDF.

If the implementation logic is complex, you can write multiple SQL statements and enclose them with BEGIN and END. RETURNS specifies the variable to be returned by the SQL function. If you do not specify RETURNS, the value of function_name is returned.

Example:

```
CREATE SQL FUNCTION MY_SUM(@a BIGINT, @b BIGINT, @c BIGINT) RETURNS @my_sum BIGINT
AS BEGIN
    @temp := @a + @b;
    @my_sum := @temp + @c;
END;
```

Query an SQL function

You can query an SQL function in the same way as querying a Java or Python UDF. Example:

```
DESC FUNCTION my_add;
```

Delete an SQL function

You can delete an SQL function in the same way as deleting a Java or Python UDF. Example:

```
DROP FUNCTION my_add;
```

Call an SQL function

You can call an SQL function in the same way as calling an existing built-in function. Example:

```
SELECT my_sum(col1, col2 ,col3) from t;
```

Use temporary SQL functions

You can use temporary SQL functions if you no longer need to save SQL functions to the metadata system of MaxCompute. The temporary SQL functions are valid only in the current script. Example:

```
FUNCTION MY_ADD(@a BIGINT) AS @a + 1;
SELECT MY_ADD(key), MY_ADD(value) FROM src;
```

Use function type parameters

You can specify built-in functions, UDFs, or SQL functions in function type parameters when you call SQL functions. Example:

```
FUNCTION ADD(@a BIGINT) AS @a + 1;
FUNCTION OP(@a BIGINT, @fun FUNCTION (BIGINT) RETURNS BIGINT) AS @fun(@a);
SELECT OP(key, ADD), OP(key, ABS) FROM VALUES (1),(2) AS t (key);
```

Returned results:

```
+------------+------------+
| _c0        | _c1        |
+------------+------------+
| 2          | 1          |
| 3          | 2          |
+------------+------------+
```

In this example, two input parameters are specified for the OP function. The @a parameter specifies a value of the BIGINT type. The @fun parameter specifies a function whose input and output parameters are both of the BIGINT type. The OP function transfers @a to the function that is specified by @fun and then to the ADD and ABS functions for processing.

Use anonymous functions

You can specify anonymous functions in function type parameters when you call SQL functions. Example:

```
FUNCTION OP(@a BIGINT, @fun FUNCTION (BIGINT) RETURNS BIGINT) AS @fun(@a);
SELECT OP(key, FUNCTION (@a) AS @a + 1) FROM VALUES (1),(2) AS t (key);
```

In this example, FUNCTION (@a) AS @a + 1 is an anonymous function. You do not need to specify a data type for the input parameter @a. The compiler will infer the data type of @a based on the parameter definition of the OP function.

1.6.7.9. Embedded UDF

1.6.7.9.1. Background information

This topic describes the background information of code-embedded user-defined functions (UDFs).

The code-embedded UDFs of MaxCompute resolve the following issues in code implementation and maintenance:

- Complicated code implementation: After you create UDFs and develop code, you must compile the code in Java and create resources and functions.
- Inconvenient code maintenance: You cannot directly view the implementation logic of the UDFs that are referenced in SQL scripts or obtain the source code of JAR packages.
- Poor code readability: To implement Java library functions by using user-defined types (UDTs), you must write Java code as expressions in long code lines. In addition, you may fail to write some Java code as expressions. Example:

  ```
  Foo f = new Foo();
  f.execute();
  f.getResult();
  ```

1.6.7.9.2. Feature summary

This topic describes the features of code-embedded user-defined functions (UDFs).

Code-embedded UDFs allow you to embed Java or Python code into SQL scripts. When you compile a script, Janino-compiler identifies and extracts the embedded code, compiles the code in Java, and then dynamically generates resources and creates temporary functions.

You can place SQL scripts and third-party code lines in the same source code file. This simplifies the usage of user-defined types (UDTs) or UDFs and facilitates routine development and maintenance.

1.6.7.9.3. Limits

This topic describes the limits on code-embedded user-defined functions (UDFs).

You can use only Janino-compiler to compile embedded Java code. The syntax of the embedded Java code must be a subset of the standard JDK syntax. Embedded Java code has the following limits:

- Lambda expressions are not supported.
- You cannot specify multiple types of exceptions in a single catch block, for example, catch(Exception1 | Exception2 e).
- Generics cannot be automatically inferred, for example, Map map = new HashMap<>();.
- Expressions for type parameter inference are ignored and you must use cast expressions to specify the parameter type, for example, (String) myMap.get(key).
- Assertions are forcibly enabled, even if the -ea option of the Java Virtual Machine (JVM) is used.
- Code that is programmed in versions later than Java 8 is not supported.

1.6.7.9.4. Examples

This topic describes how to use code-embedded UDFs and provides examples for reference.

Reference embedded code in a UDT

Example:

```
SELECT
  s,
  com.mypackage.Foo.extractNumber(s)
FROM VALUES ('abc123def'),('apple') AS t(s);
#CODE ('lang'='JAVA')
package com.mypackage;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class Foo {
  final static Pattern compile = Pattern.compile(". *?([ 0-9]+).*");
  public static String extractNumber(String input) {
    final Matcher m = compile.matcher(input);
    if (m.find()) {
      return m.group(1);
    }
    return null;
  }
}
#END CODE;
```

Description:

- #CODE indicates the beginning of the embedded code block. #END CODE indicates the end of the embedded code block. In this example, the embedded code block is placed at the end of the script and applies to the whole script.
- 'lang'='JAVA' indicates that the embedded code is in Java. JAVA can be replaced with PYTHON if you compile

code in Python.

- You can use UDT syntax in the SQL script to call Foo.extractNumber.

Define and call a Java code-embedded UDF

Example:

```
CREATE TEMPORARY FUNCTION foo AS 'com.mypackage.Reverse' USING
#CODE ('lang'='JAVA')
package com.mypackage;
import com.aliyun.odps.udf.UDF;
public class Reverse extends UDF {
  public String evaluate(String input) {
    if (input == null) return null;
    StringBuilder ret = new StringBuilder();
    for (int i = input.toCharArray().length - 1; i >= 0; i--) {
      ret.append(input.toCharArray()[i]);
    }
    return ret.toString();
  }
}
#END CODE;
SELECT foo('abdc');
```

Description:

- You can place the embedded code block next to USING or at the end of the script. If you place it next to USING, it applies only to the `CREATE TEMPORARY FUNCTION` statement.
- The function created by `CREATE TEMPORARY FUNCTION` is a temporary function. This temporary function is executed only during the current execution process and is not stored in the MaxCompute meta system.

Define and call a Java code-embedded UDTF

Example:

```
CREATE TEMPORARY FUNCTION foo AS 'com.mypackage.Reverse' USING
#CODE ('lang'='JAVA', 'filename'='embedded.jar')
package com.mypackage;
import com.aliyun.odps.udf.UDTF;
import com.aliyun.odps.udf.UDFException;
import com.aliyun.odps.udf.annotation.Resolve;
@Resolve({"string->string,string"})
public class Reverse extends UDTF {
  @Override
  public void process(Object[] objects) throws UDFException {
    String str = (String) objects[0];
    String[] split = str.split(",");
    forward(split[0], split[1]);
  }
}
#END CODE;
SELECT foo('ab,dc') AS (a,b);
```

The return value of @Resolve must be of the string[] type. However, Janino-compiler cannot identify "string->string,string" in embedded code as string[]. To enable Janino-compiler to identify "string->string,string" as string[], "string->string,string" must be enclosed in braces {}. If you create a Java UDTF by using a common method, braces {} are not required.

Define and call a Python code-embedded UDF

Example:

```
CREATE TEMPORARY FUNCTION foo AS 'embedded.UDFTest' USING
#CODE ('lang'='PYTHON', 'filename'='embedded')
from odps.udf import annotate
@annotate("bigint->bigint")
class UDFTest(object):
  def evaluate(self, a):
    return a * a
#END CODE;
SELECT foo(4);
```

Description:

- The indentation of Python code must comply with the specifications of the Python language.
- When you create a Python UDF, the class name that follows the AS clause must contain the file name of the Python source code. You can use `'filename'='embedded'` to specify a virtual file name.

1.6.8. UDTs

1.6.8.1. Scenarios and limits

This topic describes the scenarios and limits of user-defined types (UDTs).

MaxCompute introduces UDTs based on the new-generation SQL engine. UDTs allow you to reference classes or objects of third-party programming languages in SQL statements to call methods or obtain data.

UDTs are suitable for the following scenarios:

- You want to use some features that are not provided by MaxCompute but can be implemented in other programming languages.

  For example, to implement some features, you need only to call built-in Java classes once. However, MaxCompute does not provide methods to implement these features. If you use user-defined functions (UDFs) to run these tasks, the procedure is complex.
- You want to call a third-party library in SQL statements to implement the related features. In this scenario, UDFs allow you to directly use a function provided by a third-party library in a SQL statement, instead of wrapping the function inside a UDF.
- You want to directly call the source code of a third-party programming language in SQL statements. The SELECT TRANSFORM statement allows you to write scripts to SQL statements. This improves readability and facilitates code maintenance. For some programming languages, such as Java, the source code can be executed only after it is compiled. You can use UDTs to reference objects and classes of these languages in SQL statements.

Limits

- UDTs support only Java. By default, all classes of SDK for Java can be referenced by UDTs.

  > **Note** JDK 1.8 is used. A version later than JDK 1.8 may not be supported.
- All operators use the semantics of MaxCompute SQL instead of UDTs.
- UDTs cannot be used as shuffle keys in clauses, such as JOIN, GROUP BY, DISTRIBUTE BY, SORT BY, ORDER BY, or CLUSTER BY.
- DDL statements do not support UDTs. You cannot create tables that contain UDT objects. The final output cannot be UDT types.

1.6.8.2. Feature summary

This topic describes the features of user-defined types (UDTs) and provides examples for reference.

UDTs supported by many SQL engines are similar to the STRUCT type in MaxCompute. UDTs supported by MaxCompute are similar to the CREATE TYPE statement. A UDT contains both fields and methods. You do not need to use data definition language (DDL) statements to define new data types in MaxCompute. Instead, MaxCompute allows you to reference new data types directly in SQL statements. The following examples show how to use UDTs.

For example, to call the java.lang package in SQL statements, you can use one of the following methods:

- Use UDTs to call the java.lang package

```
-- Enable new data types. A new type of INTEGER (INT) is used in this example.
set odps.sql.type.system.odps2=true;
SELECT java.lang.Integer.MAX_VALUE;
```

  Based on Java conventions, you can also omit the java.lang package from the preceding statement and use the following statement:

```
set odps.sql.type.system.odps2=true;
SELECT Integer.MAX_VALUE;
```

  Returned results:

```
+------------+
| max_value |
+------------+
| 2147483647 |
+------------+
```

- Use user-defined functions (UDFs) to call the java.lang package
  i. 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;
  }
}
```

  ii. Compile the UDF into a JAR package, upload the 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';
```

  iii. Call the function in the SQL statement.

```
select integer_max_value();
```

In this example, UDTs simplify the procedure for you to use other programming languages to extend SQL features.

1.6.8.3. Implementation principles and feature description

This topic describes the implementation principles of user-defined types (UDTs) and their features.

Implementation principles

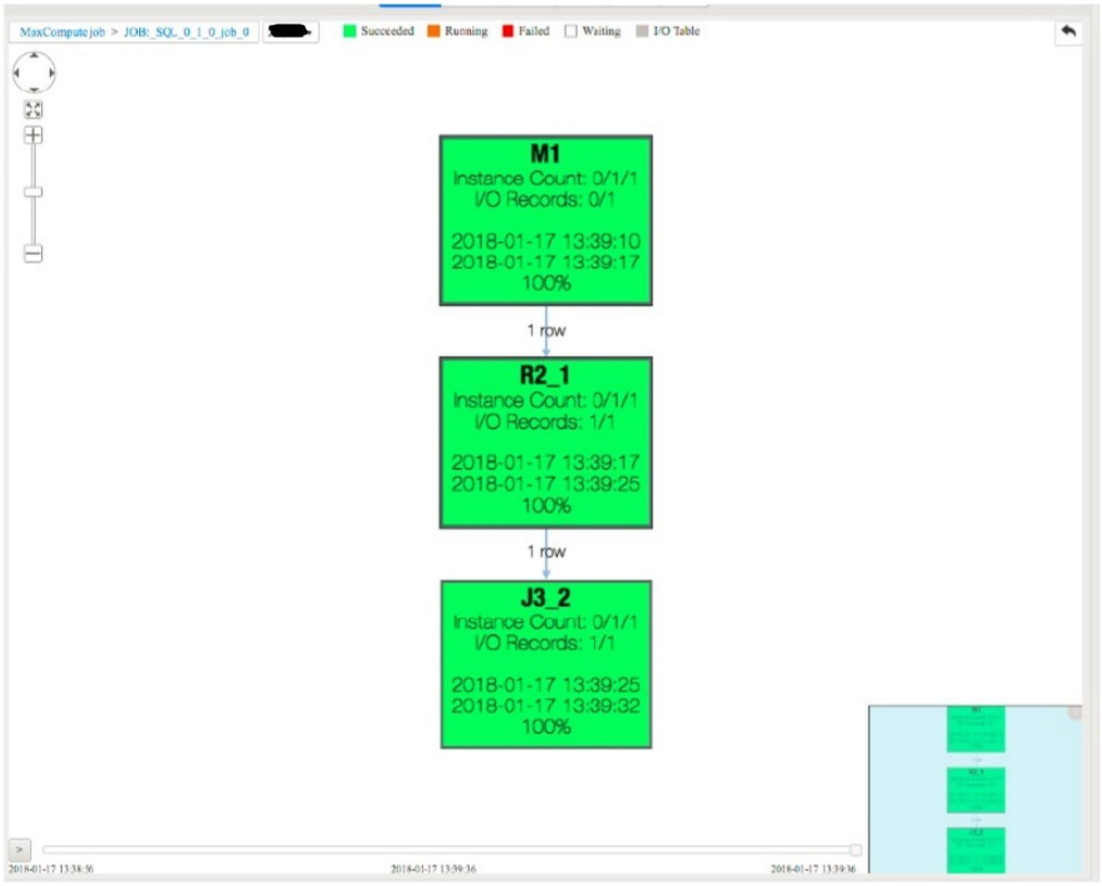
The following example shows how to run a UDT:

```
-- Sample data.
@table1 := select * from values ('100000000000000000000') as t(x);
@table2 := select * from values (100L) as t(y);
-- Code logic.
-- Create an object by using the new method.
@a := select new java.math.BigInteger(x) x from @table1;
-- Call a static method.
@b := select java.math.BigInteger.valueOf(y) y from @table2;
-- Call an instance method.
select /*+mapjoin(b)*/ x.add(y).toString() from @a a join @b b;
```

Returned results:

```
10000000000000000100
```

The following figure shows the process.



This UDT has three stages: M1, R2, and J3. If a JOIN operation is used in MapReduce, data must be reshuffled. As a result, data is processed at multiple stages. The processes and physical machines that process data vary at different stages.

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 separately called at the J3 stage. These methods are called at different stages and executed in different processes and on different physical machines. UDTs encapsulate these stages to achieve an effect similar to all the stages being implemented on the same Java Virtual Machine (JVM).

The preceding example shows that the result of a subquery supports UDT columns. The x column retrieved by variable a is of the java.math.BigInteger type rather than a built-in type. You can transfer the UDT data to another operator and then call its method. You can also use the UDT data in a data shuffle.

Feature description

- UDTs support only Java. By default, all classes of SDK for Java can be referenced by UDTs.
- 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`.

    > **Note** This flag works the same as the flag used to specify resources in the SELECT TRANSFORM statement. Therefore, this flag controls two features.

    ```
    set odps.sql.type.system.odps2=true;
    set odps.sql.session.resources=odps-test.jar;
    -- Specify the JAR package that you want to reference. You must upload the package to your p
    roject in advance.
    select new com.aliyun.odps.test.IntegerMaxValue().evaluate();
    ```

  - odps.sql.session.java.imports: specifies the default Java package. You can specify multiple packages and separate them with commas (,). This flag is similar to the IMPORT statement in Java. You can specify a class path, such as *java.math.BigInteger*, or use asterisks (*). static import is not supported.

    ```
    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 JAR package.
    select new IntegerMaxValue().evaluate();
    ```

- UDTs support resource access. In MaxCompute SQL, you can call the static method `com.aliyun.odps.udf.impl.UDTExecutionContext.get()` to obtain the ExecutionContext object. Then, you can use this object to access the current ExecutionContext class and then access resources, such as files or tables.
- UDTs support the following operations:
  - Create objects by using the `new` method.
  - Create arrays by using the `new` method. Initializer lists can be used. Example: `new Integer[] { 1, 2, 3 }`.
  - Call methods, including static methods.

  > **Note**
  > - The identifiers in UDTs contain the names of packages, classes, methods, and fields. All identifiers are case-sensitive.
  > - Anonymous classes and lambda expressions are not supported.
  > - UDTs are used in expressions. Functions that do not return values cannot be called in expressions. This issue will be resolved in later versions.

- UDTs support the following data types:
  - UDTs support SQL type conversions, such as `cast(1 as java.lang.Object)`. UDTs do not support Java type conversions, such as `(Object)1`.

- The built-in types of UDTs have a one-to-one mapping relationship with specific Java types.
  - 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 built-in function CHR 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 apply.

    > **Note** 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.

- The following type conversion rules apply 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 between two objects without inheritance follows the original conversion rules. However, such conversions may result in changes to the data. 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 result in changes to the data and even loss of data precision.
- UDTs support Java generics. For example, the compiler can determine that the value returned by `java.util.Arrays.asList(new java.math.BigInteger('1'))` is of the java.util.List<java.math.BigInteger> type based on the parameter type.

  > **Note** You must specify the type parameter in a constructor function or use java.lang.Object. This is the same as Java.

- All operators use the semantics of MaxCompute SQL. Examples:
  - Combination of strings: 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.
  - = operations: The = operator in SQL statements is used as a comparison operator. It is used to compare one expression with another. You must call the equals method in Java to check whether two objects are equivalent. The = operator cannot be used to verify the equivalence of two objects.
- UDTs do not have a clear definition of object equality. This is caused by data reshuffling. Objects may be transmitted between different processes or physical machines. During object 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 verify the equivalence of two objects.

  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 of expressions, but cannot be used as outputs.
- You can use UDTs to implement the feature provided by the SCALAR function. You can use the built-in functions COLLECT_SET and EXPLODE with UDTs to implement the features provided by aggregate and table-valued functions.

1.6.8.4. Benefits

This topic describes the benefits of user-defined types (UDTs).

UDTs deliver the following benefits:

- UDTs are easy to use. You do not need to define functions.
- UDTs support all Java Development Kit (JDK) features. This improves SQL flexibility.
- UDT code can be stored in the same file as the SQL code. This facilitates code management.
- You can directly reference the libraries of other programming languages and reuse code that you have written in other languages.

1.6.8.5. Performance advantages

This topic describes the performance advantages of user-defined types (UDTs).

UDTs run in a similar way to UDFs in terms of performance. Therefore, the performance of UDTs is almost the same as that of UDFs. The optimized computing engine improves the performance of UDTs in specific scenarios.

- If a UDT object is used in different processes, it must be serialized and deserialized. If you use a UDT to perform operations that do not require data reshuffling, such as JOIN or AGGREGATE, the overheads of serialization and deserialization are avoided.
- The runtime of UDTs is based on Codegen rather than reflection. Therefore, no performance loss occurs. Multiple UDTs can be executed in a single function call. In the following example, it seems that a UDT is called multiple times, but the UDF is actually called only once. Therefore, no additional interface overheads are caused even though the operational units of UDTs are small.

```
values[x].add(values[y]).divide(java.math.BigInteger.valueOf(2))
```

1.6.8.6. Security advantages

This topic describes the security advantages of user-defined types (UDTs).

Similar to UDFs, UDTs are restricted in the Java sandbox model. To perform restricted operations, you must cancel sandbox isolation for the operations or apply to join a sandbox whitelist.

1.6.8.7. More examples

1.6.8.7.1. Example of using Java arrays

This topic provides an example of using Java arrays of user-defined types (UDTs).

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},  -- Initialize an ArrayList to create an array that contains three e
lements.
    new Integer[][] { new Integer[] {c1, c2}, new Integer[] {c3, c4} },  -- Create a multidimensiona
l array.
    new Integer[] {c1, c2, c3} [2], -- Access the elements in the array by using indexes.
    java.util.Arrays.asList(c1, c2, c3);    -- Create a list of the List<Integer> type, which can be
used as an array of the Array<Int> type. This is another way to create a built-in array.
FROM VALUES (1,2,3,4) AS t(c1, c2, c3, c4);
```

1.6.8.7.2. Example of using JSON

This topic provides an example of using JSON for user-defined types (UDTs).

The runtime of a UDT carries a JSON dependency (V2.2.4), which can be directly used in JSON.

> **Note** In addition to JSON 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).

Example:

```
set odps.sql.type.system.odps2=true;
set odps.sql.session.java.imports=java.util.*,java,com.google.gson. *; -- To import multiple package
s at a time, separate 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. Gso
n forcibly converts the deserialization result from the List<Object> type to the List<String> type.
from @a;
```

Returned results:

```
+------------+------------+
| _c0        | _c1        |
+------------+------------+
| [1,2,3]    | 1          |
+------------+------------+
```

Compared with the built-in function GET_JSON_OBJECT, this UDT-based method is simpler. In addition, this method deserializes the content that is extracted from the JSON string to a supported data type. The deserialization improves efficiency.

1.6.8.7.3. Example of using complex data types

This topic provides an example of using complex data types of user-defined types (UDTs).

The built-in data type ARRAY maps the java.util.List method, and the built-in data type MAP maps the java.util.Map method.

- Java objects in classes that implement java.util.List or java.util.Map can be used to process complex-type data in MaxCompute SQL.
- MaxCompute can directly call java.util.List or java.util.Map to process data of the ARRAY or MAP type.

Example:

```
set odps.sql.type.system.odps2=true;
set odps.sql.session.java.imports=java.util.*;
select
    size(new ArrayList<Integer>()),        -- Call the built-in function size() to obtain the size of the ArrayList.
    array(1,2,3).size(),                   -- Call the built-in function size() of the java.util.List method for the built-in data type ARRAY.
    sort_array(new ArrayList<Integer>()),  -- Sort the data in the ArrayList.
    al[1],                                 -- java.util.List does not support indexing but can process data of the ARRAY type that supports indexing.
    Objects.toString(a),        -- Convert data from the ARRAY type to the STRING type.
    array(1,2,3).subList(1, 2)             -- Obtain a sublist.
from (select new ArrayList<Integer>(array(1,2,3)) as al, array(1,2,3) as a) t;
```

Returned results:

```
+------------+------------+------------+------------+------------+------------+
| _c0        | _c1        | _c2        | _c3        | _c4        | _c5        |
+------------+------------+------------+------------+------------+------------+
| 0          | 3          | []         | 2          | [1, 2, 3]  | [2]        |
+------------+------------+------------+------------+------------+------------+
```

1.6.8.7.4. Aggregation example

This topic provides an example of using user-defined types (UDTs) to aggregate data.

To aggregate data by using a UDT, you must use the built-in function COLLECT_SET or COLLECT_LIST to aggregate data to a list and then call the UDT to calculate the aggregate value.

The following example shows how to calculate the median of BigInteger data. You cannot directly call the built-in function MEDIAN because the data is of the 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 list.
@b := select sort_array(values) as values, values.size() cnt from @a;  -- Sort 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;
-- Obtain the final output.
select med.toString() from @c;
```

Returned results:

```
+------------+
| _c0        |
+------------+
| 2.5        |
+------------+
```

The COLLECT_LIST function cannot be used to aggregate partial data because it can only aggregate all data in a specific group. It is less efficient than the built-in aggregate functions of MaxCompute or user-defined aggregate functions (UDAFs). We recommend that you use built-in aggregate functions if possible. If all data in a group is aggregated, data skew may occur.

The UDT-based method produces a higher efficiency than UDAFs or built-in aggregate functions, such as WM_CONCAT, for aggregating all data in a group.

1.6.8.7.5. Example of using table-valued functions

This topic provides an example of table-valued functions of user-defined types (UDTs).

Table-valued functions allow you to specify multiple input rows and columns, and can generate multiple output rows and columns. To implement a table-valued function, perform the following steps:

- Specify multiple input rows or columns. For more information, see Example of aggregation.
- Call the java.util.List or java.util.Map method to generate a data collection, and then call the explode function to split the collection into multiple rows.
- Call different getter methods to obtain data from different fields in a UDT. The obtained data is returned in multiple columns.

The following example shows how to expand a JSON string.

```
@a := select '[{"a":"1","b":"2"},{"a":"1","b":"2"}]' str; -- The sample data.
@b := select new com.google.gson.Gson().fromJson(str, java.util.List.class) l from @a; -- Deserializ
e the JSON string.
@c := select cast(e as java.util.Map<Object,Object>) m from @b lateral view explode(l) t as e;  -- C
all the explode function to split the string.
@d := select m.get('a') as a, m.get('b') as b from @c; -- Return the splitting result in multiple co
lumns.
select a.toString() a, b.toString() b from @d; -- The final output. Columns a and b in the variable
d are of the Object type.
```

Returned results:

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

1.6.9. UDJ

1.6.9.1. Background information

This topic describes the background information of user-defined join (UDJ).

MaxCompute provides multiple JOIN methods, including INNER JOIN, RIGHT JOIN, OUTER JOIN, LEFT JOIN, FULL JOIN, SEMI JOIN, and ANTI-SEMI JOIN. You can use these built-in JOIN methods in most scenarios. However, these methods are insufficient if you want to perform cross join operations.

In most cases, you can use user-defined functions (UDFs) to describe your code framework. However, the current UDF, user-defined table-valued function (UDTF), and user-defined aggregate function (UDAF) frameworks can only handle one table at a time. To perform user-defined operations for multiple tables, you must use built-in JOIN methods, UDFs, UDTFs, and complex SQL statements. In such scenarios, you must use a custom MapReduce framework instead of SQL to complete the required computing tasks.

Regardless of the scenario, these operations require technological expertise and may cause the following issues:

- In scenarios where you use built-in JOIN methods, UDFs, UDTFs, and complex SQL statements: The use of multiple JOIN methods and code in SQL statements results in a logical black box, which makes it difficult to

generate an optimal execution plan.

- In scenarios where you use a custom MapReduce framework: Execution plans are hard to optimize. Most of the MapReduce code is written in Java. During the deep optimization of native runtime code, the execution of the MapReduce code is less efficient than the execution of the MaxCompute code that is generated by the LLVM code generator.

MaxCompute introduces UDJ to the UDF framework based on the MaxCompute V2.0 computing engine. You can flexibly join tables by using UDJ to perform more customized operations. This simplifies MapReduce-based underlying operations in the distributed system.

1.6.9.2. Examples of using UDJ

1.6.9.2.1. Cross join operation by using UDJ

This topic describes how to use user-defined join (UDJ) to perform cross join operations and provides an example for reference.

Assume that two log tables named payment and user_client_log exist.

- The payment table stores the payment records of users. Each payment record contains the user ID, payment time, and payment content. The following table lists the sample data.

user_id	time	pay_info
2656199	2018-02-13 22:30:00	gZhvdySOQb
8881237	2018-02-13 08:30:00	pYvotuLDIT
8881237	2018-02-13 10:32:00	KBuMzRpsko

- The user_client_log table stores the client logs of users. Each log contains the user ID, logging time, and log content. The following table lists the sample data.

user_id	time	content
8881237	2018-02-13 00:30:00	click MpkvilgWSmhUuPn
8881237	2018-02-13 06:14:00	click OkTYNUHMqZzlDyL
8881237	2018-02-13 10:30:00	click OkTYNUHMqZzlDyL

Requirement: For each record in the user_client_log table, find the payment record that has the closest time to this record in the payment table. Then, join the two records and generate results. The following table lists the results.

user_id	time	content
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

To meet this requirement, use one of the following methods:

- Use built-in JOIN methods. SQL sample code:

```
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))
```

If 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. However, you cannot call aggregate functions in the join condition. Therefore, you cannot use standard JOIN methods to complete this task.

- Use the UDJ method.
  i. Create a UDJ function.
     a. Configure the SDK of the new version.

```
<dependency>
  <groupId>com.aliyun.odps</groupId>
  <artifactId>odps-sdk-udf</artifactId>
  <version>0.30.0</version>
  <scope>provided</scope>
</dependency>
```

     b. Write UDJ code and package the code as odps-udj-example.jar.

```
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 java.util.ArrayList;
import java.util.Iterator;
/** For each record in the table on the right, find the record in the table on the left th
at has the closest time to this record.
 * Join the two records.
 */
@Resolve("->string,bigint,string")
public class PayUserLogMergeJoin extends UDJ {
  private Record outputRecord;
  /** Call the preceding code before data is processed. You can use the PayUserLogMergeJoi
n method for initialization.
   */
  @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)
    });
  }
  /** Rewrite the PayUserLogMergeJoin method to implement the connection logic.
   * @param key: the current key for joining.
   * @param left: the record group of the current key in the table on the left.
```

```
   * @param right: the record group of the current key in the table on the right.
   * @param output: generates the result of UDJ.
   */
  @Override
  public void join(Record key, Iterator<Record> left, Iterator<Record> right, Yieldable<Re
cord> output) {
    outputRecord.setString(0, key.getString(0));
    if (! right.hasNext()) {
      // The group on the right is empty. Do not perform operations on this group.
      return;
    } else if (! left.hasNext()) {
      // The group on the left is empty. Generate all records from the group on the right
but do not join the records of the two tables.
      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 records in the group on the left are iterated from the beginning to the end.
    // The iterator cannot reset the records in the group on the right.
    // Save each record of the group on the left to 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 all the records of the group on the left. Then, you can find the minimum
time difference between the records.
      for (Record pay: pays) {
        long delta = Math.abs(logTime - pay.getDatetime(0).getTime());
        if (delta < minDelta) {
          minDelta = delta;
          nearestPay = pay;
        }
      }
      // Merge the log records and payment records that are closest in time and then gener
ate results.
      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() {
  }
}
```

c. Add the odps-udj-example.jar package to MaxCompute.

```
add jar odps-udj-example.jar;
```

d. Register the UDJ function pay_user_log_merge_join in MaxCompute.

```
create function pay_user_log_merge_join
  as 'com.aliyun.odps.udf.example.udj.PayUserLogMergeJoin'
  using 'odps-udj-example.jar';
```

ii. Prepare sample data.

a. Create the payment table and the user_client_log table.

```
create table payment(user_id string,time datetime,pay_info string);
create table user_client_log(user_id string,time datetime,content string);
```

b. Insert data into the sample tables.

```
-- Insert data into 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')
;
-- Insert data into 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')
;
```

iii. Use a UDJ function in SQL.

```
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);
```

Parameters in the USING clause:

- pay_user_log_merge_join: the name of the UDJ function in SQL.
- (p.time, p.pay_info, u.time, u.content): the columns in both of the tables used in the UDJ function.
- r: 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.

Returned results:

```
+---------+------------+---------+
| 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 |
+---------+------------+---------+
```

1.6.9.2.2. Pre-sorting

This topic describes the pre-sorting feature of user-defined join (UDJ) and provides an example for reference.

An iterator is used to iterate all records in the payment table and find the payment record that has the closest time to a specific log record in the user_client_log table. 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 the limits of the memory size, 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 by using the SORT BY clause.

If the number of payment records is too large to be stored in the memory and all the data in the table has been sorted by time, you need only to compare the first element in the two lists.

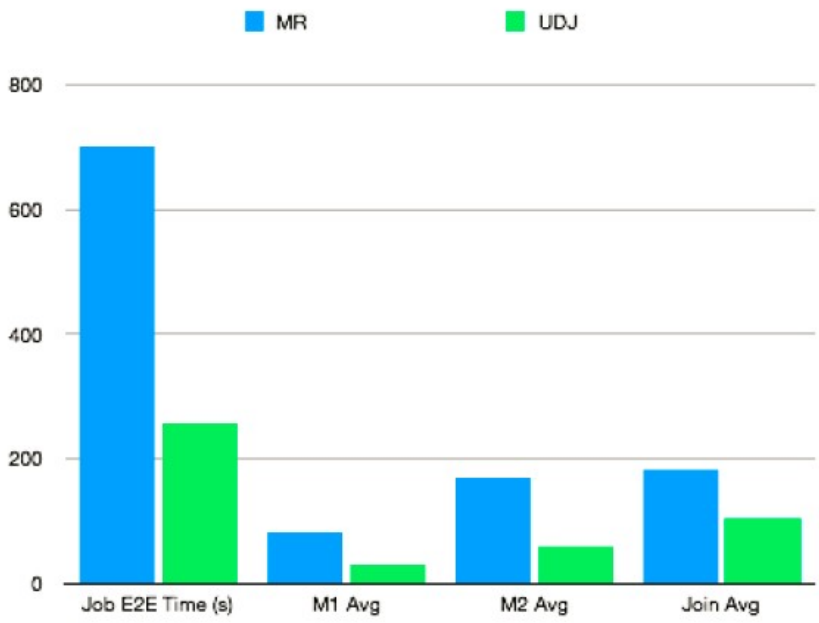
This method uses the SORT BY clause to pre-sort the UDJ data. To achieve the same effect by using the previous method, you need only to cache a maximum of three data records during pre-sorting. The following code shows an example:

```
@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 time delta between the two records decreases and the operation can continue.
      // Explore the group on the left to try to obtain a smaller time delta.
      lastPayRecord = payRecord.clone();
      prevDelta = delta;
      payRecord = left.next();
    } else {
     // The minimum time delta point is reached. Check the final record in the payment table.
     // Generate the merged result and prepare to process the next record.
     // The group on the right.
      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;
      }
    }
  }
}
```

1.6.9.3. Performance

This topic describes the performance of user-defined join (UDJ).

A real online MapReduce job is used as an example to verify the performance of UDJ. The job runs based on a complex algorithm. In this example, two tables are joined, UDJ is used to rewrite the MapReduce job, and the correctness of the UDJ results is checked. The following figure shows the MapReduce and UDJ performance under the same data concurrency.



As shown in the figure, UDJ conveniently describes the complex logic of how to handle multiple tables and greatly improves performance. The code is called only within UDJ. The entire Mapper logic in this example is executed by the native runtime engine of MaxCompute. The data exchange logic between the MaxCompute UDJ runtime engine and Java interfaces is optimized in Java code. The JOIN logic of UDJ is more efficient than that of a reducer.

1.6.10. Parameterized view

This topic describes the parameterized view feature that is supported by the MaxCompute SQL engine.

In the traditional views of MaxCompute, complex SQL scripts are encapsulated at the underlying layer. Callers can call views the same way as reading a common table without the need to understand the underlying implementation. Traditional views are widely used because they implement encapsulation and reuse. However, traditional views cannot accept parameters from callers. This reduces code reuse efficiency. For example, a caller cannot filter data in the underlying table that is read by a view nor pass other parameters. The MaxCompute SQL engine supports parameterized views and allows you to import tables or variables to customize views.

Define a parameterized view

You can use the following syntax to create a parameterized view:

```
-- view with parameters
-- param @a -a table parameter
-- param @b -a string parameter
-- 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;
```

Syntax description:

- To create a parameterized view, you must specify parameters. Therefore, you must use Script Mode SQL to create a parameterized view.
- The created view pv1 has two parameters: TABLE and STRING. The parameter value can be a table or of a basic data type.
- The parameter value can also be a subquery, for example, `SELECT * FROM view_name( (SELECT 1 FROM src WH`

`ERE a > 0), 1);` .

- When you create a view, you can set ANY for a parameter, which indicates any data type. For example, `CREATE VIEW paramed_view (@a ANY) AS SELECT * FROM src WHERE CASE WHEN @a IS NULL THEN key1 ELSE key2 END = key3;` defines that the first parameter of the view can be a value of any data type.

  However, the ANY type cannot be used in a specific operation, such as + or AND, which requires specific data types. A field of the ANY type is often used as a PassThrough column in the TABLE parameter. Example:

  ```
  CREATE VIEW paramed_view (@a TABLE(name STRING, id ANY, age BIGINT)) AS SELECT * FROM @a WHERE nam
  e = 'foo' AND age < 25;
  -- The call example.
  SELECT * FROM param_view((SELECT name, id, age FROM students));
  ```

  > **Note**
  > - After you execute the `CREATE VIEW` statement to create a view, you can run the `DESC` command to obtain the description of the view. This description contains the return type of the view.
  > - The return type of a view is recalculated when the view is called. It may be different from the return type, such as ANY, that you specify when you create the view.

- When you create a view, you can use an asterisk (*) in the TABLE parameter to retrieve any columns. The asterisk (*) can represent a specific data type or the ANY type. Example:

  ```
  CREATE VIEW paramed_view (@a TABLE(key STRING, * ANY), @b TABLE(key STRING, * STRING)) AS SELECT a
  . * FROM @a JOIN @b ON a.key = b.key;
  -- The call example.
  SELECT name, address FROM param_view((SELECT school, name, age, address FROM student), school) WHE
  RE age < 20;
  ```

  In this example, the view accepts two TABLE parameters. In the table that is specified by the first TABLE parameter, data in the first column is of the STRING type, and data in other columns can be of the ANY type. In the table that is specified by the second TABLE parameter, data in the first and other columns is all of the STRING type. Take note of the following points:

  - The varied-length part must be placed at the end of the table that is specified by the TABLE parameter. This means that the * column cannot be followed by other columns. Therefore, the table that is specified by the TABLE parameter can contain only one varied-length column.
  - The varied-length part must be placed at the end of the table that is specified by the TABLE parameter. However, the columns of an input table may not be arranged in this sequence. In this case, the columns need to be rearranged. A subquery can be used as a parameter value and must be enclosed in a pair of parentheses ().
  - No name is specified for the varied-length part of the table that is specified by the TABLE parameter. As a result, data in the varied-length part cannot be referenced or used for operation when you define the view.
  - Although you cannot use the varied-length part for operation, you can use the SELECT * statement to transfer data in the varied-length part out of the table.
  - The column of the table that is specified by the TABLE parameter may be different from the fixed-length column that you specify when you define the view. If the names are different, the compiler automatically renames the column of the table that is specified by the TABLE parameter. If the data types are different, the compiler performs an implicit conversion. If the implicit conversion fails, an error occurs.

Call a parameterized view

You can execute the following statement to call the pv1 view that you defined:

```
@a := SELECT * FROM src WHERE value >0;
--call view with table variable and 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 the name of a physical table, view, or table variable. You can also set the TABLE parameter to a table alias by using common table expressions (CTEs).
- The values of common parameters can be variables or constants.

Usage notes

- The script of a parameterized view can contain only data manipulation language (DML) statements. The statements, such as INSERT, CREATE TABLE, or PRINT, cannot be contained.
- A parameterized view can contain multiple SQL statements.

```
-- view with parameters
-- param @a -a table parameter
-- param @b -a string parameter
-- 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 Statements between BEGIN and END are the script of this view. The `@pv2 :=...` statement is similar to the RETURN statement in other programming languages. You can use the @pv2 :=... statement to assign a value to a variable of an implicit table that has the same name as the view.

- The matching rules for actual and formal view parameters are the same as those specified in a common weakly-typed language. Specifically, if a view parameter can be implicitly converted, it can match the defined parameter. For example, a value of the BIGINT type can match 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-type parameters that have the same schema as table b.
- You can explicitly declare the return type to make the code easier to read.

```
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, which indicates the type returned to the caller. The return type is specified by the TABLE (x STRING,y STRING) parameter. You can use this parameter to customize the table schema.
- The response parameter. The @ret parameter defines the name of the response parameter. Assigning a value to the response parameter is performed in the view script.

You can consider the view that contains no BEGIN/END or return variables as a simplified parameterized view.

1.6.11. CLONE TABLE

This topic describes how to use the CLONE TABLE statement. The CLONE TABLE statement is used to clone data from one table to another. This statement improves data migration efficiency.

Limits

- The schema of the destination table must be compatible with that of the source table.
- The CLONE TABLE statement can be executed for partitioned tables and non-partitioned tables. This statement cannot be executed for hash clustering tables.
- If a destination table is created before the CLONE TABLE statement is executed, data in a maximum of 10,000 partitions can be cloned at a time.
- If a destination table is not created before the CLONE TABLE statement is executed, the number of partitions from which you can clone data at a time is unlimited. This ensures atomicity.
- You can execute the CLONE TABLE statement for up to seven times in the same non-partitioned table or in the same partition of a partitioned table.
- You cannot execute the CLONE TABLE statement for projects across regions.

Syntax

```
CLONE TABLE <[src_project_name.]src_table_name> [PARTITION(spec), ...]
 TO <[dest_project_name.]desc_table_name> [IF EXISTS (OVERWRITE | IGNORE)] ;
```

Description

The CLONE TABLE statement is used to clone data from the src_table_name table to the desc_table_name table.

Note After you clone data to the desc_table_name table, we recommend that you verify the data accuracy. For example, you can execute the SELECT COUNT statement to view the number of rows in the destination table or execute the DESC statement to view the table size.

Parameters

- src_table_name: the name of the source table.
- src_project_name: the name of the project to which the source table belongs. If this parameter is not specified, the name of the current project is used by default.

- desc_table_name: the name of the destination table.
  - If a destination table is not created, the table is created by using the CREATE TABLE LIKE statement when you execute the CLONE TABLE statement.
  - If a destination table is created and IF EXISTS OVERWRITE is specified, data in the partitions of the destination table is overwritten when you execute the CLONE TABLE statement.
  - If a destination table is created and IF EXISTS IGNORE is specified, the existing partitions in the table are skipped and the data in these partitions is not overwritten when you execute the CLONE TABLE statement.
- dest_project_name: the name of the project to which the destination table belongs. If this parameter is not specified, the current project name is used by default.

Examples

Assume that the partitioned table srcpart_copy and non-partitioned table src_copy are source tables. The two tables have the following metadata:

```
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 src_copy to the destination table src_clone.

  ```
  CLONE TABLE src_copy TO src_clone;
  ```

  Returned results:

  ```
  ID = 20191023030245442540cdv2
  OK
  ```

  //After the data is cloned, query data in the destination table src_clone and check the data accuracy.

  ```
  SELECT * FROM src_clone;
  ```

- Clone data from a specified partition of the partitioned table srcpart_copy to the destination table srcpart_clone.

  ```
  CLONE TABLE srcpart_copy PARTITION(ds="2008-04-09", hr='11') TO srcpart_clone IF EXISTS OVERWRITE;
  ```

  Returned results:

  ```
  ID = 20191023030534986g4540cdv2
  OK
  ```

  //After the data is cloned, query data in the destination table srcpart_clone and check the data accuracy.

  ```
  SELECT * FROM srcpart_clone;
  ```

- Clone all data from the partitioned table srcpart_copy to the destination table srcpart_clone and skip the data in the existing partitions of the destination table.

```
CLONE TABLE srcpart_copy TO srcpart_clone IF EXISTS IGNORE;
```

Returned results:

```
ID = 20191023030619196g5540cdv2
OK
```

//After the data is cloned, query data in the destination table srcpart_clone and check the data accuracy.

```
SELECT * FROM srcpart_clone;
```

- Clone all data from the partitioned table srcpart_copy to the destination table srcpart_clone2.

```
CLONE TABLE srcpart_copy TO srcpart_clone2;
```

Returned results:

```
ID = 20191023030825186g6540cdv2
OK
```

//After the data is cloned, query data in the destination table srcpart_clone2 and check the data accuracy.

```
SELECT * FROM srcpart_clone2;
```

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 a `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

This topic describes the Constructor function ST_AsBinary and provides an example for reference.

Function declaration:

```
ST_AsBinary(ST_Geometry)
```

Description: This function returns the well-known binary (WKB) expression of the input geometry.

Example:

```
SELECT ST_AsBinary(ST_Point(1, 2)) FROM onerow;
```

Returned results:

```
WKB representation of POINT (1 2)
```

1.6.12.2.2. ST_AsGeoJson

This topic describes the Constructor function ST_AsGeoJson and provides an example for reference.

Function declaration:

```
ST_AsGeoJson(geometry)
```

Description: This function returns the GeoJSON expression of the input geometry.

Example:

```
SELECT ST_AsGeoJson(ST_Point(1.0, 2.0)) from onerow;
```

Returned results:

```
{"type":"Point", "coordinates":[1.0, 2.0]}
```

1.6.12.2.3. ST_AsJson

This topic describes the Constructor function ST_AsJson and provides examples for reference.

Function declaration:

```
ST_AsJSON(ST_Geometry)
```

Description: This function returns the JSON expression of the input geometry.

Examples:

```
SELECT ST_AsJSON(ST_Point(1.0, 2.0)) from onerow;
```

Returned results:

```
{"x":1.0,"y":2.0}
```

```
SELECT ST_AsJSON(ST_SetSRID(ST_Point(1, 1), 4326)) from onerow;
```

Returned results:

```
{"x":1.0,"y":1.0,"spatialReference":{"wkid":4326}}
```

1.6.12.2.4. ST_AsShape

This topic describes the Constructor function ST_AsShape and provides an example for reference.

Function declaration:

```
ST_AsShape(ST_Geometry)
```

Description: This function returns the ESRI shape expression of the input geometry.

Example:

```
SELECT ST_AsShape(ST_Point(1, 2)) FROM onerow;
```

Returned results:

```
Esri shape representation of POINT (1 2)
```

1.6.12.2.5. ST_AsText

This topic describes the Constructor function ST_AsText and provides an example for reference.

Function declaration:

```
ST_AsText(ST_Geometry)
```

Description: This function returns the well-known text (WKT) expression of the input geometry.

Example:

```
SELECT ST_AsText(ST_Point(1, 2)) FROM onerow;
```

Returned results:

```
POINT (1 2)
```

1.6.12.2.6. ST_GeomCollection

This topic describes the ST_GeomCollection function of the Constructors function and provides an example for reference.

Function declaration:

```
ST_GeomCollection(wkt)
```

Description: This function constructs a multi-part geometry based on the well-known text (WKT) representation defined by 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)')
-- The collection feature is not supported.
```

1.6.12.2.7. ST_GeomFromGeoJson

This topic describes the Constructor function ST_GeomFromGeoJson and provides an example for reference.

Function declaration:

```
ST_GeomFromGeoJson(json)
```

Description: This function constructs a geometry based on the input GeoJSON expression.

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

This topic describes the Constructor function ST_GeomFromJSON and provides an example for reference.

Function declaration:

```
ST_GeomFromJSON(json)
```

Description: This function constructs a geometry based on the input ESRI JSON expression.

Example:

```
SELECT ST_GeomFromJSON('{"x":0.0,"y":0.0}') FROM src LIMIT 1;
-- Construct a point.
```

1.6.12.2.9. ST_GeomFromShape

This topic describes the Constructor function ST_GeomFromShape and provides an example for reference.

Function declaration:

```
ST_GeomFromShape(shape)
```

Description: This function constructs a geometry based on the input ESRI shape expression.

Example:

```
SELECT ST_GeomFromShape(ST_AsShape(ST_Point(1, 2)));
-- Construct a point.
```

1.6.12.2.10. ST_GeomFromText

This topic describes the Constructor function ST_GeomFromText and provides an example for reference.

Function declaration:

```
ST_GeomFromText(wkt)
```

Description: This function constructs a geometry based on the input well-known text (WKT) representation defined by 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

This topic describes the Constructor function ST_GeomFromWKB and provides an example for reference.

Function declaration:

```
ST_GeomFromWKB(wkb)
```

Description: This function constructs a geometry based on the input well-known binary (WKB) expression that complies with the Open Geospatial Consortium (OGC) standards.

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

This topic describes the Constructor function ST_GeometryType and provides examples for reference.

Function declaration:

```
ST_GeometryType(geometry)
```

Description: This function returns the type of the input geometry.

Examples:

```
SELECT ST_GeometryType(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
```

Returned results:

```
ST_Point
```

```
SELECT ST_GeometryType(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
```

Returned results:

```
ST_LineString
```

```
SELECT ST_GeometryType(ST_Polygon(2,0, 2,3, 3,0)) FROM src LIMIT 1;
```

Returned results:

```
ST_Polygon
```

1.6.12.2.13. ST_LineString

This topic describes the Constructor function ST_LineString and provides an example for reference.

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

This topic describes the Constructor function ST_LineFromWKB and provides an example for reference.

Function declaration:

```
ST_LineFromWKB(wkb)
```

Description: This function constructs a two-dimensional line based on the input well-known binary (WKB) representation defined by 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

This topic describes the Constructor function ST_MultiLineString and provides an example for reference.

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;
-- Construct a two-dimensional multilinestring.
```

1.6.12.2.16. ST_MLineFromWKB

This topic describes the Constructor function ST_MLineFromWKB and provides an example for reference.

Function declaration:

```
ST_MLineFromWKB(wkb)
```

Description: This function constructs a two-dimensional multilinestring based on the input well-known binary (WKB) expression that complies with the Open Geospatial Consortium (OGC) standards.

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

This topic describes the Constructor function ST_MultiPoint and provides an example for reference.

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

This topic describes the Constructor function ST_MPointFromWKB and provides an example for reference.

Function declaration:

```
ST_MPointFromWKB(wkb)
```

Description: This function constructs a two-dimensional multipoint geometry based on the input well-known binary (WKB) representation defined by 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

This topic describes the Constructor function ST_MultiPolygon and provides an example for reference.

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 LIM
IT 1;
-- Construct a two-dimensional multipolygon.
```

1.6.12.2.20. ST_MPolyFromWKB

This topic describes the Constructor function ST_MPolyFromWKB and provides an example for reference.

Function declaration:

```
ST_MPolyFromWKB(wkb)
```

Description: This function constructs a two-dimensional multipolygon based on the input well-known binary (WKB) expression that complies with the Open Geospatial Consortium (OGC) standards.

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

This topic describes the Constructor function ST_Point and provides an example for reference.

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;
-- Construct a two-dimensional point.
```

1.6.12.2.22. ST_PointFromWKB

This topic describes the Constructor function ST_PointFromWKB and provides an example for reference.

Function declaration:

```
ST_PointFromWKB(wkb)
```

Description: This function constructs a two-dimensional point based on the input well-known binary (WKB) expression that complies with the Open Geospatial Consortium (OGC) standards.

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

This topic describes the Constructor function ST_PointZ and provides an example for reference.

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;
-- Construct a three-dimensional point.
```

1.6.12.2.24. ST_Polygon

This topic describes the Constructor function ST_Polygon and provides an example for reference.

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

This topic describes the Constructor function ST_PolyFromWKB and provides an example for reference.

Function declaration:

```
ST_PolyFromWKB(wkb)
```

Description: This function constructs a two-dimensional polygon based on the input well-known binary (WKB) representation defined by the Open Geospatial Consortium (OGC).

Example:

```
SELECT ST_PolyFromWKB(ST_AsBinary(ST_GeomFromText('polygon ((0 0, 10 0, 0 10, 0 0))'))) FROM src LIMIT 1;
-- Construct a two-dimensional polygon.
```

1.6.12.2.26. ST_SetSRID

This topic describes the Constructor function ST_SetSRID and provides an example for reference.

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

This topic describes the Accessor function ST_Area and provides an example for reference.

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;
```

Returned results:

```
9.0
```

1.6.12.3.2. ST_Centroid

This topic describes the Accessor function ST_Centroid and provides examples for reference.

Function declaration:

```
ST_Centroid(polygon)
```

Description: This function returns the centroid of the minimum bounding rectangle of the input polygon.

Examples:

```
SELECT ST_Centroid(ST_GeomFromText('polygon ((0 0, 3 6, 6 0, 0 0))')) FROM src LIMIT 1;
```

Returned results:

```
POINT(3 3)
```

```
SELECT ST_Centroid(ST_GeomFromText('polygon ((0 0, 0 8, 8 0, 0 0))')) FROM src LIMIT 1;
```

Returned results:

```
POINT(4 4)
```

1.6.12.3.3. ST_CoordDim

This topic describes the Accessor function ST_CoordDim and provides examples for reference.

Function declaration:

```
ST_CoordDim(geometry)
```

Description: This function returns the coordinate dimension of the input geometry.

Examples:

```
SELECT ST_CoordDim(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
```

Returned results:

```
2
```

```
SELECT ST_CoordDim(ST_PointZ(1.5,2.5, 3) FROM src LIMIT 1;
```

Returned results:

```
3
```

```
SELECT ST_CoordDim(ST_Point(1.5, 2.5, 3., 4.)) FROM src LIMIT 1;
```

Returned results:

```
4
```

1.6.12.3.4. ST_Dimension

This topic describes the Accessor function ST_Dimension and provides examples for reference.

Function declaration:

```
ST_Dimension(geometry)
```

Description: This function returns the spatial dimension of the input geometry.

Examples:

```
SELECT ST_Dimension(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
```

Returned results:

```
0
```

```
SELECT ST_Dimension(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
```

Returned results:

```
1
```

```
SELECT ST_Dimension(ST_Polygon(2,0, 2,3, 3,0)) FROM src LIMIT 1;
```

Returned results:

```
2
```

1.6.12.3.5. ST_Distance

This topic describes the Accessor function ST_Distance and provides an example for reference.

Function declaration:

```
ST_Distance(ST_Geometry1, ST_Geometry2)
```

Description: This function returns the distance between two geometries.

Example:

```
SELECT ST_Distance(ST_Point(0.0,0.0), ST_Point(3.0,4.0)) FROM src LIMIT 1;
```

Returned results:

```
5.0
```

1.6.12.3.6. ST_GeodesicLengthWGS84

This topic describes the Accessor function ST_GeodesicLengthWGS84 and provides examples for reference.

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.

Examples:

```
SELECT ST_GeodesicLengthWGS84(ST_SetSRID(ST_Linestring(0.0,0.0, 0.3,0.4), 4326)) FROM src LIMIT 1;
```

Returned results:

```
55km
```

```
SELECT ST_GeodesicLengthWGS84(ST_GeomFromText('MultiLineString((0.0 80.0, 0.3 80.4))', 4326)) FROM s
rc LIMIT 1;
```

Returned results:

```
45km
```

1.6.12.3.7. ST_GeometryN

This topic describes the Accessor function ST_GeometryN and provides examples for reference.

Function declaration:

```
ST_GeometryN(ST_GeometryCollection, n)
```

Description: This function returns the n^{th} geometry in the input geometry collection. n starts from 1.

Examples:

```
SELECT ST_GeometryN(ST_GeomFromText('multipoint ((10 40), (40 30), (20 20), (30 10))'), 3) FROM src
LIMIT 1;
```

Returned results:

```
ST_Point(20 20)
```

```
SELECT ST_GeometryN(ST_GeomFromText('multilinestring ((2 4, 10 10), (20 20, 7 8))'), 2) FROM src LIM
IT 1;
```

Returned results:

```
ST_Linestring(20 20, 7 8)
```

1.6.12.3.8. ST_Is3D

This topic describes the Accessor function ST_Is3D and provides an example for reference.

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 is returned.
SELECT ST_Is3D(ST_LineString(0.,0., 3.,4., 0.,4., 0.,0.)) FROM src LIMIT 1;
-- False is returned.
SELECT ST_Is3D(ST_Point(3., 4.)) FROM src LIMIT 1;
-- False is returned.
SELECT ST_Is3D(ST_PointZ(3., 4., 2)) FROM src LIMIT 1;
-- True is returned.
```

1.6.12.3.9. ST_IsClosed

This topic describes the Accessor function ST_IsClosed and provides an example for reference.

Function declaration:

```
ST_IsClosed(ST_[Multi]LineString)
```

Description: If the input linestring or linestrings are closed, this function returns True. Otherwise, it returns False.

Example:

```
SELECT ST_IsClosed(ST_LineString(0.,0., 3.,4., 0.,4., 0.,0.)) FROM src LIMIT 1;
-- True is returned.
SELECT ST_IsClosed(ST_LineString(0.,0., 3.,4.)) FROM src LIMIT 1;
-- False is returned.
```

1.6.12.3.10. ST_IsEmpty

This topic describes the Accessor function ST_IsEmpty and provides an example for reference.

Function declaration:

```
ST_IsEmpty(geometry)
```

Description: If the input geometry is empty, this function returns True. Otherwise, it returns False.

Example:

```
SELECT ST_IsEmpty(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
-- False is returned.
SELECT ST_IsEmpty(ST_GeomFromText('point empty')) FROM src LIMIT 1;
-- True is returned.
```

1.6.12.3.11. ST_IsMeasured

This topic describes the Accessor function ST_IsMeasured and provides an example for reference.

Function declaration:

```
ST_IsMeasured(geometry)
```

Description: If the input geometry has M coordinates (measures), this function returns True. Otherwise, it returns False.

Example:

```
SELECT ST_IsMeasured(ST_Polygon(1,1, 1,4, 4,4, 4,1)) FROM src LIMIT 1;
-- False is returned.
SELECT ST_IsMeasured(ST_LineString(0.,0., 3.,4., 0.,4., 0.,0.)) FROM src LIMIT 1;
-- False is returned.
SELECT ST_IsMeasured(ST_Point(3., 4.)) FROM src LIMIT 1;
-- False is returned.
SELECT ST_IsMeasured(ST_PointM(3., 4., 2)) FROM src LIMIT 1;
-- True is returned.
```

1.6.12.3.12. ST_IsSimple

This topic describes the Accessor function ST_IsSimple and provides an example for reference.

Function declaration:

```
ST_IsSimple(geometry)
```

Description: If the input geometry is a simple object, this function returns True. Otherwise, it returns False.

Example:

```
SELECT ST_IsSimple(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
-- True is returned.
SELECT ST_IsSimple(ST_LineString(0.,0., 1.,1., 0.,1., 1.,0.)) FROM src LIMIT 1;
-- False is returned.
```

1.6.12.3.13. ST_IsRing

This topic describes the Accessor function ST_IsRing and provides an example for reference.

Function declaration:

```
ST_IsRing(ST_LineString)
```

Description: This function returns True if the line string is a simple object or closed. Otherwise, it returns False.

Example:

```
SELECT ST_IsRing(ST_LineString(0.,0., 3.,4., 0.,4., 0.,0.)) FROM src LIMIT 1;
-- True is returned.
SELECT ST_IsRing(ST_LineString(0.,0., 1.,1., 1.,2., 2.,1., 1.,1., 0.,0.)) FROM src LIMIT 1;
-- False is returned.
SELECT ST_IsRing(ST_LineString(0.,0., 3.,4.)) FROM src LIMIT 1;
-- False is returned.
```

1.6.12.3.14. ST_Length

This topic describes the Accessor function ST_Length and provides an example for reference.

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;
```

Returned results:

```
5.0
```

1.6.12.3.15. ST_M

This topic describes the Accessor function ST_M and provides an example for reference.

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;
```

Returned results:

```
2
```

1.6.12.3.16. ST_MaxM

This topic describes the Accessor function ST_MaxM and provides examples for reference.

Function declaration:

```
ST_MaxM(geometry)
```

Description: This function returns the maximum M coordinate of the input geometry.

Examples:

```
SELECT ST_MaxM(ST_PointM(1.5, 2.5, 2)) FROM src LIMIT 1;
```

Returned results:

```
2
```

```
SELECT ST_MaxM(ST_LineString('linestring m (1.5 2.5 2, 3.0 2.2 1)')) FROM src LIMIT 1;
```

Returned results:

```
1
```

1.6.12.3.17. ST_MinM

This topic describes the Accessor function ST_MinM and provides examples for reference.

Function declaration:

```
ST_MinM(geometry)
```

Description: This function returns the minimum M coordinate of the input geometry.

Examples:

```
SELECT ST_MinM(ST_PointM(1.5, 2.5, 2)) FROM src LIMIT 1;
```

Returned results:

```
2
```

```
SELECT ST_MinM(ST_LineString('linestring m (1.5 2.5 2, 3.0 2.2 1)')) FROM src LIMIT 1;
```

Returned results:

```
1
```

1.6.12.3.18. ST_X

This topic describes the Accessor function ST_X and provides an example for reference.

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;
```

Returned results:

```
1.5
```

1.6.12.3.19. ST_Y

This topic describes the Accessor function ST_Y and provides an example for reference.

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;
```

Returned results:

```
2.5
```

1.6.12.3.20. ST_Z

This topic describes the Accessor function ST_Z and provides an example for reference.

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;
```

Returned results:

```
1.5
```

1.6.12.3.21. ST_MaxX

This topic describes the Accessor function ST_MaxX and provides examples for reference.

Function declaration:

```
ST_MaxX(geometry)
```

Description: This function returns the maximum X coordinate of the input geometry.

Examples:

```
SELECT ST_MaxX(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
```

Returned results:

```
1.5
```

```
SELECT ST_MaxX(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
```

Returned results:

```
3.0
```

1.6.12.3.22. ST_MaxY

This topic describes the Accessor function ST_MaxX and provides examples for reference.

Function declaration:

```
ST_MaxY(geometry)
```

Description: This function returns the maximum Y coordinate of the input geometry.

Examples:

```
SELECT ST_MaxY(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
```

Returned results:

```
2.5
```

```
SELECT ST_MaxY(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
```

Returned results:

```
2.5
```

1.6.12.3.23. ST_MaxZ

This topic describes the Accessor function ST_MaxZ and provides examples for reference.

Function declaration:

```
ST_MaxZ(geometry)
```

Description: This function returns the maximum Z coordinate of the input geometry.

Examples:

```
SELECT ST_MaxZ(ST_PointZ(1.5, 2.5, 2)) FROM src LIMIT 1;
```

Returned results:

```
2
```

```
SELECT ST_MaxZ(ST_LineString('linestring z (1.5 2.5 2, 3.0 2.2 1)')) FROM src LIMIT 1;
```

Returned results:

```
1
```

1.6.12.3.24. ST_MinX

This topic describes the Accessor function ST_MinX and provides examples for reference.

Function declaration:

```
ST_MinX(geometry)
```

Description: This function returns the minimum X coordinate of the input geometry.

Examples:

```
SELECT ST_MinX(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
```

Returned results:

```
1.5
```

```
SELECT ST_MinX(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
```

Returned results:

```
3.0
```

1.6.12.3.25. ST_MinY

This topic describes the Accessor function ST_MinY and provides examples for reference.

Function declaration:

```
ST_MinY(geometry)
```

Description: This function returns the minimum Y coordinate of the input geometry.

Examples:

```
SELECT ST_MinY(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
```

Returned results:

```
2.5
```

```
SELECT ST_MinY(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
```

Returned results:

```
2.2
```

1.6.12.3.26. ST_MinZ

This topic describes the Accessor function ST_MinZ and provides examples for reference.

Function declaration:

```
ST_MinZ(geometry)
```

Description: This function returns the minimum Z coordinate of the input geometry.

Examples:

```
SELECT ST_MinZ(ST_PointZ(1.5, 2.5, 2)) FROM src LIMIT 1;
```

Returned results:

```
2
```

```
SELECT ST_MinZ(ST_LineString('linestring z (1.5 2.5 2, 3.0 2.2 1)')) FROM src LIMIT 1;
```

Returned results:

```
1
```

1.6.12.3.27. ST_NumGeometries

This topic describes the Accessor function ST_NumGeometries and provides examples for reference.

Function declaration:

```
ST_NumGeometries(ST_GeometryCollection)
```

Description: This function returns the number of geometries in the input geometry collection.

Examples:

```
SELECT ST_NumGeometries(ST_GeomFromText('multipoint ((10 40), (40 30), (20 20), (30 10))')) FROM src
LIMIT 1;
```

Returned results:

```
4
```

```
SELECT ST_NumGeometries(ST_GeomFromText('multilinestring ((2 4, 10 10), (20 20, 7 8))')) FROM src LI
MIT 1;
```

Returned results:

```
2
```

1.6.12.3.28. ST_NumInteriorRing

This topic describes the Accessor function ST_NumInteriorRing and provides examples for reference.

Function declaration:

```
ST_NumInteriorRing(ST_Polygon)
```

Description: This function returns the number of interior rings of the input polygon.

Examples:

```
SELECT ST_NumInteriorRing(ST_Polygon(1,1, 1,4, 4,1)) FROM src LIMIT 1;
```

Returned results:

```
0
```

```
SELECT ST_NumInteriorRing(ST_Polygon('polygon ((0 0, 8 0, 0 8, 0 0), (1 1, 1 5, 5 1, 1 1))')) FROM s
rc LIMIT 1;
```

Returned results:

```
1
```

1.6.12.3.29. ST_NumPoints

This topic describes the Accessor function ST_NumPoints and provides examples for reference.

Function declaration:

```
ST_NumPoints(geometry)
```

Description: This function returns the number of points in the input geometry.

Examples:

```
SELECT ST_NumPoints(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
```

Returned results:

```
1
```

```
SELECT ST_NumPoints(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
```

Returned results:

```
2
```

```
SELECT ST_NumPoints(ST_GeomFromText('polygon ((0 0, 10 0, 0 10, 0 0))')) FROM src LIMIT 1;
```

Returned results:

```
4
```

1.6.12.3.30. ST_PointN

This topic describes the Accessor function ST_PointN and provides an example for reference.

Function declaration:

```
ST_PointN(ST_Geometry, n)
```

Description: This function returns the n^{th} 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;
```

Returned results:

```
POINT(3.0 2.2)
```

1.6.12.3.31. ST_StartPoint

This topic describes the Accessor function ST_StartPoint and provides an example for reference.

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;
```

Returned results:

```
POINT(1.5 2.5)
```

1.6.12.3.32. ST_EndPoint

This topic describes the Accessor function ST_EndPoint and provides an example for reference.

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;
```

Returned results:

```
POINT(3.0 2.0)
```

1.6.12.3.33. ST_SRID

This topic describes the Accessor function ST_SRID and provides an example for reference.

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;
```

Returned results:

```
returns SRID 0
```

1.6.12.4. Operations

1.6.12.4.1. ST_Aggr_ConvexHull

This topic describes the Operation function ST_Aggr_ConvexHull and provides an example for reference.

Function declaration:

```
ST_Aggr_ConvexHull(ST_Geometry)
```

Description: This function returns a convex hull for input geometries by using aggregate transformation.

Example:

```
SELECT ST_Aggr_ConvexHull(geometry) FROM source;
-- Return the convex hull of input geometries from the data source by using aggregate transformation
.
```

1.6.12.4.2. ST_Aggr_Intersection

This topic describes the Operation function ST_Aggr_Intersection and provides an example for reference.

Function declaration:

```
ST_Aggr_Intersection(ST_Geometry)
```

Description: This function returns the intersection of input geometries by using aggregate transformation.

Example:

```
SELECT ST_Aggr_Intersection(geometry) FROM source;
-- Return the intersection of input geometries from the data source by using aggregate transformatio
n.
```

1.6.12.4.3. ST_Aggr_Union

This topic describes the Operation function ST_Aggr_Union and provides an example for reference.

Function declaration:

```
ST_Aggr_Union(ST_Geometry)
```

Description: This function returns a union of input geometries by using aggregate transformation.

Example:

```
SELECT ST_Aggr_Union(geometry) FROM source;
-- Return the union of input geometries from the data source by using aggregate transformation.
```

1.6.12.4.4. ST_Bin

This topic describes the Operation function 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

This topic describes the Operation function ST_BinEnvelop.

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

This topic describes the Operation function ST_Boundary and provides examples for reference.

Function declaration:

```
ST_Boundary(ST_Geometry)
```

Description: This function returns the boundary of the input geometry ST_Geometry.

Examples:

```
SELECT ST_Boundary(ST_LineString(0,1, 1,0))) FROM src LIMIT 1;
```

Returned results:

```
MULTIPOINT((1 0),(0 1))
```

```
SELECT ST_Boundary(ST_Polygon(1,1, 4,1, 1,4)) FROM src LIMIT 1;
```

Returned results:

```
LINESTRING(1 1, 4 1, 1 4, 1 1)
```

1.6.12.4.7. ST_Buffer

This topic describes the Operation function 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

This topic describes the Operation function ST_ConvexHull and provides an example for reference.

Function declaration:

```
ST_ConvexHull(ST_Geometry, ST_Geometry, ...)
```

Description: This function returns the geometric object ST_Geometry as the convex hull of the specified geometry.

Example:

```
SELECT ST_AsText(ST_ConvexHull(ST_Point(0, 0), ST_Point(0, 1), ST_Point(1, 1))) FROM onerow;
```

Returned results:

```
MULTIPOLYGON (((0 0, 1 1, 0 1, 0 0)))
```

1.6.12.4.9. ST_Difference

This topic describes the Operation function ST_Difference and provides examples for reference.

Function declaration:

```
ST_Difference(ST_Geometry1, ST_Geometry2)
```

Description: This function returns a geometry that indicates the difference between ST_Geometry1 and ST_Geometry2.

Examples:

```
SELECT ST_AsText(ST_Difference(ST_MultiPoint(1, 1, 1.5, 1.5, 2, 2), ST_Point(1.5, 1.5))) FROM onerow
;
```

Returned results:

```
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))) from onerow;
```

Returned results:

```
MULTIPOLYGON (((10 0, 10 10, 0 10, 0 5, 5 5, 5 0, 10 0)))
```

1.6.12.4.10. ST_Envelope

This topic describes the Operation function ST_Envelope and provides examples for reference.

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.

Examples:

```
SELECT ST_Envelope(ST_LineString(0,0, 2,2)) from src LIMIT 1;
```

Returned results:

```
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;
```

Returned results:

```
POLYGON ((2 0, 3 0, 3 3, 2 3, 2 0))
```

1.6.12.4.11. ST_ExteriorRing

This topic describes the Operation function ST_ExteriorRing and provides examples for reference.

Function declaration:

```
ST_ExteriorRing(polygon)
```

Description: This function returns the exterior ring of a polygon as a linestring.

Examples:

```
SELECT ST_ExteriorRing(ST_Polygon(1,1, 1,4, 4,1)) FROM src LIMIT 1;
```

Returned results:

```
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;
```

Returned results:

```
LINESTRING (8 0, 0 8, 0 0, 8 0)
```

1.6.12.4.12. ST_InteriorRingN

This topic describes the Operation function ST_InteriorRingN and provides an example for reference.

Function declaration:

```
ST_InteriorRingN(ST_Polygon, n)
```

Description: This function returns the n^{th} 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;
```

Returned results:

```
LINESTRING (1 1, 5 1, 1 5, 1 1)
```

1.6.12.4.13. ST_Intersection

This topic describes the Operation function ST_Intersection and provides examples for reference.

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.

Examples:

```
SELECT ST_AsText(ST_Intersection(ST_Point(1,1), ST_Point(1,1))) FROM onerow;
```

Returned results:

```
POINT (1 1)
```

```
SELECT ST_AsText(ST_Intersection(ST_GeomFromText('linestring(0 2, 0 0, 2 0)'), ST_GeomFromText('line
string(0 3, 0 1, 1 0, 3 0)'))) FROM onerow;
```

Returned results:

```
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 oner
ow;
```

Returned results:

```
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 on
erow;
```

Returned results:

```
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 on
erow;
```

Returned results:

```
MULTIPOLYGON EMPTY or LINESTRING (2 1, 3 1, 2 1)
```

1.6.12.4.14. ST_SymmetricDiff

This topic describes the ST_SymmetricDiff function and provides examples of using this function.

Function declaration:

```
ST_SymmetricDiff(ST_Geometry1, ST_Geometry2)
```

Description: This function returns a geometry that consists of the symmetric differences of the input geometries.

Examples:

```
SELECT ST_AsText(ST_SymmetricDiff(ST_LineString('linestring(0 2, 2 2)'), ST_LineString('linestring(1
2, 3 2)'))) FROM onerow;
```

Returned result:

```
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;
```

Returned result:

```
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

This topic describes the ST_Union function and provides an example of using this function.

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))) F
ROM onerow;
```

Returned result:

```
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

This topic describes the ST_Contains function and provides an example of using this function.

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

This topic describes the ST_Crosses function and provides an example of using this function.

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

This topic describes the ST_Disjoint function and provides an example of using this function.

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

This topic describes the ST_EnvIntersects function and provides an example of using this function.

Function declaration:

```
BOOLEAN ST_EnvIntersects(ST_Geometry1, ST_Geometry2)
```

Description: If the envelopes of ST_Geometry1 and ST_Geometry2 intersect, this function returns true. Otherwise, this function returns false.

Sample statements:

```
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

This topic describes the ST_Equals function and provides an example of using this function.

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

This topic describes the ST_Intersects function and provides an example of using this function.

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

This topic describes the ST_Overlaps function and provides an example of using this function.

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

This topic describes the ST_Relate function and provides an example of using this function.

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 LI
MIT 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 LI
MIT 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

This topic describes the ST_Touches function and provides an example of using this function.

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

This topic describes the ST_Within function and provides an example of using this function.

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

This topic describes the ST_GeoHash function and provides an example of using this function.

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

This topic describes the ST_PointFromGeoHash function and provides an example of using this function.

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

This topic describes the ST_EnvelopeFromGeoHash function and provides an example of using this function.

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

This topic describes the ST_GeoHashNeighbours function and provides an example of using this function.

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

This topic describes the ST_S2CellIdsFromGeom function and provides an example of using this function.

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,-71.
161144 42.25932)'), 17) as cellid;
```

1.6.12.7.2. ST_S2CellIdsFromText

This topic describes the ST_S2CellIdsFromText function and provides an example of using this function.

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 42.2593
2)', 17) as cellid;
```

1.6.12.7.3. ST_S2CellCenterPoint

This topic describes the ST_S2CellCenterPoint function and provides an example of using this function.

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

This topic describes the ST_S2CellNeighbours function and provides an example of using this function.

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

This topic describes the ST_AreaWGS84 function and provides an example of using this function.

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

This topic describes the ST_DistanceWGS84 function and provides an example of using this function.

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

This topic describes the ST_BufferWGS84 function and provides an example of using this function.

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

This topic describes the ST_GeodesicDistance function and provides an example of using this function.

Function declaration:

```
double ST_GeodesicDistance(double lon1, double lat1, double lon2, double lat2, string method = VINCENTY)
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('POINT(1
51.960336 -24.993289)'), 'LawOfCosines');
```

1.6.12.8.5. ST_Distance_Sphere

This topic describes the ST_Distance_Sphere function and provides an example of using this function.

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)'));
```

Returned result:

```
+------------+
| _c0        |
+------------+
| 460.6965312526471 |
+------------+
```

1.6.12.8.6. ST_Area_Sphere

This topic describes the ST_Area_Sphere function and provides an example of using this function.

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 40.202114,116.259024 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 40.198586,116.257564 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 40.195646,116.255628 40.195611,116.255468 40.195653,116.255385 40.195742,116.255347 40.195849,116.255258 40.197143,116.255103 40.199576,116.255078 40.200585,116.251227 40.20059,116.251098 40.203978,116.259433 40.204111,116.259247 40.203079,116.259097 40.202114))'));
```

Returned result:

```
+------------+
| _c0        |
+------------+
| 353493.765625 |
+------------+
```

1.6.12.9. R-tree index functions

1.6.12.9.1. ST_BuildRTreeIndex

This topic describes the ST_BuildRTreeIndex function and provides an example of using this function.

Function declaration:

```
RTree ST_BuildRTreeIndex(string uniqueId, string geometryWkt)
```

Description: This function is a user-defined aggregate 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

This topic describes the ST_ContainsFromRTree function.

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

This topic describes the ST_CrossesFromRTree function.

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

This topic describes the ST_EqualsFromRTree function.

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

This topic describes the ST_IntersectsFromRTree function.

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

This topic describes the ST_OverlapsFromRTree function.

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 the ST_Overlaps query.

1.6.12.9.7. ST_TouchesFromRTree

This topic describes the ST_TouchesFromRTree function.

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

This topic describes the ST_WithinFromRTree function and provides an example of using this function.

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.

Example:

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));
```

Returned result:

```
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;
```

Returned result:

```
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 |
+---------+----------+
```

1.6.12.9.9. ST_KNNFromRTree

This topic describes the ST_KNNFromRTree function and provides an example of using this function.

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.

Example:

Create an R-tree for all points in a table and use the KNN function to find the nearest point of each 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;
```

Returned result:

```
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 expe
ctations.
```

1.6.12.10. Other functions

1.6.12.10.1. ST_IsValid

This topic describes the ST_IsValid function and provides an example of using this function.

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

This topic describes the ST_Transform function and provides an example of using this function.

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, 743265 2967450,743265.625 2967416,743238 2967416))', 2249, 4326)));
SELECT ST_AsText(ST_TransformWGS84(ST_GeomFromText('POLYGON((-71.1776848522251 42.3902896512902,-71.1776843766326 42.3903829478009, -71.1775844305465 42.3903826677917,-71.1775825927231 42.3902893647987,-71.1776848522251 42.3902896512902))')));
```

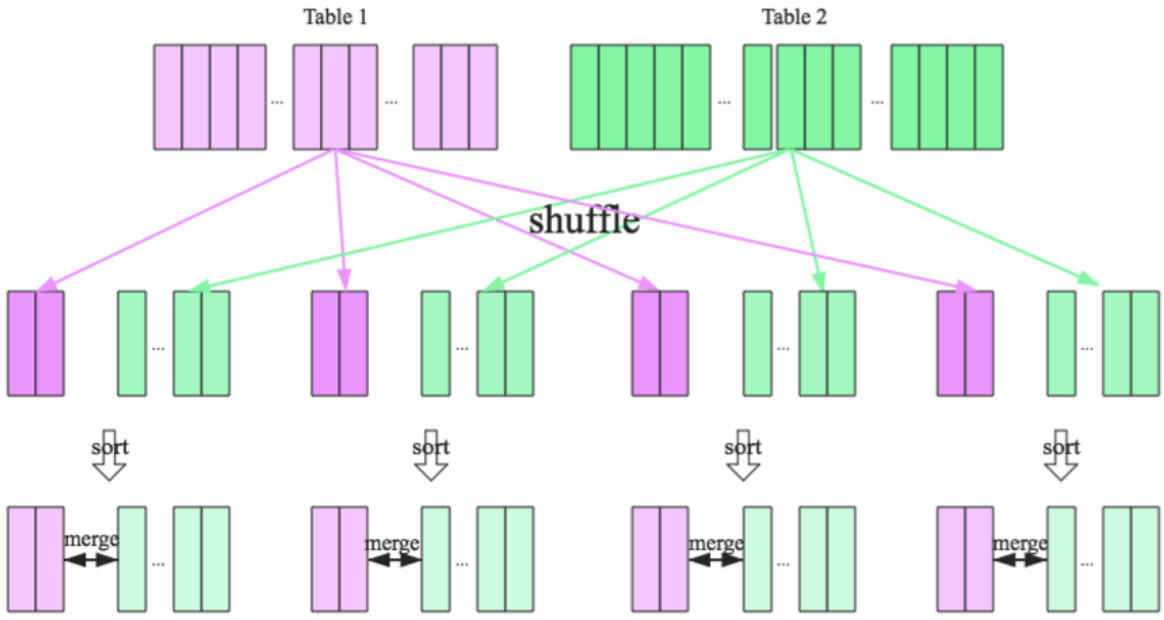
1.6.13. MaxCompute Hash Clustering

1.6.13.1. Background information

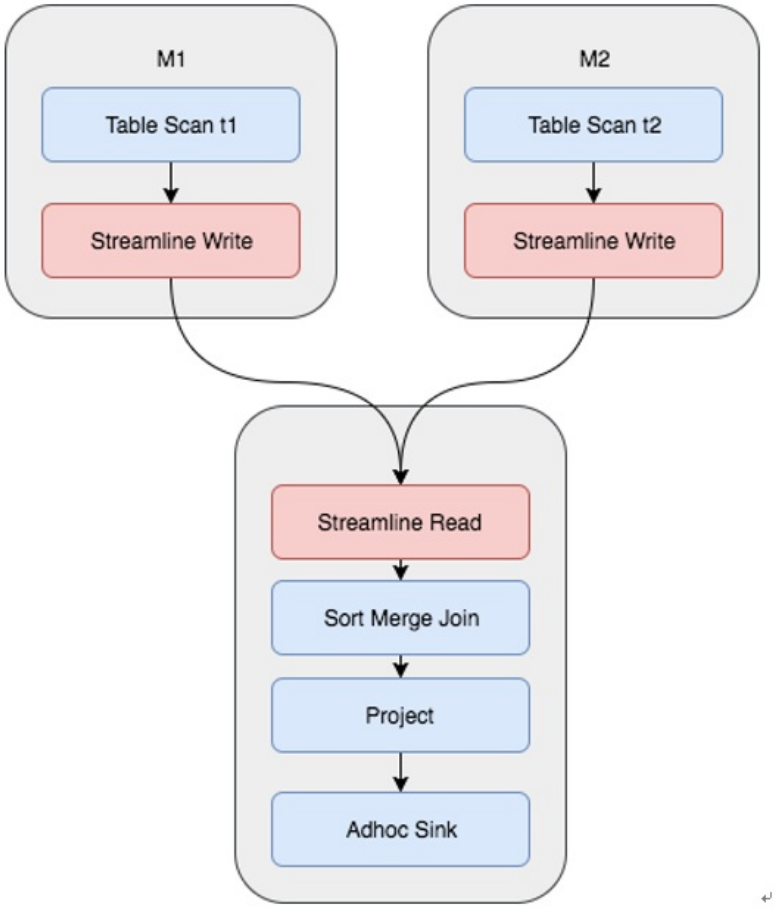
This topic describes the background of MaxCompute Hash Clustering.

JOIN operations are commonly used for queries in MaxCompute. MaxCompute provides the following methods to implement JOIN operations:

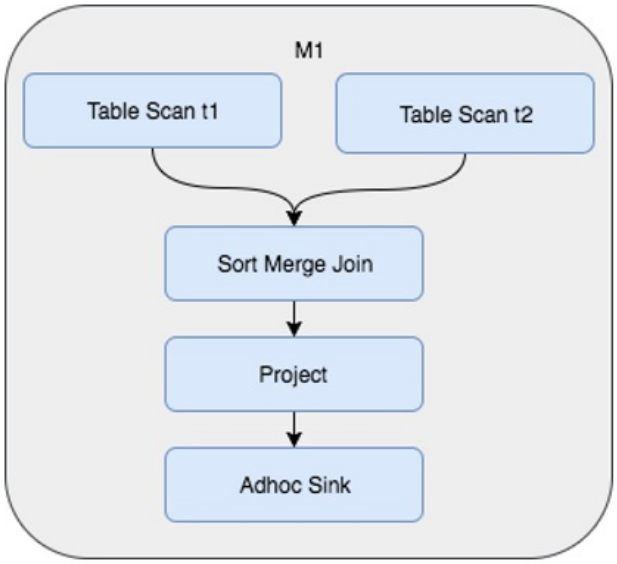
- Broadcast hash join: This method is used when a JOIN operation involves a small table. The small table is broadcasted and transferred to all instances in a join task. Then, the hash join operation is performed to join the small table with a large table.
- Shuffle hash join: This method is used when a JOIN operation involves large tables that cannot be directly broadcasted. 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 to the same instance of a join task. 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.
- 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 method is commonly used in MaxCompute because MaxCompute processes large amounts of data in most cases. If you use this method, shuffle and join operations are repeatedly performed. The physical execution plan of Job Scheduler of a JOIN operation requires multiple stages, which consumes excessive amounts of resources.



To resolve this issue, 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 repeatedly shuffled and sorted in subsequent queries. The number of stages in the physical execution plan of Job Scheduler of a JOIN operation is also 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. Then, MaxCompute optimizes the execution plan, improves the efficiency, and saves resources based on the existing storage characteristics.

1.6.13.2. Descriptions

1.6.13.2.1. Enable Hash Clustering

This topic describes how to enable the Hash Clustering feature of MaxCompute.

The Hash Clustering feature has been launched. By default, this feature is enabled. 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 at all times, contact the MaxCompute team.

> **Note** Clustered indexes improve the efficiency of queries (equivalent values or ranges) based on sort keys. You can experience the superior performance provided by Hash Clustering even if you do not add this flag.

1.6.13.2.2. Create a hash-clustered table

This topic describes how to create a hash-clustered table.

You can use the following statement to create a hash-clustered 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 a large number of 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 ensures that SQL statements can be concurrently executed 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.

1.6.13.2.3. Modify table attributes

This topic describes how to modify the attributes of a hash-clustered table.

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 |
DESC] ...])] INTO number_of_buckets BUCKETS]
ALTER TABLE table_name NOT CLUSTERED;
```

When you use the ALTER TABLE statement, take note of the following points:

- The ALTER TABLE statement can modify only 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 on 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 on 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.

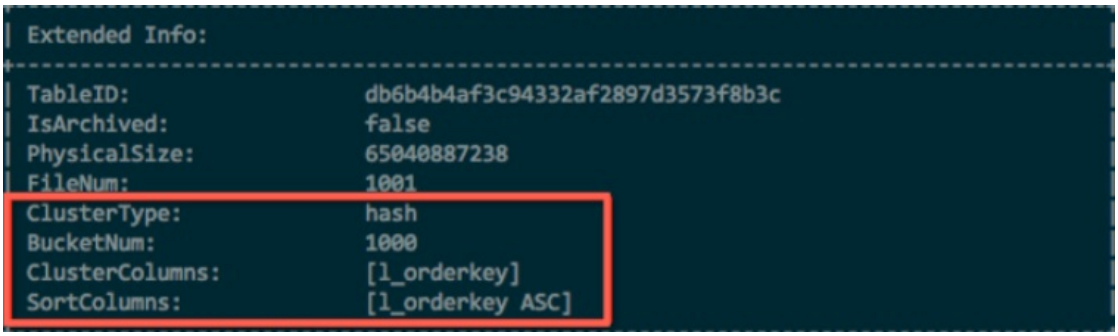
1.6.13.2.4. View and verify table attributes

This topic describes how to view and verify attributes of a hash-clustered table.

After you create a hash-clustered table, execute the following statement to view table attributes:

```
DESC EXTENDED table_name;
```

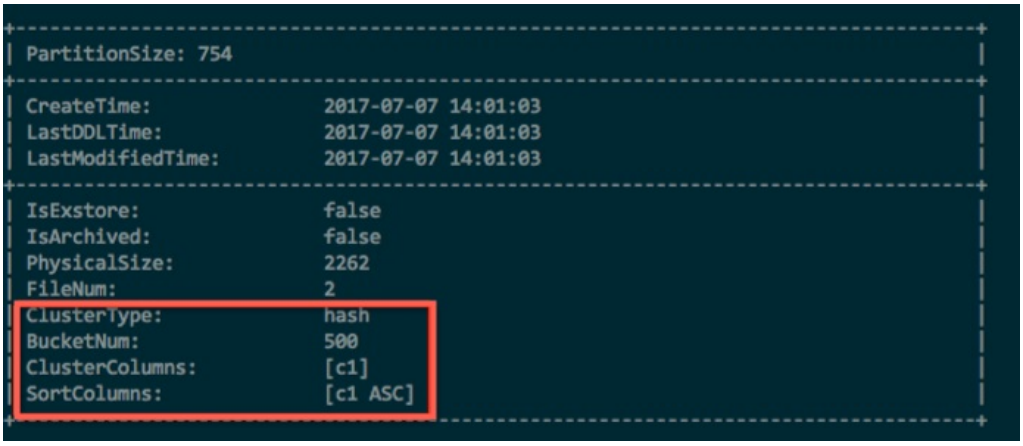
The table attributes are displayed in Extended Info.



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.



1.6.13.3. Benefits

1.6.13.3.1. Bucket pruning and index optimization

This topic describes 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.

Sample procedure:

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.

1.6.13.3.2. Aggregation optimization

This topic describes aggregation optimization.

The following code provides an example:

```
SELECT department, SUM(salary) FROM employee GROUP BY (department);
```

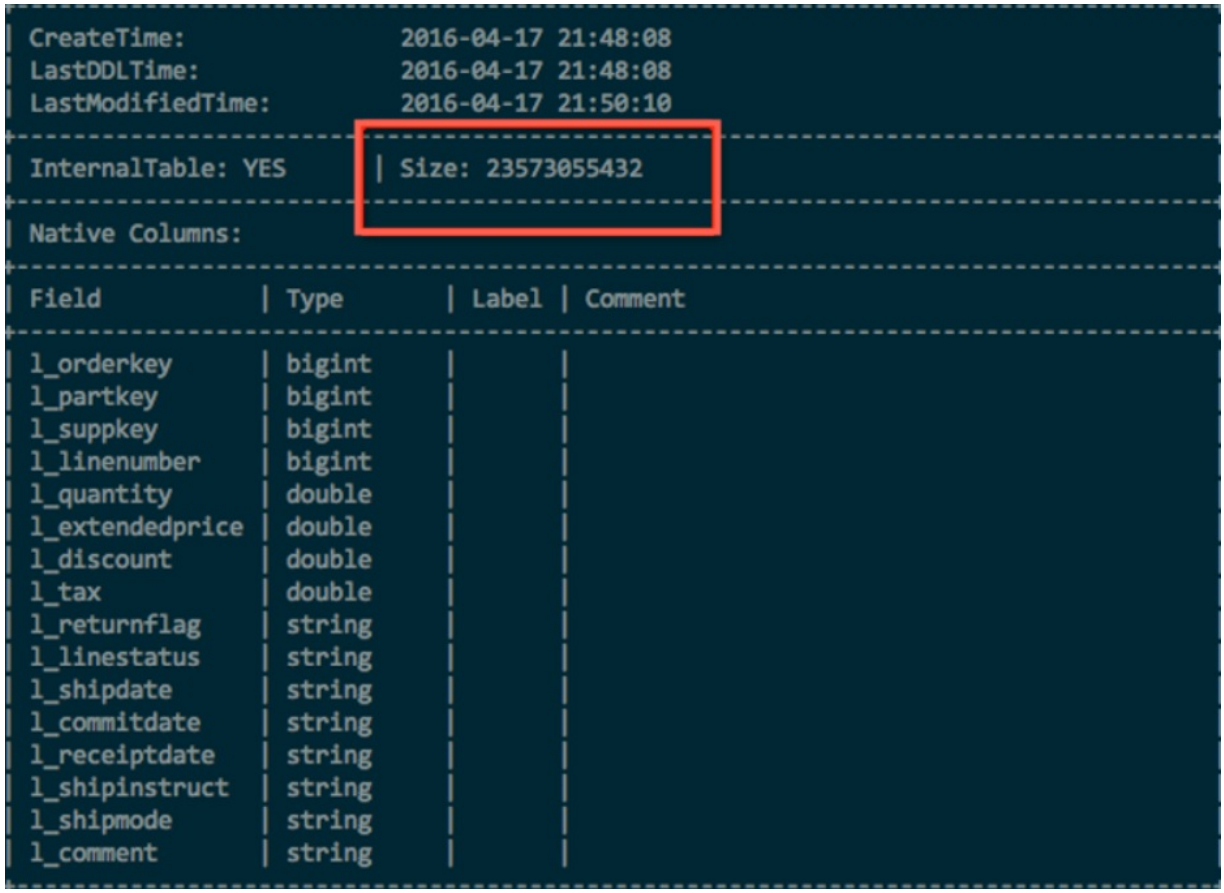
In most cases, the department column is shuffled and sorted. Then, a stream aggregate operation is performed to collect statistics on the department groups. However, if `CLUSTERED BY (department) SORTED BY (department)` is executed for the table data, the shuffle and sort operations are no longer required.

1.6.13.3.3. Storage optimization

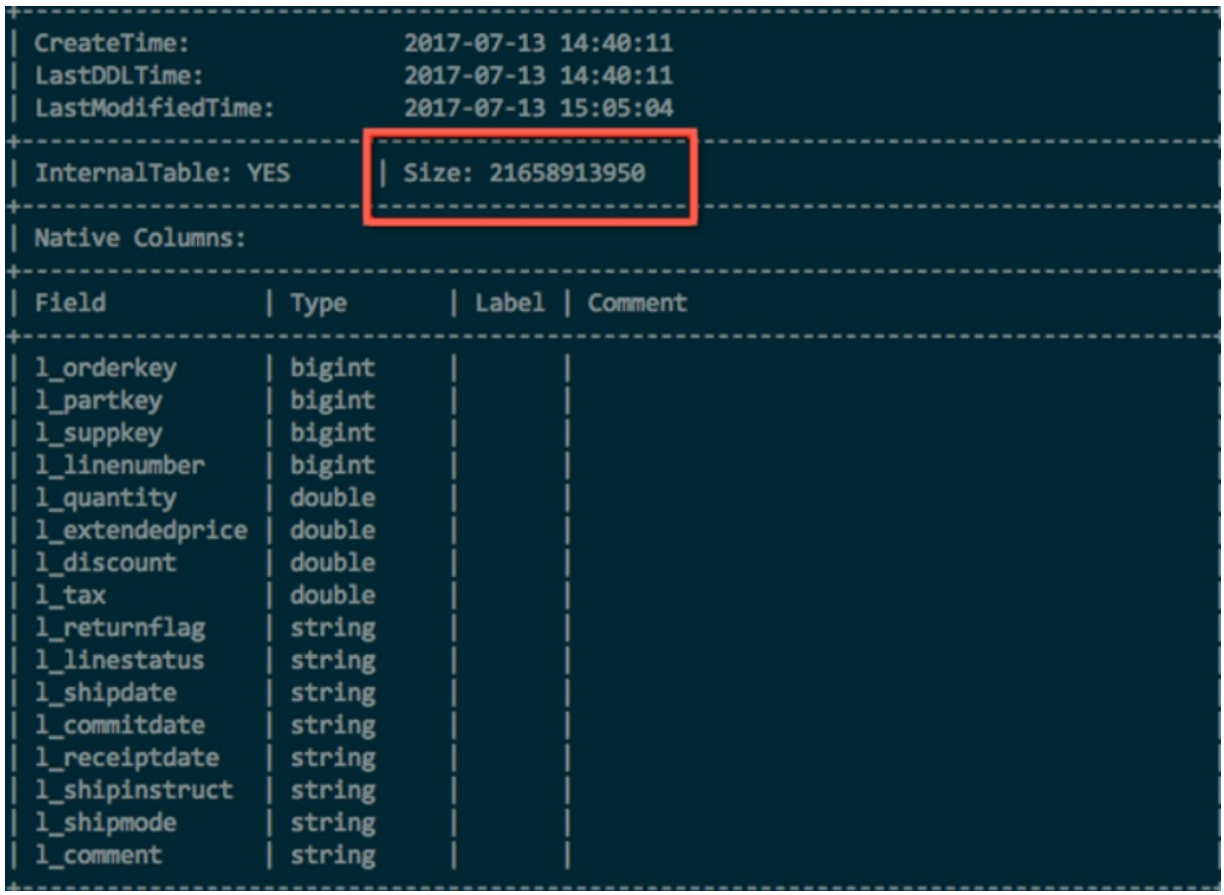
This topic describes storage optimization.

In addition to computation optimization, storage space is greatly saved if tables are shuffled and stored in a sorted manner. MaxCompute uses column store at the underlying layer. Records that have the same or similar key-value pairs are stored together by the sort function, which facilitates encoding and compression. This way, compression efficiency is improved. In some extreme cases, a sorted table can save 50% more storage space than an unsorted table. Therefore, Hash Clustering is suitable for storing tables that have long lifecycles.

The following figure shows a sample table with 100 GB of TPC-H line items and multiple data types, such as INT, DOUBLE, and STRING. If Hash Clustering is used, about 10% of the storage space is saved when 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    |       |         |
+-----------------+-----------+-------+---------+
```

1.6.13.4. ShuffleRemove

This topic describes the operations that are related to ShuffleRemove.

- Range-clustered tables support the join and aggregate operations. If a join key or group key is a range-clustered 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 specifies whether to enable this feature. By default, this feature is disabled.

- If the two hash-clustered tables you want to join have different numbers of buckets but the numbers are multiples of each other, data redistribution is not required. This improves execution efficiency.

  Usage: The `odps.optimizer.enable.hash.partial.repartitioning` flag specifies whether to enable this feature. By default, this feature is enabled.

- 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.

1.6.13.5. Limits

This topic describes limits on MaxCompute Hash Clustering.

Limits:

- The INSERT INTO statement is not supported. You can only execute the INSERT OVERWRITE statement to add data.
- Small files cannot be merged. Most data is evenly distributed in buckets when it is split. Therefore, the number of small files is not large. 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 the data uploaded by using Tunnel is unsorted.

1.6.14. Common MaxCompute SQL parameter settings

1.6.14.1. Map configurations

This topic describes Map configurations.

```
set odps.sql.mapper.cpu=100;
```

Description: specifies the number of CPUs used by each instance in a Map task. Default value: 100. Valid values: [50,800].

```
set odps.sql.mapper.memory=1024;
```

Description: specifies the memory size of each instance in a Map task. Unit: MB. Default value: 1024. Valid values: [256,12288].

```
set odps.sql.mapper.merge.limit.size=64;
```

Description: specifies the maximum size of control files to be merged. Unit: MB. Default value: 64. Valid values: [0,Integer.MAX_VALUE]. You can configure this variable to control the input of mappers.

```
set odps.sql.mapper.split.size=256;
```

Description: specifies the maximum data input volume for a Map task. Unit: MB. Default value: 256. Valid values: [1,Integer.MAX_VALUE]. You can configure this variable to control the input of mappers.

1.6.14.2. Join configurations

This topic describes Join configurations.

```
set odps.sql.joiner.instances=-1;
```

Description: specifies the number of instances of a join task. Default value: -1. Valid values: [0,2000].

```
set odps.sql.joiner.cpu=100;
```

Description: specifies the number of CPUs used by each instance of a join task. Default value: 100. Valid values: [50,800].

```
set odps.sql.joiner.memory=1024;
```

Description: specifies the memory size of each instance of a join task. Unit: MB. Default value: 1024. Valid values: [256,12288].

1.6.14.3. Reduce configurations

This topic describes Reduce configurations.

```
set odps.sql.reducer.instances=-1;
```

Description: specifies the number of instances in a Reduce task. Default value: -1. Valid values: [0,2000].

```
set odps.sql.reducer.cpu=100;
```

Description: specifies the number of CPUs used by each instance in a Reduce task. Default value: 100. Valid values: [50,800].

```
set odps.sql.reducer.memory=1024;
```

Description: specifies the memory size of each instance in a Reduce task. Unit: MB. Default value: 1024. Valid values: [256,12288].

1.6.14.4. UDF configurations

This topic describes the configurations of user-defined functions (UDFs).

```
set odps.sql.udf.jvm.memory=1024;
```

Description: specifies the maximum memory size used by the UDF JVM heap. Unit: MB. Default value: 1024. Valid values: [256,12288].

```
set odps.sql.udf.timeout=600;
```

Description: specifies the timeout period of a UDF. Unit: seconds. Default value: 600. Valid values: [0,3600].

```
set odps.sql.udf.python.memory=256;
```

Description: specifies the maximum memory size used by the UDF Python API. Unit: MB. Default value: 256. Valid values: [64,3072].

```
set odps.sql.udf.optimize.reuse=true/false;
```

Description: When this parameter is set to true, each UDF function expression can be calculated only once. This improves performance. Default value: true.

```
set odps.sql.udf.strict.mode=false/true;
```

Description: specifies whether functions return NULL or an error if dirty data is found. If the parameter is set to true, an error is returned. Otherwise, NULL is returned.

1.6.14.5. MapJoin configurations

This topic describes MapJoin configurations.

```
set odps.sql.mapjoin.memory.max=512;
```

Description: specifies the maximum memory size for a small table that is involved in a MapJoin operation. Unit: MB. Default value: 512. Valid values: [128,2048].

```
set odps.sql.reshuffle.dynamicpt=true/false;
```

Description:

- Dynamic partitioning is time-consuming in some scenarios. Disabling dynamic partitioning can accelerate SQL.
- If the number of dynamic partitions is small, disabling dynamic partitioning can prevent data skew.

1.6.14.6. Configurations of data skew

This topic describes configurations of data skew.

```
set odps.sql.groupby.skewindata=true/false;
```

Description: allows you to enable GROUP BY optimization.

```
set odps.sql.skewjoin=true/false;;
```

Description: allows you to enable JOIN optimization. This is effective only when odps.sql.skewinfo is specified.

```
set odps.sql.skewinfo;
```

Description: allows you to configure detailed information for JOIN optimization. Command syntax:

```
set odps.sql.skewinfo=skewed_src:(skewed_key)[("skewed_value")]
```

Example:

Configure a single skewed value for a single field.

```
set odps.sql.skewinfo=src_skewjoin1:(key)[("0")];
-- Output result: explain select a.key c1, a.value c2, b.key c3, b.value c4 from src a join src_skew
join1 b on a.key = b.key;
```

Configure multiple skewed values for a single field.

```
set odps.sql.skewinfo=src_skewjoin1:(key)[("0")("1")];
-- Output result: explain select a.key c1, a.value c2, b.key c3, b.value c4 from src a join src_skew
join1 b on a.key = b.key;
```

1.6.15. MapReduce-to-SQL conversion for execution

1.6.15.1. Overview

This topic provides an overview of the MapReduce-to-SQL conversion for execution feature.

MaxCompute provides a series of Java APIs for MapReduce to process data.

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.

> **Note**
> - You do not need to change the original APIs and job logic.
> - Only MapReduce jobs of the OpenMR type, which are written by using MapReduce APIs, can be converted to SQL.
> - This feature can be used only 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-clustered tables.
> - This feature supports near-real-time execution of small jobs.

1.6.15.2. Configure the MaxCompute client

This topic describes how to configure the MaxCompute client.

1. Download the latest MaxCompute client package to your computer and configure the client.

   > **Note** On the download page, you can obtain the MaxCompute client package of the required version in the **Other Resources** section.

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 run by MapReduce. The new compiler, optimizer, and execution engine are not used.

   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. In this case, the configuration takes effect on all jobs in the project. Therefore, only the project administrator can apply to 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 run by MapReduce. If SQL execution fails in sql mode, an error is reported.
   - Method 2: Enable the execution mode at the session level. This method applies only to 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 job parameters. Example:

       ```
       JobConf job = new JobConf();
       job.set("odps.mr.run.mode","hybrid")
       ```

   MaxCompute O&M personnel can enable the execution mode at the project level at a later point in time.

1.6.15.3. Configure running settings in DataWorks

This topic describes how to configure running settings in DataWorks.

Jobs that run in DataWorks are updated by the O&M personnel of MaxCompute and DataWorks. You do not need to update your client.

- Enable the conversion for a single job.

  You can add a set flag before the statement of a MapReduce job or configure job parameters for the set flag. These methods take effect at the session level and apply only to the current job.

  The following examples show how to use these methods:

- Add a set flag, such as `set odps.mr.run.mode=hybrid;` before JAR statements.
- Configure job parameters. Example:

```
JobConf job = new JobConf();
job.set("odps.mr.run.mode","hybrid")
```

- Enable the conversion at the project level by setting odps.mr.run.mode for a project.

1.6.15.4. View details

This topic describes how to view MapReduce-to-SQL conversion results and running details of SQL jobs.

You can use Logview and MaxCompute Studio to view MapReduce-to-SQL conversion results and running details of SQL jobs.

- 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 about the node. Example:

```
create temporary function mr2sql_mapper_152955927079392291755 as   'com.aliyun.odps.mapred.bridge.
LotMapperUDTF' using ;
create temporary function mr2sql_reducer_152955927079392291755 as 'com.aliyun.odps.mapred.bridge.L
otReducerUDTF' 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_sto
ck,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(
SELECT mr2sql_mapper_152955927079392291755(id,gmt_create,gmt_modified,product_id,admin_seq,sku_att
r,sku_price,sku_stock,sku_code,sku_image,delivery_time,sku_bulk_order,sku_bulk_discount,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_att
r,sku_price,sku_stock,sku_code,sku_image,delivery_time,sku_bulk_order,sku_bulk_discount,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_bulk_order,v_s
ku_bulk_discount,v_sku_image_version,v_currency_code) as (id,gmt_create,gmt_modified,product_id,ad
min_seq,sku_attr,sku_price,sku_stock,sku_code,sku_image,delivery_time,sku_bulk_order,sku_bulk_disc
ount,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,sku_i
mage,delivery_time,sku_bulk_order,sku_bulk_discount,sku_image_version,currency_code ;
```

- Logview details or summary

  A new execution engine is used to run jobs.

```
Job run mode: fuxi job
Job run engine: execution engine
```

- Logview details or JSON summary

  The JSON summary information in MapReduce contains only the input and output information about 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}]])
]
```

1.6.15.5. Perform operations on the distributed file system

This topic describes how to perform operations on the distributed file system.

Procedure

1. Specify volume files.

   You can use one 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();
   ```

1.6.16. Analysis of the mapping between SQL input and output fields

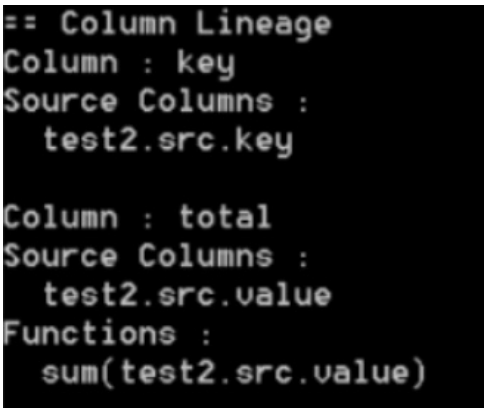
1.6.16.1. Overview

This topic describes the feature of analyzing the mapping between fields in input and output tables.

Example:

```
select key, sum(value) as total from  src group by key;
```

The result that is shown in the following figure 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.

1.6.16.2. Usage notes

This topic provides an example on how to use the feature of analyzing the mapping between fields in input and output tables.

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

Example

```
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 submit SQL jobs.

```
lineage_test>set odps.sql.task.mode=LINEAGE;
-- OK is returned. Continue to run the following command:
odps@ lineage_test>set odps.sql.select.output.format=Humanreadable;
-- OK is returned. Continue to run the following command:
lineage_test>select * from dual;
```

Returned result:

```
== Column Lineage
Column : id
Source Columns :
  test2.dual.id
```

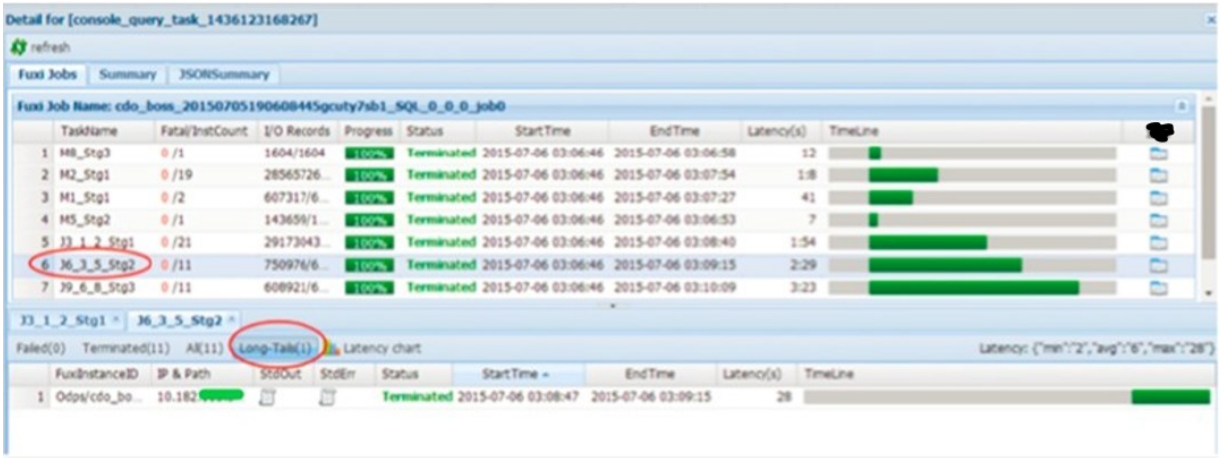
1.6.17. Common MaxCompute SQL errors and solutions

1.6.17.1. Data skew

1.6.17.1.1. Overview

This topic provides an overview of data skew issues.

In most cases, data skew occurs if the min, max, and avg values of the instance time, input records, and output records parameters are unbalanced in an instance on which jobs run. For example, if the max value is significantly larger than the avg value, data skew may occur. You can use Logview to identify data skew issues, as shown in the following figure.



The Long Tails tab of each task displays the instances on which data skew occurs. The root cause of data skew is that the amount of data processed by a specific instance far exceeds that processed by other instances. As a result, the running time of this instance is also much longer than the average running time of other instances, and the entire job slows down.

Different types of data skew issues in SQL are resolved by using different methods. The subsequent topics describe possible causes of data skew issues and provide solutions.

1.6.17.1.2. GROUP BY data skew

This topic describes possible causes of the GROUP BY data skew and provides a solution.

Possible cause: GROUP BY keys are unevenly distributed, which results in data skew on reducers.

Solution: Set the anti-skew parameter before you execute SQL statements.

```
set odps.sql.groupby.skewindata=true;
```

Note If you set this parameter to true, when the Shuffle hash algorithm is used, the system automatically adds a random factor and introduces a new task to prevent data skew.

1.6.17.1.3. DISTRIBUTE BY data skew

This topic describes possible causes of the DISTRIBUTE BY data skew and provides a solution.

Possible cause: Constants are used to execute the DISTRIBUTE BY clause for full sorting of the entire table. This results in data skew on reducers.

Solution: Avoid the preceding operation.

1.6.17.1.4. Data skew caused by JOIN operations

This topic describes the data skew caused by JOIN operations and provides a solution.

Possible cause: JOIN ON keys are unevenly distributed. For example, a key has a large number of duplicates in multiple tables that you want to join. This results in a Cartesian explosion of data in the instance on which the JOIN statement is executed and causes data skew.

Solution: You can use one of the following methods to resolve the data skew issue based on scenarios:

- If one of the tables that you want to join is a small table, execute the MAP JOIN statement, instead of the JOIN statement.
- Use separate logic to handle skewed keys. For example, if data skew occurs because a large number of null key values exist in both tables, you must filter out these null key values or use a CASE WHEN expression to replace the null key values with random values before you perform a JOIN operation.
- If you do not want to change the SQL statement, configure 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;
```

1.6.17.1.5. Data skew caused by multiple DISTINCT operations

This topic describes the data skew caused by multiple DISTINCT operations and provides a solution.

Possible cause: Multiple DISTINCT operations aggravate the GROUP BY skew.

Solution: Execute two GROUP BY clauses to alleviate the data skew issue. In most cases, do not use multiple DISTINCT operations at the same time.

1.6.17.1.6. Data skew caused by misuse of dynamic partitions

This topic describes the data skew caused by misuse of dynamic partitions and provides a solution.

Possible cause: In scenarios where dynamic partitions are used, if K Map instances and N destination partitions are deployed, K × N small files may be generated. This significantly increases the management workload of the file system. Therefore, the following configuration takes effect by default:

```
set odps.sql.reshuffle.dynamicpt=true;
```

An additional level of Reduce task is introduced. The same or a small number of other Reduce instances are used to write data to the same destination partition. This prevents a large number of small files from being generated. However, dynamic partition shuffle may cause data skew.

Solution: If the number of destination partitions is small, only an appropriate number of small files are generated. In this case, you can run the following command to set odps.sql.reshuffle.dynamicpt to false to prevent the data skew caused by dynamic partition shuffle.

```
set odps.sql.reshuffle.dynamicpt=false;
```

If the number of destination partitions is large, we recommend that you do not use dynamic partitions because a large number of small files are generated.

1.6.17.1.7. Use SKEWJOIN HINT to avoid skewed hot key values

This topic describes how to avoid skewed hot key values by using SKEWJOIN HINT. It also provides examples for reference.

Methods

- Method 1: Specify the name of a table in SKEWJOIN HINT. Note that the name specified in the hint is the alias of the table. The following statement shows an example:

  ```
  select /*+ skewjoin(a) */ * from T0 a join T1 b on a.c0 = b.c0 and a.c1 = b.c1;
  ```

- Method 2: Specify the table name and possibly skewed columns in SKEWJOIN HINT. In the following statement, the c0 and c1 columns of Table a are skewed columns.

  ```
  select /*+ skewjoin(a(c0, c1)) */ * from T0 a join T1 b on a.c0 = b.c0 and a.c1 = b.c1 and a.c2 =
  b.c2;
  ```

- Method 3: Specify the table name, columns, and skewed hot key values in SKEWJOIN HINT. If the skewed key values are of the STRING type, enclose each value with double quotation marks ("). In the following statement, (a.c0=1 and a.c1="2") and (a.c0=3 and a.c1="4") contain skewed hot key values.

  ```
  select /*+ skewjoin(a(c0, c1)((1, "2"), (3, "4"))) */ * from T0 a join T1 b on a.c0 = b.c0 and a.c
  1 = b.c1 and a.c2 = b.c2;
  ```

Differences

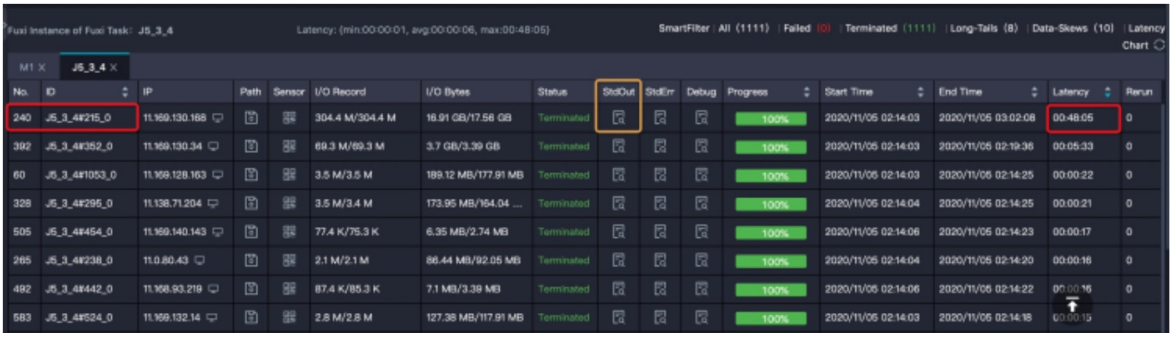
- Method 3 in which the skewvalue parameter is specified is more efficient than Method 1 and Method 2 in which the skewvalue parameter is not specified.
- .If you use Analyze and Freeride to collect TopK skewed hot key values, SKEWJOIN may automatically take effect without requiring you to specify a hint.

Examples

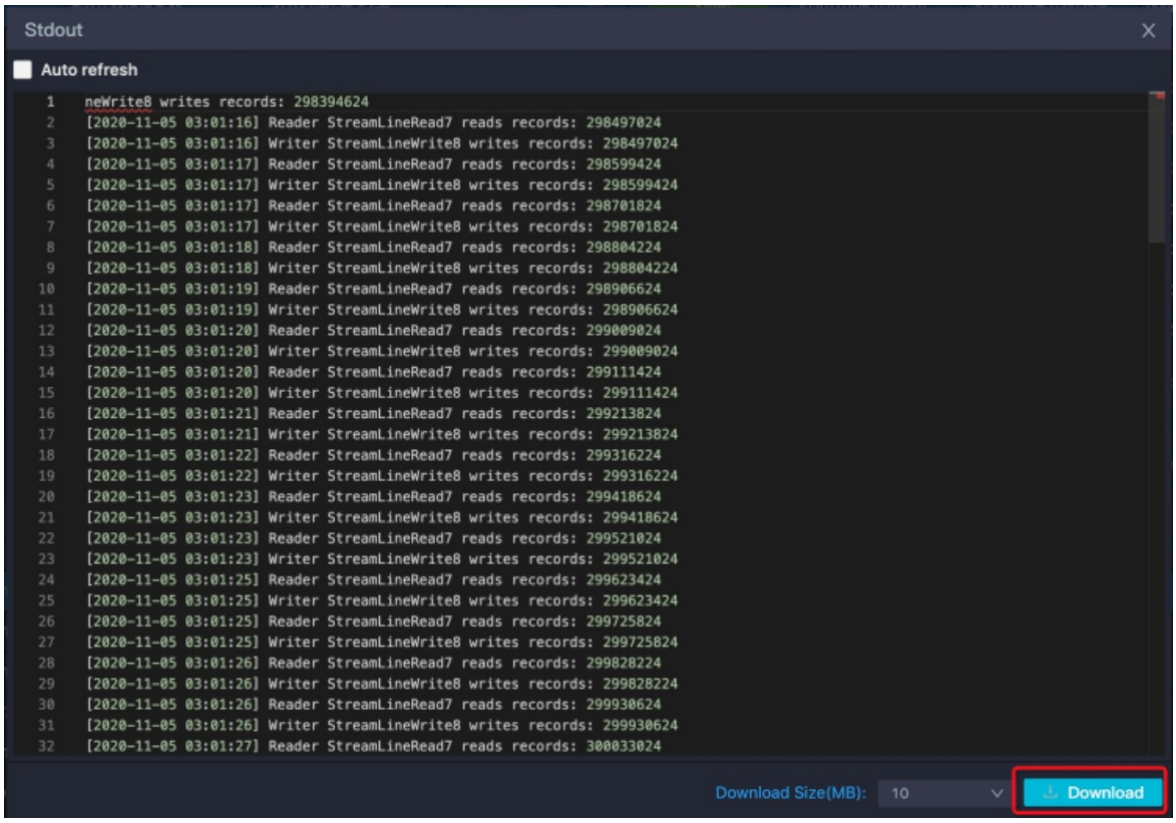
1. Identify the JOIN statement that causes data skew.

In the following screenshot captured on Logview, J5_3_4 is the Fuxi task that took the longest time to run.

Click the J5_3_4 task and query the instances of this task on the tab that appears. The query results show that the J5_3_4#215_0 instance takes the longest time to run and its I/O records and I/O bytes are much more than those of other instances.

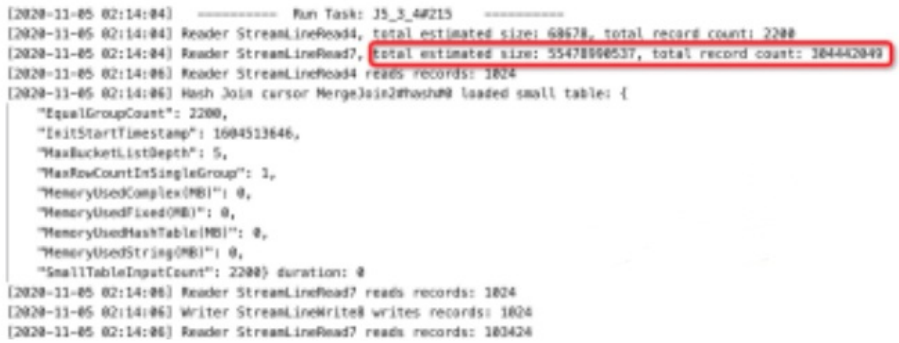


This indicates that data skew occurs on the J5_3_4#215_0 instance. However, the JOIN statement that causes the data skew needs to be further determined. You can open the stdout of the skewed instance and the stdout of a non-skewed instance. The StdOut of the skewed instance is enclosed in the yellow rectangle in the preceding figure. The Stdout dialog box shown in the following figure appears. In most cases, the web page cannot display all the content of stdout. To view all the content, you can click **Download** in this dialog box.



The following figures show that the value of record count in StreamLineRead7 of the skewed instance is much greater than that of the non-skewed instance. Therefore, data skew occurs when data in StreamLineWrite7 and SreamLineRead7 is shuffled.

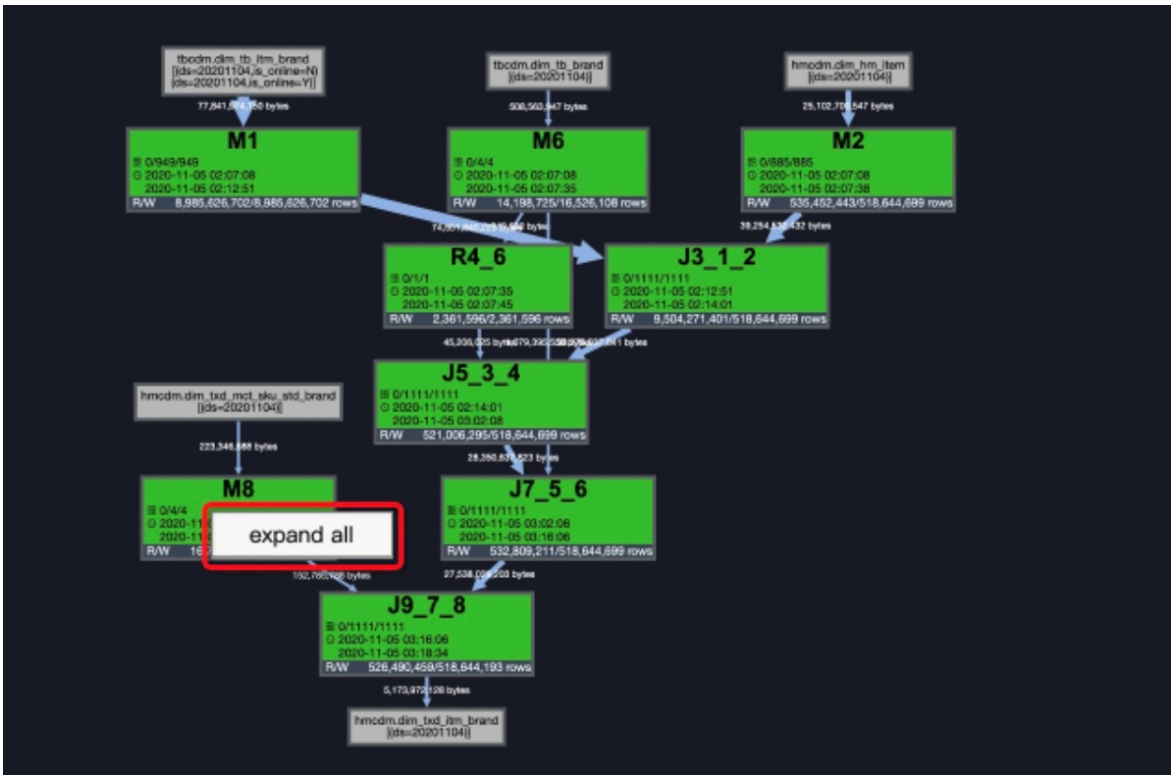
Skewed instance

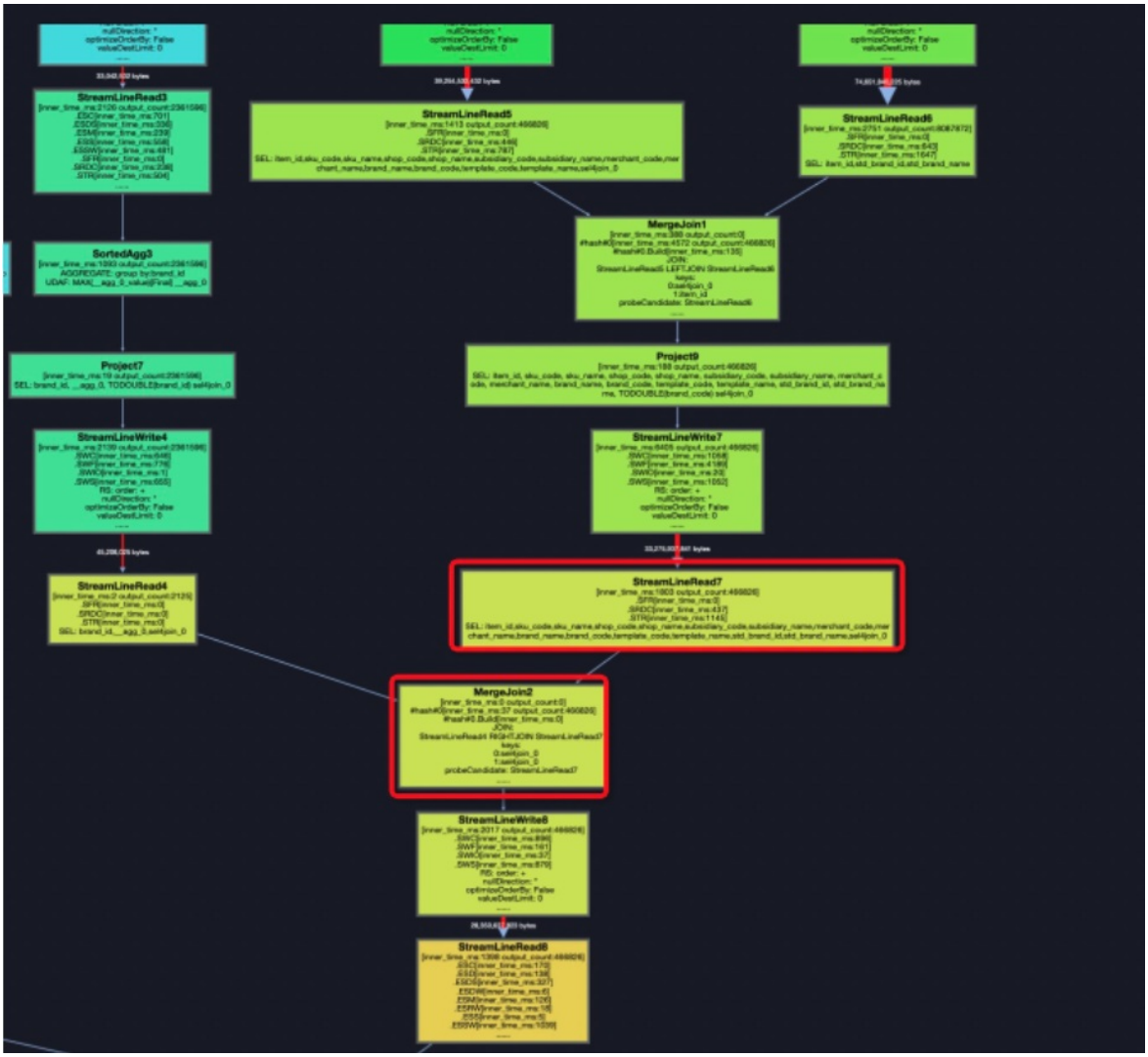


Non-skewed instance

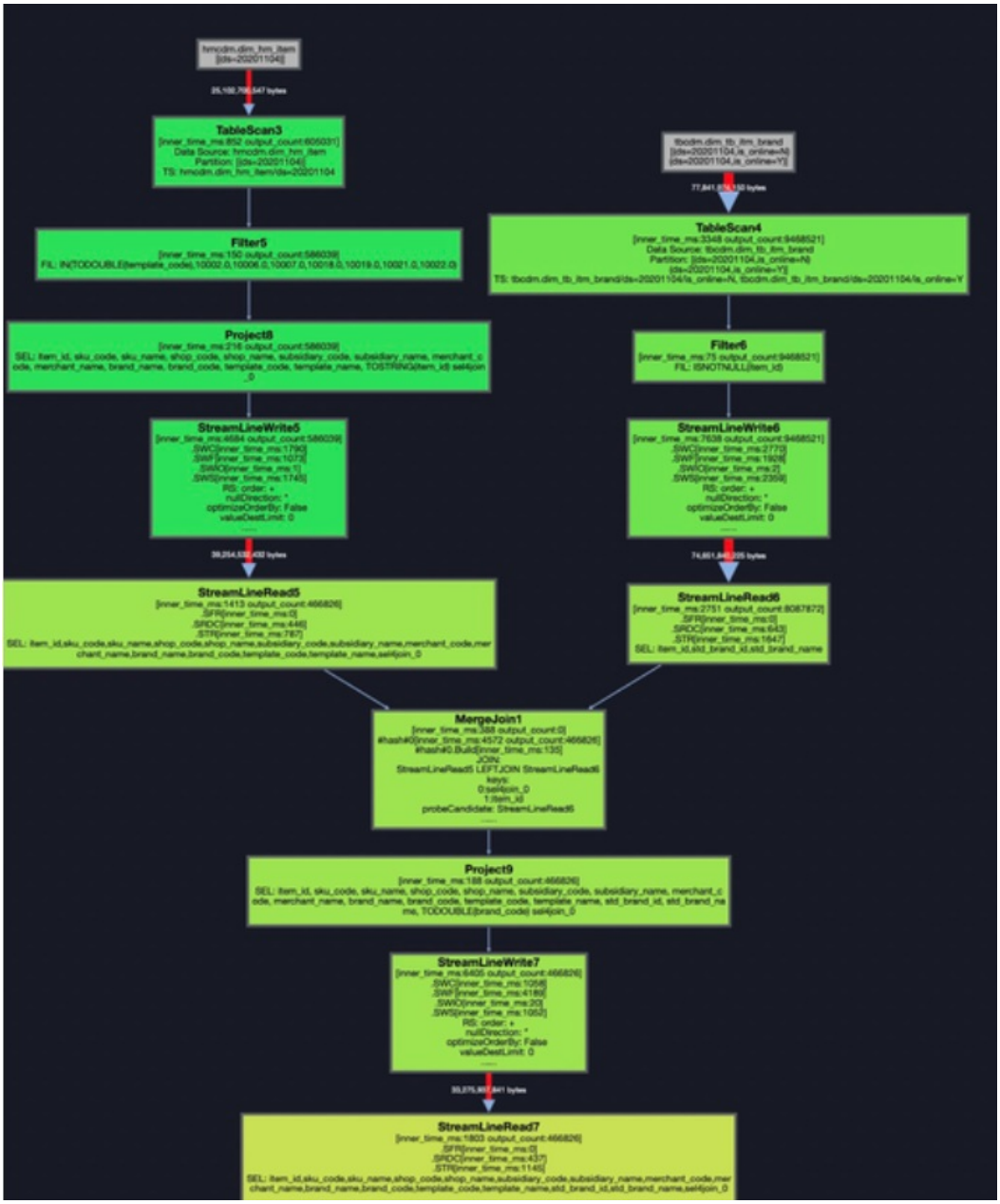
```
[2020-11-05 02:14:04]   ----------  Run Task: J5_3_4#1053   ----------
[2020-11-05 02:14:04] Reader StreamLineRead4, total estimated size: 69105, total record count: 2150
[2020-11-05 02:14:04] Reader StreamLineRead7, total estimated size: 605691502, total record count: 3534181
[2020-11-05 02:14:06] Reader StreamLineRead4 reads records: 1024
[2020-11-05 02:14:06] Hash Join cursor MergeJoin2#hash#0 loaded small table: {
    "EqualGroupCount": 2150,
    "InitStartTimestamp": 1604513646,
    "MaxBucketListDepth": 5,
    "MaxRowCountInSingleGroup": 1,
    "MemoryUsedComplex(MB)": 0,
    "MemoryUsedFixed(MB)": 0,
    "MemoryUsedHashTable(MB)": 0,
    "MemoryUsedString(MB)": 0,
    "SmallTableInputCount": 2150} duration: 0
[2020-11-05 02:14:06] Reader StreamLineRead7 reads records: 1024
[2020-11-05 02:14:06] Writer StreamLineWrite8 writes records: 1024
[2020-11-05 02:14:06] Reader StreamLineRead7 reads records: 103424
[2020-11-05 02:14:06] Writer StreamLineWrite8 writes records: 103424
[2020-11-05 02:14:07] Reader StreamLineRead7 reads records: 205824
[2020-11-05 02:14:07] Writer StreamLineWrite8 writes records: 205824
[2020-11-05 02:14:07] Reader StreamLineRead7 reads records: 308224
[2020-11-05 02:14:07] Writer StreamLineWrite8 writes records: 308224
```

On the directed acyclic graph (DAG), you can right-click the skewed instance and select **expand all** to find StreamLineWrite7 and StreamLineRead7.





The DAG shows that data skew occurs on StreamLineRead7 in MergeJoin2. MergeJoin2 is generated after the dim_hm_item and dim_tb_itm_brand tables are joined, and then the generated new table and the dim_tb_brand table are joined.



2. Locate the join based on the names of the related tables in Step 1. The result shows that data skew occurs in LEFT OUTER JOIN in the SQL statements and table t1 has data skew. To resolve this data skew issue, add the following command to the SQL statements:

```
/*+ skewjoin(t1) */
```

```
        t1.sku_code,
        t1.sku_name,
        t1.brand_code,
        t1.brand_name,
        t2.std_brand_id,
        t2.std_brand_name
    from hmcdm.dim_hm_item t1
    left outer join tbcdm.dim_tb_itm_brand t2
    on cast(t1.item_id as string)=cast(t2.item_id as string)
    and t2.ds=max_pt('tbcdm.dim_tb_itm_brand')
    where t1.ds='20201104'
    and t1.template_code in (
        10002,
        10006,
        10007,
        10018,
        10019,
        10021,
        10022
    )
) t1
left outer join(
    select brand_id as std_brand_id, max(brand_name) as std_brand_name
    from tbcdm.dim_tb_brand
    where ds=max_pt('tbcdm.dim_tb_brand')
    and is_standard_brand='Y'
    and cast(brand_id as bigint)>0
    group by brand_id
) t21
on t1.brand_code=t21.std_brand_id
left outer join(
```

Usage notes

- Supported join types: INNER JOIN allows you to specify a hint for either the right or left table of the JOIN operation. LEFT JOIN, SEMI JOIN, and ANTI JOIN allow you to specify a hint only for the left table. RIGHT JOIN allows you to specify a hint only for the right table. FULL JOIN does not support SKEWJOIN HINT.
- An Aggregate is run after SKEWJOIN HINT is added, which slows down the JOIN operation. Therefore, we recommend that you add SKEWJOIN HINT only to the JOIN statements that cause data skew.
- The data type of Left Side Join Key must be the same as that of Right Side Join Key for the JOIN statement to which SKEWJOIN HINT is added. If the data types are different, SKEWJOIN HINT becomes ineffective.
- After a hint is added, the optimizer runs an Aggregate to dynamically obtain the hot key values of the first 20 rows with the most duplicate key values. 20 is a default value. You can add the following flag parameter and specify the number of the rows whose hot key values you want to return.

  ```
  set odps.optimizer.skew.join.topk.num = xx;
  ```

- You can specify a hint only for one of the two tables in the JOIN operation.
- .In the JOIN statement to which SKEWJOIN HINT is added, left key = right key must be included. SKEWJOIN HINT cannot be added to a CARTESIAN JOIN statement.
- The following statement shows how to use SKEWJOIN HINT with other hints. Note that SKEWJOIN HINT cannot be added to a JOIN statement to which MAPJOIN HINT is added.
- On the Json Summary tab of Logview, you can search for the topk_agg field to check whether SKEWJOIN HINT takes effect. If such a field exists, SKEWJOIN HINT takes effect.

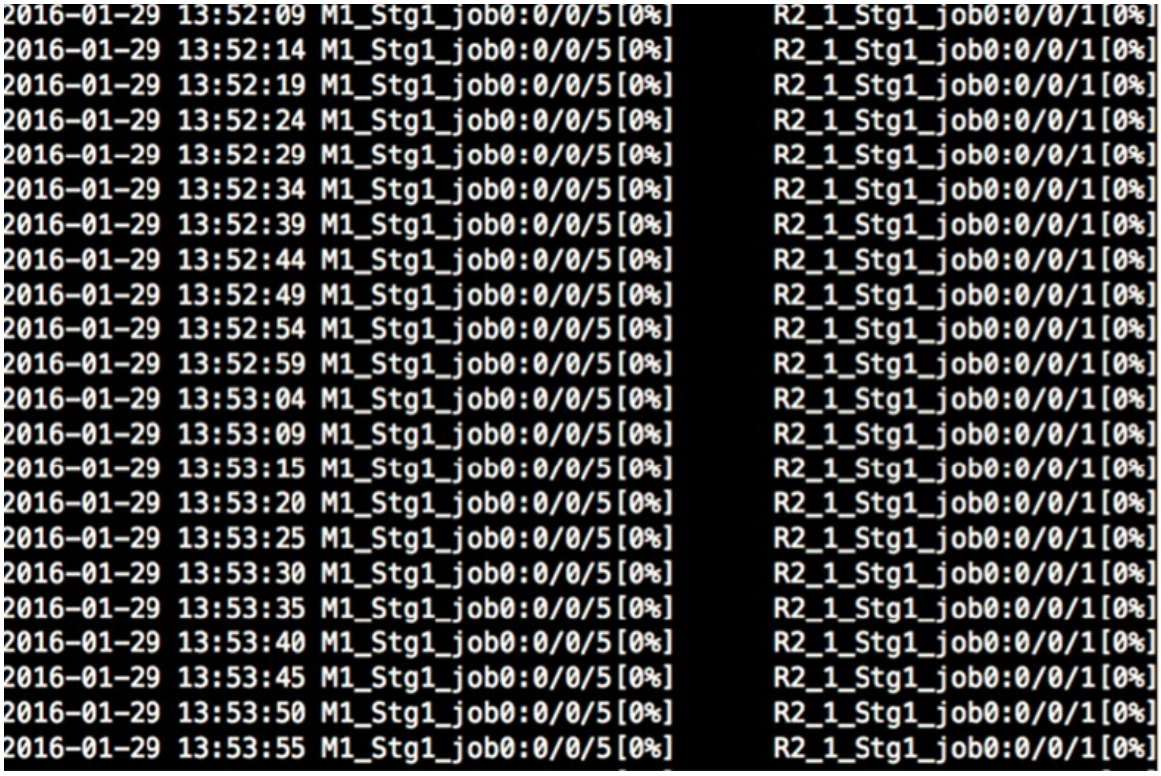
1.6.17.2. Insufficient computing resources

This topic describes the issue of insufficient computing resources and provides a solution.

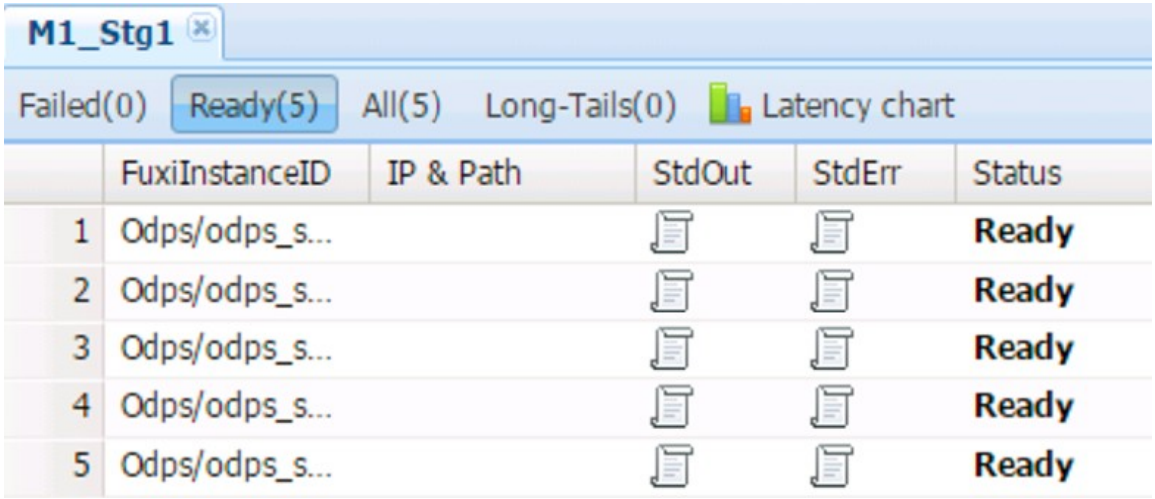
Computing resources in MaxCompute may be insufficient due to improper planning and misuse of cluster resources.

Jobs with insufficient computing resources have the following common characteristics:

- The job output stops at a stage and the progress remains unchanged. For example, in the following figure, the progress of M1_Stg1 in the job remains at 0%. R2_1_Stg1 depends on M1_Stg1. Therefore, the progress of R2_1_Stg1 remains at 0% before M1_Stg1 is complete.



- Task instances are in the Ready state on Logview, as shown in the following figure. This state indicates that resources are insufficient to run these task instances. After an instance obtains the necessary resources, its state changes to Running. If an instance is in the Waiting state, the instance waits for the completion of the task on which it depends.



Each job is split into multiple tasks based on an execution plan. These tasks form a directed acyclic graph (DAG). Multiple instances are concurrently invoked in each task to complete the computation. In most cases, the resources required to invoke an instance are a single-core CPU and 2 GB memory. To properly allocate resources, a quota group is assigned to each project. The quota group determines the upper limit of resources (CPU and memory) that are available for all jobs in the project. If the resources used by the jobs that concurrently run reach the upper limit of the quota group, the jobs stop running due to insufficient resources.

You can use one of the following methods to resolve this issue:

- Run jobs in off-peak hours.
- Contact O&M personnel to increase the resource capacity of the quota group for the project.

1.6.17.3. Methods to optimize storage in MaxCompute SQL

This topic describes the methods to optimize storage in MaxCompute SQL.

Reasonably configure partitioned tables

MaxCompute supports the concept of partitioning in a table. A partition refers to a partition space that is specified when a table is created. That is, a few fields in the table serve as partition key columns. In most cases, you can consider a partition as a directory in a file system. Each value of a partition key 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 a partition you want to access, the system reads data only from that partition and does not scan the entire table. This reduces costs and improves efficiency.

Example of a partitioned table:

```
create table src (key string, value bigint) partitioned by (pt string);
select * from src where pt='20160901';
-- Specifies the partitioning format. MaxCompute takes only the data in the 20160901 partition as th
e input when MaxCompute generates a query plan.
```

Example of a non-partitioned table:

```
select * from src where key = 'MaxCompute';
-- The entire table is scanned in a query plan.
```

You can configure partitions by date or geographical region. You can also configure 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 level-1 partitions are specified by sale_date, and
level-2 partitions are specified by region.
```

Reasonably configure the table lifecycle

Storage resources in MaxCompute are valuable. You can configure the lifecycle of a table based on data usage. MaxCompute deletes data that exceeds the lifecycle threshold at the earliest opportunity to save storage space.

Example:

```
create table test3 (key boolean) partitioned by (pt string, ds string) lifecycle 100;
-- Create a table with a lifecycle of 100 days. If the last modification of the table or a partition occurred more than 100 days ago, MaxCompute deletes the table or partition.
```

Notice The lifecycle takes a partition as the smallest unit. For a partitioned table, if some partitions reach the lifecycle threshold, they will be directly deleted. Partitions that have not reached the lifecycle threshold are not affected.

You can run the following command to change the lifecycle of a created table:

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

Archive cold data

Some data needs to be retained permanently or for a long period of time. However, the frequency of accessing the data decreases over time. If the use frequency decreases to an extent, you can archive the data as RAID files. Data is not stored as three copies. The Cauchy Reed-Solomon algorithm is used to store data as six copies of the original data plus three parity blocks. This improves the effective storage ratio from 1:3 to 1:1.5. MaxCompute uses the bzip2 algorithm to archive tables with a higher data compression ratio than other algorithms. These two algorithms can be combined to save the storage space by more than 70%.

Format of an archiving command:

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

Example:

```
alter table my_log partition(ds='20170101') archive;
-- Archive the data with ds set to 20170101.
```

Merge small files

A large number of small files are generated during the Reduce computing process or during Tunnel-based real-time data collection. This causes the following issues:

- More instance resources are required because the number of files a single instance can process is limited. This results in a waste of resources and affects the overall execution performance.
- The workload on the file system is high, and the disk utilization is affected.

You can use one of the following methods to merge small files:

- ALTER command: Run the following command in the console command line to merge small files:

  ```
  ALTER TABLE tablename [PARTITION] MERGE SMALLFILES;
  ```

- SQL statement: After SQL statements are executed, set `odps.task.merge.enabled` to true and run a Fuxi job to merge small files.

1.6.17.4. UDF OOM

This topic describes the UDF OOM issue and provides a solution.

An OOM error is reported when some jobs are running. Detailed error information:

```
FAILED: ODPS-0123144: Fuxi job failed - WorkerRestart errCode:9,errMsg:SigKill(OOM), usually caused by OOM(out of memory)
```

You can configure UDF runtime parameters to fix the error. Example:

```
odps.sql.mapper.memory=3072;
set odps.sql.udf.jvm.memory=2048;
set odps.sql.udf.python.memory=1536;
```

1.6.18. Limits of MaxCompute SQL statements

This topic describes all the limits of MaxCompute SQL statements.

Limit	Maximum value/Limit	Requirement	Description
Table name length	128 bytes	Size	A table or column name can contain only letters, digits, and underscores (_). It must start with a letter.
Comment length	1,024 bytes	Size	A comment is a valid string that cannot exceed 1,024 bytes in length.
Column definitions in a table	1,200	Quantity	A table can contain a maximum of 1,200 column definitions.
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.
Destination `INSERT` objects	256	Quantity	In a `MULTI-INSERT` operation, you can insert data into a maximum of 256 tables at a time.
`UNION ALL`	256	Quantity	A `UNION ALL` operation can be performed on a maximum of 256 tables.
`MAPJOIN`	128	Quantity	A `MAPJOIN` operation can be performed on a maximum of 128 small tables.
`MAPJOIN` memory	512 MB	Size	The memory size for all small tables on which a `MAPJOIN` operation is performed cannot exceed 512 MB.
Window functions	5	Quantity	A SELECT statement can contain a maximum of five window functions.
`PT IN SUBQUERY`	1,000	Quantity	A PT IN SUBQUERY statement can generate a maximum of 1,000 rows.
Length of an SQL statement	2 MB	Size	An SQL statement cannot exceed 2 MB in length. This limit is suitable for the scenarios where you use an SDK to call SQL statements.
Conditions of a `WHERE` clause	256	Quantity	A `WHERE` clause can contain a maximum of 256 conditions.

Limit	Maximum value/Limit	Requirement	Description
Length of a column record	8 MB	Size	A column record in a table cannot exceed 8 MB in length.
Parameters in an IN clause	1,024	Quantity	This item specifies the maximum number of parameters in an IN clause, for example, `IN (1,2,3….,1024)`. If the number of parameters in an `IN` clause is too large, the compilation performance is affected. We recommend that you use no more than 1,024 parameters, but this is not a fixed upper limit.
`jobconf.json`	1 MB	Size	The maximum size of the `jobconf.json` file is 1 MB. If a table contains a large number of partitions, the size of the `jobconf.json` file 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 cannot be modified.
Java UDFs	Not allowed to be `abstract` or `static`	Operation	Java UDFs cannot be `abstract` or `static`.
Partitions that can be queried	10,000	Quantity	A maximum of 10,000 partitions can be queried.
SQL execution plans	1 MB	Size	The size of the execution plan generated by MaxCompute SQL cannot exceed 1 MB. Otherwise, `FAILED: ODPS-0010000:System internal error - The Size of Plan is too large` is returned.

Notice The preceding limits cannot be manually modified.

1.6.19. Appendix

1.6.19.1. Escape characters

This topic describes the escape characters in MaxCompute SQL.

In MaxCompute SQL, the backslash (\) is an escape character that invokes an alternative representation on the special characters in a character string. This escape character may also indicate a literal interpretation of the characters that follow the escape character. If the backslash (\) in a string constant is followed by three valid octal digits in the range from 001 to 177, the system converts the ASCII values to the corresponding characters. The following table describes the mapping between escape characters and represented special characters.

Escape character	Represented special character
\b	Backspace
\t	Tab
\n	Newline
\r	Carriage-return
\'	Single quotation mark
\"	Double quotation marks
\ \	Backslash
\;	Semicolon
\Z	Control-Z
\0 or \00	Ending character

Examples:

- Use an escape character to indicate a literal interpretation of the characters that follow the escape character.

  String constants in MaxCompute SQL can be expressed in single quotation marks (' ') or double quotation marks (" "). You can include double quotation marks (" ") in a string that is enclosed in single quotation marks (' ') or include single quotation marks (' ') in a string that is enclosed in double quotation marks (" "). Examples:

  ```
  "I'm a happy manong."
  'I\'m a happy manong.'
  ```

- Use an escape character to represent special characters. Examples:

  ```
  select length('a\tb');
  -- 3 is displayed in the returned result. This indicates that the string contains three characters and that \t is treated as one character.
  ```

  ```
  select 'a\ab',length('a\ab');
  -- aab, 3 is displayed in the returned result. \a is treated as Letter a.
  ```

1.6.19.2. LIKE usage

This topic describes how to use the LIKE operator for character matching.

In LIKE character matching, a percent sign (%) is a wildcard character that matches an arbitrary number of characters. An underscore (_) is a wildcard character that matches a single character.

To match percent signs (%) or underscores (_), escape them with double backslashes (\\). Specifically, \\% or _ is used for the match.

```
'abcd' like 'ab%'
-- True.
'abcd' like 'ab_'
-- False.
'ab_cde' like 'ab\\_c%';
-- True.
```

> **Note** MaxCompute SQL statements support only UTF-8 character sets. If data is encoded in another format, the calculation result may be incorrect.

1.6.19.3. Regular expressions

This topic describes regular expressions in MaxCompute SQL.

Regular expressions in MaxCompute SQL use the Perl Compatible Regular Expressions (PCRE) standards and are matched by character.

RLIKE

You can use the RLIKE statement to match regular expressions. The following table lists the supported metacharacters.

Metacharacter	Description
^	Match the beginning of a line.
$	Match the end of a line.
.	Match any characters.
*	Match zero or more instances of the preceding character or character pattern.
+	Match one or more instances of the preceding character or character pattern.
?	Match zero or one instance of the preceding character or character pattern.
?	Match a modifier. If this character follows one of other delimiters (*, +, ?, {n}, {n,}, or {n,m}), the match pattern is non-greedy. In the non-greedy pattern, as few characters as possible are matched with the searched character string. In the default greedy pattern, as many characters as possible are matched with the searched character string.
A|B	Match A or B.
(abc)*	Match zero or more instances of the abc sequence.
{n} or {m,n}	The number of matches.
[ab]	Match any character (a or b) in the brackets. Fuzzy match is supported.
[a-d]	Match any of the following characters: a, b, c, and d.
[^ab]	Match any character that is not a or b. ^ indicates 'non'.
[::]	For more information, see the following table.
\	The escape character.
\n	n indicates a digit from 1 to 9 and is backward referenced.
\d	Digits.
\D	Non-digit characters.

POSIX character groups

Character group	Description	Valid value
[[:alnum:]]	Letters and digits	[a-zA-Z0-9]
[[:alpha:]]	Letters	[a-zA-Z]
[[:ascii:]]	ASCII characters	[\x00-\x7F]
[[:blank:]]	Spaces and tab characters	[\t]
[[:cntrl:]]	Control characters	[\x00-\x1F\x7F]
[[:digit:]]	Digits	[0-9]
[[:graph:]]	Characters other than whitespace characters	[\x21-\x7E]
[[:lower:]]	Lowercase letters	[a-z]
[[:print:]]	[:graph:] and whitespace characters	[\x20-\x7E]
[[:punct:]]	Punctuation marks	[][!"#$%&'()*+,./:;<=>?@\^_`{|}~-]
[[:space:]]	Whitespace characters	[\t\r\n\v\f]
[[:upper:]]	Uppercase letters	[A-Z]
[[:xdigit:]]	Hexadecimal characters	[A-Fa-f0-9]

Escape characters

The system uses a backslash (\) as the escape character. A backslash (\) in a regular expression indicates a second escape. For example, a regular expression is used to match the `a+b` string. The plus sign (`+`) is a special character in the expression and must be escaped. In the regular expression engine, the string is expressed as `a\\+b` . The system must perform an escape on the expression. Therefore, the expression that can match the string is `a\\\+b` .

Assume that the test_dual table exists. Example:

```
select 'a+b' rlike 'a\\\+b' from test_dual;
```

Returned result:

```
+------+
| _c1  |
+------+
| true |
+------+
```

The \ character is a special character in the regular expression engine. To match this character, it is expressed as \\ in the engine. Then, the system performs an escape on the expression. As a result, the \ character is expressed as \\\.

```
select 'a\\b', 'a\\b' rlike 'a\\\b' from test_dual;
```

Returned result:

```
+-----+------+
| _c0 | _c1  |
+-----+------+
| a\b | false |
+-----+------+
```

Note If a MaxCompute SQL statement includes `a\\b`, `a\b` is displayed in the output because MaxCompute escapes the expression.

If a string includes tab characters, the system reads `\t` and stores it as one character. Therefore, it is a common character in regular expressions.

```
select 'a\tb', 'a\tb' rlike 'a\tb' from test_dual;
```

Returned result:

```
+---------+------+
| _c0     | _c1  |
+---------+------+
| a     b | true |
+---------+------+
```

1.6.19.4. Reserved words and keywords

This topic describes all reserved words and keywords in MaxCompute SQL.

Notice

- When you name a table, a column, or a partition, do not use reserved words or keywords. Otherwise, an error may occur.
- Reserved words are not case-sensitive.

```
    %    &    &&    (    )    *    +
    -    .    /    ;    <    <=    <>
    =    >    >=    ?    ADD    AFTER    ALL
    ALTER    ANALYZE    AND    ARCHIVE    ARRAY    AS    ASC
    BEFORE    BETWEEN    BIGINT    BINARY    BLOB    BOOLEAN    BOTH    DECIMAL
    BUCKET    BUCKETS    BY    CASCADE    CASE    CAST    CFILE
    CHANGE    CLUSTER    CLUSTERED    CLUSTERSTATUS    COLLECTION    COLUMN    COLUMNS
    COMMENT    COMPUTE    CONCATENATE    CONTINUE    CREATE    CROSS    CURRENT
    CURSOR    DATA    DATABASE    DATABASES    DATE    DATETIME    DBPROPERTIES
    DEFERRED    DELETE    DELIMITED    DESC    DESCRIBE    DIRECTORY    DISABLE
    DISTINCT    DISTRIBUTE    DOUBLE    DROP    ELSE    ENABLE    END    EXCEPT
    ESCAPED    EXCLUSIVE    EXISTS    EXPLAIN    EXPORT    EXTENDED    EXTERNAL
    FALSE    FETCH    FIELDS    FILEFORMAT    FIRST    FLOAT    FOLLOWING
    FORMAT    FORMATTED    FROM    FULL    FUNCTION    FUNCTIONS    GRANT
    GROUP    HAVING    HOLD_DDLTIME    IDXPROPERTIES    IF    IMPORT    IN
    INDEX    INDEXES    INPATH    INPUTDRIVER    INPUTFORMAT    INSERT    INT
    INTERSECT    INTO    IS    ITEMS    JOIN    KEYS    LATERAL
    LEFT    LIFECYCLE    LIKE    LIMIT    LINES    LOAD    LOCAL
    LOCATION    LOCK    LOCKS    LONG    MAP    MAPJOIN    MATERIALIZED
    MINUS    MSCK    NOT    NO_DROP    NULL    OF    OFFLINE    OFFSET
    ON    OPTION    OR    ORDER    OUT    OUTER    OUTPUTDRIVER
    OUTPUTFORMAT    OVER    OVERWRITE    PARTITION    PARTITIONED    PARTITIONPROPERTIES    PARTITIO
NS
    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    SEQUENCEFILE    SERDE    SERDEPROPERTIES    SET    SHARED    SHOW
    SHOW_DATABASE    SMALLINT    SORT    SORTED    SSL    STATISTICS    STATUS    STORED
    STREAMTABLE    STRING    STRUCT    TABLE    TABLES    TABLESAMPLE    TBLPROPERTIES
    TEMPORARY    TERMINATED    TEXTFILE    THEN    TIMESTAMP    TINYINT    TO
    TOUCH    TRANSFORM    TRIGGER    TRUE    TYPE    UNARCHIVE    UNBOUNDED    UNDO
    UNION    UNIONTYPE    UNIQUEJOIN    UNLOCK    UNSIGNED    UPDATE    USE
    USING    UTC    UTC_TMESTAMP    VIEW    WHEN    WHERE    WHILE    DIV
```

1.6.19.5. Settings of new data types

This topic describes the precautions for configuring new data types in MaxCompute SQL.

If you want to read a table that includes new data types, you are not required to add the `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 reported.
- 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 reported because the implicit data type conversion is invalid.

1.6.19.6. ACID semantics of MaxCompute parallel write jobs

This topic describes the atomicity, consistency, isolation, durability (ACID) semantics of MaxCompute parallel write jobs.

Terms

- Operation: a single job submitted in MaxCompute.

- Data object: an object that stores data, such as a non-partitioned table or a partition.
- INTO job: an SQL job that contains the INTO keyword, such as INSERT INTO or DYNAMIC INSERT INTO.
- OVERWRITE job: an SQL job that contains the OVERWRITE keyword, such as INSERT OVERWRITE or DYNAMIC INSERT OVERWRITE.
- Data upload by using Tunnel: an INTO or OVERWRITE job.

Description of ACID semantics

- Atomicity: An operation is completely performed or not performed at all. An operation is not partially performed.
- Consistency: The integrity of data objects is maintained when an operation is performed.
- Isolation: An operation is independent of other parallel operations.
- Durability: After an operation is complete, modified data is permanently valid and not lost even if a system failure occurs.

Scenarios of ACID semantics for MaxCompute

- Atomicity
  - When multiple jobs conflict with each other, MaxCompute ensures that only one job succeeds.
  - The atomicity of the CREATE, OVERWRITE, and DROP operations on a single table or partition is ensured.
  - The atomicity of cross-table operations, such as MULTI-INSERT, cannot be ensured.
  - In extreme cases, the following operations may not be atomic:
    - A `DYNAMIC INSERT OVERWRITE` operation that is performed on more than 10,000 partitions.
    - An INTO operation. The atomicity of INTO operations cannot be ensured because data cleansing fails during a transaction rollback. However, the data cleansing failure does not cause loss of original data.
- Consistency
  - The consistency can be ensured for OVERWRITE jobs.
  - If an INTO job fails due to a conflict, data from the failed job may remain.
- Isolation
  - For non-INTO operations, MaxCompute ensures that read operations are submitted.
  - For INTO operations, some read operations may not be submitted.
- Durability
  - MaxCompute ensures data durability.

ACID semantics of transactional tables

- If existing ACID semantics are used, INTO operations ensure that read operations are submitted and no data remains after an operation fails due to a conflict.
- The atomicity of the UPDATE and DELETE operations on a single non-partitioned table or a partition is ensured. For example, if two update jobs run in parallel to modify the data of a partition, only one job succeeds. The partition is not partially updated by one job. It is also not updated by the two jobs with part of the updated data remained.

Conflicts among parallel operations: When two jobs run in parallel and write data to the same destination table, a conflict may occur. In this case, the job that ends earlier will succeed, and the job that ends later will fail due to the conflict.

The following table describes the conflicts that may occur when two jobs run in parallel and write data to the same non-partitioned table or partition. The first column lists the job that ends earlier. The other columns list the jobs that end later.

Before/After	INSERT OVERWRITE	INSERT INTO	UPDATE/DELETE	MERGE SMALLFILES
INSERT OVERWRITE	Both jobs succeed. The result data of the INSERT OVERWRITE operation that ends later will overwrite that of the INSERT OVERWRITE operation that ends earlier.	Both jobs succeed. The result data of the INSERT INTO operation is appended to that of the INSERT OVERWRITE operation.	The INSERT OVERWRITE operation modifies the data of the non-partitioned table or partition. An error is reported in the UPDATE operation.	The INSERT OVERWRITE operation modifies the data of the non-partitioned table or partition. An error is reported in the MERGE SMALLFILES operation.
INSERT INTO	Both jobs succeed. The result data of the INSERT OVERWRITE operation overwrites that of the INSERT INTO operation.	Both jobs succeed. The result data of the two INSERT INTO operations is merged.	The INSERT INTO operation modifies the data of the non-partitioned table or partition. An error is reported in the UPDATE operation that ends later.	The INSERT INTO operation modifies the data of the non-partitioned table or partition. An error is reported in the MERGE SMALLFILES operation that ends later.
UPDATE/DELETE	Both jobs succeed. The result data of the INSERT OVERWRITE operation overwrites that of the UPDATE operation.	Both jobs succeed. The result data of the INSERT INTO operation is appended to that of the UPDATE operation.	The UPDATE operation that ends earlier modifies the data of the non-partitioned table or partition. An error is reported in the UPDATE operation that ends later.	The UPDATE operation modifies the data of the non-partitioned table or partition. An error is reported in the MERGE SMALLFILES operation that ends later.
MERGE SMALLFILES	Both jobs succeed. The result data of the INSERT OVERWRITE operation overwrites that of the MERGE SMALLFILES operation.	Both jobs succeed. The result data of the INSERT INTO operation is appended to that of the MERGE SMALLFILES operation.	The MERGE SMALLFILES operation modifies the data of the non-partitioned table or partition. An error is reported in the UPDATE operation that ends later.	The MERGE SMALLFILES operation that ends earlier modifies the data of the non-partitioned table or partition. An error is reported in the MERGE SMALLFILES operation that ends later.

The following table simplifies the preceding rules and describes only whether an error is reported for an operation after data changes.

Operation	Error reported upon data changes
INSERT OVERWRITE	No
INSERT INTO	No
UPDATE/DELETE	Yes
MERGE SMALLFILES	Yes

No conflict occurs for INSERT operations in the event of data changes. An error is reported in the UPDATE/DELETE or MERGE SMALLFILES operation if the destination non-partitioned table or partition has data changes.

> **Note**
> - Minor or major compaction is classified as MERGE SMALLFILES operations.
> - In extreme scenarios, if multiple jobs are all in the metadata update phase, an error may be reported due to a conflict among these parallel update operations.

1.7. MaxCompute Tunnel

1.7.1. Overview

This topic describes how to upload data to MaxCompute or download data from MaxCompute. This topic also describes related limits.

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. A variety of data upload and download tools are derived from the SDKs.

The tools can meet the data upload and download requirements of most common scenarios in which data is migrated to the cloud.

Tunnel service connections

DataHub and Tunnel use different endpoints in different network environments. You must also select different endpoints to connect to the service.

Limits

- Limits on Tunnel-based data uploads:
  - You cannot run Tunnel commands to upload or download data of the ARRAY, MAP, or STRUCT type.
  - 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. Selection of tools to migrate data to the cloud

MaxCompute provides a variety of data upload and download tools, which can be used in different scenarios to migrate data to the cloud. This topic describes the selection of data transmission tools in three typical scenarios.

Hadoop data migration

You can use MaxCompute Migration Assist (MMA), Sqoop, and DataWorks to migrate Hadoop data.

- If you use DataWorks, DataX is required.
- If you use Sqoop, a MapReduce job runs on the original Hadoop cluster for distributed data transmission to MaxCompute.
- If you use MMA, Meta Carrier is required to access your Hive metastore service and capture Hive metadata. Then, MMA generates DDL statements that are used to create MaxCompute tables and partitions, as well as Hive UDTF SQL statements for data migration.

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, such as 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 RDS database.

Log collection

You can use tools such as Flume, Fluentd, and Logstash to collect logs.

1.7.3. 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 usage scenarios. This topic describes these tools.

Alibaba Cloud services

- Data Integration of DataWorks (Tunnel)

  Data Integration of DataWorks is a stable, efficient, and scalable data synchronization platform. 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 sources: MaxCompute, ApsaraDB RDS (MySQL, SQL Server, and PostgreSQL), Oracle, FTP, AnalyticDB, OSS, ApsaraDB for Memcache, and PolarDB-X.

- MaxCompute client (Tunnel)

  Based on the batch data tunnel SDK, the client provides built-in Tunnel commands for data uploads and downloads.

- DTS (Tunnel)

  Data Transmission Service (DTS) is an Alibaba Cloud data service that supports data exchange among multiple data sources, such as Relational Database Management System (RDBMS), NoSQL, and Online Analytical Processing (OLAP). It provides data transmission features, such as data migration, real-time data subscription, and real-time data synchronization.

  DTS supports data synchronization from ApsaraDB RDS and MySQL instances to MaxCompute tables. Other data sources are not supported.

Open source products

The projects that correspond to each product are open-sourced. You can visit aliyun-maxcompute-data-collectors to view details.

- Sqoop (Tunnel)

  Sqoop 1.4.6 in the community is further developed to provide enhanced MaxCompute support. It imports data from relational databases such as MySQL and data from HDFS or Hive to MaxCompute tables. It also exports data from MaxCompute tables to relational databases such as MySQL.

- Kettle (Tunnel)

  Kettle is an open source extract, transform, load (ETL) tool that is developed in Java. It runs on Windows, UNIX, or Linux, and provides graphic interfaces for you to define the data transmission topology by using drag-and-drop components.

- Apache Flume (DataHub)
  - Apache Flume is a distributed and reliable system. It efficiently collects large volumes of log data from different data sources and then aggregates and stores the data in a centralized data storage. It supports multiple Source and Sink plug-ins.
  - 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 analytics.
  - 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 an Oracle database to DataHub in real time and archive the data in MaxCompute tables.

- MMA

  MaxCompute Migration Assist (MMA) uses Meta Carrier to access your Hive metastore service and capture Hive metadata. It then generates DDL statements that are used to create MaxCompute tables and partitions, as well as Hive UDTF SQL statements for data migration.

1.7.4. Tunnel SDK overview

1.7.4.1. Overview

Data upload and download tools provided by MaxCompute are developed based on Tunnel SDK. This topic describes the major APIs of Tunnel SDK.

The usage of the SDK varies based on its version. For more information, see SDK Java Doc.

API	Description
TableTunnel	The entry class that is used to access MaxCompute Tunnel.
TableTunnel.UploadSession	A session that uploads data to a MaxCompute table.
TableTunnel.DownloadSession	A session that downloads data from a MaxCompute table.
InstanceTunnel	The entry class that is used to access MaxCompute Tunnel.
InstanceTunnel.DownloadSession	A session that downloads data from a MaxCompute SQL instance. The SQL instance must start with the SELECT keyword and is used to query data.

Note

- If you use Maven, you can search for odps-sdk-core in the Maven repository to find the latest version of the SDK for Java. You can configure the Maven dependency in the following way:

```
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>odps-sdk-core</artifactId>
    <version>0.36.2</version>
</dependency>
```

- The endpoint of MaxCompute Tunnel supports automatic routing based on the MaxCompute endpoint settings.

1.7.4.2. TableTunnel

This topic describes the TableTunnel API.

TableTunnel is an entry class of the MaxCompute Tunnel service. You can use TableTunnel to upload or download only table data. Views cannot be uploaded or downloaded.

Definition

The following code defines the TableTunnel API.

```
public class TableTunnel {
public DownloadSession createDownloadSession(String projectName, String tableName);
public DownloadSession createDownloadSession(String projectName, String tableName, PartitionSpec par
titionSpe c);
public UploadSession createUploadSession(String projectName, String tableName);
public UploadSession createUploadSession(String projectName, String tableName, PartitionSpec partiti
onSpec);
public DownloadSession getDownloadSession(String projectName, String tableName, PartitionSpec partit
ionSpec, String id);
public DownloadSession getDownloadSession(String projectName, String tableName, String id);
public UploadSession getUploadSession(String projectName, String tableName, PartitionSpec partitionS
pec, String id);
public UploadSession getUploadSession(String projectName, String tableName, String id); public void
setEndpoint(String endpoint);
}
```

Description

- The lifecycle of a TableTunnel instance starts from the time it is created to the time data upload or download is complete.
- TableTunnel provides a method to create UploadSession and DownloadSession objects. TableTunnel.UploadSession is used to upload data. 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.
- In an upload session, each RecordWriter matches an HTTP request and is identified by a unique block ID. The block ID is the name of the file that corresponds to the RecordWriter.
- If you use the same block ID to enable a RecordWriter multiple times in the same session, the data uploaded after the RecordWriter calls the close() method for the last time overwrites all the data that is previously uploaded. This feature can be used to retransmit a data block that fails to be uploaded.
- In UploadSession of TableTunnel:
  - If the boolean overwrite parameter is not specified, the INSERT INTO statement is used.
  - If the boolean overwrite parameter is set to True, the INSERT OVERWRITE statement is used.
  - If the boolean overwrite parameter is set to False, the INSERT INTO statement is used.

  Descriptions of INSERT OVERWRITE and INSERT INTO:

  - INSERT INTO: Upload sessions of the same table or partition do not affect each other. Data uploaded in each session is saved in different directories.
  - INSERT OVERWRITE: All data in a table or partition is overwritten by the data in the current upload session. If you use this statement to upload data, do not perform concurrent operations on the same table or partition.

Implementation process

1. The RecordWriter.write() method uploads your data as files to a temporary directory.
2. The RecordWriter.close() method moves the files from the temporary directory to a data directory.
3. The session.commit() method moves all files from the data directory to the directory in which the required table is saved, and updates the table metadata. This way, the data moved to a table in the current job is visible to other MaxCompute jobs such as SQL and MapReduce jobs.

Limits

- The value of a block ID must be greater than or equal to 0 but less than 20000. The size of the data that can be uploaded in a block cannot exceed 100 GB.

- A session is uniquely identified by its ID. The lifecycle of a session is 24 hours. If your session times out because large amounts of data are transmitted, you must transmit your 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.

  > **Note** HTTP has an 8 KB buffer. When you call the RecordWriter.write() method, your data may be saved to the buffer and no inbound traffic flows over the HTTP connection. In this case, you can call the TunnelRecordWriter.flush() method to forcibly flush data out of the buffer.

- If you use a RecordWriter to write logs to MaxCompute, the write operation may time out due to unexpected traffic fluctuations. To avoid such issues, take note of the following points:
  - We recommend that you do not use a RecordWriter for each data record. If you use a RecordWriter for each data record, a large number of small files are generated, because each RecordWriter corresponds to a file. This affects the performance of MaxCompute.
  - If the size of cached code reaches 64 MB, we recommend that you use a RecordWriter to write multiple data records at the same time.
- The lifecycle of a RecordReader is 300 seconds.

1.7.4.3. InstanceTunnel

This topic describes the InstanceTunnel API.

InstanceTunnel is an entry class to access the MaxCompute Tunnel service. You can use InstanceTunnel to download the execution results of an SQL instance that executes a SELECT statement.

Definition

The following code shows the definition of the InstanceTunnel API:

```
public class InstanceTunnel{
 public DownloadSession createDownloadSession(String projectName, String instanceID);
 public DownloadSession createDownloadSession(String projectName, String instanceID, boolean limitEn
abled);
 public DownloadSession getDownloadSession(String projectName, String id);
 }
```

Parameters:

- projectName: the name of a project.
- instanceID: the ID of an instance.

Limits

InstanceTunnel provides an easy way to obtain instance execution results. However, it is subject to the following permission limits to ensure data security:

- If the number of data records is less than or equal to 10,000, all users who have read permissions on the specified instance can use InstanceTunnel to download the data records. The same rule applies to data queries by calling a RESTful API.
- If the number of data records is greater than 10,000, only users who have the read permissions on all the source tables from which the specified instance queries data can use InstanceTunnel to download the data records.

1.7.4.4. UploadSession

This topic describes the UploadSession API.

This API is used to upload data to MaxCompute tables.

Definition

The following code defines the UploadSession API:

```
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);
    public RecordWriter openBufferedWriter();
    public RecordWriter openBufferedWriter(boolean compress);
}
```

Notice

- Block IDs that are used within the same upload session must be unique. After you use a block ID to enable RecordWriter, write multiple data records at the same time, call close, and then call commit to complete data upload in an upload session, you cannot use this block ID to enable another RecordWriter to write data.
- The maximum size of a block is 100 GB. We recommend that the volume of data written to each block be greater than 64 MB. Otherwise, the computing performance deteriorates significantly.
- The lifecycle of a session on the server is 24 hours.
- When you upload data, a network action is triggered each time RecordWriter writes 8 KB of data. If no network actions are triggered within 120 seconds, the server closes the connection and RecordWriter becomes unavailable. You must enable a new RecordWriter to write data.
- We recommend that you use the openBufferedWriter operation to upload data. This operation does not show the blockId value but contains an internal data cache. If a block fails to be uploaded, this operation automatically retries the upload process.
- The overwrite mode is added by using the commit method. You can use the overwrite mode to submit data. If you submit data in overwrite mode, the submitted data overwrites the existing data in the table or partition.

  Notice Undefined behavior occurs when you submit data in overwrite mode in multiple concurrent sessions. This may affect data accuracy. To avoid this issue, you must determine the number of concurrent sessions in which you submit data in overwrite mode.

Description

- Lifecycle: indicates the period from the time an upload instance is created to the time data is uploaded.
- Upload instance. You can call the Constructor method or use TableTunnel to create an upload instance.
  - Request mode: synchronous.
  - The server creates a session for the upload instance and generates a unique upload ID to identify the upload instance. You can run the `getId` command on the client to obtain the upload ID.

- Data upload
  - Request mode: synchronous.
  - Call the openBufferedWriter operation to generate a RecordWriter instance. The blockId parameter identifies the data that is uploaded this time and describes the data position in the whole table. The value of blockId is in the range of [0,20000]. If the upload fails, you can upload the data again based on the block ID.
- Upload status
  - Request mode: synchronous.
  - Call the `getStatus` method to obtain the current upload status.
  - Call the `getBlockList` method to obtain the blocks that are uploaded. Compare the result with the list of block IDs that were previously sent to the server and re-upload the blocks that fail to be uploaded.
- Upload termination
  - Request mode: synchronous.
  - Call the Commit(Long[] blocks) method. The blocks parameter indicates the blocks that are uploaded. The server verifies the block list.
  - Verification enhances data accuracy. If the provided list of block IDs is different from the list on the server, an error is returned.
  - If the commit operation fails, try again.
- State description
  - UNKNOWN: This is the initial state when the server creates a session.
  - NORMAL: The upload session is created.
  - CLOSING: When you call the complete method to end an upload session, the server changes the state to CLOSING.
  - CLOSED: The data upload is complete. The data is moved to the directory where the result table is saved.
  - EXPIRED: The upload session times out.
  - CRITICAL: An error occurs.

1.7.4.5. DownloadSession

This topic describes the DownloadSession API.

This API is used to download data from MaxCompute tables.

Definition

The following code defines the DownloadSession API:

```
public class DownloadSession {
    DownloadSession(Configuration conf, String projectName, String tableName,
                    String partitionSpec) throws TunnelException
    DownloadSession(Configuration conf, String projectName, String tableName,
                    String partitionSpec, String downloadId) throws TunnelException
    public String getId()
    public long getRecordCount()
    public TableSchema getSchema()
    public TableTunnel.DownloadStatus getStatus()
    public RecordReader openRecordReader(long start, long count)
    public RecordReader openRecordReader(long start, long count, boolean compress)
}
```

Description

- Lifecycle: indicates the period from the time a download instance is created to the time data is downloaded.
- Download instance: You can call the constructor method or use TableTunnel to create a download instance.
  - Request mode: synchronous.
  - The server creates a session for the download instance and generates a unique download ID to identify the download instance. You can call the `getId` method on the client to obtain the download ID.
  - This operation results in high overheads. The server creates indexes for data files. If a large number of data files exists, it takes a long time to create indexes for the data files.
  - The server returns the total number of records. You can start multiple download sessions at the same time to download data based on the total number of data records.
- Data download
  - Request mode: asynchronous.
  - Call the openRecordReader API to generate a RecordReader instance. The start parameter identifies the start position of the data record in this download session. The value of this parameter starts from 0 and must be greater than or equal to 0. The count parameter identifies the number of data records that are downloaded in this session. The value of the count parameter must be greater than 0.
- Download status
  - Request mode: synchronous.
  - Call the `getStatus` method to obtain the download status.
- Status description
  - UNKNOWN: This is the initial state when the server creates a download session.
  - NORMAL: The download object is created.
  - CLOSED: The download is complete.
  - EXPIRED: The download session times out.

1.7.4.6. TunnelBufferedWriter

This topic describes the TunnelBufferedWriter API.

This API is used to upload data.

The upload process is complex due to limits on block management and connection timeout on the server. The Tunnel SDK provides an enhanced RecordWriter of TunnelBufferWriter to simplify the upload process.

Definition

The following code defines the TunnelBufferedWriter API:

```
public class TunnelBufferedWriter implements RecordWriter {
    public TunnelBufferedWriter(TableTunnel.UploadSession session, CompressOption option) throws IOE
xception;
    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;
}
```

Description

- Lifecycle: indicates the period from the time RecordWriter is created to the time data is upload.
- TunnelBufferedWriter instance: You can call openBufferedWriter of UploadSession to create a

TunnelBufferedWriter instance.

- Data upload: If you call Write, data records are first written to the local cache. After the cache is full, multiple data records are submitted to the server at a time to avoid a connection timeout. If data upload fails, the system automatically retries the upload operation.
- Upload termination: You can call close and then commit of UploadSession to terminate the upload process.
- Buffer control: You can call setBufferSize to change the memory (in bytes) occupied by the buffer. We recommend that you set the memory size to a value greater than or equal to 64 MB. This prevents excessive small files from being generated on the server, which may affect the processing performance. The value ranges from 1 MB to 1000 MB. The default value is 64 MB.
- Retry policy settings: The following policies are provided: EXPONENTIAL_BACKOFF, LINEAR_BACKOFF, and CONSTANT_BACKOFF. The following code snippet sets the number of Write retries to 6. To avoid unnecessary retries, you can perform each retry after an exponential interval, such as 4s, 8s, 16s, 32s, 64s, and 128s. By default, the interval starts from 4s.

```
RetryStrategy retry
  = new RetryStrategy(6, 4, RetryStrategy.BackoffStrategy.EXPONENTIAL_BACKOFF)
writer = (TunnelBufferedWriter) uploadSession.openBufferedWriter();
writer.setRetryStrategy(retry);
```

Note We recommend that you retain the preceding settings.

1.7.5. Tunnel SDK example

1.7.5.1. Example of simple uploads

This topic provides an example of simple uploads.

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, open a writer to start writing data to a block.
      // If you write a small amount of data to a block, a large number of small files are generated
. This severely deteriorates computing performance. We strongly recommend that you write more than 6
4 MB of data to a block at a time. The amount of data that can be written to the same block must be
less than 100 GB.
      // You can calculate the total amount of data that is written to a block by using the followin
g formula: Total amount of data = Average amount of data in each write operation × Number of write o
perations. The total amount of data must be greater than 64 MB, but less than 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 d
ata is written.
        // If no network transmission is performed within 120s, the server closes the connection. In
this case, the writer becomes unavailable and you must rewrite data.
        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();
    }
  }
}
```

1.7.5.2. Example of simple downloads

This topic provides an example of simple downloads.

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 accessKey>";
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.getColumns().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();
}
}
```

1.7.5.3. Example of multi-thread uploads

This topic provides an example of multi-thread uploads.

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 d
ata is written.
        // If no network transmission is performed within 120s, the server closes the connection. In
this case, the writer becomes unavailable and you must rewrite data.
        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 data is prepared, open a writer to start writing data in multi-thread mode to a block
.
      // If you write a small amount of data to a block, a large number of small files are generated
. This severely deteriorates computing performance. We strongly recommend that you write more than 6
4 MB of data to a block at a time. The amount of data that can be written to the same block must be
```

```
less than 100 GB.
      // You can calculate the total amount of data that is written to a block by using the followin
g formula: Total amount of data = Average amount of data in each write operation × Number of write o
perations. The total amount of data must be greater than 64 MB, but less than 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();
    }
  }
}
```

1.7.5.4. Example of multi-thread downloads

This topic provides an example of multi-thread downloads.

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
java.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.acc
ount.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.TunnelExcep
tion;
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.ge
tColumns().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>> callers =
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();
}
}
}
```

1.7.5.5. Example of uploading data by using BufferedWriter

This topic provides an example on how to upload data by using BufferedWriter of the MaxCompute Tunnel SDK.

Example:

```
// Initialize the code of MaxCompute and MaxCompute Tunnel.
RecordWriter writer = null;
TableTunnel.UploadSession uploadSession = tunnel.createUploadSession(projectName, tableName);
try {
  int i = 0;
  // Construct a BufferedWriter for MaxCompute Tunnel SDK.
  writer = uploadSession.openBufferedWriter();
  Record product = uploadSession.newRecord();
  for (String item : items) {
    product.setString("name", item);
    product.setBigint("id", i);
    // Call the write() method of BufferedWriter to write data.
    writer.write(product);
    i += 1;
  }
} finally {
  if (writer ! = null) {
    // Close BufferedWriter of MaxCompute Tunnel SDK.
    writer.close();
  }
}
// Commit the upload session to end the data upload.
uploadSession.commit();
```

1.7.5.6. Example of uploading data by using BufferedWriter in multithreading mode

This topic provides an example on how to upload data by using BufferedWriter of Tunnel SDK in multithreading 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 MaxCompute 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();
}
```

1.7.5.7. Examples of uploading and downloading data of complex data types

This topic provides examples of uploading and downloading data of complex data types by using the SDK of MaxCompute Tunnel.

Upload data of complex data types

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);
// Pass data into a record.
record.setArray(0, arrayData);
record.setMap(1, mapData);
record.setStruct(2, new SimpleStruct((StructTypeInfo) schema.getColumn(2).getTypeInfo(), structData)
);
// Write the record to RecordWriter.
recordWriter.write(record);
```

Download data of complex data types

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);
```

Upload and download data of complex data types

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>";
  // Partition 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<nam
e:STRING,age:BIGINT>}.
      UploadSession uploadSession = tunnel.createUploadSession(project, table, partitionSpec);
      // Obtain the table schema.
      TableSchema schema = uploadSession.getSchema();
      // Open 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);
      // Pass data into a record.
      record.setArray(0, arrayData);
      record.setMap(1, mapData);
      record.setStruct(2, new SimpleStruct((StructTypeInfo) schema.getColumn(2).getTypeInfo(), struc
tData));
      // Write the record to RecordWriter.
      recordWriter.write(record);
      // Close RecordWriter.
      recordWriter.close();
      // Commit the upload session to complete 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<nam
e:STRING, age:BIGINT>}.
      DownloadSession downloadSession = tunnel.createDownloadSession(project, table, partitionSpec);
      schema = downloadSession.getSchema();
      // Open RecordReader.
      RecordReader recordReader = downloadSession.openRecordReader(0, 1);
      // Use RecordReader to 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);
      System.out.println("download success!") ;
    } catch (TunnelException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

1.7.6. MaxCompute Streaming Tunnel

This topic describes the MaxCompute Streaming Tunnel service.

Description

MaxCompute Streaming Tunnel provides a new set of API operations and backend services. It provides an easy way for using streaming services, such as stream processing, tunnel, and data synchronization, as upstream services to write data to MaxCompute. MaxCompute Streaming Tunnel is introduced to resolve various issues that occur when streaming services call the original Batch API operation to write data. The issues include a large number of directory and file fragments, and a large number of request failures caused by highly concurrent uploads.

MaxCompute Streaming Tunnel provides the following features:

- Streaming semantics APIs: new APIs that simplify the development of distributed services.
- Point-to-point upload: provides higher performance and stability when network access is allowed.
- Automatic partition creation: allows you to create partitions for multiple distributed services at the same time.
- Limited data deduplication: deduplicates the data that you want to write based on the specified time or data volume.

> **Note** The automatic partition creation and limited data deduplication features will be available in later versions.

MaxCompute Streaming Tunnel outperforms other streaming services in the following aspects:

- Row-store file format: allows you to append data to a file to resolve the file fragmentation issue.
- Asynchronous data merging: improves the data storage efficiency without interrupting services.
- Separation of data requests from metadata operations: separates data write links from metadata operations (DDL operations). This ensures that DDL operations do not become a bottleneck in high queries per second (QPS) scenarios.
- Fine-grained throttling and resource allocation.

Limits

- Partition locking: MaxCompute Streaming Tunnel is used based on MaxCompute metadata. When MaxCompute writes streaming data to a table or partition, the table or partition is locked. During this period, data modification operations, such as INSERT INTO and INSERT OVERWRITE, are not allowed. After the data write operation is complete, the table or partition is unlocked, and all operations can be performed.
- Delayed detection of DDL operations: Data links are separated from DDL operations. As a result, MaxCompute

Streaming Tunnel detects DDL operations 30 seconds to 60 seconds after these operations are complete. For example, a data write operation to a table or partition is successful even after the table or partition has been deleted by using the DROP TABLE or DROP PARTITION statement. To prevent this issue, make sure that you execute the DROP TABLE or DROP PARTITION statement only after the data write operation is complete.

- Increased volume of stored hot data: To support asynchronous merging, MaxCompute Streaming Tunnel saves two copies of data that is written in the last hour. One copy is the original data, and the other copy is the merged data. Therefore, the volume of stored data increases to some extent on the day data is written.

Example

You can use the Maven repository to manage and configure the version of the SDK that you want to use. Sample configurations in the Maven repository:

```
<dependency>
  <groupId>com.aliyun.odps</groupId>
  <artifactId>odps-sdk-core</artifactId>
  <version>0.36.2</version>
</dependency
```

Sample code:

```
public class UploadSample {
        private static String accessId = "<your access id>";
        private static String accessKey = "<your access Key>";
        private static String odpsUrl = "http://service.odps.aliyun.com/api";
        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);
                        PartitionSpec partitionSpec = new PartitionSpec(partition);
                        StreamUploadSession uploadSession = tunnel.createStreamUploadSession(projec
t,
                                        table, partitionSpec);
                        System.out.println("Session Status is : "
                                        + uploadSession.getStatus().toString());
                        TableSchema schema = uploadSession.getSchema();
                        StreamRecordPack pack = uploadSession.newRecordPack();
                        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++) {
                                pack.append(record);
                        }
                        pack.flush();
                        System.out.println("upload success!") ;
                } catch (TunnelException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }
 }
```

1.7.7. Appendix

1.7.7.1. FAQ related to data upload and download by using MaxCompute Tunnel

This topic provides answers to some frequently asked questions about data upload and download by using MaxCompute Tunnel.

What is MaxCompute Tunnel?

MaxCompute Tunnel is a data tunnel that is used to upload data to or download data from MaxCompute. View data cannot be uploaded or downloaded by using MaxCompute Tunnel.

Are duplicate block IDs allowed in an upload session?

No, 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 in an upload session, you cannot use this block ID to open another RecordWriter. Valid values of block IDs: [0, 20000).

What is the maximum size of a block?

The maximum size of a block is 100 GB. We recommend that data of greater than 64 MB be written to each block. Each block corresponds to a file. We recommend that you write at least 64 MB of data to each block.

Can a session be shared among processes or threads? What is the lifecycle of a session?

Each session has a 24-hour lifecycle on the server. A session can be used within the 24 hours after it is created, and can be shared among processes or threads. The block ID of each session must be unique. You can perform the following steps to upload data:

1. Create a session.
2. Estimate the amount of data.
3. Assign blocks to threads, for example, assign blocks with the IDs of 0 to 100 to Thread 1 and assign blocks with the IDs of 100 to 200 to Thread 2.
4. Prepare data.
5. Upload data.
6. Commit all blocks to which data is written.

What do I do if read or write operations time out or an I/O exception occurs?

If you upload data, a network action is triggered each time the RecordWriter writes data of 8 KB. If no network action is triggered within 120 seconds, the server closes the connection, and the RecordWriter stops writing data. You must start another RecordWriter to write data.

If you download data, the RecordReader works in the similar way as the RecordWriter. If no network I/O occurs for a long period of time, the server closes the connection. We recommend that you perform read operations without calling other system API operations.

Which SDKs are supported by MaxCompute Tunnel?

MaxCompute Tunnel supports SDK for Java and SDK for C++.

Does MaxCompute Tunnel allow multiple clients to upload the same table at the same time?

Yes, MaxCompute Tunnel allows multiple clients to upload the same table at the same time.

Is MaxCompute Tunnel suitable for batch upload or streaming upload?

MaxCompute Tunnel is suitable for batch upload. It is not suitable for streaming upload.

Do I need to create partitions before I upload data by using MaxCompute Tunnel?

Yes, MaxCompute Tunnel does not automatically create partitions.

What is the relationship between Dship and MaxCompute Tunnel?

Dship is a tool that uploads and downloads data by using MaxCompute Tunnel.

Is data uploaded by using MaxCompute Tunnel appended to an existing file or is the data in the file overwritten by the uploaded data?

The uploaded data is appended to the existing file.

What is the routing feature of MaxCompute Tunnel?

The routing feature allows the SDK of MaxCompute Tunnel to obtain the endpoint of MaxCompute Tunnel by setting the MaxCompute endpoint. Therefore, you can run the SDK of MaxCompute Tunnel after you set the MaxCompute endpoint.

When data is uploaded by using MaxCompute Tunnel, what is the appropriate size of data in each block?

The size of data in each block is determined based on a number of factors, such as the network conditions, real-time performance requirements, data usage, and small files in clusters. In most cases, if the amount of data is large and data is continuously uploaded, the data size can be 64 MB to 256 MB. If data is uploaded once a day, the data size can be about 1 GB.

The timeout error message appears when data is downloaded by using MaxCompute Tunnel. Why?

This issue is usually caused by an incorrect endpoint. You can check the endpoint configuration by using different methods, for example, use Telnet to check network connectivity.

The following error message appears when data is downloaded by using MaxCompute Tunnel. Why?

```
You have NO privilege 'odps:Select' on {acs:odps:*:projects/XXX/tables/XXX}. project 'XXX' is protec
ted
```

The reason is that data protection is enabled for the project from which data is downloaded. The data download you performed aims to transfer data from the project to another project. You can perform data download only after the project owner authorizes you to perform this operation.

The following error message appears when data is uploaded by using MaxCompute Tunnel. Why?

```
ErrorCode=FlowExceeded, ErrorMessage=Your flow quota is exceeded. **
```

The maximum number of concurrent upload or download requests is exceeded. By default, the quota for concurrent requests allowed by MaxCompute Tunnel is 2,000. A request counts against the quota after the request is sent and complete. If you encounter similar errors, we recommend that you use one of the following solutions:

- Make the system work in sleep mode and try again after the system awakes.
- Increase the quota for the project. Before you perform this operation, contact the administrator to estimate the traffic pressure.
- Report the issue to the project owner to locate and control the requests that consume most resources.

1.7.7.2. Common tunnel error codes

This topic describes common tunnel error codes.

The following table describes the common tunnel error codes.

ErrorCode	Cause	Recommended solution
NoSuchPartition	The partition does not exist.	Tunnel commands cannot be used to create partitions. Create partitions before you upload or download data.
InvalidProjectTable	The project name or table name is invalid.	Rename the project or table to ensure that a valid project or table name is used.
NoSuchProject	The specified project does not exist.	Check whether the project name is valid.
NoSuchTable	The specified table does not exist.	Check whether the table name is valid.
StatusConflict	The session expires or has been committed.	Recreate a session.
MalformedDataStream	The data format is invalid.	If the network connection is closed, reconnect it. If the schema is inconsistent with the table schema, make them consistent.
InvalidPartitionSpec	The partition information is invalid.	Check partition information. An example of a valid partition is pt='1',ct='2017'.
InvalidRowRange	The number of rows exceeds the upper limit or is 0.	Check related parameters.
Unauthorized	The account information, such as the AccessKey ID or AccessKey secret, is invalid, or the time gap between the local device and the server is more than 15 minutes.	Configure a valid AccessKey ID or AccessKey secret or make sure that the maximum time gap between the local device and the server is 15 minutes.
DataStoreError	A storage error occurs.	Contact the administrator.
NoPermission	You are not authorized to perform this operation or you have configured an IP address whitelist.	Check whether the granted permissions are correct.
MissingPartitionSpec	The partition information is not specified. If you want to perform operations on a partitioned table, the partition information must be specified.	Specify the partition information.

ErrorCode	Cause	Recommended solution
TableModified	Table data is modified by other tasks during data uploading or downloading.	Recreate a session.
FlowExceeded	The parallelism exceeds the quota.	Check and control parallelism. Before you increase the parallelism, contact the project owner or administrator to evaluate the traffic pressure.
InvalidResourceSpec	The information of the project, table, or partition is inconsistent with that specified for the session.	Check related information and try again later.
MethodNotAllowed	The method to export views is not supported.	Use other methods.
InvalidColumnSpec	The column name is invalid.	Rename the column to ensure that a valid column name is specified for data downloading.
DataVersionConflict	Cross-cluster replication is performed.	Try again later.
InternalServerError	An internal error occurs.	Try again later or contact the administrator.

1.8. MaxCompute MapReduce

1.8.1. Overview

1.8.1.1. MapReduce

This topic describes the MapReduce programming interfaces supported by MaxCompute.

MaxCompute provides the following types of MapReduce programming interfaces:

- MaxCompute MapReduce: MaxCompute native programming interfaces. These interfaces run fast and are easy to develop a program without exposing file systems.
- Extended MaxCompute MapReduce (MR2): an extension of MaxCompute MapReduce. This type of interface supports the logic to schedule complex jobs. The implementation method of these interfaces is the same as that used by the programming interfaces of MaxCompute MapReduce.
- Hadoop-compatible MapReduce: the programming interfaces that are highly compatible with Hadoop MapReduce. This type of interface is not compatible with MR2.

> **Note**
>
> - The basic concepts, job submission, input and output, and resource usage of the three types of interfaces are similar. The only difference lies in SDKs for Java.
> - You cannot use MapReduce to read data from or write data to external tables.

Scenarios

MapReduce supports the following scenarios:

- Search: web crawling, inverted index, and PageRank.
- Analysis of web access logs:

  - Analyze and summarize the characteristics of user behavior, such as web browsing and online shopping. The analysis can be used to deliver personalized recommendations.
  - Analyze user access behavior.
- Statistical analysis of texts:
  - Word count and term frequency-inverse document frequency (TFIDF) analysis of popular novels.
  - Statistical analysis of references to academic papers and patent documents.
  - Wikipedia data analysis.
- Mining of large amounts of data: mining of unstructured data, spatio-temporal data, and image data.
- Machine learning: supervised learning, unsupervised learning, and classification algorithms, such as decision trees and support vector machines (SVMs).
- Natural language processing (NLP):
  - Training and forecast based on big data.
  - Construction of a co-occurrence matrix, mining of frequent itemset data, and duplicate document detection based on existing libraries.
- Advertisement recommendations: forecast of the click-through rate (CTR) and conversion rate (CVR).

Procedure

A MapReduce program processes data in two stages in sequence: the map and reduce stages. You can specify the logic to process data in the map and reduce stages. However, the logic must comply with the conventions of the MapReduce framework. The following procedure shows how the MapReduce framework processes data:

1. Before you perform map operations, make sure that input data is partitioned. After partitioning, the input data is divided into equally sized blocks that are called partitions. Each partition is processed as the input of a single map worker.
2. After partitioning, multiple map workers work at the same time. Each map worker reads its respective partition data, computes the data, and exports the result to a reduce worker.

   > **Note** To generate data, a map worker must specify a key for each output record. The key determines the reduce worker for which the data record is targeted. Multiple keys may correspond to a single reduce worker. Data records with the same key are sent to the same reduce worker. A single reduce worker may receive data records with different keys.

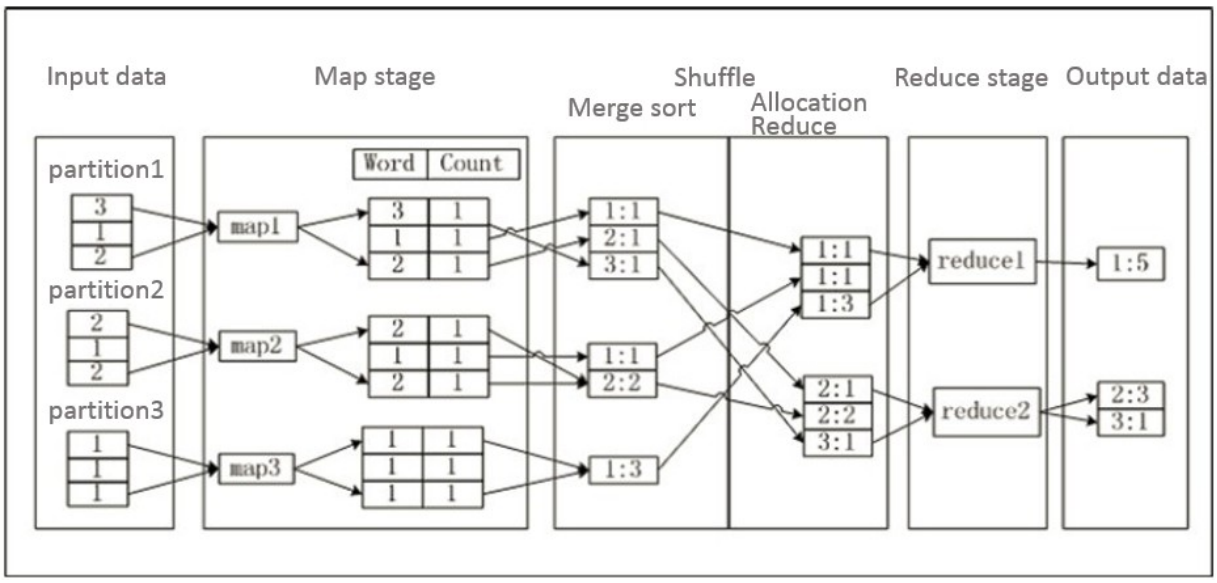
3. Before MapReduce enters the reduce stage, the MapReduce framework sorts data based on key values to make data records with the same key adjacent. If you specify a combiner, the MapReduce framework calls the combiner to combine data records with the same key.

   > **Note** You can define the logic of the combiner. Unlike the typical MapReduce framework, MaxCompute requires the input and output parameters of the combiner to be consistent with reduce workers. This process is generally called shuffle.

4. When the MapReduce program enters the Reduce stage, data records with the same key are sent to the same reduce worker. A single reduce worker may receive data records from multiple map workers. Each reduce worker performs the reduce operation on multiple data records with the same key. After the reduce operation, all data records with the same key are converted into a single value.
5. The MapReduce framework generates the result.

The following example describes the concepts of MaxCompute MapReduce for WrodCount at different stages.

Assume that a file named a.txt exists and each line of the file contains a digit. You want to count the number of times each digit appears. Each digit is called a word, and the number of times it is used represents the count. The following figure shows how MaxCompute MapReduce counts the words.



1. Partitions the a.txt file and uses data in each partition as the input of a single map worker.
2. For the map processing input, the Count parameter is set to 1 for each obtained word. The <Word, Count> pair is output as a word data key.
3. In the early phase of shuffle, the output of each map worker is sorted by key value (word value). After data records are sorted, the records are combined. To perform this operation, you must accumulate the count values that share the same key value to generate a <Word, Count> pair. This is a sorting and combining process.
4. In the late phase of shuffle, data is sent to the reduce workers. The reduce workers sort the received data records based on the key values.
5. In the reduce stage, each reduce worker uses the same logic as combiner to process data, and accumulates the count value with the same key value (word value) to obtain the output result.

Note All the MaxCompute data is stored in tables. Therefore, the input and output data of MaxCompute MapReduce can be saved only as tables. You cannot specify the output format, and no interfaces similar to file systems are provided.

1.8.1.2. MapReduce 2

This topic describes MapReduce 2 (MR2) supported by MaxCompute.

In MapReduce, data must be stored in a MaxCompute table or distributed file system, such as a Hadoop Distributed File System (HDFS), after each round of MapReduce operations. Typically, MapReduce runs multiple MapReduce jobs at the same time. After each job is complete, data is written to disks. However, data may need to be read only once in subsequent map operations to prepare for the shuffle operation. In this case, redundant I/O operations are performed on the disk.

The computing scheduling logic of MaxCompute supports more complicated programming models. Reduce operations can be consecutively performed without having a map operation in between. MaxCompute supports MR2 to allow reduce operations to be consecutively performed after a map operation.

MR2 supported by MaxCompute changes the underlying scheduling and I/O model to avoid redundant I/O operations when you run a job.

Hadoop ChainMapper and ChainReducer also support map or reduce operations in a chained fashion. However, they are essentially different from MR2.

Hadoop ChainMapper and ChainReducer follow the working mechanism used by MapReduce. They allow more than one map operation to be performed after a map or reduce operation. A map or reduce operation cannot be followed by reduce operations. This way, you can reuse the preceding map operation logic to split a map or reduce operation into multiple map stages. This does not change the underlying scheduling or I/O model.

1.8.1.3. Compatibility with Hadoop MapReduce

This topic describes the background information of compatibility with Hadoop MapReduce. It also describes how to use the Hadoop MapReduce plug-in.

MaxCompute provides a set of programming models and interfaces of Hadoop MapReduce. The input and output of the interfaces are data in MaxCompute tables. The data is organized as records, which demonstrate how the data is processed.

Programming interfaces of MaxCompute MapReduce differ from those of Hadoop MapReduce. To migrate Hadoop MapReduce jobs to MaxCompute MapReduce, you must rewrite the MapReduce code, compile and debug the code by calling MaxCompute MapReduce interfaces, package the final code into a JAR file, and then upload the file to MaxCompute. This process is tedious and labor-intensive for development and testing. It was expected that the original Hadoop MapReduce code can be used on MaxCompute with few modifications or without configurations.

To achieve the ideal solution, MaxCompute provides a plug-in to adapt Hadoop MapReduce to MaxCompute MapReduce. The plug-in enables Hadoop MapReduce jobs to be compatible with MaxCompute MapReduce at the binary level. You can directly run the original Hadoop MapReduce JAR files on MaxCompute after you configure relevant settings. Code rewriting is not required. The plug-in is under testing and does not support custom comparators or key types.

The following procedure describes how to use the Hadoop MapReduce plug-in with the WordCount program.

> **Note**
> - For more information about compatibility with Hadoop MapReduce, see Compatibility with Hadoop MapReduce.
> - For more information about the Hadoop MapReduce SDK, see the MapReduce official documentation.

Download the Hadoop MapReduce plug-in

Download the Hadoop MapReduce plug-in file, openmr_hadoop2openmr-1.0.jar.

> **Note** This JAR file contains the dependencies of Hadoop 2.7.2. To avoid version conflicts, you must not include Hadoop dependencies in the JAR files of your jobs.

Prepare the JAR file

Compile and export the WordCount JAR file wordcount_test.jar. Sample source code of the WordCount program:

```
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. Execute the following statements to create the input table wc_in and the output table wc_out:

```
create table if not exists wc_in(line string);
create table if not exists wc_out(key string, cnt bigint);
```

2. Prepare the data that you want to import from the data.txt file into the wc_in table.

   Sample data in the data.txt file:

```
hello maxcompute
hello mapreduce
```

3. Run the following Tunnel command on the MaxCompute client to import the preceding data from the data.txt file to the wc_in table:

```
tunnel upload data.txt wc_in;
```

Configure the mapping between HDFS directories and MaxCompute tables

Configure the mapping between HDFS directories and MaxCompute tables in the wordcount-table-res.conf file. Sample configurations in the wordcount-table-res.conf file:

```
{
  "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 wordcount-table-res.conf file is a JSON file that describes the mapping between HDFS directories and MaxCompute tables. You must configure the input and output data. Each HDFS directory requires you to configure the following parameters: resolver, tableInfos, and matchMode.

Parameters:

- resolver: specifies how to process data in files. The following built-in resolvers can be used: com.aliyun.odps.mapred.hadoop2openmr.resolver.TextFileResolver and com.aliyun.odps.mapred.hadoop2openmr.resolver.BinaryFileResolver. After you specify the resolver parameter, you must configure the required properties for the resolver to parse data.
  - TextFileResolver: regards the input or output as plaintext if data is of the plaintext type. If you specify the resolver parameter for input, you must configure the text.resolver.columns.combine.enable and text.resolver.seperator properties. If 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 to list keys and values.
  - BinaryFileResolver: converts binary data into a data type that is supported by MaxCompute, such as BIGINT, BOOLEAN, and DOUBLE. If you specify the resolver parameter for output, you must configure the binary.resolver.input.key.class and binary.resolver.input.value.class properties. binary.resolver.input.key.class specifies the key class of the intermediate result, and binary.resolver.input.value.class indicates the value class of the intermediate result.
- tableInfos: the MaxCompute table that maps to an HDFS directory. Only the tblName parameter can be configured. You must configure the partSpec and label parameters the same as those in the preceding example.
- matchMode: specifies how MaxCompute tables map to HDFS directories. It can be set to exact or fuzzy. If this parameter is set to fuzzy, you can use a regular expression to map tables to HDFS directories.

Submit a job

Run the following command to submit a job on the MaxCompute client, odpscmd:

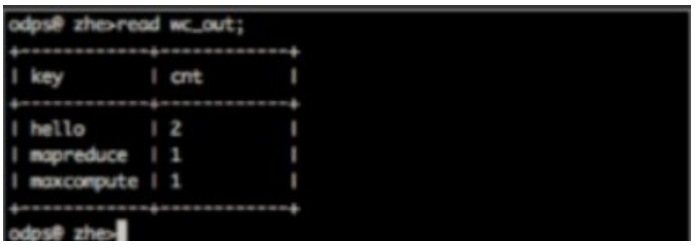
```
jar -DODPS_HADOOPMR_TABLE_RES_CONF=./wordcount-table-res.conf -classpath hadoop2openmr-1.0.jar,wordc
ount_test.jar com.aliyun.odps.mapred.example.hadoop.WordCount /foo/bar;
```

Note

- wordcount-table-res.conf: the configuration file that includes the wc_in and wc_out tables. The tables are mapped to the /foo/bar directory.
- wordcount_test.jar: the JAR file of the Hadoop MapReduce plug-in.
- com.aliyun.odps.mapred.example.hadoop.WordCount: the class name of the job that you want to run.
- /foo/bar: the HDFS directory, which is mapped to the wc_in and wc_out tables in the JSON configuration file.
- After you configure the mapping, you must use the Data Integration service of DataWorks to import the HDFS input file to the wc_in table for MapReduce computing, and export the wc_out table to the HDFS output directory */bar*.
- To run the preceding command, make sure that you have stored the hadoop2openmr-1.0.jar, wordcount_test.jar, and wordcount-table-res.conf files in the /foo/bar directory of odpscmd. If you do not store the files in the /foo/bar directory, modify the configuration and the -classpath parameter in the command.

The following figure shows the running process of the job.

After the job is complete, you can view the information in the wc_out table to check whether the job succeeds and whether the results meet your expectations.



1.8.2. Limits

This topic describes the limits of MapReduce.

- The memory consumed by a map or reduce worker can be 256 MB to 12 GB. By default, a single map or reduce worker occupies 2,048 MB memory of the MapReduce framework, and occupies 1,024 MB heap memory of a JVM.
- The number of resources referenced by a single task cannot exceed 256. A partitioned table is considered one resource. The number of input data records for a single task cannot exceed 1,024 and the number of output data records for a single task cannot exceed 256. The number of custom counters in a single task cannot exceed 64. The total bytes of resources referenced by a single task cannot exceed 2 GB.
- The number of map workers in a job is calculated by the MapReduce framework based on the split size. If no input table is specified, you can use odps.stage.mapper.num to specify the number of map workers. The number is in the range of 1 to 100,000.
- By default, the number of reduce workers in a job is one fourth of the number of map workers. You can set the number to a value that ranges from 0 to 2000. If reducers process much more data than mappers, the data processing in the reduce stage is time-consuming. A maximum of 2,000 reducers can be created.
- The maximum number of retries that are allowed for a map or reduce worker is 3. Exceptions that do not allow retries may cause jobs to fail.
- If the MapReduce program works in local mode, the default number of map workers is 2 and the default number of reduce workers is 1. The maximum number of map workers or reduce workers cannot exceed 100. The default number of download records is 100 and the number cannot exceed 10,000.
- The number of times that a map or reduce worker repeatedly reads a resource cannot exceed 64.
- The MapReduce framework determines the number of map workers based on the split size.

- Each column of the STRING type in a MaxCompute table cannot exceed 8 MB in length.
- If a map or reduce worker does not read or write data, or stops sending heartbeats by using `context.progress()`, the default timeout period is 600 seconds.
- When a MapReduce task references a table resource, the supported data types are BIGINT, DOUBLE, STRING, DATETIME, and BOOLEAN. An error is reported for tables of other data types.
- MapReduce does not read OSS data.
- MapReduce does not support the data types that are added to MaxCompute V2.0.

1.8.3. Features

1.8.3.1. Run command

This topic describes the command that is used to run MapReduce jobs.

The MaxCompute client provides a jar command for running MapReduce jobs.

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, seperate by comma
```

Parameters:

- -conf <configuration file>: the JobConf configuration file.
- -classpath <local_file_list>: the classpath that is used to run a MapReduce job in local mode. This parameter specifies the relative and absolute paths of the JAR package where the main function is located.
- -D <prop_name>=<prop_value>: the Java property of <mainClass> for a MaxCompute job that runs in local mode. You can specify multiple Java properties for a MaxCompute job.
- -local: specifies that a MapReduce job runs in local mode. This parameter is used for program debugging.
- -resources <resource_name_list>: the resources that are used to run a 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 data from other MaxCompute resources, you must add the names of these resources to resource_name_list. Resource names in resource_name_list are separated by commas (,). If you use cross-project resources, add `PROJECT_NAME/resources/` before the resource name, for example, `resources otherproject/resources/resfile`.

You can use the -conf <configuration file> parameter to specify the JobConf file. This file contains the settings of JobConf in the SDK. Example of a JobConf file:

```
<configuration>
<property>
<name>import.filename</name>
<value>resource.txt</value>
</property>
</configuration>
```

In the preceding example, the JobConf file defines the import.filename variable. The value of this variable is resource.txt. You can obtain the value of this variable by calling the JobConf interface in MapReduce. You can also call the JobConf interface 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
```

1.8.3.2. Concepts

1.8.3.2.1. Map/Reduce

This topic describes Map and Reduce, which are basic concepts in MaxCompute MapReduce.

If a map or reduce task runs, the setup(), map() or reduce(), and cleanup() methods are called. The setup() method is called before the map() or reduce() method. Each worker calls the setup() method only once. The cleanup() method is called after the map() or reduce() method. Each worker calls the cleanup() method only once.

> **Note** For more information about the usage examples, see Sample programs.

1.8.3.2.2. Sorting

This topic describes sorting, a basic concept in MaxCompute MapReduce.

Some columns in the key records generated by a mapper can be used as sort columns. These columns do not support a custom comparator. You can select a few sort columns as group columns. These columns do not support a custom group comparator. Sort columns are used to sort your data, whereas group columns are used for secondary sorting.

> **Note** For detailed examples, see Secondary sorting example.

1.8.3.2.3. Partition

This topic describes partition, a basic concept in MaxCompute MapReduce.

MaxCompute allows you to configure partition key columns and custom partitioners. Partition key columns take precedence over custom partitioners. Partitioners are used to allocate data generated by a mapper to different reducers based on the partition logic.

1.8.3.2.4. Combiner

This topic describes the combiner function, a basic concept in MaxCompute MapReduce.

The combiner function combines adjacent records at the shuffle stage. You can determine whether to use the combiner function based on your business logic.

The combiner function is the optimization of the MapReduce computing framework. The combiner logic is the same as the reducer logic. After a mapper generates data, the framework combines the data with the same key at the map stage.

> **Note** For detailed examples, see Sample programs.

1.8.3.3. Submit a job

This topic describes how to run a jar command on the MaxCompute client to submit a MapReduce job.

The MaxCompute client provides a jar command to submit MapReduce jobs. Command 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 comma
        -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
```

Parameters:

- -conf <configuration file>: the JobConf file. This file can affect the settings of JobConf in an SDK.

  Template of a JobConf file:

  ```
  <configuration>
         <property>
            <name>import.filename</name>
            <value>resource.txt</value>
         </property>
      </configuration>
  ```

  In the preceding template, the JobConf file defines the import.filename variable. The value of this variable is resource.txt. You can obtain the value of this variable over the JobConf interface in MapReduce. You can also call the JobConf interface in an SDK to obtain the value of the variable. For more information, see Resource usage example.

  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;
  ```

- -resources <resource_name_list>: the resources that are used to run a MapReduce job. Typically, resource_name_list specifies the name of the resource in which the Map or Reduce function is located.

  > **Note** If the Map or Reduce function reads data from other MaxCompute resources, you must add the names of these resources to resource_name_list. The resource names are separated by commas (,). If you use cross-project resources, add `PROJECT/resources/` before resource_name_list, for example, `-resources otherproject/resources/resfile`.

- -classpath <local_file_list>: the classpath that is used to run a MapReduce job in local mode. This parameter specifies the relative and absolute paths of the JAR packages that encapsulates the main function. Package names are separated by default file delimiters. In most cases, the Windows operating system uses semicolons (;) as the default file delimiter, and the Linux operating system uses commas (,) as the default file delimiter. 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 encapsulated into the same package. If you run the related program, mapreduce-examples.jar is specified in the following parameters: -resources <resource_name_list> and -classpath <local_file_list>. However, -resources <resource_name_list> references the Map or Reduce function and is run in a distributed environment, whereas -classpath <local_file_list> references the main function and is run in local mode with the specified JAR package saved in a local directory.

- -D <name>=<value>: the Java attribute of <mainClass> when you run a MapReduce job in local mode. You can define multiple Java attributes.
- -l: specifies that the MapReduce job is executed in local mode. This parameter is used for program debugging.

Example:

```
jar -conf \home\admin\myconf -resources a.txt,example.jar -classpath ..\lib\example.jar:.\other_lib.
jar -D java.library.path=.\native;
```

1.8.3.4. Inputs and outputs

This topic describes the inputs and outputs of MapReduce jobs in MaxCompute.

- The inputs and outputs of MapReduce jobs in MaxCompute support built-in data types of MaxCompute, including BIGINT, DOUBLE, STRING, DATETIME, and BOOLEAN. User-defined data types are not supported.
- MapReduce supports input data from multiple tables with different schemas. You can use the map function to obtain the table information that corresponds to the current record.
- MapReduce supports null values as input data but does not support views as input data.
- A reduce job can write data to different tables or different partitions of a table. The destination tables or partitions can have different schemas. You can specify labels to distinguish outputs. If you want to use the default output, you do not need to specify a label. MapReduce does not support a function without output returned.

> **Note** For more information about input and output examples, see Example: Input and output data to multiple objects.

1.8.3.5. Use resources

This topic describes how MapReduce uses resources.

You can use the Map or Reduce function to read data from MaxCompute resources. A Map or Reduce worker loads resources to the memory for you to write code.

> **Note** For a detailed example, see Resource usage example.

1.8.3.6. Job running in local mode

This topic describes the differences between the local mode and distributed mode in which MapReduce jobs run. It also provides examples of MapReduce jobs in local mode.

Introduction to the local mode

Before you run a job in local mode, you can specify the -local option in the JAR command to simulate the running of the job. This way, you can perform local debugging on the job.

During job running, the client downloads the metadata and data of the input table, metadata of the output table, and resources that are required for local debugging from MaxCompute. The downloaded data is saved to a local directory named *warehouse*.

After the job is completed, the computing results are saved to a file in the *warehouse* directory. If the input table and required resources are downloaded to the *warehouse* directory, MapReduce directly references the data and files in the directory next time, instead of downloading the data again.

Differences between the local mode and distributed mode

A MapReduce job that runs in local mode starts multiple map and reduce tasks to process data. These tasks run in sequence rather than in parallel. The simulated running process is different from an actual distributed running process in the following aspects:

- Rows in the input table: A maximum of 100 rows of data can be downloaded in local mode.
- Resource usage: In distributed mode, MaxCompute limits the size of resources that can be referenced. However, no limits are imposed on the size of resources in local mode.
- Security: MaxCompute MapReduce and user-defined functions (UDFs) are limited by a Java sandbox in distributed mode. However, no limits are imposed in local mode.

Examples

The following code shows an example of a MapReduce job in local mode:

```
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_input_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 more information about the sample code of WordCount, see WordCount example.

If this is the first time you run a local debugging command, a directory named warehouse is created in the current path after the command is executed. The following code shows the directory structure of *warehouse*.

```
<warehouse>
   |____my_project (project directory)
          |____ <__tables__>
          |       |__wc_in (table data directory)
          |       |      |____ data (file)
          |       |      |
          |       |      |____ <__schema__> (file)
          |       |__wc_out (table data directory)
          |               |____ data (file)
          |               |
          |               |____ <__schema__> (file)
          |
          |____ <__resources__>
                  |
                  |___table_resource_name (table resource)
                  |         |____<__ref__>
                  |
                  |___ file_resource_name (file resource)
```

- Directories at the same level as *my_project* indicate projects. Directories at the same level as *wc_in* and *wc_out* indicate data tables. The table data that you read or write by using the JAR command is downloaded to directories at this level.
- The schema file stores the metadata of a table. The following code defines the file format:

  ```
  project=local_project_name
  table=local_table_name
  columns=col1_name:col1_type,col2_name:col2_type
  partitions=p1:STRING,p2:BIGINT    -- In this example, you do not need to specify this field.
  ```

  > **Note** Separate the name and data type of a column with a colon (:). Separate columns with commas (,). The project and table names, projectname.tablename, must be declared at the beginning of the schema file. Separate the declaration and column definition with a comma (,), for example, project_name.table_name,col1_name:col1_type,col2_name:col2_type,.....

- The data file in the *tables* directory stores the table data. The number of columns and column data must match the definition in the schema file. Separate columns with commas (,).

For example, the schema file in the *wc_in* directory contains the following data:

```
my_project.wc_in,key:STRING,value:STRING
```

In this case, the data file contains the following data:

```
0,2
```

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 program, the client directly uses the data in the *wc_in* directory, instead of downloading it again.

> **Note** Data can be downloaded from MaxCompute only for MapReduce jobs that run in local mode.

For example, the schema file in the *wc_out* directory contains the following data:

```
my_project.wc_out,key:STRING,cnt:BIGINT
```

In this case, the data file contains the following data:

```
0,1
2,1
```

The client downloads the metadata of the wc_out table from MaxCompute and saves the data to the schema file. After a job is completed, the results are saved to the data file.

> **Note**
>
> - You can also edit the schema and data files and save the files in table directories.
> - If you run a job in local mode and the client detects that the table directory exists, the client does not download the information of this table from MaxCompute. The local table directory can include a table that does not exist in MaxCompute.

1.8.4. SDK introduction

1.8.4.1. Overview of major MapReduce APIs

This topic describes major MapReduce APIs.

If you use Maven, you can obtain Maven dependencies from the Maven Central Repository.

```
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>odps-sdk-mapred</artifactId>
    <version>0.36.2-public</version>
</dependency>
```

API	Description
MapperBase	The base class that custom Map functions must inherit. A mapper converts the records in the input table to key-value pairs and passes the key-value pairs to a reducer. Alternatively, a mapper can directly write the key-value pairs to a result table by skipping the Reduce stage. The jobs that skip the Reduce stage and directly return computing results are called map-only jobs.
ReducerBase	The base class that custom Reduce functions must inherit. A reducer reduces a set of values that are associated with a key.
TaskContext	Describes the context of a task. The task context is an input parameter of multiple member functions of MapperBase and ReducerBase.
JobClient	Submits and manages jobs. Jobs can be submitted in blocking mode (synchronous) or non-blocking mode (asynchronous).
RunningJob	Defines a running job. The objects of this class are used to track the instances on which MapReduce jobs are running.
JobConf	Describes the configuration of a MapReduce job. The JobConf object is defined in the main function. Then, a job client submits a job to MaxCompute based on the JobConf object.

1.8.4.2. API description

1.8.4.2.1. MapperBase

This topic describes the main function interfaces of the MapperBase API.

The following table describes the main function interfaces of MapperBase.

Interface	Description
void cleanup(TaskContext context)	Calls the related method at the end of the map stage after the map method is called.
void map(long key, Record record, TaskContext context)	Calls the map method to process records in an input table.
void setup(TaskContext context)	Calls the related method at the beginning of the map stage before the map method is called.

1.8.4.2.2. ReducerBase

This topic describes the main function interfaces of the ReducerBase API.

The following table describes the main function interfaces of ReducerBase.

Interface	Description
void cleanup(TaskContext context)	Calls the related method at the end of the reduce stage after the reduce method is called.
void reduce(Record key, Iterator<Record > values, TaskContext context)	Calls the reduce method to process records in an input table.
void setup(TaskContext context)	Calls the related method at the beginning of the reduce stage before the reduce method is called.

1.8.4.2.3. TaskContext

This topic describes the main function interfaces of the TaskContext API.

The following table describes the main function interfaces of TaskContext.

Interface	Description
TableInfo[] getOutputTableInfo()	Gets output table information.
Record createOutputRecord()	Creates records for the default output table.
Record createOutputRecord(String label)	Creates records for the output table with the specified label.
Record createMapOutputKeyRecord()	Creates records for keys in the key-value pairs that are generated at the map stage.
Record createMapOutputValueRecord()	Creates records for values in the key-value pairs that are generated at the map stage.
void write(Record record)	Writes records to the default output table. The interface can be called multiple times at the reduce stage.

Interface	Description
void write(Record record, String label)	Writes records to the output table with the specified label. The interface can be called multiple times at the reduce stage.
void write(Record key, Record value)	Writes records to the intermediate results. The interface can be called multiple times at the map stage.
BufferedInputStream readResourceFileAsStream(String resourceName)	Reads a file resource.
Iterator<Record > readResourceTable(String resourceName)	Reads a table resource.
Counter getCounter(Enum<? > name)	Obtains the counter with the specified name.
Counter getCounter(String group, String name)	Obtains the counter with the specified name in the specified group.
void progress()	Sends heartbeat information to the MapReduce framework. If your task takes an extended period of time to process data and you do not need to call the framework during this time period, you can call this interface to avoid a task timeout. The default timeout period for a task is 600 seconds.

Note

- The TaskContext API has a progress interface, which prevents it from being forced out due to timeout if a worker runs for a long time. This interface sends heartbeats to the framework rather than reporting the worker progress.
- The default timeout period for a worker is 10 minutes in MaxCompute MapReduce. You cannot change the timeout period. If a worker does not send heartbeat information by calling the progress interface in 10 minutes, the framework terminates the worker and the map or reduce task fails. Therefore, we recommend that you periodically call the progress interface in a map or reduce task to prevent the framework from terminating workers unexpectedly.

1.8.4.2.4. JobConf

This topic describes the main function interfaces of the JobConf API.

The following table describes the main function interfaces of JobConf.

Interface	Description
void setResources(String resourceNames)	Declares the resources that are used in the current job. A mapper or reducer can read only the resources that have been declared in the TaskContext object.
void setMapOutputKeySchema(Column[] schema)	Sets the attributes of keys that are passed from the mapper to the reducer.
void setMapOutputValueSchema(Column[] schema)	Sets the attributes of values that are passed from the mapper to the reducer.

Interface	Description
void setOutputKeySortColumns(String[] cols)	Sets the columns for sorting the keys that are passed from the mapper to the reducer.
void setOutputGroupingColumns(String[] cols)	Sets the columns for grouping the keys.
void setMapperClass(Class<? extends Mapper > > > theClass)	Sets a mapper for a job.
void setPartitionColumns(String[] cols)	Sets the partition key columns for a job. By default, the partition key columns are all columns of the keys that are generated by the mapper.
void setReducerClass(Class<? extends Reducer > theClass)	Sets a reducer for a job.
void setCombinerClass(Class<? extends Reducer > theClass)	Sets a combiner for a job. A combiner combines records with the same key. It is similar to a reducer but works at the map stage.
void setSplitSize(long size)	Sets the split size. Unit: MB. The default split size is 256 MB.
void setNumReduceTasks(int n)	Sets the number of reduce tasks. By default, the number of reduce tasks is one-fourth of the number of map tasks.
void setMemoryForMapTask(int mem)	Sets the memory available to a worker in a map task. Unit: MB. The default memory size is 2048 MB.
void setMemoryForReduceTask(int mem)	Sets the memory available to a worker in a reduce task. Unit: MB. The default memory size is 2048 MB.
void setOutputSchema(Column[] schema, String label)	Sets the output attribute of a designated label. If data is inserted into multiple objects, each output corresponds to a label.

Note

- The grouping columns are selected from the sort columns. The sort columns and partition key columns must exist in keys.
- At the map stage, the hash values of records from a mapper are calculated based on the specified partition key columns. The hash values help determine the reducers to which records are passed. The records are sorted based on the sort columns before the records are passed to reducers.
- At the reduce stage, input records are grouped based on the grouping columns. Then, a group of records that share the same key are passed to the reduce method as input.

1.8.4.2.5. JobClient

This topic describes the main function interfaces of the JobClient API.

The following table describes the main function interfaces of JobClient.

Interface	Description

Interface	Description
static RunningJob runJob(JobConf job)	Submits a MapReduce job in blocking (synchronous) mode and returns a RunningJob object.
static RunningJob submitJob(JobConf job)	Submits a MapReduce job in non-blocking (asynchronous) mode and returns a RunningJob object.

1.8.4.2.6. RunningJob

This topic describes the main function interfaces of the RunningJob API.

The following table describes the main function interfaces of RunningJob.

Interface	Description
String getInstanceID()	Obtains the ID of a job instance. You can use the job instance ID to view operational logs and manage jobs.
boolean isComplete()	Checks whether a job is completed.
boolean isSuccessful()	Checks whether a job instance is successful.
void waitForCompletion()	Waits for a job instance to end. The method is used for jobs that are submitted in asynchronous mode.
JobStatus getJobStatus()	Checks the running status of a job instance.
void killJob()	Ends the current job.
Counters getCounters()	Obtains the counter information.

1.8.4.2.7. InputUtils

This topic describes the main function interfaces of the InputUtils API.

The following table describes the main function interfaces of InputUtils.

Interface	Description
static void addTable(TableInfo table, JobConf conf)	Adds an input table to a task. The method can be called multiple times. New tables are appended to the input queue.
static void setTables(TableInfo [] tables, JobConf conf)	Adds multiple input tables to a task.

1.8.4.2.8. OutputUtils

This topic describes the main function interfaces of the OutputUtils API.

The following table describes the main function interfaces of OutputUtils.

Interface	Description
static void addTable(TableInfo table, JobConf conf)	Adds an output table to a task. The method can be called multiple times. New tables are appended to the output queue.

Interface	Description
static void setTables(TableInfo [] tables, JobConf conf)	Adds multiple output tables to a task.

1.8.4.2.9. Pipeline

This topic describes the main interfaces of the Pipeline API.

Pipeline is the main class of MR2. You can call the Pipeline.builder method to build a pipeline. The following code shows the main interfaces of the Pipeline class:

```
public Builder addMapper(Class<? extends Mapper> mapper)
public Builder addMapper(Class<? extends Mapper> mapper,
       Column[] keySchema, Column[] valueSchema, 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[] valueSchema, String[] sortCols,
       SortOrder[] order, String[] partCols,
       Class<? extends Partitioner> theClass, String[] groupCols)
public Builder setOutputKeySchema(Column[] keySchema)
public Builder setOutputValueSchema(Column[] valueSchema)
public Builder setOutputKeySortColumns(String[] sortCols)
public Builder setOutputKeySortOrder(SortOrder[] order)
public Builder setPartitionColumns(String[] partCols)
public Builder setPartitionerClass(Class<? extends Partitioner> theClass)
public Builder setOutputGroupingColumns(String[] cols)
```

The following example shows how to call the Pipeline.builder method to build a pipeline:

```
Job job = new Job();
Pipeline pipeline = Pipeline.builder()
 .addMapper(TokenizerMapper.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 create a MapReduce job in which a mapper is followed by two reducers in the Main function. If you are familiar with the basic features of MapReduce, MR2 is easy to use.

> **Note**
>
> - Before you use MR2, we recommend that you learn how to use MapReduce.
> - You can create a MapReduce job in which a mapper is followed by only one reducer by using JobConf.

1.8.4.3. Compatibility with Hadoop MapReduce

This topic describes the compatibility between specific MaxCompute MapReduce interfaces and 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.
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.

Type	Interface	Compatible with Hadoop MapReduce
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.
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.

Type	Interface	Compatible with Hadoop MapReduce
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.
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.

Type	Interface	Compatible with Hadoop MapReduce
Job	setCacheFiles(URI[] files)	No. An exception is reported.
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.
Job	setOutputValueClass(Class<? > theClass)	Yes.
Job	setPartitionerClass(Class<? extends Partitioner> cls)	Yes.

Type	Interface	Compatible with Hadoop MapReduce
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.
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.

Type	Interface	Compatible with Hadoop MapReduce
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.
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.

Type	Interface	Compatible with Hadoop MapReduce
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.
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.

Type	Interface	Compatible with Hadoop MapReduce
JobConf	deleteLocalFiles()	No. An exception is reported.
JobConf	deleteLocalFiles(String subdir)	No. An exception is reported.
JobConf	normalizeMemoryConfigValue(long val)	Empty implementation.
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.
JobConf	setMapRunnerClass(Class<? extends MapRunnable> theClass)	No. An exception is reported.

Type	Interface	Compatible with Hadoop MapReduce
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.
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.

Type	Interface	Compatible with Hadoop MapReduce
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.
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.

Type	Interface	Compatible with Hadoop MapReduce
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.
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.

Type	Interface	Compatible with Hadoop MapReduce
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.
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.

Type	Interface	Compatible with Hadoop MapReduce
DistributedCache	getSymlink(Configuration conf)	No. An exception is reported.
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.
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.

Type	Interface	Compatible with Hadoop MapReduce
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.

1.8.5. Data types

This topic describes the data types supported by MapReduce.

MapReduce supports the following data types: BIGINT, DOUBLE, STRING, DATETIME, BOOLEAN, and DECIMAL. The following table lists the mappings between MaxCompute data types and Java data types.

MaxCompute SQL Type	Java Type
BIGINT	LONG
DOUBLE	DOUBLE
DECIMAL	BIGDECIMAL
BOOLEAN	BOOLEAN
STRING	STRING
DATETIME	DATE

1.8.6. Sample programs

1.8.6.1. WordCount example

This topic provides a 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.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();
        }
        // The combiner implements the same interface as that of the reducer. The combiner allows yo
u to immediately run a local reduce job on the mapper to reduce the output of the mapper.
        @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);
        // Configure the schema that defines the intermediate output of the mapper as key-value pair
s. The intermediate output of the mapper exists as records.
        job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
        job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));
        // Configure information about input and output tables.
        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

This topic provides a MapOnly example of MapReduce.

For MapOnly jobs, a mapper directly generates key-value pairs to MaxCompute tables. You need only to specify output tables. You do not need to specify the key-value metadata for the output of a mapper.

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);
            // The main function executes the following logic only if option.mapper.setup is set to
true in the JobConf file:
            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);
            // The main function executes the following logic only if option.mapper.map is set to tr
ue in the JobConf file:
            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);
            // The main function executes the following logic only if option.mapper.cleanup is set t
o true in the JobConf file:
            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);
        // For MapOnly jobs, the number of reducers must be explicitly set to 0.
        job.setNumReduceTasks(0);
        // Configure information about input and output tables.
        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]);
            // You can specify key-value pairs in the JobConf file, and use getJobConf of the contex
t to query the configurations in a mapper.
            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. MultipleInOut example

This topic provides a MultipleInOut example.

MaxCompute jobs can read data from multiple input tables and write data to multiple output tables. To read data from multiple input tables, you must make sure that the input tables have the same number of columns and the same data types. These requirements do not apply to output tables.

Example:

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
```

```
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.
     **/
public class MultipleInOut {
    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.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;
        private Record result1;
        private Record result2;
        @Override
            public void setup(TaskContext context) throws IOException {
            // Create a record for each output and add labels to distinguish outputs.
            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);
                // If you do not specify a label, the default output is used.
                context.write(result);
```

```
                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, "out2");
            result2.set(1, 1L);
            context.write(result2, "out2");
        }
    }
    // Convert partition strings such as "ds=1/pt=2" to MAP.
    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.
```

```
        // 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. MultiJobs example

This topic provides a MultiJobs 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
     *
     **/
public class MultiJobs {
    public static class InitMapper extends MapperBase {
        @Override
            public void setup(TaskContext context) throws IOException {
            Record record = context.createOutputRecord();
            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 that are defined in the main function from JobConf.
            long expect = context.getJobConf().getLong("multijobs.expect.value", -1);
            long v = -1;
            int count = 0;
            // Read the data from the output table of the previous job.
            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);
            // Set the counter. The counter value can be obtained in the main function after the job
is complete.
            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"));
        // Explicitly set the number of reducers to 0 for map-only jobs.
        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);
            // Explicitly set the number of reducers to 0 for map-only jobs.
            decJob.setNumReduceTasks(0);
            RunningJob rJob = JobClient.runJob(decJob);
            iterCount--;
            // If the specified number of iterations is reached, exit the loop.
            if (rJob.getCounters().findCounter("multijobs", "value").getValue() == 0) {
                break;
            }
        }
        if (iterCount ! = 0) {
            throw new IOException("Job failed.") ;
        }
    }
}
```

1.8.6.5. SecondarySort example

This topic provides a SecondarySort 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 on the first number.
     *
     **/
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.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) 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

This topic provides an example of using resources in MapReduce.

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
     *
     **/
public class Upload {
    public static class UploadMapper extends MapperBase {
        @Override
            public void setup(TaskContext context) throws IOException {
            Record record = context.createOutputRecord();
            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);
        // Specify the resource name, which can be obtained in the map stage by using the JobConf in
terface.
        job.set("import.filename", args[0]);
        // Explicitly set the number of reducers to 0 for MapOnly jobs.
        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 one of the following methods to set the JobConf file:

- Use the JobConf interface in the SDK. This method is used in this example.
- Use the -conf parameter in a jar command to specify a new JobConf file.

1.8.6.7. Counter usage example

This topic provides an example of using counters in MapReduce.

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 custom counters and perform related operations.

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
     *
     **/
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);
```

```
            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);
        // If the job is successful, the values of the custom counters in the job are returned.
        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

This topic provides a 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.
     *
     **/
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 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 IOEx
ception {
            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 IOEx
ception {
            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);
        // Set the regular expression for 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);
        // Specify the output of grepJob as the input of sortJob.
        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" });
        @SuppressWarnings("unused")
            RunningJob rjSort = JobClient.runJob(sortJob);
    }
}
```

1.8.6.9. Join example

This topic provides a Join example.

The MaxCompute MapReduce framework does not support Join operations. However, you can join data by using a custom map or reduce function.

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 org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
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.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
     * Join, mr_Join_src1/mr_Join_src2(key bigint, value string), mr_Join_out(key
     * bigint, value1 string, value2 string)
     *
     */
public class Join {
    public static final Log LOG = LogFactory.getLog(Join.class);
    public static class JoinMapper extends MapperBase {
        private Record mapkey;
        private Record mapvalue;
        private long tag;
        @Override
            public void setup(TaskContext context) throws IOException {
            mapkey = context.createMapOutputKeyRecord();
            mapvalue = context.createMapOutputValueRecord();
            tag = context.getInputTableInfo().getLabel().equals("left") ? 0 : 1;
        }
        @Override
            public void map(long key, Record record, TaskContext context)
            throws IOException {
            mapkey.set(0, record.get(0));
            mapkey.set(1, tag);
            for (int i = 1; i < record.getColumnCount(); i++) {
                mapvalue.set(i - 1, 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();
        }
        // Each input of the reduce function is the records that have the same key.
        @Override
            public void reduce(Record key, Iterator<Record> values, TaskContext context)
            throws IOException {
            long k = key.getBigint(0);
            List<Object[]> leftValues = new ArrayList<Object[]>();
            // Records are sorted based on the combination of the key and tag. This ensures that rec
ords in the left table are passed to the reduce function first when the reduce function performs the
Join operation.
            while (values.hasNext()) {
                Record value = values.next();
                long tag = (Long) key.get(1);
                // Data in the left table is first cached in memory.
                if (tag == 0) {
                    leftValues.add(value.toArray().clone());
                } else {
                    // Data in the right table is joined with all data in the left table.
                    // The sample code has poor performance and is only used as an example. We recom
mend that you do not use the code in your production environment.
                    for (Object[] leftValue : leftValues) {
                        int index = 0;
                        result.set(index++, k);
                        for (int i = 0; i < leftValue.length; i++) {
                            result.set(index++, leftValue[i]);
                        }
                        for (int i = 0; i < value.getColumnCount(); i++) {
                            result.set(index++, value.get(i));
                        }
                        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("value:string"));
        job.setPartitionColumns(new String[]{"key"});
        job.setOutputKeySortColumns(new String[]{"key", "tag"});
        job.setOutputGroupingColumns(new String[]{"key"});
        job.setNumReduceTasks(1);
        InputUtils.addTable(TableInfo.builder().tableName(args[0]).label("left").build(), job);
        InputUtils.addTable(TableInfo.builder().tableName(args[1]).label("right").build(), job);
        OutputUtils.addTable(TableInfo.builder().tableName(args[2]).build(), job);
        JobClient.runJob(job);
```

```
    }
}
```

1.8.6.10. Sleep example

This topic provides a Sleep example.

Example:

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.conf.JobConf;
public class Sleep {
  private static final String SLEEP_SECS = "sleep.secs";
  public static class MapperClass extends MapperBase {
    // No data is entered, the map function is not executed, and the related logic can be written on
ly into setup.
    @Override
    public void setup(TaskContext context) throws IOException {
      try {
        // Obtain the number of sleep seconds set in JobConf.
        Thread.sleep(context.getJobConf().getInt(SLEEP_SECS, 1) * 1000);
      } catch (InterruptedException e) {
        throw new RuntimeException(e);
      }
    }
  }
  public static void main(String[] args) throws Exception {
    if (args.length ! = 1) {
      System.err.println("Usage: Sleep <sleep_secs>");
      System.exit(-1);
    }
    JobConf job = new JobConf();
    job.setMapperClass(MapperClass.class);
    // This instance is also a MapOnly job and the number of reducers must be set to 0.
    job.setNumReduceTasks(0);
    // The number of mappers must be specified by the user because no input table is provided.
    job.setNumMapTasks(1);
    job.set(SLEEP_SECS, args[0]);
    JobClient.runJob(job);
  }
}
```

1.8.6.11. Unique example

This topic provides a 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
     *
     **/
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];
        }
        // The input group of Reduce is determined by the value of the setOutputGroupingColumns para
meter. If this parameter is not specified, the default value MapOutputKeySchema is used.
        // 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

This topic provides a 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.
     *
     **/
public class Sort {
 static int printUsage() {
  System.out.println("sort <input> <output>");
  return -1;
 }
 /**
       * Implements 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);
  // For global sorting, the number of reducers is set to 1. All the data is transferred to the same
reducer.
  // This method applies only to the scenarios when small amounts of data are processed. If large am
ounts of data need to be processed, use other methods, such as TeraSort.
  jobConf.setNumReduceTasks(1);
  jobConf.setMapOutputKeySchema(SchemaUtils.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 job took " + (end_time.getTime() - startTime.getTime()) / 1000 + " seconds
.") ;
 }
}
```

1.8.6.13. Examples of using partitioned tables as input

This topic provides examples of using partitioned tables as 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", "654321");
    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 example 2, the MaxCompute SDK and MapReduce SDK are combined to implement a MapReduce task that reads data from specific partitions.
- The code in the example cannot be compiled for execution. It is only an example of the main function.
- The applicable function is user logic that determines whether the partition can be used as the input of a MapReduce job.

1.8.6.14. Pipeline example

This topic provides a Pipeline example.

Example:

```
package com.aliyun.odps.mapred.open.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 WordCountPipelineTest {
    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 value;
        @Override
            public void setup(TaskContext context) throws IOException {
            value = 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);
            }
            value.set(0, count);
            context.write(key, value);
        }
    }
    public static class IdentityReducer extends ReducerBase {
        private Record result;
        @Override
            public void setup(TaskContext context) throws IOException {
            result = context.createOutputRecord();
        }
        @Override
            public void reduce(Record key, Iterator<Record> values, TaskContext context)
            throws IOException {
            while (values.hasNext()) {
                result.set(0, key.get(0));
                result.set(1, values.next().get(0));
                context.write(result);
            }
        }
    }
```

```
    public static void main(String[] args) throws OdpsException {
        if (args.length ! = 2) {
            System.err.println("Usage: WordCountPipeline <in_table> <out_table>");
            System.exit(2);
        }
        Job job = new Job();
        /***
         * During pipeline construction, if you do not specify OutputKeySortColumns, PartitionColumn
s, and OutputGroupingColumns for a mapper, the framework uses OutputKey of the mapper as the default
values of these parameters.
         ***/
        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("word", OdpsType.STRING) })
            .setOutputValueSchema(
            new Column[] { new Column("count", OdpsType.BIGINT)})
            .addReducer(IdentityReducer.class).createPipeline();
        // Add the pipeline to JobConf. If you want to configure a combiner, use JobConf.
        job.setPipeline(pipeline);
        // Configure the input and output tables.
        job.addInput(TableInfo.builder().tableName(args[0]).build());
        job.addOutput(TableInfo.builder().tableName(args[1]).build());
        // Submit the job and wait for the job to complete.
        job.submit();
        job.waitForCompletion();
        System.exit(job.isSuccessful() == true ? 0 : 1);
    }
}
```

1.9. MaxCompute Graph

1.9.1. Graph overview

1.9.1.1. Overview

This topic describes the components of MaxCompute Graph and the supported operations.

MaxCompute Graph is a processing framework designed for iterative graph computing. Graph computing jobs use graphs to build models. A graph is a collection of vertices and edges that have values.

MaxCompute Graph allows you to perform the following operations on graphs:

- Change the value of a vertex or edge.
- Add or remove a vertex.
- Add or remove an edge.

> **Note**
>
> - When you edit a vertex or edge, you must use code to maintain the relationships between vertices and edges.
> - A directed graph has the following types of edges.
>   - Outgoing edge: a directed edge on which the current vertex is the origin.
>   - Incoming edge: a directed edge on which the current vertex is the destination.

MaxCompute Graph performs iterations to edit graphs, enable graphs to evolve, and obtain analysis results. Typical applications include the PageRank, single source shortest path (SSSP) algorithm, and k-means clustering algorithm. You can use SDK for Java provided by MaxCompute Graph to compile graph computing programs.

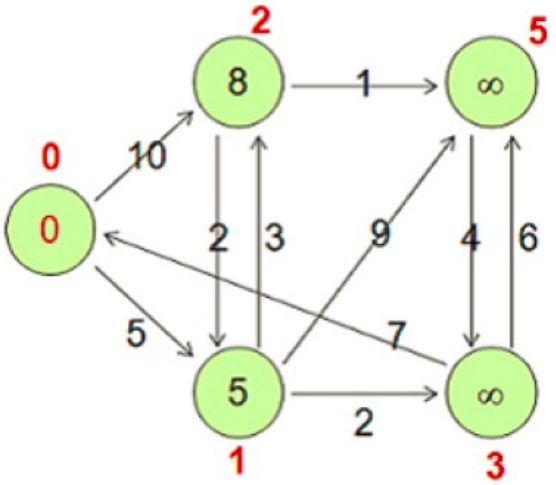
1.9.1.2. Data structure of MaxCompute Graph

This topic describes the data structure of MaxCompute Graph.

MaxCompute Graph processes directed graphs. MaxCompute provides only a two-dimensional table storage structure. Therefore, you must resolve graph data into two-dimensional tables and store them in MaxCompute.

During graph computing and analytics, use custom GraphLoader to convert two-dimensional table data into vertices and edges that are applicable to MaxCompute Graph. You can determine how to resolve graph data into two-dimensional tables based on your business requirements.

The structure of a vertex is <ID, Value, Halted, Edges>. ID specifies the identifier of the vertex. Value specifies the value of the vertex. Halted specifies whether the iteration of the vertex is terminated. Edges specifies all the edges that start from the vertex. The structure of an edge is <DestVertexID, Value>. DestVertexID specifies the identifier of the destination vertex. Value specifies the value of the edge. The following figure shows the data structure of MaxCompute Graph.



The preceding figure is composed of the vertexes listed in the following table.

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>]>

Vertex	<ID, Value, Halted, Edges>
v5	<5, Long.MAX_VALUE, false, [<3, 4 >]>

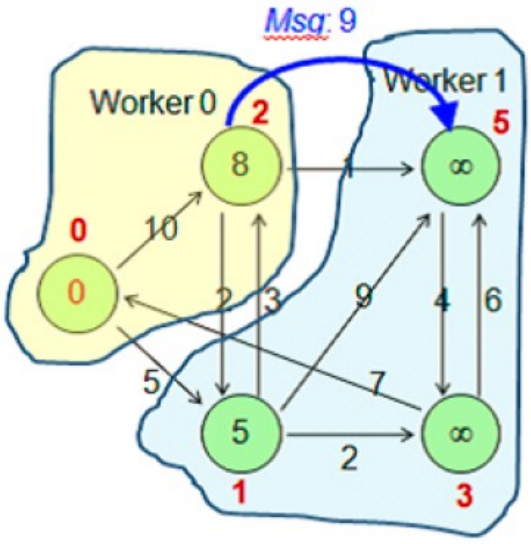
1.9.1.3. Graph logic

1.9.1.3.1. Graph loading

A Graph program performs the following operations: graph loading, iterative computing, and iteration termination. This topic describes the graph loading steps in a Graph program.

Graph loading consists of the following steps:

- Graph loading: MaxCompute Graph calls the custom GraphLoader to resolve the records in the input table into vertices or edges.
- Partitioning: MaxCompute Graph calls the custom Partitioner to partition the vertices and distributes the partitions to the related workers. By default, MaxCompute Graph partitions the vertices based on the hash values of the vertex IDs modulo the number of workers.



In the preceding figure, the number of workers is 2. Vertex 0 and Vertex 2 are distributed to Worker 0 because the result of vertex IDs modulo 2 is 0. Vertex 1, Vertex 3, and Vertex 5 are distributed to Worker 1 because the result of vertex IDs modulo 2 is 1.

1.9.1.3.2. Iterative computing

A Graph program performs the following operations: graph loading, iterative computing, and iteration termination. This topic describes the iterative computing steps in a Graph program.

An iteration is called a superstep. During an iteration, the program traverses all non-halted vertices or all vertices that receive messages, and calls the compute(ComputeContext context, Iterable messages) method. For non-halted vertices, the value of the Halted parameter is false. Halted vertices are automatically activated after they receive messages.

The compute(ComputeContext context, Iterable messages) method can be used to perform the following operations:

- Process the messages that are sent by the previous superstep to the current vertex.
- Edit a graph as required:

  - Change the values of vertices or edges.
  - Send messages to some vertices.
  - Add or remove vertices or edges.
- Aggregate information to obtain global information by using the aggregator.
- Set the current vertex to the halted or non-halted state.
- Enable MaxCompute Graph to asynchronously send messages to the related workers in each superstep. The messages are then processed in the next superstep.

1.9.1.3.3. Iteration termination

A Graph program performs the following operations: graph loading, iterative computing, and iteration termination. This topic describes the iteration termination steps in a Graph program.

An iteration ends if one of the following conditions is met:

- All vertices are in the halted state (the value of the Halted parameter 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 mechanism

This topic describes the implementation mechanism and related API operations of Aggregator. It also describes how to apply Aggregator by using k-means clustering.

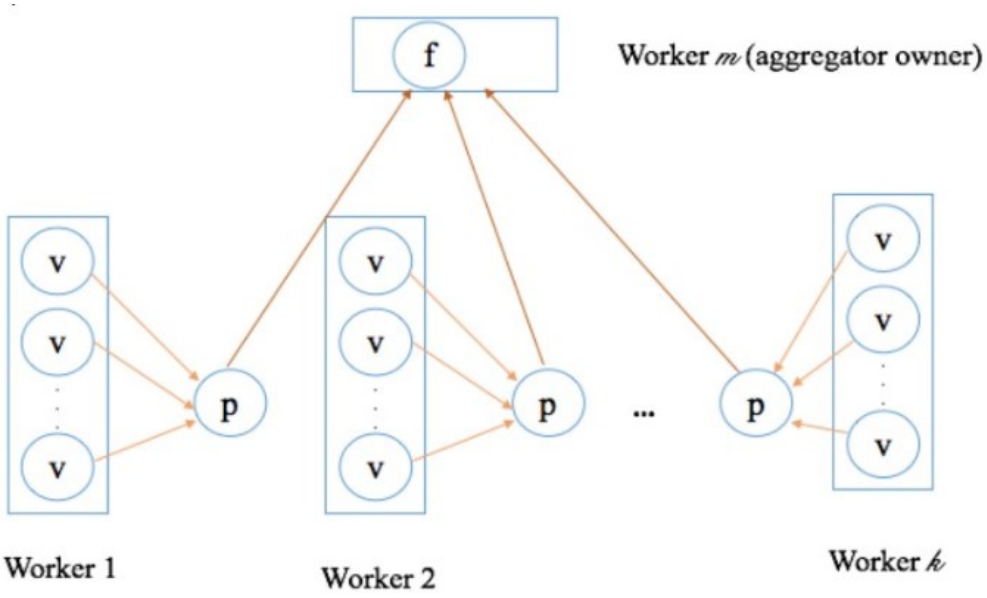
Aggregator is a common feature in MaxCompute Graph jobs and is especially suited for solving machine learning issues. In MaxCompute Graph, Aggregator is used to aggregate and process global information.

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 each superstep starts. It is used to initialize AggregatorValue. In the first superstep (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 the Aggregator owner resides. 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, and partial indicates objects that you want to aggregate. partial is used to update value.

For example, three workers w0, w1, and w2 generate partial aggregation results p0, p1, and p2. If p1, p0, and p2 are sent in sequence to the worker where the Aggregator owner resides, the merge operations are performed in the following sequence:

i. merge(p1, p0) is first executed to aggregate p1 and p0 as p1.

ii. merge(p1, p2) is executed to aggregate p1 and p2 as p1. p1 is the global aggregation result in this superstep.

Therefore, if only one worker exists, the merge method is not required. In this case, merge() is not called.

- terminate(context, value)

  After the worker where the Aggregator owner resides 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. If true is returned after convergence is completed, jobs are immediately ended. This applies to machine learning scenarios.

K-means clustering example

This section uses k-means clustering as an example to demonstrate how to use Aggregator.

> **Note** For 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 k-means 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 that is closest to a specific center. For example, sums(i,j) indicates the sum of dimension j of the sample that is 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 centers file, 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 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 following methods can be used to implement global aggregation on the worker where the Aggregator owner resides:

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 add 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.centroids
);
    // print old centroids and new centroids for debugging
    System.out.println("\nsuperstep: " + context.getSuperstep() +
                       "\nold centriod:\n" + value.centroids + " new centriod:\n" + newCentrio
ds);
    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, teminate() calls calculateNewCentroids() based on sums and counts to calculate the average value and obtain a new center. Then, isConverged() is called to check 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 is set to false, data loaded by each worker is not partitioned based on the partitioner. Data is loaded and maintained by the same worker.

Summary

Basic steps to implement Aggregator:

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 the Aggregator owner resides.
5. The worker where the Aggregator owner resides executes merge multiple times to implement global aggregation.
6. The worker where the Aggregator owner resides 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

This topic describes how to run a MaxCompute Graph job.

The MaxCompute client provides a JAR command for you to run MaxCompute Graph jobs. This command is used in the same way as the JAR command in MapReduce.

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 in this command.

Parameters

Parameter	Description
-conf <configuration file>	The JobConf configuration file.
-classpath <local_file_list>	The classpath used to run a job in local mode. This parameter specifies the local paths of the JAR package where the Main function is located. The paths include both the relative and absolute paths.
-D <prop_name>=<prop_value>	The Java property of <mainClass> for local execution. You can specify multiple properties.
-local	Specifies that the Graph jobs run in local mode. It is used for program debugging.
-resources <resource_name_list>	Declares the resources used for running the Graph job. In most cases, you must specify the names of the resources that the Graph job uses in resource_name_list. **Note** If you read other MaxCompute resources when you run the Graph job, you must add the names of those resources to <resource_name_list>. Multiple resources must be separated by commas (,). If you use cross-project resources, add PROJECT_NAME/resources/ before <resource_name_list>. 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 by using the MaxCompute client. The PageRank algorithm is used in the following 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 the cache resource. These resources correspond to those spec
ified 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 files correspond to thos
e specified by -libjars in the JAR command.
job.addCacheResourceToClassPath("mapreduce-examples.jar");
// Set the configuration item that corresponds to odps_config.ini in the client. Replace it with the
actual one 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

This topic describes how to define the input and output of a MaxCompute Graph job.

The input and output of MaxCompute Graph jobs must be tables. You cannot customize the input or output format.

Multiple tables and partitions can be used as input. Example of 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 the col2 and col0 columns of the input table. Use record.get(0) to obtain col2 in the l
oad() method of GraphLoader. Both are read in the same sequence.
```

Note

- Partition filtering is not supported.
- The addInput framework reads records from the input table and transfers the records to the user-defined GraphLoader to load graph data.

Multiple tables and partitions can be used as output, and each of them must be marked with a label. Example of 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").b
uild(), true);
// true indicates that the code overwrites partitions specified by tableinfo. The value true is simi
lar to the INSERT OVERWRITE statement. The value false is similar to the INSERT INTO statement.
```

Note A Graph job can use the write() method of WorkContext to write data to an output table during runtime. If you want to write data to multiple tables, you must mark each table with a label, such as output1 in the preceding example.

1.9.2.3. Read data from resources

1.9.2.3.1. Use GraphJob to specify resources to be read

This topic describes how to use GraphJob to read resources in MaxCompute Graph.

In addition to the JAR command, you can use the following two methods of GraphJob to specify the resources to be read by Graph:

```
void addCacheResources(String resourceNames)
void addCacheResourcesToClassPath(String resourceNames)
```

1.9.2.3.2. Use resources in Graph

This topic describes how to use WorkerContext to read resources in MaxCompute Graph.

You can use a WorkerContext object to read resources in MaxCompute Graph.

```
public byte[] readCacheFile(String resourceName) throws IOException;
public Iterable<byte[]> readCacheArchive(String resourceName) throws IOException;
public Iterable<byte[]> readCacheArchive(String resourceName, String relativePath)throws IOException
;
public Iterable<WritableRecord> readResourceTable(String resourceName);
public BufferedInputStream readCacheFileAsStream(String resourceName) throws IOException;
public Iterable<BufferedInputStream> readCacheArchiveAsStream(String resourceName) throws IOExceptio
n;
public Iterable<BufferedInputStream> readCacheArchiveAsStream(String resourceName, String relativePa
th) throws IOException;
```

Note

- You can use the setup() method of WorkerComputer to read resources and save the resources in WorkerValue. Then, you can use getWorkerValue to obtain the resources.
- If you want to read resources and process them at the same time, you can use the preceding stream APIs. This method reduces memory consumption.

1.9.3. Graph SDK

This topic describes the commonly used classes of Graph SDK.

Class	Description
GraphJob	Defines, submits, and manages a MaxCompute Graph job. The class inherits JobConf.
Vertex	Defines a vertex in a graph. A vertex has the following properties: id, value, halted, and edges. You can specify this class by using the setVertexClass method of GraphJob.
Edge	Defines an edge in a graph. An edge has the following properties: destVertexId and value. MaxCompute Graph uses the adjacency list as the data structure. The outgoing edges of a vertex are specified by the edges property of the vertex.
GraphLoader	Loads a graph. You can specify this class by using the setGraphLoaderClass method of GraphJob.
VertexResolver	Customizes the logic for handling conflicts during graph topology modification. You can specify this class by using the setLoadingVertexResolverClass and setComputingVertexResolverClass methods of GraphJob. The setLoadingVertexResolverClass method specifies the class that is used during graph loading, whereas the setComputingVertexResolverClass method specifies the class that is used during iterative computing.
Partitioner	Specifies how to distribute vertices to workers. This way, computing can be performed by multiple workers at the same time. You can specify this class by using the setPartitionerClass method of GraphJob. By default, the HashPartitioner class is used. HashPartitioner computes the hash value of a vertex ID and divides the hash value by the number of workers to obtain the remainder of the hash value. This remainder specifies the worker to which the vertex is distributed.
WorkerComputer	Customizes the operations to perform when a worker starts and stops. You can specify this class by using the setWorkerComputerClass method of GraphJob.
Aggregator	Processes and summarizes global information. You can specify one or more Aggregator classes by using the setAggregatorClass(Class...) method of GraphJob.
Combiner	Summarizes the output records. You can specify this class by using the setCombinerClass method of GraphJob.
Counters	Defines a counter that is used for counting workers. In the operating logic of a job, you can use the WorkerContext class to obtain the counter and count workers. The framework then automatically calculates the sum of the workers.
WorkerContext	Defines the context that encapsulates the features provided by the framework, such as the features that allow workers to modify the graph topology, send messages, write results, and read resources.

1.9.4. Development and debugging

1.9.4.1. Development process

This topic describes the development process of MaxCompute Graph.

Graph development plug-ins are unavailable in MaxCompute. You can use Eclipse to develop MaxCompute Graph programs. We recommend that you perform the following steps to develop a MaxCompute Graph program:

1. Compile Graph code and perform local debugging to test basic functions.
2. Perform cluster debugging to verify the result.

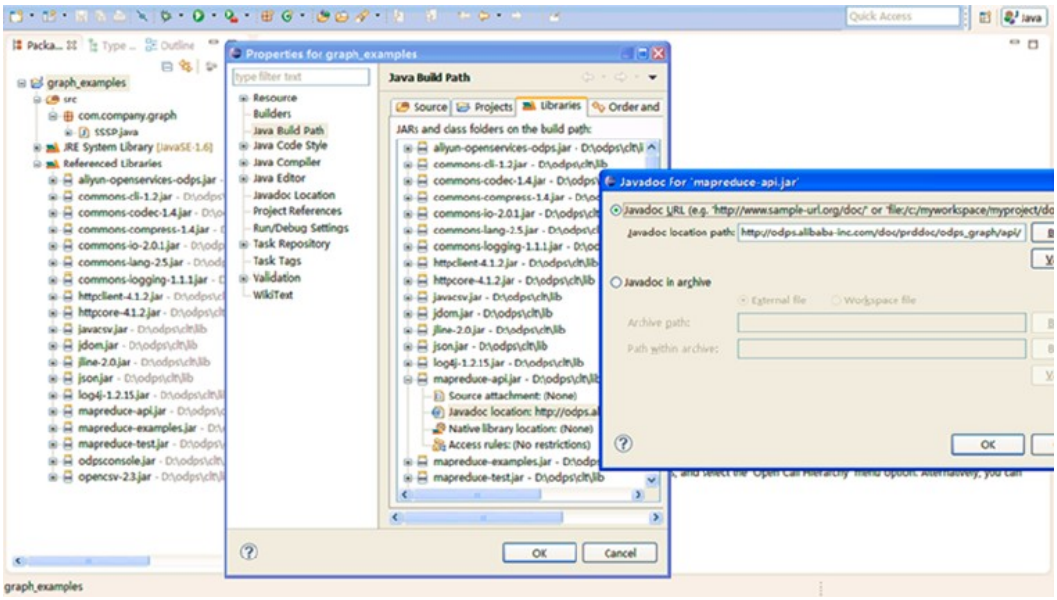
1.9.4.2. Development example

This topic uses the single source shortest path (SSSP) algorithm as an example to describe how to use Eclipse to develop and debug a MaxCompute Graph program.

Procedure

1. Create a Java project named graph_examples.
2. Add the JAR package in the lib directory on the MaxCompute client to **Java Build Path** of the Eclipse project.

   The following figure shows a configured Eclipse project.



3. Develop a MaxCompute Graph program.

   In the actual development process, an example program, such as SSSP, is first copied and then modified. In this example, only the package path is changed to *package com.aliyun.odps.graph.example*.

4. Compile and package the code. In an Eclipse environment, right-click the source code directory (the src directory in the preceding figure) and choose **Export > Java > JAR file** to generate a JAR package. Then, select the path to store the JAR package, such as *D:\odps\clt\odps-graph-example-sssp.jar*.
5. Use the MaxCompute client to run SSSP. For more information, see Compile and run a Graph job.

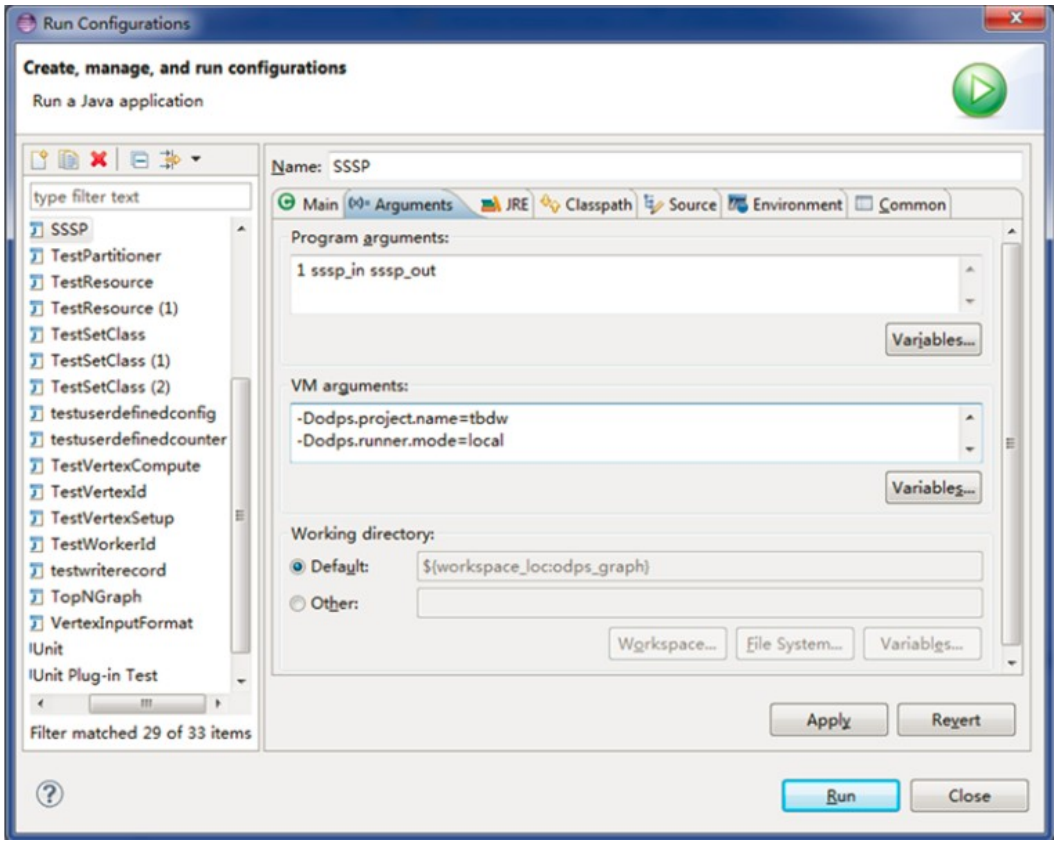
1.9.4.3. Perform debugging on the local computer

MaxCompute Graph supports the local debugging mode. This topic describes how to use Eclipse for breakpoint debugging.

Procedure

1. Download a Maven package named odps-graph-local.
2. Select the Eclipse project, right-click the main program file that contains the main function of a Graph job, and choose **Run As > Run Configurations** to configure parameters.
3. On the **Arguments** tab, set Program arguments to 1 sssp_in sssp_out as the input parameter of the main program.
4. On the **Arguments** tab, set VM arguments to the following content:

```
-Dodps.runner.mode=local
-Dodps.project.name=<project.name>
-Dodps.end.point=<end.point>
-Dodps.access.id=<access.id>
-Dodps.access.key=<access.key>
```



5. In local mode in which odps.end.point is not specified, create the sssp_in and sssp_out tables in the warehouse directory and add the following data to the sssp_in table:

```
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 directory, see Run MapReduce tasks locally.

6. Click **Run** to run SSSP on the local machine.

> **Note**
>
> Configure the parameters based on the settings in conf/odps_config.ini on the MaxCompute client. The preceding parameters are commonly used. Descriptions of other parameters:
>
> - odps.runner.mode: The value is local. This parameter is required for the local debugging feature.
> - odps.project.name: specifies the current project. This parameter is required.
> - odps.end.point: specifies the endpoint of MaxCompute. This parameter is optional. If this parameter is not specified, metadata and data of tables or resources are only read from the warehouse directory. An exception is reported if such data does not exist in the directory. If this parameter is specified, metadata and data are read from the warehouse directory first and then from the remote MaxCompute server if such data does not exist in the directory.
> - odps.access.id: specifies the AccessKey ID used to access MaxCompute. This parameter is valid only if odps.end.point is specified.
> - odps.access.key: specifies the AccessKey secret used to access MaxCompute. This parameter is valid only if odps.end.point is specified.
> - odps.cache.resources: specifies the resources you want to use. This parameter functions the same as `-resources` of the jar command.
> - odps.local.warehouse: specifies the local path to warehouse. The default value is *./warehouse*.

Output of the local SSSP debugging in Eclipse:

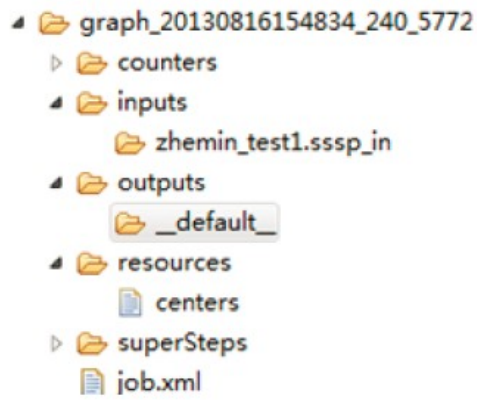
```
Counters: 3
         com.aliyun.odps.graph.local.COUNTER
                 TASK_INPUT_BYTE=211
                 TASK_INPUT_RECORD=5
                 TASK_OUTPUT_BYTE=161
                 TASK_OUTPUT_RECORD=5
 graph task finish
```

> **Notice** In this example, the sssp_in and sssp_out tables must exist in the local warehouse directory. For more information about the sssp_in and sssp_out tables, see Compile and run a Graph job.

1.9.4.4. Temporary directory of a local job

This topic describes the temporary directory created when MaxCompute Graph runs a local debugging job.

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 of a local Graph job contains the following directories and files:

- counters: stores the counter information that is generated during the job running.
- inputs: stores the input data of the job. Input data is read from the local warehouse directory first. If no data is available, MaxCompute SDK reads data from the server if odps.end.point is specified. The -Dodps.mapred.local.record.limit parameter specifies the number of records that can be read during each input operation. The default value is 10. The maximum value is 10000.
- outputs: stores the output data of the job. If an output table exists in the local warehouse, the results in outputs will overwrite data in that table after the job is executed.
- resources: stores the resources that are used by the job. Similar to input data, resource data is read from the local warehouse directory first. If no data is available, MaxCompute SDK reads data from the server if odps.end.point is specified.
- job.xml: stores job configurations.
- superstep: stores persistent messages from each iteration.

> **Note** If you want to produce 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

This topic describes how to submit a job to a cluster for testing.

After local debugging, you can submit the job to a cluster for 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 run a job. Then, check the operational log and command output.

> **Note** For more information about how to run a Graph job in a cluster, see Compile and run a Graph job.

1.9.4.6. Performance optimization

This topic describes how to configure job parameters to optimize the performance of Graph.

The following table describes the job parameters that affect the performance of Graph.

Parameter	Description
setSplitSize(long)	The split size of an input table. The unit is MB. The value must be greater than 0. The default value is 64.
setNumWorkers(int)	The number of workers for a job. The value ranges from 1 to 1000. The default value is 1. The number of workers is determined by the input bytes of the job and split size.
setWorkerCPU(int)	The number of CPU resources for a map job. 100 resources are equivalent to one CPU core. The value ranges from 50 to 800. The default value is 200.
setWorkerMemory(int)	The size of memory resources for a map job. The unit is MB. The value ranges from 256 to 12288. The default value is 4096.
setMaxIteration(int)	The maximum number of iterations. The default value is -1. If the value is less than or equal to 0, the job does not stop when the maximum number of iterations is reached.

Parameter	Description
setJobPriority(int)	The job priority. The value ranges from 0 to 9. The default value is 9. A greater value indicates a lower priority.

We recommend that you optimize the performance by using one or more of the following methods:

1. Use setNumWorkers to increase the number of workers.
2. Use setSplitSize to reduce the split size and increase the data loading speed.
3. Increase the CPU or memory resources for workers.
4. Set the maximum number of iterations. For applications that do not require high precision, you can reduce the number of iterations to accelerate the execution process.

setNumWorkers and setSplitSize can be used together to accelerate data loading. Assume that the value of setNumWorkers equals that of workerNum, the value of setSplitSize equals that of splitSize, and the total number of input bytes is the value of inputSize. The number of split data records is calculated by using the following formula: splitNum = inputSize/splitSize. Relationship between workerNum and splitNum:

- If the value of splitNum is equal to that of workerNum, each worker loads one split data record.
- If the value of splitNum is greater than that of workerNum, each worker loads one or more split data records.
- If the value of splitNum is less than that of workerNum, each worker uploads one or no split data record.

Therefore, you can adjust the settings of workerNum and splitNum to obtain a suitable data loading speed. In the first two cases, data is loaded faster. In the iteration phase, you need only to adjust the setting of workerNum. If you set runtime partitioning to False, we recommend that you use setSplitSize to adjust the number of workers or make sure that the conditions in the first two cases are met. In the third case, some workers do not load split data records. Therefore, you can insert `set odps.graph.split.size=<m>; set odps.graph.worker.num=<n>;` before the JAR command to achieve the same effect as setNumWorkers and setSplitSize.

Another common performance issue is data skew. As indicated by counters, some workers process much more split data records or edges than other workers.

Data skew occurs when the number of split data records, edges, or messages that correspond to some keys is much greater than that of other keys. These keys are processed by a small number of workers, which results in longer runtime. To address this issue, use one or more of the following methods:

- Use a combiner to aggregate the messages of the split data records that correspond to the keys to reduce the number of messages generated.

  Developers can define a combiner to reduce the memory and network traffic consumed by message storage, which shortens the job execution duration.

- Improve the business logic.

  If the data volume is large, reading data in a disk may take up the processing time. Therefore, you can reduce the data bytes to be read to increase the overall throughput. This improves job performance. To reduce data bytes, use one of the following methods:

  - Reduce data input: For some decision-making applications, processing sampled data affects only the precision of the results, not the overall accuracy. In this case, you can perform special data sampling and import the data to the input table for processing.
  - Avoid reading fields that are not used: The TableInfo class of the MaxCompute Graph framework supports reading specific columns that are transferred by using column name arrays, rather than reading the entire table or partition. This reduces the input data volume and improves job performance.

1.9.4.7. Built-in JAR packages

This topic describes the built-in JAR packages that are loaded to a Java Virtual Machine (JVM) by default.

By default, the following JAR packages are loaded to a JVM that runs Graph programs. 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

> **Note** In the classpath of the JVM, the preceding built-in JAR packages are placed before your JAR packages, which may result in a version conflict. For example, your program calls a specific class function in commons-codec-1.5.jar, but the function is not included in commons-codec-1.3.jar. In this case, you can call a similar function in commons-codec-1.3.jar or wait until MaxCompute is updated to the required version.

1.9.5. Limits

This topic describes the limits of MaxCompute Graph.

- Each job can reference a maximum 256 resources. Each table or archive is considered as one unit.
- The total size of resources referenced by a job cannot exceed 512 MB.
- The number of the inputs of a job cannot exceed 1,024. The number of input tables cannot exceed 64. The number of the outputs of a job cannot exceed 256.
- Labels that are specified for outputs cannot be null or empty strings. A label cannot exceed 256 characters in length and can contain only letters, digits, underscores (_), number signs (#), periods (.), and hyphens (-).
- The number of custom counters in a job cannot exceed 64. group name and counter name of counters cannot contain number signs (#). The total length of the two names cannot exceed 100 characters.
- The number of workers for a job is calculated by the framework. The maximum number of workers is 1,000. If the number of workers exceeds this value, an error is reported.
- By default, a worker consumes 200 CPU resources. 100 resources are equivalent to one CPU core. The number of CPU resources consumed by a worker ranges from 50 to 800.
- By default, a worker consumes 4,096 MB memory. The memory consumed by a worker ranges from 256 MB to 12,288 MB.
- A worker can repeatedly read a resource for a maximum of 64 times.
- The default value of split size is 64 MB. You can set the value based on your requirements. The value of this parameter must be greater than 0 and less than or equal to the result of the 9223372036854775807>>20 operation.
- GraphLoader, Vertex, and Aggregator in MaxCompute Graph are limited by the Java sandbox when they are run in a cluster. However, the main program of Graph jobs is not limited by the Java sandbox. For more information, see Java sandbox limits.

1.9.6. Sample programs

1.9.6.1. SSSP

This topic provides an example of Single Source Shortest Path (SSSP).

Dijkstra's algorithm is a common algorithm that is used to calculate the SSSP in a directed graph.

How the Dijkstra's algorithm works:

- Initialization: The path from s to s is 0 (d[s] = 0), and the path from u to s is infinite (d[u] = ∞).
- Iteration: If an edge from u to v exists, the shortest path from s to v is updated to d[v]=min(d[v], d[u]+weight(u, v)). The iteration does not end until the paths from all vertices to s do not change.

> **Note** Shortest path: For a weighted directed graph G = (V,E), multiple 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 working mode of the algorithm shows that the algorithm is suitable for MaxCompute Graph. Each vertex maintains the current shortest path to the source vertex. If the path changes, the new path is added with the edge weight, and a message is sent to notify adjacent vertices. In the next iteration, the adjacent vertices update the shortest paths based on the received message. If the shortest path between each vertex and the source vertex does not change, the iteration ends.

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 path from this vertex to the source vertex startVe
rtexId.
* The compute() method uses the iteration formula d[v] = min(d[v], d[u] + weight(u, v)) to update th
e vertex value.
* The cleanup() method writes the vertex and its shortest path to the source vertex to the result ta
ble.
**/
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<LongWritable> 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 IOException {
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 messageToCombine) 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 from 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 IOException
{
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, and Combiner, and specify inp
ut and 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 + " second
s");
}
}
```

1.9.6.2. PageRank

This topic provides a PageRank example.

PageRank is a classic algorithm used to rank web pages. The input of the algorithm is a digraph G. Each vertex represents a web page, and the edge between two vertices represents the link between two web pages. Basic principles of the algorithm lie in the following aspects:

- Initialization: A vertex value indicates the rank value of PageRank. Vertex values are of the DOUBLE type. During initialization, the values of all vertices are 1/TotalNumVertices.
- Iteration formula: PageRank(i) = 0.15/TotalNumVertices + 0.85 × sum. In this formula, sum indicates the sum of the PageRank(j)/out_degree(j) values. j indicates all vertices that point to vertex i.

The PageRank algorithm is suitable for MaxCompute Graph programs. Each vertex j maintains its PageRank value and sends the PageRank(j)/out_degree(j) value to its adjacent vertices (for voting) in each iteration. In the next iteration, each vertex recalculates the PageRank value by 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);
/**
* Define PageRankVertex, where:
* The vertex value indicates the current PageRank value of the vertex (web page).
* The compute() method uses the following iteration formula to update the vertex value: PageRank(i)
= 0.15/TotalNumVertices + 0.85 × sum.
* The cleanup() method writes the vertex value 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<DoubleWritable>
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
vertex. The first column of the table is the source vertices and other columns are the destination v
ertices.**/
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 number
of iterations (> 30 by default), and the 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

This topic provides a k-means clustering example.

K-means clustering is a basic clustering algorithm that is widely used. How k-means clustering works: Clustering is performed around k points in space, and the closest vertices are classified. The values of the clustering centers are updated in sequence by using iterations until the optimal clustering result is obtained.

Procedure to divide the sample set into k classes:

1. Select the initial centers of k classes.
2. In the ith iteration, select a sample, calculate its distance to k centers, and then classify the sample into the class of the center with the shortest distance.
3. Use the mean method to update the center value of the class.
4. For all the k centers, if the value remains unchanged or is less than a threshold after the update, 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 the KmeansVertex class. The compute() method is simple. It calls the aggregate() method o
f the context object and passes in the value of the current vertex. The value is of the TUPLE type a
nd 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) th
rows IOException { context.aggregate(getValue());
}
}
/** Define the KmeansVertexReader class, load a graph, and parse each record in the table as a verte
x. The transmitted value of recordNum is used as the vertex ID. The vertex value is a tuple that con
sists of all columns in the record. **/
public static class KmeansVertexReader extends
GraphLoader<Text, Tuple, NullWritable, NullWritable> {
@Override
public void load(LongWritable recordNum, WritableRecord record, MutationContext<Text, Tuple, NullWri
table, 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();
}
}
/**
* Define the KmeansAggregator class. This class encapsulates the main logic of the k-means clusterin
g algorithm.
* createInitialValue is the initial value (the center point for each of the k classes) that is creat
ed for each iteration. In the first iteration (superstep 0), the value of this parameter is the init
ial center point. In other iterations, the value is the new center point when the previous iteration
ends.
* The aggregate() method calculates the distance from each vertex to the centers of different classe
s, classifies the vertex into the class of the nearest center, and updates sum and count of the clas
s.
* The merge() method combines sums and counts collected by each worker.
* The terminate() method calculates a new center point based on sum and count of each class. If the
distance between the original and new center points is less than a threshold or the number of iterat
ions reaches the upper limit, the iteration ends, and False is returned. The final center point is w
ritten to the result 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(KmeansAggrValue value, 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)).toArra
y());
}
// 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/Aggregator, the max
imum number of iterations (> 30 by default), and the input and output tables. **/
public static void main(String[] args) throws IOException {
if (args.length < 2)
printUsage();
GraphJob job = new GraphJob();
job.setGraphLoaderClass(KmeansVertexReader.class);
job.setRuntimePartitioning(false);
// Define job.setRuntimePartitioning(false). For the k-means clustering algorithm, vertices do not n
eed to be distributed for graph loading. RuntimePartitioning is set to False to improve the performa
nce of graph loading.
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. BiPartiteMatchiing

This topic provides a BiPartiteMatchiing example.

A bipartite graph is a graph where vertices are divided into two sets and each edge connects a vertex in one set to a vertex in the other set. In a bipartite graph, a matching is a set of pairwise non-adjacent edges in which no two edges share a common vertex. Bipartite matching is often used for information matching in scenarios with clear supply and demand relationships, such as online dating websites.

Implementation process:

1. From the first vertex on the left, select another unmatched vertex by following an alternating path to find an augmenting path.
2. If the unmatched vertex is found, an augmenting path is found.
3. Update path information, increase the number of matched edges by 1, and stop searching.
4. If no augmenting path is found, begin with another 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 IOExcept
ion {
```

```
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:
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 components

This section provides an example of strongly connected components.

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.

The algorithm for strongly connected components is based on the parallel coloring algorithm. For more information, see Optimizing Graph Algorithms on Pregel-like Systems. Each vertex contains two fields:

- colorID: stores the color of Vertex v during forward traversal. At the end of computing, vertices with the same colorID belong to the same strongly connected component.
- transposeNeighbors: stores neighbor IDs of Vertex v in the transpose graph of the input graph.

The algorithm is implemented in the following steps:

1. 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.
2. 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.
3. 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.
4. 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 vertices 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 one or more vertices remain active, 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> ms
gs) 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 SCCAgg
rValue();
}
@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)ite
m;
if ((value.getStage() == STAGE_FW_REST || value.getStage() == STAGE_BW_REST)&& v.isUpdated()) { valu
e.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);
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 IOException {
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 + " second
s");
}
}
```

1.9.6.6. Connected components

This section provides an example of connected components.

Two vertices are connected if a path exists between them. If each vertex in an undirected graph G is connected to all the other vertices in the graph, G is called a connected graph. 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 generates 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<LongWritabl
e> 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 IOException
{
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 + " second
s");
}
}
```

1.9.6.7. Topological sorting

This topic provides an example of topological sorting in MaxCompute Graph.

For a directed edge (u,v), all vertex sequences that satisfy u < v are called topological sequences. Topological sorting is an algorithm that is used to calculate the topological sequence of a directed graph.

Steps to implement the algorithm:

1. Identify a vertex without incoming edges and generate an output record.
2. Delete the vertex and all its outgoing edges from the graph.
3. Repeat the preceding steps until output records are generated for all the vertices without incoming edges.

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<LongWritabl
e> 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 IOException
{
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 messageToCombi
ne) 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);
}
// Format of the input table:
// 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 value is -1, the vertex does not have any outgoing edges.
// Format of the output table:
// 0 0
// 1 1
// 1 2
// 2 3
// The first column is the supstep value, in which the topological sequence is hidden. The second co
lumn is vertexid.
// TopologySortAggregator is used to determine whether the graph has loops.
// If the input graph has a loop and the indegree of all active vertices is not 0, the iteration end
s.
// You can use records in the input and output tables to determine whether 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 + " second
s");
}
}
```

1.9.6.8. Linear regression

This topic provides a linear regression example.

In statistics, linear regression is a statistical analysis method used to determine the dependency between two or more variables. Linear regression is different from the classification algorithm that predicts discrete values. The regression algorithm can predict continuous values.

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 in the following steps:

1. Initialize the weight vector to provide the descent speed and iterations or iteration convergence condition.
2. Calculate the least square error for each sample.
3. Calculate the sum of the least square errors and update the weight based on the descent speed.
4. 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> mess
ages) 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, Tuple,
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;
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).toSt
ring());
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)-h0(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. Triangle count

This topic provides a triangle count example.

The triangle count algorithm calculates the number of triangles that pass through each vertex in a graph. The algorithm is implemented in the following steps:

1. Send the ID of each vertex to all its outgoing neighbors.
2. Store information about incoming and outgoing neighbors, and send the information to outgoing neighbors.
3. Calculate the number of endpoint intersections for each edge, calculate the sum, and write the output results to a table.
4. Sum up the output results in the table and divide the sum by 3 to obtain the number of triangles that pass through each vertex.

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.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 IOExc
eption {
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;
```

```
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 + " second
s");
}
}
```

1.9.6.10. Edge table import

This topic provides an example of importing an edge table.

Example:

```
import java.io.IOException;
import com.aliyun.odps.conf.Configuration;
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;
/**
 * This example describes how to compile a Graph job program to load data of different data types. It covers how GraphLoader
 * and VertexResolver are used together to build a graph.
 *
 * A MaxCompute Graph job uses MaxCompute tables as the input. For example, a job uses two tables as the input. One stores information about vertices, and the other stores information about edges.
 * Format of the table that stores information about vertices:
 * +------------------------+
 * | VertexID | VertexValue |
 * +------------------------+
 * |       id0|            9|
 * +------------------------+
 * |       id1|            7|
 * +------------------------+
 * |       id2|            8|
 * +------------------------+
 *
 * Format of the table that stores information about edges:
 * +-----------------------------------+
 * | VertexID | DestVertexID| EdgeValue|
 * +-----------------------------------+
 * |       id0|          id1|         1|
 * +-----------------------------------+
 * |       id0|          id2|         2|
 * +-----------------------------------+
 * |       id2|          id1|         3|
 * +-----------------------------------+
 *
 * The two tables show that id0 has two outgoing edges that point to id1 and id2. id2 has one outgoing edge that points to id1, and id1 has no outgoing edges.
 *
 * For data of this type, in GraphLoader::load(LongWritable, Record, MutationContext), MutationContext#addVertexRequest(Vertex) can be used to add vertices to the graph.
 * link MutationContext#addEdgeRequest(WritableComparable, Edge) can be used to add edges to the graph. In
 * link VertexResolver#resolve(WritableComparable, Vertex, VertexChanges, boolean),
 * vertices and edges added by using the load() method are combined to a vertex object. This object is used as the return value and added to the graph for computing.
 *
 **/
public class VertexInputFormat {
  private final static String EDGE_TABLE = "edge.table";
  /**
   * Resolve records to vertices and edges. Each record indicates a vertex or edge based on its source.
   * <p>
   * The following process is similar to the map stage of com.aliyun.odps.mapreduce.Mapper.
   * Enter a record to generate key-value pairs.
   * The keys are vertex IDs, and the values are vertices or edges that are written based on context. These key-value pairs are aggregated by using LoadingVertexResolver based on the vertex IDs.
   *
   * Note: The vertices or edges added here are requests sent based on the record content and are no
```

```
t used in computing.
   * Only the vertices or edges added by using VertexResolver are used for computing.
   **/
  public static class VertexInputLoader extends
      GraphLoader<LongWritable, LongWritable, LongWritable, LongWritable> {
    private boolean isEdgeData;
    /**
     * Configure VertexInputLoader.
     *
     * @param conf
     *          Indicates the configured parameters of a job. These parameters are configured in the
main function of GraphJob or by running the SET command on the client.
     * @param workerId
     *          Indicates the serial number of the running worker. The number starts from 0 and can
be used to build a unique vertex ID.
     * @param inputTableInfo
     *          Indicates the information about the input table that is loaded to the running worker
. The information can be used to determine the data type of the input data (the format of the record
).
     **/
    @Override
    public void setup(Configuration conf, int workerId, TableInfo inputTableInfo) {
      isEdgeData = conf.get(EDGE_TABLE).equals(inputTableInfo.getTableName());
    }
    /**
     * Resolve the record to edges based on the record content and send a request to add them to the
graph.
     *
     * @param recordNum
     *          Indicates the serial number of the record. The number starts from 1 and is separatel
y counted in each worker.
     * @param record
     *          Indicates the records in the input table. The table contains three columns, which in
dicate the source vertex, destination vertex, and edge weight.
     * @param context
     *          Indicates the context. The context is used when you send a request to add resolved e
dges to the graph.
     **/
    @Override
    public void load(
        LongWritable recordNum,
        WritableRecord record,
        MutationContext<LongWritable, LongWritable, LongWritable, LongWritable> context)
        throws IOException {
      if (isEdgeData) {
        /**
         * Data comes from the table that stores information about edges.
         *
         * 1. The first column indicates the IDs of source vertices.
         **/
        LongWritable sourceVertexID = (LongWritable) record.get(0);
        /**
         * 2. The second column indicates the IDs of destination vertices.
         **/
        LongWritable destinationVertexID = (LongWritable) record.get(1);
        /**
         * 3. The third column indicates edge weights.
         **/
        LongWritable edgeValue = (LongWritable) record.get(2);
```

```
        /**
         * 4. Create an edge based on a destination vertex ID and an edge weight.
         **/
        Edge<LongWritable, LongWritable> edge = new Edge<LongWritable, LongWritable>(
            destinationVertexID, edgeValue);
        /**
         * 5. Send a request to add the edge to a source vertex.
         **/
        context.addEdgeRequest(sourceVertexID, edge);
        /**
         * 6. If each record indicates a bidirectional edge, repeat Step 4 and Step 5.
          * Edge<LongWritable, LongWritable> edge2 = new
         * Edge<LongWritable, LongWritable>( sourceVertexID, edgeValue);
         * context.addEdgeRequest(destinationVertexID, edge2);
         **/
      } else {
        /**
         * Data comes from the table that stores information about vertices.
         *
         * 1. The first column indicates the IDs of vertices.
         **/
        LongWritable vertexID = (LongWritable) record.get(0);
        /**
         * 2. The second column indicates the values of vertices.
         **/
        LongWritable vertexValue = (LongWritable) record.get(1);
        /**
         * 3. Create a vertex based on an ID and a value.
         **/
        MyVertex vertex = new MyVertex();
        /**
         * 4. Initialize the vertex.
         **/
        vertex.setId(vertexID);
        vertex.setValue(vertexValue);
        /**
         * 5. Send a request to add the vertex.
         **/
        context.addVertexRequest(vertex);
      }
    }
  }
  /**
   * Summarize key-value pairs generated by using GraphLoader::load(LongWritable, Record, MutationCo
ntext).
   * This process is similar to the reduce stage of com.aliyun.odps.mapreduce.Reducer. For a unique
vertex ID, all actions, such as
   * adding or removing 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 con
flict occurs when multiple same vertex objects or duplicate edges are added.
   * All the IDs that are requested to be generated by using the load() method are called.
   **/
  public static class LoadingResolver extends
      VertexResolver<LongWritable, LongWritable, LongWritable, LongWritable> {
    /**
     * Process a request to add or remove 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 a request is sent to add vertex objects with the same ID, the objects are collected to the returned list.
     * VertexChanges::getAddedEdgeList() corresponds to
     * MutationContext::addEdgeRequest(WritableComparable, Edge).
     * If a request is sent to add edge objects with the same source vertex ID, the objects are collected to the returned list.
     * VertexChanges::getRemovedVertexCount() corresponds to
     * MutationContext::removeVertexRequest(WritableComparable).
     * If a request is sent to remove vertices with the same ID, the number of total removal requests is returned.
     * VertexChanges#getRemovedEdgeList() corresponds to
     * MutationContext#removeEdgeRequest(WritableComparable, WritableComparable).
     * If a request is sent to remove edge objects with the same source vertex ID, the objects are collected to the returned list.
     *
     * You can process the changes in the ID and state whether the ID is used in computing in the return value.
     * If the returned vertex is not null, the ID is used in subsequent computing. If the returned vertex is null, the ID is not used in subsequent computing.
     *
     * @param vertexId
     *          Indicates the ID of the vertex that is requested to be added or the source vertex ID of the edge that is requested to be added.
     * @param vertex
     *          Indicates an existing vertex object. The value of this parameter is always null in the data loading phase.
     * @param vertexChanges
     *          Indicates a collection of vertices or edges that are requested to be added or removed for the ID.
     * @param hasMessages
     *          Indicates whether messages are sent to the ID. The value of this parameter is always false in the data loading phase.
     **/
    @Override
    public Vertex<LongWritable, LongWritable, LongWritable, LongWritable> resolve(
        LongWritable vertexId,
        Vertex<LongWritable, LongWritable, LongWritable, LongWritable> vertex,
        VertexChanges<LongWritable, LongWritable, LongWritable, LongWritable> vertexChanges,
        boolean hasMessages) throws IOException {
      /**
       * 1. Obtain the vertex object for computing.
       **/
      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 information about 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 possible duplicate, deduplicate it based on algorithm requirements.
```

```
       **/
      if (vertexChanges.getAddedEdgeList() ! = null) {
        for (Edge<LongWritable, LongWritable> edge : vertexChanges
            .getAddedEdgeList()) {
          computeVertex.addEdge(edge.getDestVertexId(), edge.getValue());
        }
      }
      /**
       * 3. Return the vertex object and add it to the final graph for computing.
       **/
      return computeVertex;
    }
  }
  /**
   * Determine the actions of the vertex that is used in computing.
   *
   **/
  public static class MyVertex extends
      Vertex<LongWritable, LongWritable, LongWritable, LongWritable> {
    /**
     * Write the edge of the vertex to the result table based on the format of the input table. Make
sure that the formats and data of the input and output tables are the same.
     *
     * @param context
     *          Indicates the runtime context.
     * @param messages
     *          Indicates the input messages.
     **/
    @Override
    public void compute(
        ComputeContext<LongWritable, LongWritable, LongWritable, LongWritable> context,
        Iterable<LongWritable> messages) throws IOException {
      /**
       * Write the ID and value of the vertex to the result table that stores information about vert
ices.
       **/
      context.write("vertex", getId(), getValue());
      /**
       * Write the edge of the vertex to the result table that stores information about edges.
       **/
      if (hasEdges()) {
        for (Edge<LongWritable, LongWritable> edge : getEdges()) {
          context.write("edge", getId(), edge.getDestVertexId(),
              edge.getValue());
        }
      }
      /**
       * Perform only one 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>");
    }
    /**
     * Use GraphJob to configure a Graph job.
     */
    GraphJob job = new GraphJob();
    /**
     * 1. Specify input graph data and the table that stores information about 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. This process is similar to
the map stage. The generated key is the vertex ID, and the generated value is the edge.
     */
    job.setGraphLoaderClass(VertexInputLoader.class);
    /**
     * 3. Specify the data loading phase to generate the vertex that is used in computing. This proc
ess is similar to the reduce stage. In the reduce stage, edges that are generated in the map stage a
re combined into a vertex.
     */
    job.setLoadingVertexResolverClass(LoadingResolver.class);
    /**
     * 4. Specify the actions of the vertex that is used in computing. The vertex.compute() method i
s used for each iteration.
     */
    job.setVertexClass(MyVertex.class);
    /**
     * 5. Specify the result table of the Graph job and write the computing results to the result ta
ble.
     */
    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();
  }
}
```

1.9.6.11. Vertex table import

This topic provides an example of importing a vertex table.

Example:

```
import java.io.IOException;
import com.aliyun.odps.conf.Configuration;
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;
/**
 * This example describes how to compile a Graph job program to load data of different data types. It covers how GraphLoader and VertexResolver are used together to build a graph.
 * A MaxCompute Graph job uses MaxCompute tables as the input. For example, a job uses two tables as the input. One stores information about vertices, and the other stores information about edges.
 * Format of the table that stores information about vertices:
 * +------------------------+
 * | VertexID | VertexValue |
 * +------------------------+
 * |       id0|            9|
 * +------------------------+
 * |       id1|            7|
 * +------------------------+
 * |       id2|            8|
 * +------------------------+
 *
 * Format of the table that stores information about edges:
 * +-----------------------------------+
 * | VertexID | DestVertexID| EdgeValue|
 * +-----------------------------------+
 * |       id0|          id1|         1|
 * +-----------------------------------+
 * |       id0|          id2|         2|
 * +-----------------------------------+
 * |       id2|          id1|         3|
 * +-----------------------------------+
 *
 * The two tables show that id0 has two outgoing edges that point to id1 and id2. id2 has one outgoing edge that points to id1, and id1 has no outgoing edges.
 *
 * For data of this type, in GraphLoader::load(LongWritable, Record, MutationContext),
 * MutationContext#addVertexRequest(Vertex) can be used to add vertices to the graph.
 * link MutationContext#addEdgeRequest(WritableComparable, Edge) can be used to add edges to the graph. In
 * link VertexResolver#resolve(WritableComparable, Vertex, VertexChanges, boolean)
 * vertices and edges added by using the load() method are combined to a vertex object. This object is used as the return value and added to the graph for computing.
 *
 **/
public class VertexInputFormat {
  private final static String EDGE_TABLE = "edge.table";
  /**
   * Resolve records to vertices and edges. Each record indicates a vertex or edge based on its source.
   *
   * The following process is similar to the map stage of com.aliyun.odps.mapreduce.Mapper. Enter a record to generate key-value pairs.
   * The keys are vertex IDs, and the values are vertices or edges that are written based on context. These key-value pairs are aggregated by using LoadingVertexResolver based on the vertex IDs.
   *
   * Note: The vertices or edges added here are requests sent based on the record content and are not used in computing.
   * Only the vertices or edges added by using VertexResolver are used for computing.
   **/
  public static class VertexInputLoader extends
      GraphLoader<LongWritable, LongWritable, LongWritable, LongWritable> {
```

```
    private boolean isEdgeData;
    /**
     * Configure VertexInputLoader.
     *
     * @param conf
     *          Indicates the configured parameters of a job. These parameters are configured in the
Main function of GraphJob or by running the SET command in the console.
     * @param workerId
     *          Indicates the serial number of the running worker. The number starts from 0 and can
be used to build a unique vertex ID.
     * @param inputTableInfo
     *          Indicates the information about the input table that is loaded to the running worker
. The information can be used to determine the data type of the input data (the format of the record
).
     **/
    @Override
    public void setup(Configuration conf, int workerId, TableInfo inputTableInfo) {
      isEdgeData = conf.get(EDGE_TABLE).equals(inputTableInfo.getTableName());
    }
    /**
     * Resolve the record to edges based on the record content and send a request to add them to the
graph.
     *
     * @param recordNum
     *          Indicates the serial number of the record. The number starts from 1 and is separatel
y counted in each worker.
     * @param record
     *          Indicates the records in the input table. The table contains three columns, which in
dicate the source vertex, destination vertex, and edge weight.
     * @param context
     *          Indicates the context. The context is used when you send a request to add resolved e
dges to the graph.
     **/
    @Override
    public void load(
        LongWritable recordNum,
        WritableRecord record,
        MutationContext<LongWritable, LongWritable, LongWritable, LongWritable> context)
        throws IOException {
      if (isEdgeData) {
        /**
         * Data comes from the table that stores information about edges.
         *
         * 1. The first column indicates the IDs of source vertices.
         **/
        LongWritable sourceVertexID = (LongWritable) record.get(0);
        /**
         * 2. The second column indicates the IDs of destination vertices.
         **/
        LongWritable destinationVertexID = (LongWritable) record.get(1);
        /**
         * 3. The third column indicates edge weights.
         **/
        LongWritable edgeValue = (LongWritable) record.get(2);
        /**
         * 4. Create an edge based on a destination vertex ID and an edge weight.
         **/
        Edge<LongWritable, LongWritable> edge = new Edge<LongWritable, LongWritable>(
            destinationVertexID, edgeValue);
```

```
        /**
         * 5. Send a request to add the edge to a source vertex.
         **/
        context.addEdgeRequest(sourceVertexID, edge);
        /**
         * 6. If each record indicates a bidirectional edge, repeat Step 4 and Step 5.
          * Edge<LongWritable, LongWritable> edge2 = new
         * Edge<LongWritable, LongWritable>( sourceVertexID, edgeValue);
         * context.addEdgeRequest(destinationVertexID, edge2);
         **/
      } else {
        /**
         * Data comes from the table that stores information about vertices.
         *
         * 1. The first column indicates the IDs of vertices.
         **/
        LongWritable vertexID = (LongWritable) record.get(0);
        /**
         * 2. The second column indicates the values of vertices.
         **/
        LongWritable vertexValue = (LongWritable) record.get(1);
        /**
         * 3. Create a vertex based on an ID and a value.
         **/
        MyVertex vertex = new MyVertex();
        /**
         * 4. Initialize the vertex.
         **/
        vertex.setId(vertexID);
        vertex.setValue(vertexValue);
        /**
         * 5. Send a request to add the vertex.
         **/
        context.addVertexRequest(vertex);
      }
    }
  }
  /**
   * Summarize key-value pairs generated by using GraphLoader::load(LongWritable, Record, MutationCo
ntext).
   * This process is similar to the reduce stage of com.aliyun.odps.mapreduce.Reducer. For a unique
vertex ID, all actions, such as
   * adding or removing 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 con
flict occurs when multiple same vertex objects or duplicate edges are added.
   * All the IDs that are requested to be generated by using the load() method are called.
   **/
  public static class LoadingResolver extends
      VertexResolver<LongWritable, LongWritable, LongWritable, LongWritable> {
    /**
     * Process a request to add or remove 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 a request is sent to add vertex objects with the same ID, the object
s are collected to the returned list.
     * VertexChanges::getAddedEdgeList() corresponds to
```

```
     * MutationContext::addEdgeRequest(WritableComparable, Edge)
     * If a request is sent to add edge objects with the same source vertex ID, the objects are collected to the returned list.
     * VertexChanges::getRemovedVertexCount() corresponds to
     * MutationContext::removeVertexRequest(WritableComparable)
     * If a request is sent to remove vertices with the same ID, the number of total removal requests is returned.
     * VertexChanges#getRemovedEdgeList() corresponds to
     * MutationContext#removeEdgeRequest(WritableComparable, WritableComparable)
     * If a request is sent to remove edge objects with the same source vertex ID, the objects are collected to the returned list.
     *
     * You can process the changes in the ID and state whether the ID is used in computing in the return value.
     * If the returned vertex is not null, the ID is used in subsequent computing. If the returned vertex is null, the ID is not used in subsequent computing.
     *
     * @param vertexId
     *          Indicates the ID of the vertex that is requested to be added or the source vertex ID of the edge that is requested to be added.
     * @param vertex
     *          Indicates an existing vertex object. The value of this parameter is always null in the data loading phase.
     * @param vertexChanges
     *          Indicates a collection of vertices or edges that are requested to be added or removed for the ID.
     * @param hasMessages
     *          Indicates whether messages are sent to the ID. The value of this parameter is always false in the data loading phase.
     **/
    @Override
    public Vertex<LongWritable, LongWritable, LongWritable, LongWritable> resolve(
        LongWritable vertexId,
        Vertex<LongWritable, LongWritable, LongWritable, LongWritable> vertex,
        VertexChanges<LongWritable, LongWritable, LongWritable, LongWritable> vertexChanges,
        boolean hasMessages) throws IOException {
      /**
       * 1. Obtain the vertex object for computing.
       **/
      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 information about 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 possible duplicate, deduplicate it based on algorithm requirements.
       **/
      if (vertexChanges.getAddedEdgeList() ! = null) {
        for (Edge<LongWritable, LongWritable> edge : vertexChanges
            .getAddedEdgeList()) {
          computeVertex.addEdge(edge.getDestVertexId(), edge.getValue());
```

```
        }
      }
      /**
       * 3. Return the vertex object and add it to the final graph for computing.
       **/
      return computeVertex;
    }
  }
  /**
   * Determine the actions of the vertex that is used in computing.
   *
   **/
  public static class MyVertex extends
      Vertex<LongWritable, LongWritable, LongWritable, LongWritable> {
    /**
     * Write the edge of the vertex to the result table based on the format of the input table. Make
sure that the formats and data of the input and output tables are the same.
     *
     * @param context
     *          Indicates the runtime context.
     * @param messages
     *          Indicates the input messages.
     **/
    @Override
    public void compute(
        ComputeContext<LongWritable, LongWritable, LongWritable, LongWritable> context,
        Iterable<LongWritable> messages) throws IOException {
      /**
       * Write the ID and value of the vertex to the result table that stores information about vert
ices.
       **/
      context.write("vertex", getId(), getValue());
      /**
       * Write the edge of the vertex to the result table that stores information about edges.
       **/
      if (hasEdges()) {
        for (Edge<LongWritable, LongWritable> edge : getEdges()) {
          context.write("edge", getId(), edge.getDestVertexId(),
              edge.getValue());
        }
      }
      /**
       * Perform only one 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>");
    }
    /**
     * Use GraphJob to configure a Graph job.
     */
```

```
    GraphJob job = new GraphJob();
    /**
     * 1. Specify input graph data and the table that stores information about 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. This process is similar to
the map stage. The generated key is the vertex ID, and the generated value is the edge.
     */
    job.setGraphLoaderClass(VertexInputLoader.class);
    /**
     * 3. Specify the data loading phase to generate the vertex that is used in computing. This proc
ess is similar to the reduce stage. In the reduce stage, edges that are generated in the map stage a
re combined into a vertex.
     */
    job.setLoadingVertexResolverClass(LoadingResolver.class);
    /**
     * 4. Specify the actions of the vertex that is used in computing. The vertex.compute() method i
s used for each iteration.
     */
    job.setVertexClass(MyVertex.class);
    /**
     * 5. Specify the result table of the Graph job and write the computing results to the result ta
ble.
     */
    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

This topic describes MaxCompute SDK for Java.

MaxCompute SDK for Java provides a variety of APIs to support development on MaxCompute.

For more information about the APIs, see *MaxCompute Developer Guide*.

1.11. PyODPS

1.11.1. Quick start

PyODPS is MaxCompute SDK for Python. It provides easy-to-use Python programming interfaces. Similar to pandas, PyODPS provides fast, flexible, and expressive data structures. You can use the data processing feature of PyODPS, which is similar to that of pandas, by calling the DataFrame API provided by PyODPS. This topic describes how to use PyODPS in your projects.

Background information

You can develop most programs in MaxCompute by using SQL statements. However, you must use Python to develop complex business logic and user defined functions (UDFs). For example, you must use Python in the following scenarios:

- Interface connection: An external service must provide required information to complete authentication before the service can access any data record in MaxCompute tables by using HTTP interfaces.
- Asynchronous invocation: In most cases, the system creates a node for each of thousands of tasks with similar data processing logic. It is difficult to manage these nodes, and excessive resources are occupied at the same time. PyODPS allows you to use queues to asynchronously run SQL tasks at a high concurrency. Queues also help you manage all nodes in a centralized manner.
- UDF development: If the built-in functions of MaxCompute cannot meet your requirements, you can develop UDFs. For example, you can develop a UDF to format data of the DECIMAL type with thousands separators.
- SQL performance improvement: To check whether database transactions follow the first in, first out (FIFO) rule, you must compare each new record with historical records in the only transaction table. In most cases, this transaction table contains a large number of records. If you use SQL statements to complete the comparison, the time complexity is $O(N^2)$. In addition, the SQL statements may fail to return the expected result. To resolve this issue, you can call the cache method in Python code to traverse records in the transaction table only once. In this case, the time complexity is O(N) and the efficiency is significantly improved.
- Other scenarios: You can use Python to develop an amount allocation model. For example, you can develop a model to allocate USD 10 to three persons. In this model, you must define the logic for handling the cash over and short.

Procedure

This topic describes how to use a PyODPS node in DataWorks for development.

1. Log on to the DataWorks console.
2. Create a PyODPS node.
3. Edit the PyODPS node.

i. Write the code of the PyODPS node.

Write the test code in the code editor of the PyODPS node. In this example, write the following code in the code editor. It covers a full range of table operations.

```
from odps import ODPS
import sys
reload (sys)
# Set UTF-8 as the default encoding format. You must execute this statement if the data cont
ains Chinese characters.
sys.setdefaultencoding('utf8')
# Create a non-partitioned table named my_new_table, which contains the fields with the spec
ified names and of the specified data types.
table = o.create_table('my_new_table', 'num bigint, id string', if_not_exists=True)
# Write data to the my_new_table table.
records = [[111, 'aaa'],
          [222, 'bbb'],
          [333, 'ccc'],
          [444, 'Chinese']]
o.write_table(table, records)
# Read data from the my_new_table table.
for record in o.read_table(table):
    print record[0],record[1]
# Read data from the my_new_table table by executing an SQL statement.
result = o.execute_sql('select * from my_new_table;',hints={'odps.sql.allow.fullscan': 'true
'})
# Obtain the execution result of the SQL statement.
with result.open_reader() as reader:
    for record in reader:
        print record[0],record[1]
# Delete the table.
table.drop()
```

ii. Run the code.

After you write the code, click the icon. After the code is run, you can view the running result of the PyODPS node on the **Run Log** tab. The result in the following figure indicates that the running is successful.

1.11.2. Installation guide

This topic describes how to install PyODPS.

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. If you want to speed up the download, we recommend that you use Alibaba Cloud images.

Prerequisites

The following requirements are met before you install PyODPS:

- 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 accelerate Tunnel-based data upload:

- Greenlet. Recommended version: 0.4.10 or later.
- Cython. Recommended version: 0.19.0 or later.

The following installation commands are for your reference:

```
pip install greenlet>=0.4.10  # Optional. It accelerates Tunnel-based data upload.
pip install cython>=0.19.0  # Optional. We recommend that you do not install Cython if you use a Win
dows 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 accelerate Tunnel-based data upload. For more information about the versions of Visual C++ and Cython, see WindowsCompilers.

Installation procedure

1. Run the following command to install PyODPS:

   ```
   pip install pyodps
   ```

2. Run the following command to check whether the installation is successful:

   ```
   python -c "from odps import ODPS"
   ```

3. If the Python version is not the default version, 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
   ```

1.11.3. Platform instructions

1.11.3.1. 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 can schedule and run nodes. You do not need to manually create a MaxCompute entry object.

To migrate PyODPS nodes from a data development platform to a manually deployed PyODPS environment, read the instructions in the following topics:

1.11.3.2. Use local PyODPS

If you want 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. This topic describes how to deploy a local PyODPS environment.

1. Install PyODPS in a local environment. For more information, see Installation instructions.
2. Create a MaxCompute entry object in the local environment.

   You can execute the following statement on the data development platform to generate a template of the statement for creating a MaxCompute entry object. Then, you can modify the template to obtain the required statement.

   ```
   print("\nfrom odps import ODPS\no = ODPS(%r, '<access-key>', %r, '<endpoint>')\n" % (o.account.a
   ccess_id, o.project))
   ```

3. Place the obtained statement at the beginning of all code.

1.11.3.3. Use PyODPS in DataWorks

This topic describes how to use PyODPS in DataWorks and the limits.

Create a PyODPS node

You can create a PyODPS node for a workflow.

> **Note** For more information, see Create a PyODPS node.

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

You can execute SQL statements in the PyODPS node. For more information, see SQL.

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. You can use 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 add `tunnel=True` to open_reader to enable InstanceTunnel for the current open_reader operation. You can add `limit=False` to open_reader to 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 invalid.

DataFrame

- Perform operations on DataFrames.

  To perform operations on DataFrames 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 pr
  ocess each data 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` to True. By default, this parameter is set to True in DataWorks. The system displays details such as the Logview URL during the running process.

Obtain scheduling parameters

Different from SQL nodes in DataWorks, a PyODPS node does not replace strings such as ${param_name} in the code. Instead, it adds a dictionary named args as a global variable before it runs the code. You can obtain the scheduling parameters from the dictionary. This way, the Python code is not affected. For example, on the **Scheduling configuration** tab of a PyODPS node in DataWorks, you can specify ds=${yyyymmdd} in the **Parameters** field in the Basic properties section. Then, you can specify the following commands in the code of the node 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

- The Python version of a PyODPS node 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. Do not write unnecessary Python data processing code in PyODPS nodes.
- Writing and debugging code in DataWorks is inefficient. We recommend that you install an IDE locally to write code.
- To avoid excess pressure on the gateway of DataWorks, DataWorks limits the CPU utilization and memory usage. If the system displays Got killed, the memory usage exceeds the limit, and the system terminates the related processes. Therefore, we recommend that you do not perform local data operations. However, the limits on the memory usage and CPU utilization do not apply to SQL or DataFrame nodes, except to_pandas, that are initiated by PyODPS.
- 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.
  - DataFrame user defined functions (UDFs) 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 that are 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 must use the try-finally structure to implement related features.

1.11.4. Basic operations

1.11.4.1. Overview

PyODPS supports basic operations on MaxCompute objects. You can use Python-compliant programming methods to perform operations on MaxCompute.

PyODPS allows you to perform basic operations on the following MaxCompute objects: projects, tables, SQL, task instances, resources, and functions.

1.11.4.2. Projects

This topic describes how to perform basic operations on projects by using PyODPS.

You can perform the following basic operations for a project:

- Obtain a project. You can call the get_project() method of a MaxCompute entry object to obtain a specific project.

  ```
  project = o.get_project('my_project')  # Obtain the specified project.
  project = o.get_project()              # Obtain the current project.
  ```

  The method requires a project name. If you do not specify a project name for the method, the method returns the current project.
- Check whether a project exists. You can call the exist_project() method to check whether a specific project exists.

1.11.4.3. Tables

This topic describes how to perform basic operations on tables by using PyODPS.

Basic operations

You can perform the following basic operations on tables:

- Obtain all tables in the current project by calling the list_tables() method of a MaxCompute entry object.

  ```
  # Obtain all tables in the current project.
  for table in o.list_tables():
  ```
- Check whether a specific table exists by calling the exist_table() method of a MaxCompute entry object.
- Obtain a specific table by calling the get_table() method of a MaxCompute entry object.

```
t = o.get_table('table_name')
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
'table_name 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('table_name', project='other_project')
```

Create a table schema

You can use one of the following methods to create a table schema:

- Create a schema based on table columns and optional 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 data types of non-partition fields.
[bigint, double]
```

- Create a schema by calling the Schema.from_lists() method. This method is more convenient, but you cannot directly set comments for 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 call the create_table() method to create a table in two ways:

- 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)  # Create the table only if no
table with the same name exists.
table = o.create_table('my_new_table', schema, lifecycle=7)  # Set the lifecycle of the table.
```

- Create a table by specifying the names and data types of the fields to be contained in the table.

```
# Create a non-partitioned table.
table = o.create_table('my_new_table', 'num bigint, num2 double', if_not_exists=True)
# Create a partitioned table with the specified table columns and partition fields.
table = o.create_table('my_new_table', ('num bigint, num2 double', 'pt string'), if_not_exists=Tru
e)
```

By default, when you create a table, you can use only 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 must 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, for example, the table schema, you can call the reload() method to synchronize the update.

```
table.reload()
```

Insert a record to a table

A record refers to a single row in a table. You can call the new_record() method to insert a record to a specific table.

```
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 table 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 the 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 use one of the following methods to obtain data from a table:

- Call the read_table() method of a MaxCompute entry object to read data from a table.

  ```
  for record in o.read_table('test_table', partition='pt=test'):
  # Process a record.
  ```

- Call the head() method to obtain less than 10,000 data records from the beginning of a table.

  ```
  t = o.get_table('table_name')
  # Process each record.
  for record in t.head(3):
  ```

- Call the open_reader() method.
  - 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 the statement multiple times until all records are read. The number of records is specified by count. You can change the code 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 the statement multiple times until all records are read. The number of records is specified by count. You can change the code to parallel-operation code.
    # Process a record.
    ```

Write data to a table

- Call the write_table() method of a MaxCompute entry object to write data to a table.

```
records = [[111, 'aaa', True],                # A list can be specified.
           [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 the write_table() method, MaxCompute generates a file on the server. This operation is time-consuming. In addition, if an excessive number of files are generated, the efficiency of subsequent queries is affected. We recommend that you write multiple records at a time or provide a generator object if you use the write_table() method.
> - If you call the write_table() method to write data to a table, new data will be appended to existing data. PyODPS does not provide options to overwrite existing data. You must manually delete the data that you want to overwrite. For a non-partitioned table, you must call the table.truncate() method to delete data. For a partitioned table, you must delete partitions first and then create partitions again.

- Call the open_writer() method to write data to a table.

```
with t.open_writer(partition='pt=test') as writer:
records = [[111, 'aaa', True],                # A list can be specified.
           [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 partiti
oning mode.
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)
```

If the specified partition does not exist, set the create_partition parameter to True 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 specified.
                [222, 'bbb', False],
                [333, 'ccc', True],
                [444, 'Chinese', False]]
   writer.write(records)  # records can be iterable objects.
```

- Use multiple processes to concurrently write data to a table.

  If multiple processes concurrently write data to a table, all processes use the same session ID but write data to different blocks. Each block corresponds to a file on the server. After all the processes finish writing data, the main process submits the data.

```
import random
from multiprocessing import Pool
from odps.tunnel import TableTunnel
def write_records(session_id, block_id):
    # Create a session with the specified session ID.
    local_session = tunnel.create_upload_session(table.name, upload_id=session_id)
    # Create a writer with the specified block ID.
    with local_session.open_record_writer(block_id) as writer:
        for i in range(5):
            # Generate data and write the data to the correct block.
            record = table.new_record([random.randint(1, 100), random.random()])
            writer.write(record)
if __name__ == '__main__':
    N_WORKERS = 3
    table = o.create_table('my_new_table', 'num bigint, num2 double', if_not_exists=True)
    tunnel = TableTunnel(o)
    upload_session = tunnel.create_upload_session(table.name)
    # All processes use the same session ID.
    session_id = upload_session.id
    pool = Pool(processes=N_WORKERS)
    futures = []
    block_ids = []
    for i in range(N_WORKERS):
        futures.append(pool.apply_async(write_records, (session_id, i)))
        block_ids.append(i)
    [f.get() for f in futures]
    # Submit the data in all the blocks.
    upload_session.commit(block_ids)
```

Delete a table

You can call the delete_table() method to delete an existing table.

```
o.delete_table('my_table_name', if_exists=True)  # Delete a table only if the table exists.
 t.drop() # Call the drop() method to delete a table if the table exists.
```

Create a DataFrame

PyODPS provides a DataFrame framework, which allows you to conveniently query and manage MaxCompute data. You can call the to_df() method to convert a table to a DataFrame.

```
table = o.get_table('my_table_name')
 df = table.to_df()
```

Manage partitions

- 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 level-2 partitions.
```

- Check whether a partition exists.

```
table.exist_partition('pt=test,sub=2015')
```

- Obtain a partition.

```
partition = table.get_partition('pt=test')
 print(partition.creation_time)
2015-11-18 22:22:27
 partition.size
0
```

- Create a partition.

```
t.create_partition('pt=test', if_not_exists=True)  # Create a partition only if no partition with
the same name exists.
```

- Delete a partition.

```
t.delete_partition('pt=test', if_exists=True)  # Delete a partition only if the partition exists.
partition.drop()  # Call the drop() method to delete a partition if the partition exists.
```

Upload and download data by using MaxCompute Tunnel

MaxCompute Tunnel is the data tunnel of MaxCompute. You can use Tunnel to upload data to or download data from MaxCompute.

> **Note** We recommend that you use the write and read interfaces of tables instead of Tunnel. In a Cython environment, PyODPS compiles C programs during installation to accelerate the Tunnel upload and download.

- Example of data upload

```
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])
```

- Example of data download

```
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 does not support the upload of external tables.

1.11.4.4. SQL

This topic describes how to perform basic operations related to SQL statements by using PyODPS.

PyODPS supports MaxCompute SQL queries and provides methods to obtain execution results.

Execute SQL statements

You can call the execute_sql() and run_sql() methods of a MaxCompute entry object to execute SQL statements for creating task instances.

Parameter description:

- statement: the SQL statement to execute.
- hints: the runtime parameters. The hints parameter is of the DICT type.

Examples:

- Execute SQL statements.

```
o.execute_sql('select * from dual')  # Execute the statement in synchronous mode. Other instances 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 instances are blocked until the SQL statement is executed.
```

- Set runtime parameters for SQL statements.

```
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 settings.
```

Obtain the execution results of SQL statements

You can call the `open_reader` method to obtain execution results of SQL statements When the query results are being read, the following situations may occur:

- The SQL statements return structured data.

```
with o.execute_sql('select * from dual').open_reader() as reader:
    for record in reader:
    # Process each record.
```

- If 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 you specify `options.tunnel.use_instance_tunnel =True` when you call the open_reader method, PyODPS calls InstanceTunnel by default. However, if you are using an earlier version of MaxCompute or an error occurs when PyODPS calls InstanceTunnel, 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 InstanceTunnel does not meet your expectation, set this option to False. When you call the open_reader method, you can specify whether to call InstanceTunnel or the 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. 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 execution result of the SQL statement 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 the open_reader method to call InstanceTunnel 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 obtained based on data slicing.
    for record in reader[:10]:
        print(record)
```

Set an alias

If the resource referenced by a UDF changes dynamically, you can set an alias for the old resource and use the alias as the name of the new resource. This way, you do not need to delete the UDF or create another 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')
# Set the alias of resource res1 as the name of resource res2 without the 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, you must specify biz_id for the SQL statement to execute. 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')
```

1.11.4.5. Task instances

This topic describes how to perform basic operations on task instances by using PyODPS.

Basic operations

Tasks are implemented as MaxCompute instances. You can call `list_instances` to retrieve all the instances in the project. You can use `exist_instance` to determine if an instance exists, and use `get_instance` to retrieve instances.

```
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 URL of an instance

For an SQL task instance, you can call the `get_logview_address` method to obtain its Logview URL.

```
# Obtain the Logview URL based on an existing instance object.
instance = o.run_sql('desc pyodps_iris')
print(instance.get_logview_address())
# Obtain the Logview URL based on an instance ID.
instance = o.get_instance('20160426055209459g9k5pvyi2')
print(instance.get_logview_address())
```

For an XFlow task instance, you must enumerate its sub-instances and obtain the Logview URL of each sub-instance.

```
instance = o.run_xflow('AppendID', 'algo_public', {'inputTableName': 'input_table', 'outputTableName
': 'output_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 an instance

An instance can be in the Running, Suspended, or Terminated state. You can obtain the status of an instance from the status property. You can call the `is_terminated` method to check whether the execution of the current instance is complete 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('20160426055209459g9k5pvyi2')
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 execution of the instance is completed. The `wait_for_success` method functions similarly. The difference is that if the execution fails, an exception is reported.

Perform operations on sub-instances

A running instance may contain one or more sub-instances.

You can call the `get_task_names` method to obtain the names of all sub-instances.

```
instance.get_task_names()
['SQLDropTableTask']
```

After you obtain a sub-instance name, you can call `get_task_result` to obtain the execution result of the sub-instance. This method returns the execution result of each sub-instance 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 specific sub-instance when the 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%]
```

1.11.4.6. Resources

This topic describes how to perform basic operations on resources by using PyODPS.

PyODPS supports file and table resources.

Resources commonly apply to UDFs and MapReduce on MaxCompute.

You can call the `list_resources` method to query all resources, the `exist_resource` method to check whether a resource exists, and the `delete_resource` method to delete a resource. You can also call the `drop` method to delete a resource.

Manage file resources

PyODPS supports basic file types, and the .py, .jar, and .archive types.

- Create a file resource

  You can specify a resource name, a file type, a file-like object or a string object to create a file resource.

  ```
  resource = o.create_resource('test_file_resource', 'file', file_obj=open('/to/path/file'))  # Use
  a file-like object to create a file resource.
  resource = o.create_resource('test_py_resource', 'py', file_obj='import this')  # Use a string to
  create a file resource.
  ```

- Read and modify a file resource

  You can call the `open` method for a file resource or call the `open_resource` method 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.

```
with resource.open('r') as fp:  # Open the specified file in read mode.
     content = fp.read()  # Read all content.
     fp.seek(0)  # Return to the beginning of the file.
     lines = fp.readlines()  # Read multiple lines.
     fp.write('Hello World')  # Error. Data cannot be written to a file in read mode.
 with o.open_resource('test_file_resource', mode='r+') as fp:  # Open the file in read/write mode.
     fp.read()
     fp.tell()  # Locate the current position.
     fp.seek(10)
     fp.truncate()  # Truncate the file to the specified length.
     fp.writelines(['Hello\n', 'World\n'])  # Write multiple lines to the file.
     fp.write('Hello World')
     fp.flush() # Manually call the method to submit the update to MaxCompute.
```

PyODPS supports the following opening modes:

- r: read mode. The file can be opened, but data cannot be written to it.
- w: write mode. Data can be written to the file, but data in the file cannot be read. If a file is opened in write mode, the file content is cleared first.
- a: append mode. Data can be added to the end of the file.
- r+: read/write mode. You can read data from and write data to the file.
- w+: This mode is similar to the r+ mode. The only difference is that the file content is cleared first.
- a+: This mode is similar to the r+ mode. The only difference is that data can be written only 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.

Manage 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 data from and write data to a table

```
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)
```

1.11.4.7. Functions

This topic describes how to perform basic operations on functions by using PyODPS.

You can create user defined functions (UDFs) and use them in MaxCompute SQL.

Create a function

You can call the create_function() method of a MaxCompute entry object to create a function. Example:

```
# Reference a resource 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 a resource 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

You can call the delete_function() method of a MaxCompute entry object to delete a function. You can also call the drop() method to delete a function. Example:

```
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. Example:

```
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.
```

1.11.5. DataFrame

This topic describes how to create and manage a DataFrame object. This topic also provides information about how to use DataFrame to process data.

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. This way, the same code can be executed on pandas.

For the complete DataFrame document, see DataFrame.

DataFrame object operations

The following example demonstrates how to create a DataFrame object.

> **Note** The data used in the example is downloaded from movielens 100K. You can download the data as needed.

Assume that the following tables exist: pyodps_ml_100k_movies that contains movie-related data, pyodps_ml_100k_users that contains user-related data, and pyodps_ml_100k_ratings that contains 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 need only to specify 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 the DataFrame object and the data types of the 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 have a quick preview on the first N data records.

```
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. If you do not want to view all the fields, you can perform the following operations as needed:
   - Specify the fields that you want to query.

```
users[['user_id', 'age']].head(5)
   user_id  age
0        1   24
1        2   53
2        3   23
3        4   24
4        5   33
```

   - 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
```

   - Exclude some fields and add new fields based on 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. You can 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, you can 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. The number of records returned by this method is limited by `options.df.odps.sort.limit`.

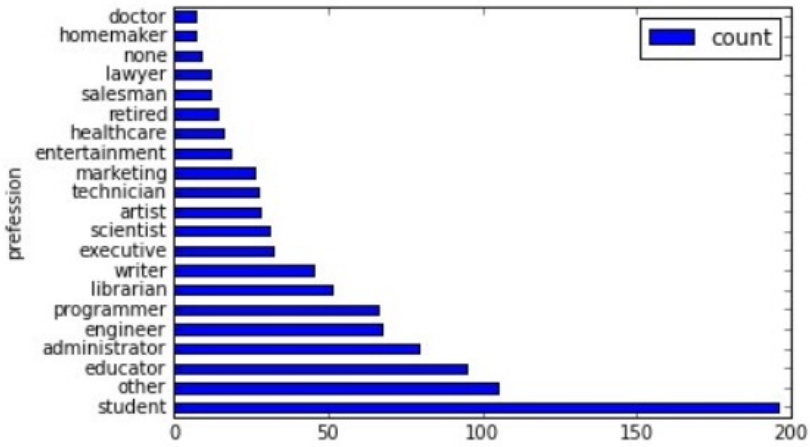
```
users.occupation.value_counts()[:10]
      occupation  count
0        student    196
1          other    105
2       educator     95
3  administrator     79
4       engineer     67
5     programmer     66
6      librarian     51
7         writer     45
8      executive     32
9      scientist     31
```

8. Show data in a more intuitive graph.

```
%matplotlib inline
```

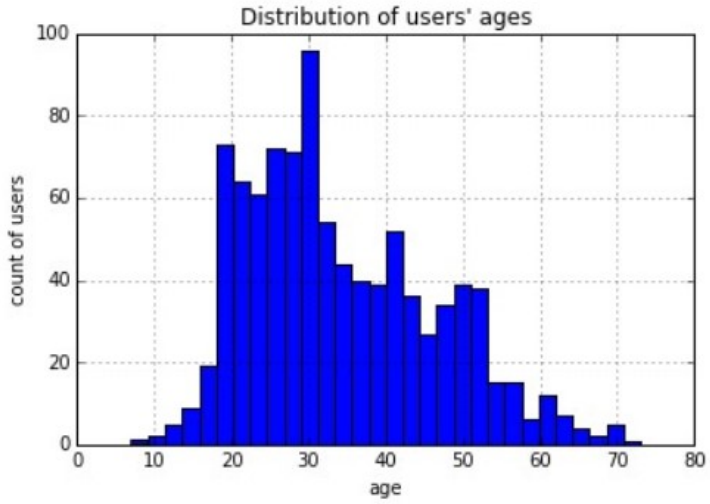
9. Use a horizontal column chart to visualize data.

```
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_gro
up')]
```

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.ratin
g.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.

> **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 **Match by location** and click Import Data.

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 MaxComp
ute 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 exten
ds horizontally.
print iris.apply(handle,axis=1).head(5)
```

1.11.6. User experience enhancement

1.11.6.1. Command line

This topic describes command line enhancement.

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 configure an account, you do not need to enter the account information again.

3. Call the enter method to create a room object on a 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 entry object is not automatically updated when you change the setup of the room. You must call `enter()` again to retrieve the new room object.

After you configure an account and call 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()
```

Result:

```
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. Example:

```
room.drop('stored-table')
room.display()
default name         desc
iris          Iris dataset
```

- You can call the `teardown` method to delete a room. If no parameters are specified, the default room is deleted.

```
teardown()
```

1.11.6.2. IPython

This topic describes IPython enhancement.

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
```

  The following result is returned:

```
<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')
```

  The following result is returned:

```
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 run the following command to display the stored objects as a table:

  ```
  %stores
  ```

  The following result is returned:

  ```
  default name      desc
  iris             Iris dataset
  ```

Object name completion

PyODPS enhances the code completion feature that is provided by IPython. When you write a statement 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.

- You can 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 also completes the names of objects in different projects. Syntax:

  ```
  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 an 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 an 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 commands to upload pandas DataFrame objects to MaxCompute tables.

You need only to use the `%persist` command. The first parameter df is the variable name. The second parameter pyodps_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
```

1.11.6.3. Jupyter Notebook

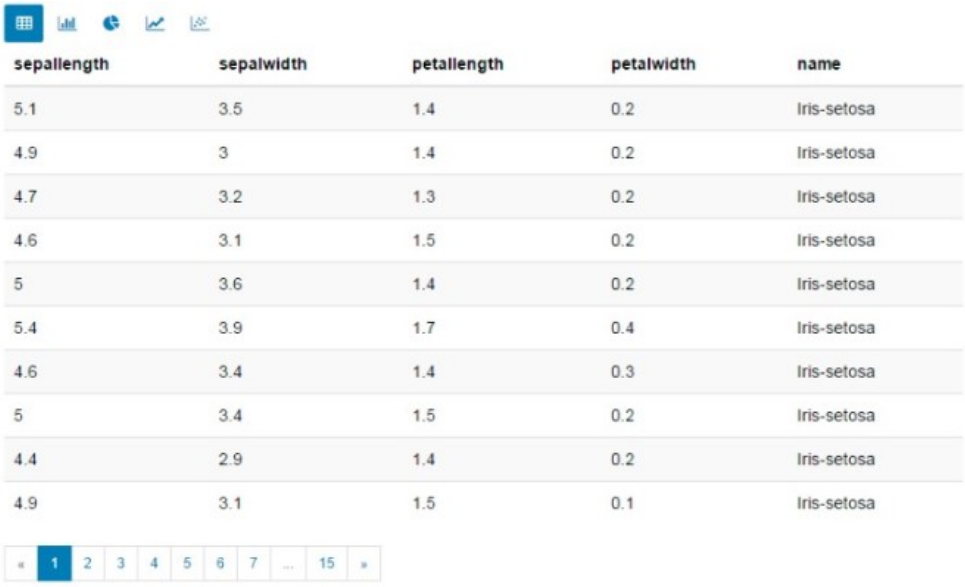
This topic describes Jupyter Notebook enhancement.

PyODPS enhances the result exploration and progress display features of Jupyter Notebook.

Result exploration

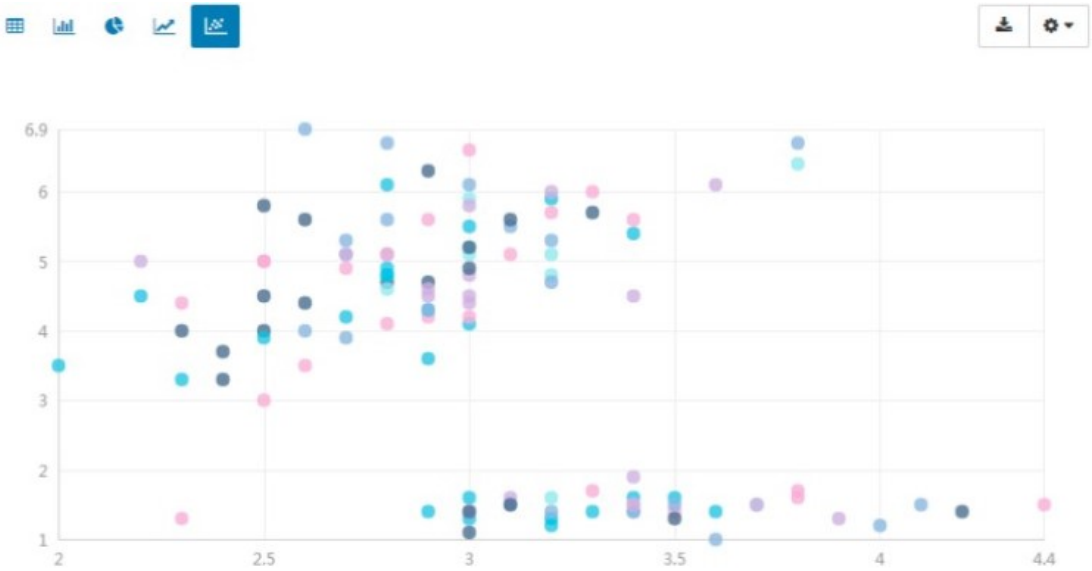
PyODPS provides a data exploration feature in Jupyter Notebook for SQLCell and DataFrame. You can use interactive data exploration tools to browse local data and create graphs.

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, the Previous button, or the Next button to browse the data.

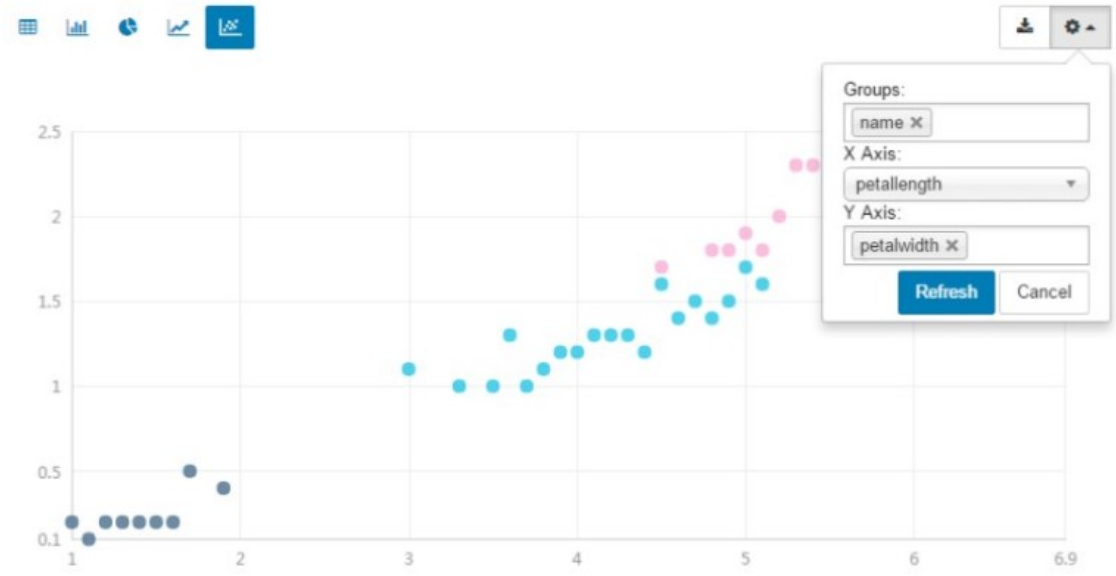


sepallength	sepalwidth	petallength	petalwidth	name
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa

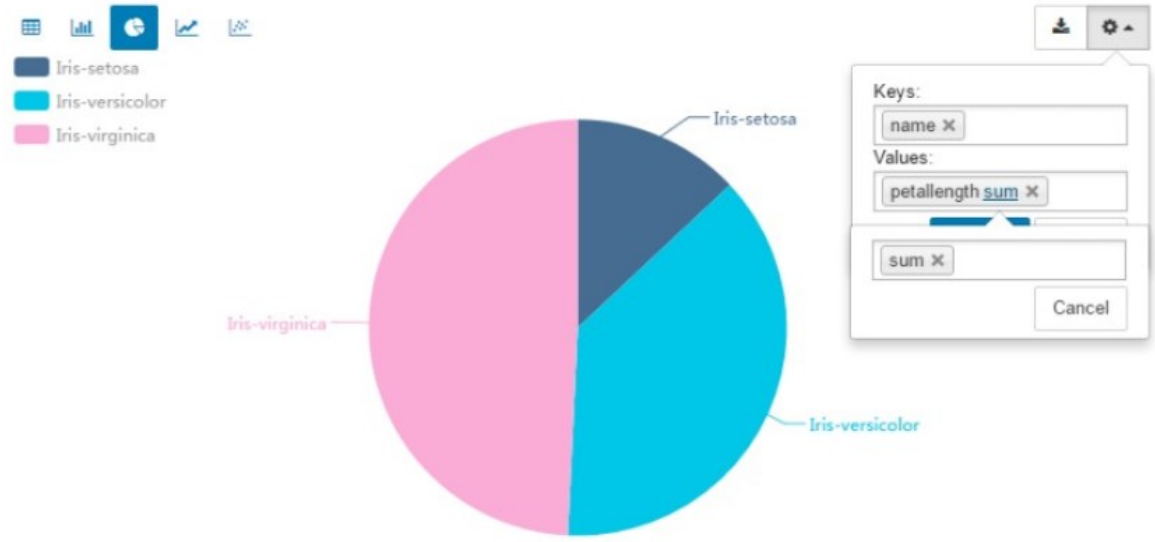
2. You can select other display modes on the top of the table to display the result in a column chart, pie chart, line chart, or scatter chart. The following figure shows a scatter chart created based on the default fields, which are the first three fields.



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 petalwidt. The result is shown in the following graph. The petallength-petalwidth settings display the data in a manner that is easy to understand.



For column 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 name of the value field to select another function. For line charts, the values on the x-axis cannot be null. If a value is null, the graph may not be correctly displayed.



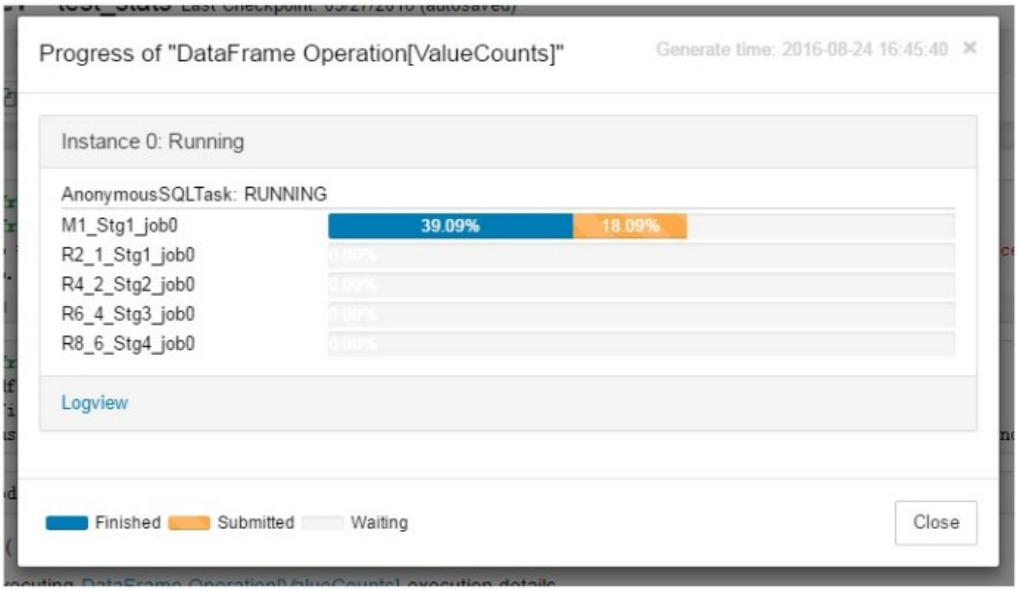
4. After you make the graph, click the **Download** icon to save it.

> **Note** To use this feature, you must install pandas and ipywidgets.

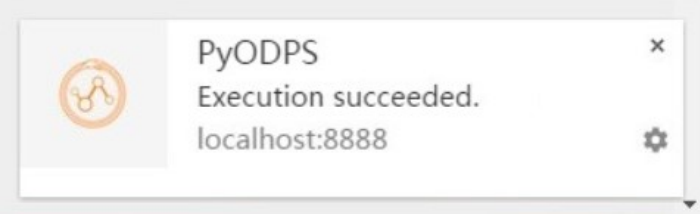
Progress display

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.



1.11.7. Configurations

This topic describes the configuration options provided by PyODPS.

PyODPS provides a series of configuration options. You can obtain them by using `odps.options`. Example:

```
from odps import options
# Set the lifecycle option to specify 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 the STRING type.
options.tunnel.string_as_binary = True
# When you execute PyODPS DataFrames in 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 configurations

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 that is 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.	5
read_timeout	The read timeout period.	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

Option	Description	Default value
notebook_repr_widget	Specifies whether to use interactive graphs.	True
sql.settings	Global hints for MaxCompute SQL.	None
sql.use_odps2_extension	Specifies whether to enable MaxCompute 2.0 language extension.	False

Data upload and download configurations

Option	Description	Default value
tunnel.endpoint	The endpoint of MaxCompute Tunnel.	None
tunnel.use_instance_tunnel	Specifies whether to use InstanceTunnel to obtain execution results.	True
tunnel.limit_instance_tunnel	Specifies whether to limit the number of data records obtained by using InstanceTunnel.	None
tunnel.string_as_binary	Specifies whether to use bytes instead of Unicode for data of the STRING type.	False

DataFrame configurations

Option	Description	Default value
interactive	Specifies whether DataFrames are used in an interactive environment.	Depending on the detection value
df.analyze	Specifies whether to enable functions that are not built in MaxCompute.	True
df.optimize	Specifies whether to enable full DataFrame optimization.	True
df.optimizes.pp	Specifies whether to enable DataFrame predicate pushdown 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 a pair of grave accents (``) to mark field and table names in the backend of MaxCompute SQL.	True
df.libraries	The resource name of the third-party library that is used for DataFrame operations.	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 that are added during a sort operation of DataFrames.	10000

Machine learning configurations

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 Predictive Model Markup Language (PMML) files of models.	False
ml.model_volume	The name of the volume used by ModelTransfer.	pyodps_volume

1.11.8. API overview

This topic provides the links to PyODPS API documentation, which provides parameter descriptions and examples of each function:

- Definitions
- DataFrame Reference

1.11.9. FAQ

This topic provides the best practices and some frequently asked questions about PyODPS, which help you efficiently develop PyODPS programs.

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 invalid 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 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. You can also store DataFrame as a table and use `read_table` or Tunnel to read data.

1.12. Java sandbox limits

This topic describes the limits imposed by Java sandboxes on MaxCompute MapReduce and user-defined function (UDF) programs in distributed environments.

> **Note** The main programs of MapReduce jobs are not subject to these limits.

Limits:

- Direct access to local files is not allowed. You can access files only by using interfaces provided by MaxCompute MapReduce or MaxCompute Graph.
- Direct access to distributed file systems is not allowed. You can use only MaxCompute MapReduce or MaxCompute Graph to access tables.
- Java Native Interface (JNI) calls are not allowed.
- Java threads cannot be created, and Linux commands cannot be executed by sub-threads.
- Network access operations such as acquiring local IP addresses are not allowed.
- Java reflection limit: The suppressAccessChecks permission is prohibited. You cannot set a private attribute or method accessible to read private attributes or call private methods.

  An access denied error is returned when you perform one of the preceding operations.

1.13. Volume lifecycle management

1.13.1. Overview

This topic provides an overview of the lifecycle of a MaxCompute volume.

In the previous versions of MaxCompute, a partition in a volume does not have a lifecycle and can exist indefinitely. Users must manage the lifecycles of volumes. In some cases, user management of volume lifecycles 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 in the current version solves this problem.

For more information about the volume lifecycle management feature, see the subsequent topic *Volume lifecycle operations*.

1.13.2. Manage the lifecycle of a volume

This topic describes how to manage the lifecycle of a MaxCompute volume.

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

This topic describes the definition and core features of Spark on MaxCompute.

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 ensures the same user experience as native Spark 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 Spark and MaxCompute jobs and ensure high performance at low costs. Spark can work with MaxCompute to create more efficient data processing solutions. Spark community applications can run seamlessly on 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

This topic describes the project resources that you must obtain before you can use Spark on MaxCompute.

Before you use Spark on MaxCompute, you must obtain the following project resources:

- Spark on MaxCompute release package: You must download the latest Spark on MaxCompute release package.
- Spark on MaxCompute plug-in: This is an open source plug-in. You can download it from Aliyun Cupid SDK.

After you prepare the preceding project resources, you must 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

This topic describes the structure of the obtained folders and files after you decompress the Spark on MaxCompute release package.

Decompress the downloaded Spark on MaxCompute release package. The following code shows the structure of the obtained folders and files:

```
.
|-- 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

This topic describes how to set the environment variables required by Spark on MaxCompute.

Set environment variables based on your business requirements. 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

This topic describes how to configure the Spark-defaults.conf file, which is required when you use Spark on MaxCompute.

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 information is the default configuration in the file. You need only to specify the blank parameters based on your account information.

```
# 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 the basic operations for you to get started with Spark on MaxCompute.

1. Obtain and decompress the Spark on MaxCompute release 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. Configure the spark-defaults.conf file.

   ```
   cp $SPARK_HOME/conf/spark-defaults.conf.template $SPARK_HOME/conf/spark-defaults.conf
   ```

   Specify the blank parameters in $SPARK_HOME/conf/spark-defaults.conf.

```
# 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
```

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-shaded.jar
```

If the following output is displayed, 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: FINISHED)
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=20180209075148695gbkp8e01&t=spark&id=application_1518162700322_1635034099&metaname=20180209075148695gbkp8e01&token=YkhjNXJWZ0dvdzVScXVFQWpCQWMra1RSZHVFPSxPRFBTX09CTzoxMzY1OTM3MTUwNzcyMjEzLDE1MTg0MjE5MzUseyJTdGF0ZW1lbnQiOlt7IkFjdGlvbiI6WyJvZHBzOlJlYWQiXSwiRWZmZWN0IjoiQWxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnByb2plY3RzL29kcHNfc21va2VfdGVzdC9pbnN0YW5jZXMvMjAxODAyMDkwNzUxNDg2OTVnYmtwOGUwMSJdfV0sIlZlcnNpb24iOiIxIn0=
     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-79a8-49f7-931d-0533663b5d85
```

1.14.5. Demo

This topic provides a demo on how to use Spark on MaxCompute.

The demo consists of the following steps:

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:

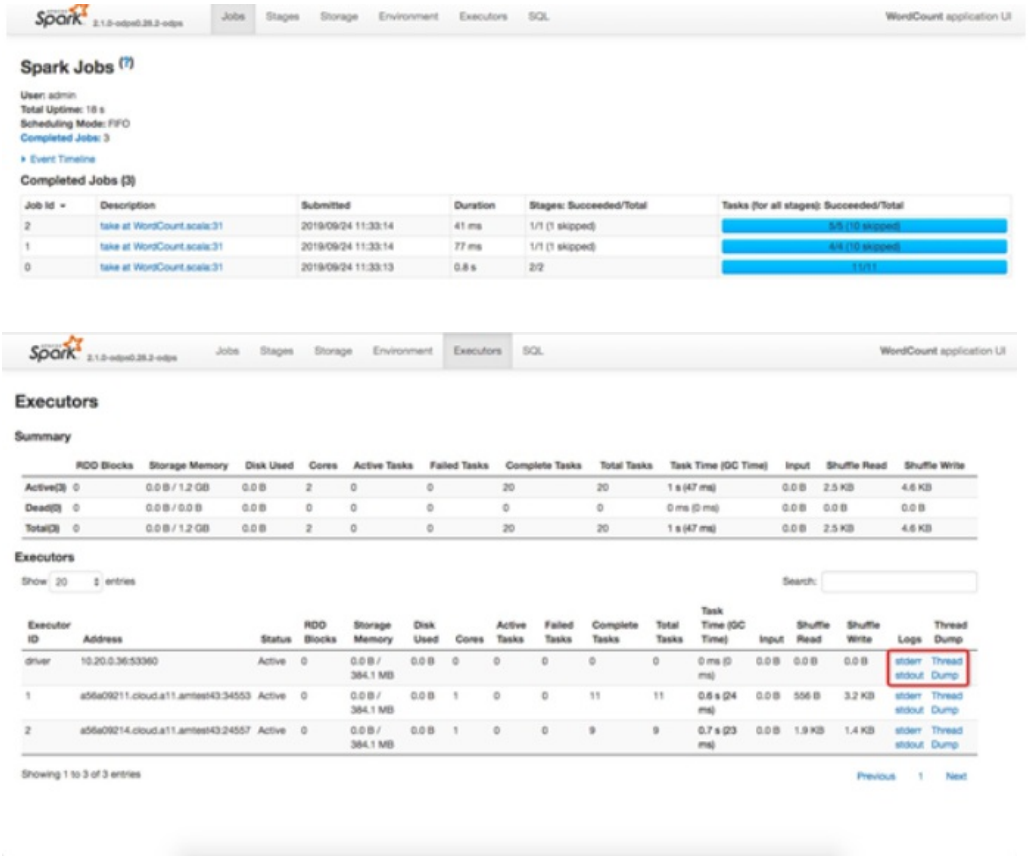
   ```
   bin/spark-submit
   --class com.aliyun.odps.spark.examples.WordCount --master yarn-cluster
   examples/spark-examples-2.0.0-SNAPSHOT-shaded.jar
   ```

3. Access the link in the command output.

   Logview

SparkUI



Spark History

1.14.6. Common cases

1.14.6.1. WordCount example

This topic provides an example of using Spark on MaxCompute to run WordCount.

You must download the GitHub project and compile the project before you can run related 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 generated. These JAR packages will be used to run the demos in this and the subsequent topics.

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(p
rintln)
    } 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

This topic provides an example of using Spark on MaxCompute to access OSS.

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 and write example

This topic provides an example of using Spark on MaxCompute to read data from and write data to a MaxCompute table and convert data to a Spark resilient distributed dataset (RDD).

Notice The project and table specified in the demo must exist or be changed to the specific project and 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, true)
      // dynamic save rdd into partition table with multiple partition spec
      val transfer_2 = (v: Tuple2[String, String], record: Record, part: PartitionSpec, schema: TableSchema) => {
        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, true)
    } 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-shaded.jar
```

You can use one of the following methods to adjust the MaxCompute table read concurrency:

- Change the value of the spark.hadoop.odps.input.split.size parameter. A larger value indicates fewer 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

This topic provides an example of using SQLContext in Spark on MaxCompute to read data from and write data to a MaxCompute Table.

> **Notice**
>
> - The project and table specified in the demo must exist or be changed to the specific project and 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-shaded.jar
```

1.14.6.5. Example of the self-developed client mode

This topic provides an example of the self-developed client mode in Spark on MaxCompute.

For security purposes, machines in MaxCompute cannot be directly connected. Therefore, the yarn-client mode 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.j
ar:$SPARK_HOME/jars/* \
com.aliyun.odps.spark.examples.SparkClientNormalFT /path/to/aliyun-cupid-sdk/examples/client-jobexam
ples/target/client-jobexamples_2.11-1.0.0-SNAPSHOT.jar
```

1.14.6.6. MaxCompute Table PySpark example

This topic provides an example of using PySpark in Spark on MaxCompute to read data from and write data to a MaxCompute table.

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-SNAPSHOT.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

This topic provides an example of using MLlib in Spark on MaxCompute.

We recommend that you use OSS for read and 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:" + feature.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.j
ar
```

1.14.6.8. PySpark interactive execution example

This topic provides an example of PySpark interactive execution in Spark on MaxCompute.

PySpark can run only on a host that can be directly connected to a computing cluster.

Install Python 2.7 and set a path:

```
export PATH=/path/to/python/bin/:$PATH
```

Run the following start command:

```
bin/pyspark --master yarn-client
```

Perform interactive execution:

```
df=spark.sql("select * from spark_user_data")
df.show()
```

1.14.6.9. Spark-shell interactive execution example (read tables)

This topic provides an example of Spark-shell interactive execution (read tables) in Spark on MaxCompute.

Spark-shell can run only on a host that can be directly connected to a computing cluster.

Run the following start command:

```
bin/spark-shell --master yarn
```

Perform 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 (Spark MLlib and OSS read/write)

This topic provides an example of Spark-shell interactive execution (Spark MLlib and OSS read/write) in Spark on MaxCompute.

Spark-shell can run only 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=***
```

Run the following start command:

```
bin/spark-shell --master yarn --jars cupid/hadoop-aliyun-package-3.0.0-alpha2-odps-jar-with-dependencies.jar
```

Perform 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. Example of SparkR interactive execution

This topic provides an example of the SparkR interactive execution in Spark on MaxCompute.

SparkR can be run only on a machine that can directly connect to a computing cluster. In addition, you must install R in the */home/admin/R* directory and specify the path:

```
export PATH=/home/admin/R/bin/:$PATH
```

The following command is used to start SparkR.

```
bin/sparkR --master yarn --archives ./R/R.zip
```

The following commands are run in interactive mode.

```
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. Example of PageRank with Apache Spark GraphX

This topic provides an example of PageRank with Apache Spark GraphX in Spark on MaxCompute.

Spark on MaxCompute supports Apache Spark 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 a 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. Example of Spark Streaming-NetworkWordCount

This topic provides an example of Spark Streaming-NetworkWordCount in Spark on MaxCompute.

Native Spark Streaming is supported. To use NetworkWordCount, you must install netcat on your on-premises machine, and then run the following command:

```
$ nc -lk 9999
```

The input in the command terminal 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 a 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-shaded.j
ar localhost 9999
```

1.14.7. Maven dependencies

This topic describes Maven dependencies of Spark on MaxCompute.

The GitHub project described in the Common cases topic can be used as a template for your quick start. For custom development, use the following pom.xml file.

> **Note** Make sure that the Spark community edition is 2.3.0 and 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 plug-in has been released to the Maven repository. You must add the following dependencies to the pom.xml file:

```
<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>
```

Dependencies in the Maven repository:

- Core code of the Cupid platform: encapsulates the APIs that are used to submit Cupid tasks and the APIs that parent and child processes use to read data from and write data to tables.

  ```
  <dependency>
          <groupId>com.aliyun.odps</groupId>
          <artifactId>cupid-core_2.11</artifactId>
          <version>1.0.0</version>
  </dependency>
  ```

- Datasource: encapsulates Spark-related APIs that are used to read data from and write data to MaxCompute tables.

  ```
  <dependency>
          <groupId>com.aliyun.odps</groupId>
          <artifactId>cupid-datasource_2.11</artifactId>
          <version>1.0.0</version>
  ```

- Client: encapsulates an SDK in 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 mode

This topic describes the running mode of Spark on MaxCompute.

Spark on MaxCompute supports three running modes: local, cluster, and DataWorks.

Local mode

The local mode facilitates code debugging for applications. In local mode, you can use Spark on MaxCompute the same way as that described in the Apache Spark community. You can also use Tunnel to read and write data from and to MaxCompute tables. In this mode, you can use either an integrated development environment (IDE) or the command line to run Spark on MaxCompute.

If Spark on MaxCompute works in this mode, you must add the configuration `spark.master=local[N]`. N indicates the CPU cores required to implement this mode.

In local mode, table read and write operations are implemented by using Tunnel. Therefore, you must add a Tunnel configuration item to the spark-defaults.conf file.

> **Note** Enter the endpoint based on the region and network environment in which your MaxCompute project resides.

The following code provides an example on how to use the command line to run Spark on MaxCompute in local 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 must specify the Main method as the entry point of a custom application. A Spark job ends when Main succeeds or fails. This mode is suitable for offline jobs. You can use Spark on MaxCompute in this mode with DataWorks to schedule jobs.

The following code provides an example on how to use the command line to run Spark on MaxCompute in cluster 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 that is in cluster mode in DataWorks to integrate and schedule other types of nodes.

Procedure:

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 for the job. The resource file is the JAR file that is uploaded and published in the business flow.

   You can specify configuration items, such as the number of executors and the memory size, for the job that you want to submit.

   You must also specify the endpoint configuration item `spark.hadoop.odps.cupid.webproxy.endpoint` of Spark on MaxCompute.

> **Note** Enter the endpoint of the region in which your MaxCompute project resides, for example, http://service.cn.maxcompute.aliyun-inc.com/api.

4. You can manually run the Spark node to view the task log and obtain the URLs of Logview and Jobview from the log for further analysis and diagnosis.
5. After the Spark job is defined, orchestrate and schedule services of different types in the business flow as required.

1.14.8.2. Spark Streaming tasks

This topic describes the configurations that are required to run Spark Streaming tasks.

MaxCompute supports Spark Streaming. To support Spark Streaming tasks that run for a long period of time, you must add the following special configurations to the spark-defaults.conf file:

```
spark.hadoop.odps.cupid.engine.running.type=longtime
# Set the running type of a task to longtime so that the task is not reclaimed.
spark.hadoop.odps.cupid.job.capability.duration.hours=25920
# Specify the running duration.
spark.yarn.maxAppAttempts=10
# Specify the maximum number of retries for a failover.
spark.streaming.receiver.writeAheadLog.enable=true
# Determine whether to enable the write-ahead logging mode. This feature prevents data loss but redu
ces data processing efficiency.
```

1.14.8.3. Job diagnosis

This topic describes job diagnosis of Spark on MaxCompute.

After you submit a job, you must check the job log to determine whether the job is submitted and executed as expected. MaxCompute provides Logview and Spark web UI for you to diagnose jobs.

The following example demonstrates how to submit a job in spark-submit mode. Logs are also generated when you use DataWorks to run Spark jobs.

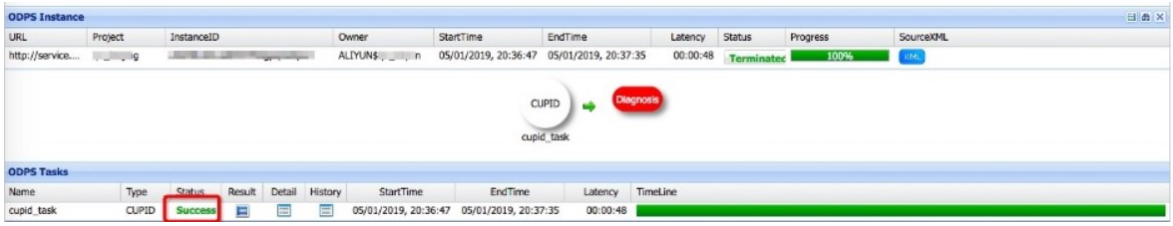
```
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class  SparkPi /tmp/spark-2.x-demo/target/AliSpark-2.x-quic
kstart-1.0-SNAPSHOT-shaded.jar
```

After the job is submitted, MaxCompute creates an instance and displays the Logview URL of the instance in the log.

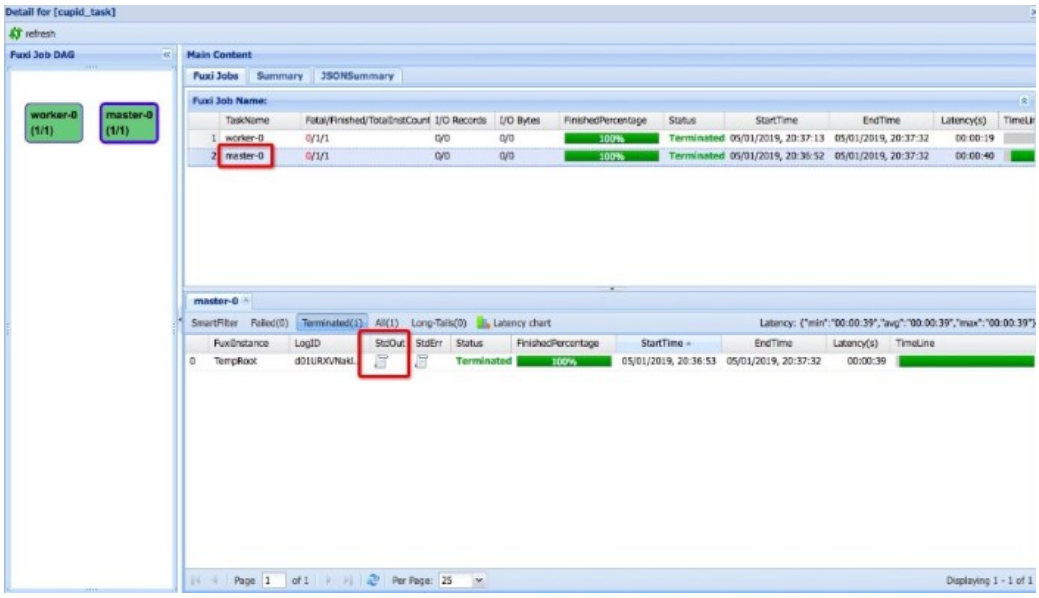
```
19/01/05 20:36:47 INFO YarnClientImplUtil: logview url: http://logview.odps.aliyun.com/logview/?h=ht
tp://service.cn.maxcompute.aliyun.com/api&p=qn_beijing&i=xxx&token=xxx
-- <The operation succeeds if an output similar to the following result is displayed.>
19/01/05 20:37:34 INFO Client:
   client 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.al
iyun-inc.com/api&p=project_name&i=xxx&t=spark&id=application_xxx&metaname=xxx&token=xxx
```

Use Logview to diagnose a job

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.

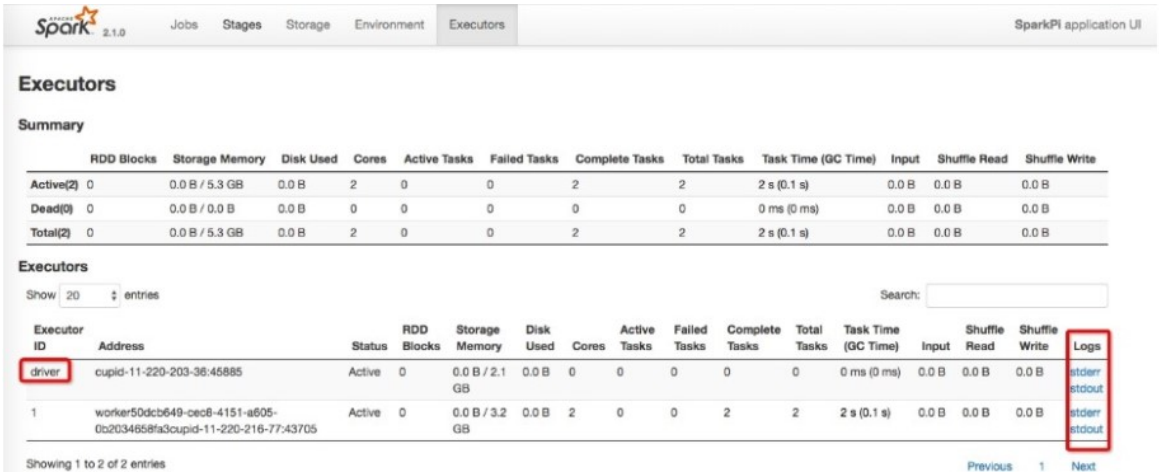


3. Click the icon in the **StdOut** column that corresponds to TempRoot to view the output of SparkPi.

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.

1. Access the URL in a browser to track the running status of your Spark job.



2. Click **StdOut** in the Logs column that corresponds to the driver to view the output of the Spark job.

1.14.9. APIs supported by Spark

1.14.9.1. Spark Shell

This topic describes the Spark shell API of Spark on MaxCompute.

Run the following commands to start the application of the Spark shell API:

```
$cd $SPARK_HOME
-- Go to the Spark shell directory.
$bin/spark-shell --master yarn
-- Specify the 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

This topic describes the SparkR API of Spark on MaxCompute.

Run the following commands to start the application of the SparkR API:

```
$mkdir -p /home/admin/R && unzip ./R/R.zip -d /home/admin/R/
# Create the R directory and decompress the R.zip package in this directory.
$export PATH=/home/admin/R/bin/:$PATH
-- Configure environment variables.
$bin/sparkR --master yarn --archives ./R/R.zip
-- Specify the 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

This topic describes the Spark SQL API of Spark on MaxCompute.

Run the following commands to start the application of the Spark SQL API:

```
$cd $SPARK_HOME
-- Go to the Spark SQL directory.
$bin/spark-sql --master yarn
-- Specify the running mode and start the application.
```

Example:

```
show tables;
select * from spark_user_data limit 3;;
quit;
```

1.14.9.4. Spark JDBC

This topic describes the Spark Java Database Connectivity (JDBC) API of Spark on MaxCompute.

Run the following commands to start the application of the Spark JDBC API:

```
$sbin/stop-thriftserver.sh
-- Stop a thread.
$sbin/start-thriftserver.sh
-- Restart a thread.
$bin/beeline
-- Start the application.
```

Example:

```
!connect jdbc:hive2://localhost:10000/odps_smoke_test
show tables;
select * from mr_input limit 3;
!quit
```

1.14.10. Dynamic resource allocation of Spark

This topic describes the dynamic resource allocation (DRA) feature of Spark on MaxCompute.

Background information

Spark provides a large number of parameters to implement a wide range of semantics. You can use the default values for most parameters, but some parameters require manual configurations. The spark.executor.instances parameter is the most complex.

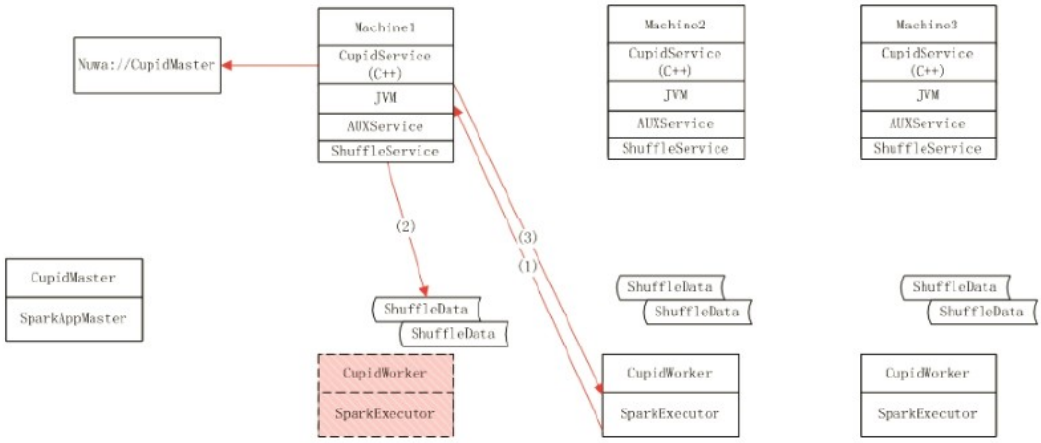
If this parameter is set to a small value, operations may run slowly or fail due to out of memory (OOM) or no free disk space. If this parameter is set to a large value, resources are wasted.

Even the optimal value obtained 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 cause waste. If long tail latency occurs, idle resources are occupied by other executors even for simple jobs.

Solution

The best solution is to allocate resources as required. DRA is a solution that addresses this issue. CupidService developed by the Cupid team supports native DRA. The following figure shows the implementation process of DRA.

> **Note** For more information about the native DRA in the Apache Spark community, see Spark help and Spark configuration.



Enable DRA

You must add the following configurations to enable DRA. You do not need to modify the code.

```
spark.hadoop.odps.cupid.shuffleservice.enable=true  // Required. This parameter specifies whether to enable the shuffle service in CupidService.
spark.hadoop.odps.cupid.disk.driver.enable=true  // Required. This parameter is a dependent item.
spark.dynamicAllocation.enabled=true // Required. This parameter specifies whether to enable DRA in Spark.
spark.shuffle.service.enabled=true // Required. This parameter is a dependent item.
spark.shuffle.service.port=7338 // Required. This parameter indicates the port used for the shuffle service.
spark.authenticate=true // Required. This parameter specifies whether to enable authentication.
spark.dynamicAllocation.maxExecutors=128 // Optional. This parameter indicates the maximum number of executors.
spark.dynamicAllocation.minExecutors=1 // Optional. This parameter indicates the minimum number of executors.
spark.dynamicAllocation.initialExecutors=1 // Optional. This parameter indicates the initial number of executors.
spark.dynamicAllocation.executorIdleTimeout=60s // Optional. This parameter indicates the waiting period before idle executors are released.
```

If DRA is enabled, spark.executor.instances is optional and equivalent to `spark.dynamicAllocation.initialExecutors`.

1.14.11. FAQ of Spark on MaxCompute

This topic provides answers to some frequently asked questions about Spark on MaxCompute.

How do I migrate code of the open source Spark to Spark on MaxCompute?

The migration method depends on whether your job needs to access MaxCompute tables or Object Storage Service (OSS):

- If the job does not need to access MaxCompute tables or OSS, run your JAR package directly. Note that you must set the dependency of Spark or Hadoop to provided.
- If the job needs to access MaxCompute tables, you need only to configure related dependencies and repackage them.
- If the job needs to access OSS, you need only to configure related dependencies and repackage them.

How do I use Spark on MaxCompute to access services in a VPC?

Spark on MaxCompute does not allow access to services in a VPC. If you want to access a service in a VPC, submit a ticket to contact the MaxCompute technical support team.

The following error message appears, which indicates that the ID and key in the spark-defaults.conf file are invalid. What do I do?

Error:

```
Stack:
com.aliyun.odps.OdpsException: ODPS-0410042:
Invalid signature value - User signature dose not match
```

Check whether the ID and key in the spark-defaults.conf file are consistent with the AccessKey ID and AccessKey secret that you obtain from the Apsara Uni-manager Management Console.

The following error message appears, which indicates that I do not have permissions. What do I do?

Error:

```
Stack:
com.aliyun.odps.OdpsException: ODPS-0420095:
Access Denied - Authorization Failed [4019], You have NO privilege 'odps:CreateResource' on {acs:odp
s:*:projects/*}
```

The project owner must grant the READ and CREATE permissions on resources to you.

The following error message appears, which indicates that the project does not support Spark jobs. What do I do?

Error:

```
Exception in thread "main" org.apache.hadoop.yarn.exceptions.YarnException: com.aliyun.odps.OdpsExce
ption: 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 resides. In addition, check whether the configurations in the spark-defaults.conf file are consistent with those described in the MaxCompute documentation. If the Spark on MaxCompute service is provided in the region and the configurations in the spark-defaults.conf file are correct, submit a ticket.

When a task is running, the following error message appears, which indicates that local storage space is insufficient. What do I do?

Error:

```
No space left on device
```

Spark on MaxCompute 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 storage space is determined by the `spark.hadoop.odps.cupid.disk.driver.device_size` parameter. The default storage space is 20 GB and the maximum storage space is 100 GB. If the error persists after you increase the storage space to 100 GB, conduct further analysis.

The most common cause is data skew. Data is unevenly distributed among blocks during the shuffle and cache processes. If this is the 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

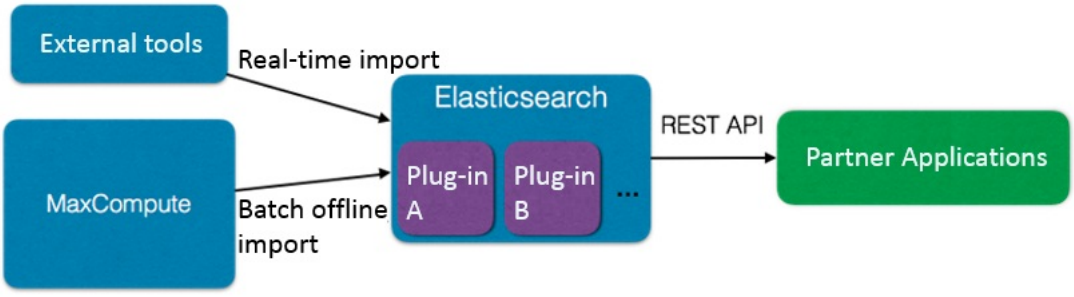
This topic provides an overview of Elasticsearch on MaxCompute and its core features.

Elasticsearch on MaxCompute is an enterprise-class full-text index system developed by Alibaba Cloud to retrieve large amounts of data. It provides near-real-time (NRT) search for government agencies and enterprises. It has the following benefits:

- Provides scalable full-text index services and supports native Elasticsearch API operations.
- Supports data import from multiple heterogeneous data sources based on the API development.
- Supports cluster- and business-level O&M management.

As a combination of Elasticsearch and MaxCompute, Elasticsearch on MaxCompute provides efficient, core massive data search engine services by leveraging unified scheduling and management of MaxCompute. In addition, Elasticsearch on MaxCompute can be used with the plug-ins of open source Elasticsearch to provide a wide range of index features.

Elasticsearch on MaxCompute allows you to use tools to import data from external sources in real time. You can also import offline data from MaxCompute. After the imported data is indexed, Elasticsearch on MaxCompute provides index services by using RESTful APIs. The following figure shows the usage of Elasticsearch on MaxCompute.



1.15.2. Workflow

1.15.2.1. Overview

This topic describes the workflow of Elasticsearch on MaxCompute.

Elasticsearch on MaxCompute is developed 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 required, including the number of nodes, disk space, memory size, and custom settings. The resources consumed by Elasticsearch are counted towards your MaxCompute quota.

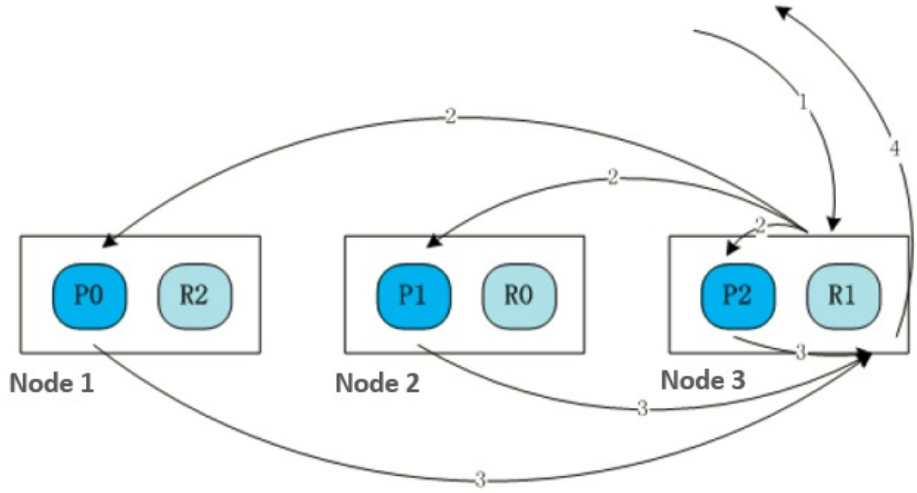
For more information about the process for starting Elasticsearch, see Elasticsearch typical practice.

The following topics describe the workflows of the features of Elasticsearch on MaxCompute.

1.15.2.2. Distributed search workflow

This topic describes the distributed search workflow of Elasticsearch on MaxCompute.

The following figure shows the distributed search workflow.



In the preceding figure, the cluster consists of three nodes. The index has three shards: P0, P1, and P2. These shards are distributed across the three nodes. Each shard is replicated in 1:1 mode. Three replicas are generated: R0, R1, and R2.

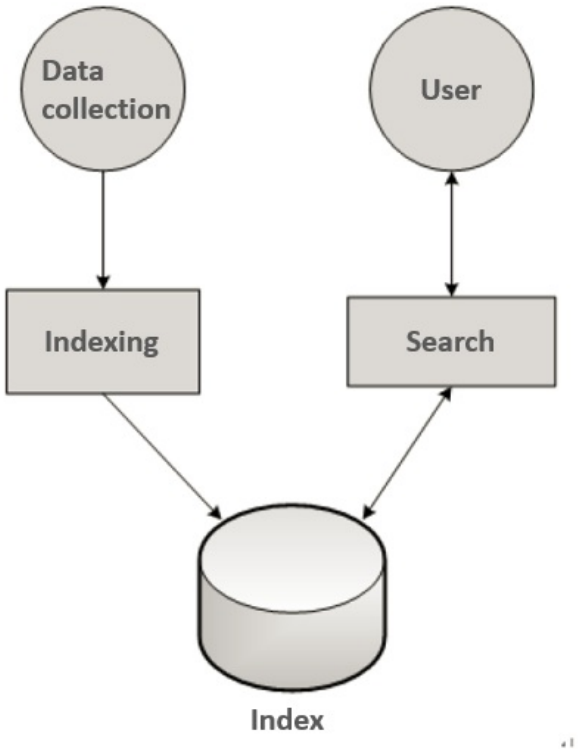
1. A user sends a search request to Node 3.
2. After Node 3 receives the request, it sends a search request (2) to P0, P1, and P2 based on the recorded index shard information.
3. The nodes in which P0, P1, and P2 are located search for the requested information in the specified shards. A search result message (3) is sent to Node 3.
4. Node 3 collects the search results from other nodes and returns the search results to the user in an acknowledgment message (4).

> **Note**
> - The search speed is increased because multiple nodes perform the search at the same time.
> - The performance of distributed search improves with the increase of nodes.

1.15.2.3. Full-text index workflow

This topic describes the full-text index workflow of Elasticsearch on MaxCompute.

The following figure shows the full-text index workflow.



1. The data collection module collects both structured and unstructured data, converts the data into the field+value format, and submits the data to the indexing module.
2. After the indexing module receives the field+value data, it tokenizes and creates an inverted index based on the predefined indexing method of the field and saves the data to the index. The type, indexing method, and tokenization method of each field are configured on the retrieval management platform.
3. A user sends a search request. After the search processing module receives and processes the request, the search index, field, and query statement are obtained. The specified record list is included in the inverted index data.
4. The indexing module returns data that meets the requirements such as sorting rules and the number of requests.

1.15.2.4. Authentication process

This topic describes the authentication process of Elasticsearch on MaxCompute.

Authentication process:

1. Elasticsearch on MaxCompute allows you to perform retrieval management and O&M on MaxCompute. To perform these operations, you must log on to the retrieval management or O&M platform that is provided by Elasticsearch on MaxCompute. During logon, you are redirected to the authentication module for authentication. If the authentication fails, you are not allowed to access the related platform.
2. The administrator can use the MaxCompute client to add Elasticsearch users and grant permissions for the users.
3. The system authenticates all users who attempt to access index libraries. After you pass the authentication, you are allowed to retrieve data or perform operations on data in the libraries.

1.15.3. Quick start

This topic describes how to use Elasticsearch on MaxCompute. It guides you through the basic use of Elasticsearch on MaxCompute.

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, each node is assigned 8 GB of memory and 20 GB of disk space.

  > **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 demonstrates how to quickly start a small Elasticsearch cluster based on the default configuration. In the following example, the name of the Elasticsearch cluster you want to start is es_first_cluster.

1. Download the MaxCompute client that supports Elasticsearch. Configure the AccessKey pair, project, and endpoint.
2. Run odpscmd and run the following command to start the Elasticsearch cluster:

   ```
   server create es_first_cluster type elasticsearch_mdu;
   ```

   Wait for several minutes. If OK is returned, the Elasticsearch cluster is started. You must create an Elasticsearch user to access the cluster.
3. In odpscmd, create an Elasticsearch user with the ALL_ACCESS permission.

   ```
   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. In the following example, the MaxCompute project that is used to access the Elasticsearch cluster is prj1. Run the following command to return information about the Elasticsearch cluster:

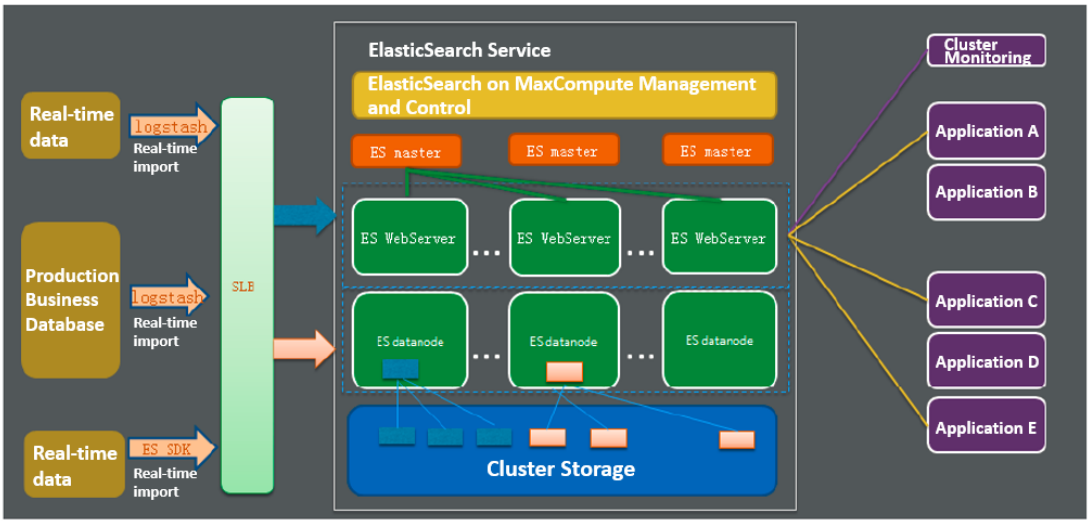
   ```
   curl -u admin:123456 http://search.aliyun.com:9200/prj1.es_first_cluster;
   ```

> **Notice** 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 restored**. We recommend that you proceed with caution.

1.15.4. Support for Elasticsearch applications

1.15.4.1. Typical practice of Elasticsearch on MaxCompute

This topic describes the typical practice of Elasticsearch on MaxCompute.



Elasticsearch on MaxCompute allows you to start an Elasticsearch cluster in a MaxCompute cluster by submitting a job. MaxCompute projects do not modify native Elasticsearch code. The Elasticsearch on MaxCompute runs in the same mode as native Elasticsearch clusters.

1.15.4.2. Limits on Elasticsearch on MaxCompute in VPCs

This topic describes the limits on Elasticsearch on MaxCompute in VPCs.

Elasticsearch on MaxCompute is an enterprise-class retrieval system that is developed by Alibaba Cloud to retrieve large amounts of data. This system also complies with data isolation and security requirements. Therefore, the limits on Elasticsearch on MaxCompute in VPCs are imposed based on the limits of MaxCompute in VPCs.

Elasticsearch on MaxCompute has the following limits in VPCs:

- The classic network, VPCs, and the Internet are isolated from each other. Users can only access the endpoints and virtual IP addresses (VIPs) of their own networks.
- Projects for which VPC IDs or IP address whitelists are not configured are accessible to the three types of networks by using domain names.
- If an Elasticsearch cluster is started in a MaxCompute project, the cluster and project share the same VPC list. The VPC list is a whitelist of VPCs.
- If you start an Elasticsearch cluster, this cluster automatically occupies all resources. If you start more Elasticsearch instances, you must scale up the MaxCompute instance or scale down the Elasticsearch cluster.

Scenario: When you deploy MaxCompute in Apsara Stack, one project is created and one Elasticsearch cluster is started for each project by default. You can start your own Elasticsearch cluster in your project, apply for a domain name and VIP after the cluster is started, and then perform VPC verification in the Elasticsearch frontend.

1.15.5. Special notes

1.15.5.1. Find the Elasticsearch service domain name

This topic describes how to find the domain name of the Elasticsearch service during O&M.

1. Find your MaxCompute cluster in the Apsara Infrastructure Management Framework.
2. Navigate to the MaxCompute cluster resource usage page.
3. Find the domain name of the Elasticsearch service.

1.15.5.2. Import table data from MaxCompute to Elasticsearch

This topic describes how to import table data from MaxCompute to Elasticsearch.

Before you use Elasticsearch on MaxCompute to search for the data in a MaxCompute table, you must import the table data from MaxCompute to an Elasticsearch cluster. To meet these requirements, Alibaba Cloud develops jobs to export data from MaxCompute to Elasticsearch based on MaxCompute MapReduce. This allows you to import data from MaxCompute to Elasticsearch by using simple configurations.

You can easily control parallelism based on the distributed scheduling capability of MaxCompute. You can also add MapReduce jobs to the scheduled tasks on DataWorks.

The following example shows the data import process:

1. Download the JAR package of a 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 the 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 worker nodes run in parallel to export data from the table specified by <TABLE_NAME> [PA
RTITION_SPEC] to the Elasticsearch cluster.
```

The following table describes the parameters in the es_mr.conf file:

Parameter	Example value	Required	Default value	Description
es.resource	my_index/my_type	Yes	N/A	The index and type of the Elasticsearch cluster into which data is imported.
es.nodes	N/A	Yes	N/A	The endpoint of Elasticsearch.
es.nodes.client.only	True	No	False	Data is sent only to client-only nodes.
es.col.field.mapping	odps_col1:es_field1,odps_col3:es_field2	Yes	N/A	The mapping between the MaxCompute columns you want to import and the Elasticsearch fields.

Parameter	Example value	Required	Default value	Description
es.batch.size.bytes	1 MB	No	1 MB	The size of data that is transmitted at a time.
es.batch.size.entries	1000	No	1000	The number of data entries that are transmitted at a time.
es.net.http.auth.user	Admin	No	N/A	The username that is used to access the Elasticsearch cluster.
es.net.http.auth.pass	123456	No.		The password that is used to access the Elasticsearch cluster.
es.mapping.routing	field_routing	No	N/A	The routing field name, which is in the <CONSTANT> format for a constant.
es.mapping.id	field_id	No	N/A	The ID field of a document.

1.16. Flink on MaxCompute

This topic provides a demo of Flink on MaxCompute.

In the current MaxCompute version, Flink on MaxCompute is only for trial use. This 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 parameters 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 of this parameter is the domain name without sparkui.

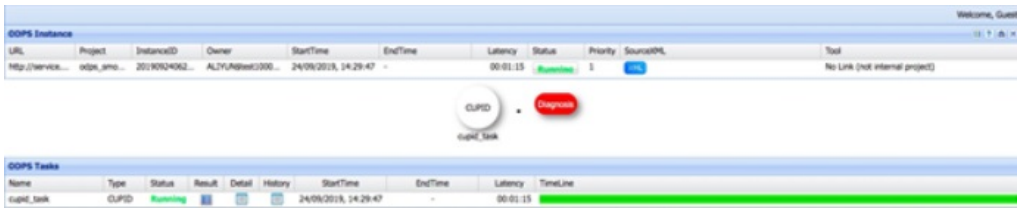
2. Start the program.

   Save the flink-java-project-0.1.jar package to the decompressed directory of Flink and run the following commands:

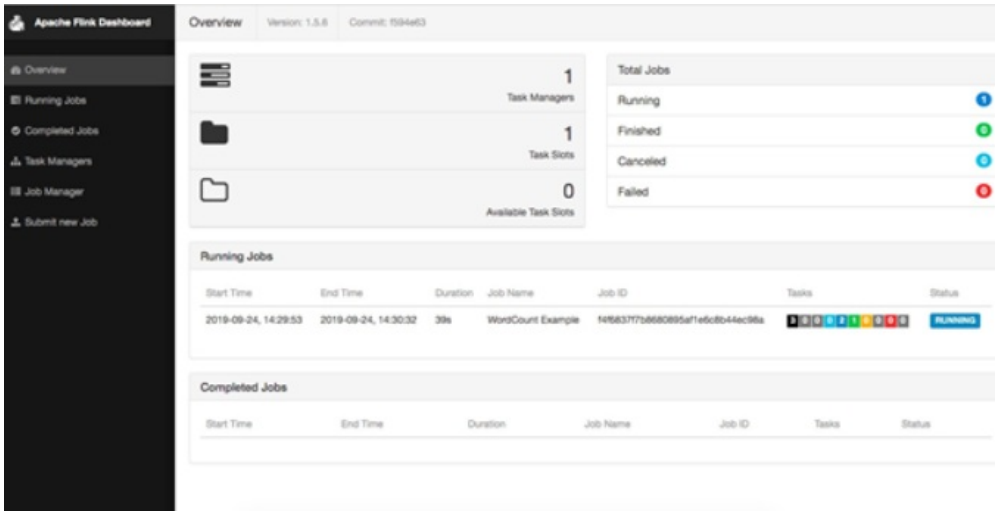
```
/bin/flink
run -c org.apache.flink.odps.WordCount -m yarn-cluster -yn 1
flink-java-project-0.1.jar --projectName odps_smoke_test --inputTable wc_in
--outputTable wc_out --sleepTime 100
```

3. Log on to Logview and Flink.

   Logview



   Flink



> **Note** MaxCompute is compatible with Flink in the following aspects:
>
> - Group windows of Flink: The TUMBLE, HOP, and SESSION built-in functions are supported.
> - User-defined functions (UDFs), user-defined aggregate functions (UDAFs), and user-defined table-valued functions (UDTFs) in Flink: MaxCompute supports UDFs that run in Flink. You can specify `odps.sql.enable.flink.udf` to enable this feature.
> - SQL syntax in Flink: The || operator can be used 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

This topic describes how to use external tables to access and process unstructured data from internal and external data sources.

MaxCompute SQL cannot directly process external data, such as non-structured data from Object Storage Service (OSS). This type of data must be imported into MaxCompute tables by using relevant tools. This 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 data definition language (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 the same way as 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 for MySQL, ApsaraDB RDS, Apsara File Storage for HDFS, and Taobao Distributed Data Layer (TDDL)
- External data sources: open source Hadoop Distributed File System (HDFS), ApsaraDB for MongoDB, and ApsaraDB for 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. Overview

This topic describes how to access and process Object Storage Service (OSS) data sources in MaxCompute.

As a core computing component of the Alibaba Cloud big data platform, MaxCompute provides powerful computing capabilities. It can schedule a large number of nodes for parallel computing, and effectively manage the failover and retry mechanisms in a distributed computing environment. MaxCompute SQL implements a variety of logic to process data based on simple semantics. It is widely used within and outside Alibaba Group. It allows interoperability among different data sources and is crucial for building a data ecosystem for Alibaba Cloud.

Some examples are provided to demonstrate how MaxCompute accesses and processes OSS data.

1.17.2.1.2. Use a built-in extractor to read data from OSS

1.17.2.1.2.1. Overview

This topic describes how to use a built-in extractor of MaxCompute to read data from Object Storage Service (OSS).

You can use the built-in extractor of MaxCompute. This provides an easy way to read data that is stored in the specified format from OSS. You need only to create an external table as a source table for data queries.

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 data file is saved in */demo/SampleData/CSV/AmbulanceData/vehicle.csv*. Some examples are provided to demonstrate how to use the built-in extractor to read data from OSS.

1.17.2.1.2.2. Create an external table

This topic describes how to create an external table when a built-in extractor is used. It also provides an example for reference.

Execute the following statements to create an external table:

```
CREATE EXTERNAL TABLE IF NOT EXISTS ambulance_data_csv_external
(
vehicleId INT,
recordId INT,
patientId INT,
calls INT,
locationLatitute DOUBLE,
locationLongtitue DOUBLE,
recordTime STRING,
direction STRING
)
STORED BY 'com.aliyun.odps.CsvStorageHandler'
WITH SERDEPROPERTIES (
 'odps.properties.rolearn'='acs:ram::xxxxx:role/aliyunodpsdefaultrole'
)
LOCATION 'oss://oss-cn-shanghai-internal.aliyuncs.com/oss-odps-test/Demo/';
```

Note

- STORED BY: specifies the name of the built-in storage handler. com.aliyun.odps.CsvStorageHandler is a built-in storage handler that processes CSV files. It defines how to read data from and write data to CSV files. You can specify this parameter as required. The logic of reading and writing CSV files is implemented by the system.
- WITH SERDEPROPERTIES: specifies the properties of the external table that you want to create. odps.properties.rolearn specifies the Alibaba Cloud Resource Name (ARN) for the AliyunODPSDefaultRole role in Resource Access Management (RAM). You must specify this parameter if you use Security Token Service (STS) to authorize MaxCompute to access OSS.
- LOCATION: specifies the OSS directory in which the data files you want to read are saved. By default, the system reads all the data files in the directory.
- An external table contains only related OSS directories. If you delete this table, the data in the directory specified by LOCATION is not deleted.

You can execute the following statement to view the structure of the created external table:

```
DESC EXTENDED <table_name>;
```

In the output, you can view basic table information, which is similar to the information returned for a created internal table. In addition to the basic table information, you can view the storage handler and the OSS directory.

1.17.2.1.2.3. Query an external table

This topic describes how to query an external table when a built-in extractor is used. It also provides an example for reference.

After an external table is created, you can use it the same way that you use a standard table.

In this example, the vehicle.csv file in the */demo/SampleData/CSV/AmbulanceData/* directory contains the following data:

```
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 be managed only by using MaxCompute SQL, not MaxCompute MapReduce.
- If you want to obtain data over HTTPS at the underlying layer, add `set odps.sql.unstructured.data.oss.use.https=true;` 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 the following built-in storage handlers: CsvStorageHandler, TsvStorageHandler, and TextStorageHandler.

1.17.2.1.2.4. MSCK REPAIR TABLE

This topic describes the MSCK REPAIR TABLE statement of MaxCompute and provides an example for reference.

The MSCK REPAIR TABLE statement of MaxCompute can be used to add partitions to external tables. The following example shows the syntax of the MSCK REPAIR TABLE statement:

```
MSCK [REPAIR] TABLE external_table_name [ADD PARTITIONS];
```

Syntax description:

- When you import data into Object Storage Service (OSS), make sure that the OSS directory is in the *oss://xxx/table-location/ptname1=ptvalue1/ptname2=ptvalue2/xxx* format.
- You must specify the partition structure when you create an external table.
- After you execute the `MSCK [REPAIR] TABLE external_table_name [ADD PARTITIONS];` statement, MaxCompute SQL automatically parses the OSS directory to identify partitions and adds the partitions to the external table.

Example:

1. Upload data to OSS. The following figure shows the OSS directory.

2. Execute the following statement to create an external table named orc_pt_v0. The structure of the pt partition is specified in the external table.

```
CREATE EXTERNAL TABLE orc_pt_v0
(
name STRING
)
PARTITIONED BY (pt bigint)
STORED BY 'com.aliyun.odps.CsvStorageHandler'
WITH SERDEPROPERTIES (
'odps.properties.rolearn'='acs:ram::xxxxx:role/aliyunodpsdefaultrole'
)
LOCATION 'oss://xxx/odps-ext-reg-perf/orc-pt-v0';
```

3. Execute the following statement to add three partitions to the external table orc_pt_v0:

```
MSCK REPAIR TABLE orc_pt_v0 ADD PARTITIONS;
-- In this case, the MSCK REPAIR TABLE 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);
```

1.17.2.1.2.5. Read data from the CSV or TSV files compressed in the GZIP format

This topic describes how to read data from the CSV or TSV files that are compressed in the GZIP format.

MaxCompute can use only a built-in extractor to read data from the CSV and TSV files that are compressed in the GZIP format. The main difference between reading non-compressed and compressed files is the property specified by SERDEPROPERTIES.

> **Note** If a data file stored in Object Storage Service (OSS) is an archived object, you must restore the data file first.

The following example shows how 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'
WITH SERDEPROPERTIES (
 'odps.properties.rolearn'='acs:ram::xxxxx:role/aliyunodpsdefaultrole'
 [,'odps.text.option.gzip.input.enabled'='true']
 [,'name3'='value3']
)
LOCATION 'oss://oss-cn-hangzhou-zmf.aliyuncs.com/oss-odps-test/Demo/SampleData/CSV/AmbulanceData/';
```

The following table describes the properties supported by SERDEPROPERTIES.

Property	Valid value	Default value	Description
odps.text.option.gzip.input.enabled	• True • False	False	Specifies whether to compress the file before data reading.
odps.text.option.gzip.output.enabled	• True • False	False	Specifies whether to compress the file before data writing.
odps.text.option.header.lines.count	Non-negative integer	0	Specifies the number of rows to skip in the file.
odps.text.option.null.indicator	String	Empty string	Specifies the strings in the file to be parsed as NULL in an SQL statement. For example, if you specify `\N` to represent NULL, `a,\N,b` is parsed as `a, NULL, b`.
odps.text.option.ignore.empty.lines	• True • False	True	Specifies whether to ignore blank rows.
odps.text.option.encoding	• UTF-8 • UTF-16 • US-ASCII • GBK	UTF-8	Specifies the encoding format of the file.
odps.text.option.delimiter	Single character	Comma (,)	Specifies the column delimiter of the file.

Property	Valid value	Default value	Description
odps.text.option.use.quote	• True • False	False	Specifies whether to recognize column delimiters in a CSV file if it uses double quotation marks (") as column delimiters. If fields in the CSV file contain specified symbols for separating multiple values, these fields must be enclosed in double quotation marks ("). The symbols include a carriage return, line feed (CRLF) pair, double quotation mark ("), or comma (,). If a field contains a pair of double quotation marks (`""`), replace the double quotation marks (`""`) with two pairs of double quotation marks ("") for escaping.

> **Note**
>
> If you want to read data from and write data to an external table that is associated with compressed OSS data, configure both odps.text.option.gzip.input.enabled and odps.text.option.gzip.output.enabled as True when you create the external table.

1.17.2.1.3. Use a custom extractor to read data from OSS

1.17.2.1.3.1. Overview

This topic describes how to use a custom extractor to read data from Object Storage Service (OSS).

If OSS data is in a complex format and cannot be processed by the built-in extractor, you must use a custom extractor to read the OSS data.

For example, a text file is stored in OSS. The columns of data records in the file are separated by vertical bars (|). The data file vehicle.csv is saved in the */demo/SampleData/CustomTxt/AmbulanceData/* directory. Some examples are provided to demonstrate how to use a custom extractor to read data from OSS.

1.17.2.1.3.2. Define a storage handler

This topic describes how to define a storage handler when a custom extractor is used. It also provides an example for reference.

You can customize the logic to parse data. StorageHandler is the unified entrance of your custom logic. You can use StorageHandler to specify the types of custom extractors and Outputer. StorageHandler provides only a simple definition. For example, you can customize a SpeicalTextStorageHandler:

```
package com.aliyun.odps.udf.example.text;
public class SpeicalTextStorageHandler extends OdpsStorageHandler {
@Override
public Class<? extends Extractor> getExtractorClass() {
return TextExtractor.class;
}
@Override
public Class<? extends Outputer> getOutputerClass() {
return TextOutputer.class;
}
}
```

Note TextStorageHandler that is built-in to MaxCompute can process the text data in which columns are separated by vertical bars (|). This example demonstrates how to customize a storage handler by using an SDK to process specially formatted data, especially in scenarios in which you use an extractor.

1.17.2.1.3.3. Define an extractor

This topic describes how to define a custom extractor when you want to use this extractor. It also provides an example for reference.

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 the next() method is cal
led. This InputStream can read all data from 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 by using the delimiter. For more information
about the implementation process, see Complete implementation of TextExtractor.
-- 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

This topic describes how to compile and package code when you use a custom extractor.

You can compile and package Java code, and run the following command to upload the package to MaxCompute. The procedure is the same as that for a common Java user-defined function (UDF).

```
add jar odps-udf-example.jar;
```

1.17.2.1.3.5. Create an external table

This topic describes how to create an external table when a custom extractor is used. It also provides an example for reference.

After you upload a JAR package, you must run the following command to create an external table. This command is similar to that you run when a built-in extractor is used. The difference is that a custom storage handler is used in this command.

```
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 storage handler.
WITH SERDEPROPERTIES('delimiter'='|')
-- SERDEPROPERITES can be used to specify parameters. These parameters are transferred to an extract
or by using DataAttributes.
LOCATION 'oss://<*your-id*>:<*your-secret-key*>@oss-cn-shanghai-internal.aliyuncs.com/oss-odps-test/
Demo/SampleData/CustomTxt/AmbulanceData/'
USING 'odps-udf-example.jar';
-- Specify the JAR package in which the class definition is located.
```

1.17.2.1.3.6. Query an external table

This topic describes how to query an external table when a custom extractor is used. It also provides an example for reference.

In this example, the vehicle.csv file in the */demo/SampleData/CustomTxt/AmbulanceData/* directory contains the following data:

```
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 custom extractor to read data from OSS:

```
SELECT recordId, patientId, direction;
FROM ambulance_data_txt_external;
WHERE patientId > 25;
```

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 |
+------------+------------+-----------+
```

1.17.2.1.4. Use a custom extractor to read external unstructured data

This topic describes how to use a custom extractor to read external unstructured data. It also provides an example for reference.

The preceding topics describe 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 a custom extractor to process non-text files that are stored in OSS.

Audio files in the WAV format are used in this example.

1. Customize the SpeechSentenceSnrExtractor main logic. Call the setup method to read parameters, initialize the parameters, and import the audio processing model. You can use the resource function to import the model.

```
public SpeechSentenceSnrExtractor(){
this.utteranceLabels = new HashMap<String, UtteranceLabel>();
}
@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);
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() method 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 results to a record in the [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-odps-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. View the processing results:

```
------------------------------------------------------------
| 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 A custom extractor allows you to execute 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

This topic describes the data partitioning feature of external tables.

LOCATION is used to specify an OSS directory, which is associated with an external table. MaxCompute reads all data in the OSS directory, including all files in the subdirectories. Due to a large amount of data accumulated in the directory, a full-text scan is performed. This operation results in extra I/O operations and prolongs the time required to process data. Two solutions are provided to address this issue:

- Reduce the amount of data: Plan data storage directories properly. Create multiple external tables for data from different parts, with LOCATION of each external table pointing to a subset of data.
- Partition data: Similar to internal tables, external tables support data partitioning. You can create partitions to manage the data.

Some examples are provided to demonstrate how to use the data partitioning feature of external tables.

1.17.2.1.5.2. Standard organization method and directory structure of partition data in OSS

This topic describes the standard organization method and directory structure of partition data in Object Storage Service (OSS). It also provides an example for reference.

Unlike the data in the internal tables of MaxCompute, the data stored in external storage, such as OSS, cannot be managed in MaxCompute. If you want to use the partitioned table feature of MaxCompute, make sure that the directories of data files 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_data/';
```

> **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. To make the partitions take effect, you must specify the OSS storage directory in the format shown in the preceding example. The following example shows a valid directory layout:

```
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 prepare offline data, use a tool to upload the offline data to OSS. In this case, you can specify the data directory format. For the partitioned table feature of the external table to work properly, we recommend that you use the directory format in the preceding example for the uploaded data.

3. You can execute the ALTER TABLE ADD PARTITION statement to import the partition information to MaxCompute. The following example shows sample statements:

```
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. If the data is ready and the partition information is imported to MaxCompute, you can execute SQL statements to manage the partitions in the external table in OSS.
   - Execute the following statement to count the number of unique IP addresses in the log generated on June 1, 2017:

```
SELECT count(distinct(ip)) FROM log_table_external;
WHERE year = '2017' AND month = '06' AND day = '01';
```

> **Note** In this example, MaxCompute accesses only logfile and logfile.1 in the *log_data/year=2016/month=06/day=01* sub-folder but not all files in the *log_data* folder for the external table log_table_external. This prevents unnecessary I/O operations.

- Similarly, you can execute the following statement to analyze data from the second half of year 2017:

```
SELECT count(distinct(ip)) FROM log_table_external;
WHERE year = '2017' AND month > '06';
```

> **Note** In this example, only the logs for the second half of year 2017 stored in OSS are accessed.

1.17.2.1.5.3. Custom directories of partition data in OSS

This topic describes the custom directories in which partition data is saved in Object Storage Service (OSS). It also provides an example for reference.

If you have historical data stored in OSS but the data is not saved in the partitionKey1=value1\partitionKey2=value2\...format, you can still access the data by using the partitioning feature of MaxCompute. MaxCompute provides a way to import partitions from custom directories.

Example:

1. The directories in which partition data is saved contain only partition values but do not contain partition keys. The following example shows the layout of the data directories:

```
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 command 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_data_cu
stomized/2017/06/01/';
```

> **Note** The LOCATION parameter specifies an OSS directory. This directory maps to the partition that is added to the external table by using ADD PARTITION. This way, MaxCompute can access data in the OSS directory even if the directory is not in the partitionKey1=value1\partitionKey2=value2\...format. In addition, you can access the partition data in the subdirectory.

1.17.2.1.5.4. Access fully-customized non-partitioned data subsets

This topic describes how to access fully-customized non-partitioned data subsets.

In some special cases, you may need to access an arbitrary subset of a file in an OSS directory, and the format of the directory where the files in this file subset are saved is not deterministic. The unstructured data processing framework of MaxCompute is able to handle this situation, but is not discussed in this topic.

For more information about advanced operations, contact the MaxCompute technical team.

1.17.2.1.6. Output OSS data

1.17.2.1.6.1. Create an external table

This topic describes how to create an external table when you write data to OSS. It also provides an example for reference.

To write data to OSS, you must execute 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 execute MaxCompute SQL statements such as INSERT INTO or OVERWRITE. The following example demonstrates how to create an external table by using the built-in storage handler TsvStorageHandler of MaxCompute.

```
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 is used to read data from or write data to the external table.
> - Data storage format: A storage handler is used to define the data access mode. In this example, the built-in storage handler com.aliyun.odps.TsvStorageHandler of MaxCompute is used to read or write data from or to TSV files. You can also use the MaxCompute SDK to define storage handlers.

1.17.2.1.6.2. Write data to a TSV text file by using an INSERT statement on an external table

This topic describes how to write data to a TSV text file by using an INSERT statement on an external table. It also provides an example for reference.

After you associate an OSS object with an external table, you can execute a standard INSERT OVERWRITE or INSERT INTO statement on the external table to write data to the OSS file. The data source can be either the data stored in MaxCompute internal tables or the external data that is imported into MaxCompute by using an external table.

> **Note**
>
> - MaxCompute internal table: You can execute an INSERT statement on an external table to write data from a MaxCompute internal table to an external storage medium.
> - External data that is imported into MaxCompute by using an external table: You can import external data to MaxCompute by using an external table, use the data for computations, and then export the results to an external storage directory or external data store. For example, you can import Tablestore data into MaxCompute and then export the data to OSS.

In this example, you have a MaxCompute internal table named tpch_lineitem and want to export some of the data in the table to OSS in the TSV format. After you create an external table, you can execute the INSERT OVERWRITE statement to export the 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;
```

In the preceding example, eight columns are selected from the rows that meet the conditions l_discount = 0.07 and l_tax = 0.01 in the internal table tcph_lineitem and then written to OSS in the TSV format. The selected columns in the internal table correspond to the schema of the external table tpch_lineitem_tsv_external. After this operation is complete, you can view the related TSV data file in OSS.

> **Notice**
>
> Data exported from MaxCompute to OSS is stored in a special file structure.
>
> - If you use MaxCompute to execute the INSERT INTO or INSERT OVERWRITE statement on an external table and write data to an OSS directory, all data is saved in a .odps folder in the directory specified by LOCATION.
> - The .meta file in the .odps folder is an extra macro data file written by MaxCompute. This file records valid data in the folder. If the INSERT operation succeeds, all data in the folder is valid. You are only required to parse the macro data if a job fails.
> - If a job fails or is terminated, execute the INSERT OVERWRITE statement again until it is complete. This prevents the .meta file from being parsed.
> - If you want to parse the .meta file, contact Alibaba Cloud technical team.

The number of files that are generated during the internal processing of TSV and CSV files of MaxCompute 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 want to force the generation of a TSV file, you can append DISTRIBUTE BY l_discount to the INSERT OVERWRITE statement. Then, a reduce stage with only one reducer is added at the end so that only one TSV file is generated.

1.17.2.1.6.3. Write data to an unstructured file by using an INSERT statement on an external table

This topic describes how to write data to an unstructured file by using an INSERT statement on an external table.

MaxCompute provides the Outputer interface for data output. You can call this interface to write user data to a custom unstructured data file by using OutputStream. Further details are not provided in this topic.

If you have related requirements, contact the MaxCompute technical team.

1.17.2.1.6.4. Migrate data between different storage media by using MaxCompute

This topic describes how to migrate data between different storage media by using MaxCompute. It also provides an example for reference.

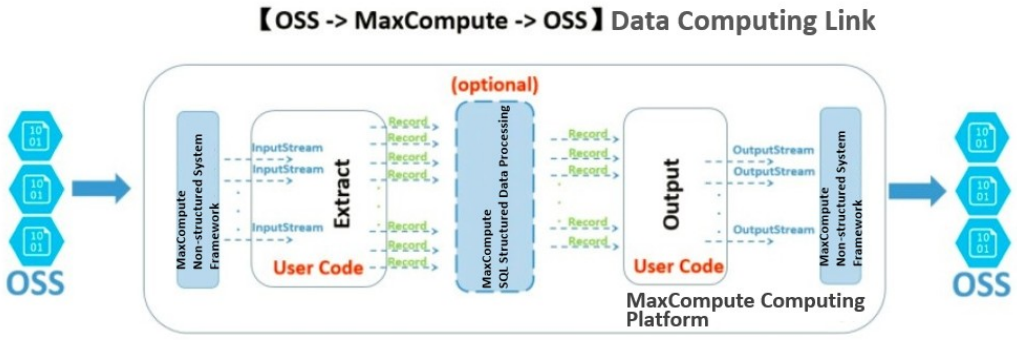
External tables act as an interface between MaxCompute and external storage media. External tables can be used to read data from or write data to various external storage media. Based on the feature of external tables, various links can be established to compute and store data. For example:

- MaxCompute reads data from an Object Storage Service (OSS) bucket that is associated with External table A, performs complex computations, and then generates results to the directory of the OSS bucket that is associated with External table B.
- MaxCompute reads data from Tablestore that is associated with External table A, performs complex computations, and then generates results to the directory of the OSS bucket that is associated with External table B.

> **Note** In the preceding examples, the data source for the SELECT statement is an external table, rather than a MaxCompute internal table.

The following figure shows the flowchart for data migration.

MaxCompute is used as a central computing platform. It reads data from an OSS bucket and then writes the data to another OSS bucket (in a different location or within a different OSS account).



Based on the data flow and processing logic shown in the preceding figure, the unstructured data processing framework can be considered a coupled data ingress and egress at both ends of MaxCompute.

1. The external data from an OSS bucket is converted based on the unstructured framework, and provided to the InputStream class of Java. The extract logic is only used 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 that is built in MaxCompute, and may generate new records.

3. The records obtained after computations are transferred to your custom output logic for further computations. Finally, the required information is extracted from the records, transferred by using OutputStream, and written to OSS.

> **Note** You can perform the preceding steps based on your business requirements.

1.17.2.1.7. STS mode authorization for OSS

This topic describes the Security Token Service (STS) mode authorization for Object Storage Service (OSS), and provides an example for reference.

When you create an external table, the Location-based OSS access account allows you to enter AccessKey ID and AccessKey secret in plaintext mode. However, the account information may be exposed. To prevent account information from being exposed, MaxCompute provides a more secure way to access OSS.

MaxCompute integrates RAM and STS of Alibaba Cloud to enhance account security. You can use one of the following methods to grant permissions:

- 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 compile the following policy content:

```
-- If the owners of MaxCompute and OSS use the same Alibaba Cloud account:
{
"Statement": [
{
 "Action": "sts:AssumeRole",
 "Effect": "Allow",
 "Principal": {
   "Service": [
     "odps.aliyuncs.com"
   ]
 }
}
],
"Version": "1"
}
-- If the owners of MaxCompute and OSS use different Alibaba Cloud accounts:
{
"Statement": [
{
 "Action": "sts:AssumeRole",
 "Effect": "Allow",
 "Principal": {
   "Service": [
     "ID of the Alibaba Cloud account that owns the MaxCompute project@odps.aliyuncs.com"
   ]
 }
}
],
"Version": "1"
}
```

ii. Grant the AliyunODPSRolePolicy permission, which is required 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 must specify the RAM information when you create an OSS external table later.

1.17.2.2. Tablestore data source

1.17.2.2.1. Overview

This topic describes how to access and process data of Tablestore data sources in MaxCompute

As the core computing component of the Alibaba Cloud big data platform, MaxCompute meets most distributed computing requirements within and outside Alibaba Group. MaxCompute SQL provides powerful support for the quick processing and storage of exabytes of offline data. As big data businesses continue to grow, 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 how to import data from Tablestore to MaxCompute, which implements seamless interoperability between data sources.

In online service scenarios, NoSQL KV storage (such as Bigtable or HBase) features flexible schema, easy scalability, and high real-time performance, compared with traditional databases. Alibaba Cloud Tablestore is a large-scale NoSQL data storage service based on the Apsara distributed operating system. It stores and allows real-time access to large volumes of key-value pairs. Tablestore is widely used by all business units in Alibaba Group and the Alibaba Cloud ecosystem. In particular, Tablestore features, such as row-level real-time update and override writing, supplement the append-only operations of MaxCompute tables. As a storage-oriented service, Tablestore does not provide sufficient computing capabilities to concurrently process large volumes of data. In this case, MaxCompute allows you to create external tables to access Tablestore data sources.

Examples are provided to demonstrate how MaxCompute accesses and processes Tablestore data.

> **Note** This topic assumes that you have a basic knowledge of Tablestore operations. If you are not familiar with Tablestore or are new to the concept of key-value tables, you can first learn some basic Tablestore concepts, such as primary keys, partition keys, and attribute columns.

1.17.2.2.2. Use MaxCompute to read and calculate data in Tablestore

1.17.2.2.2.1. Create an external table

This topic describes how to create an external table for Tablestore data sources. It also provides an example for reference.

MaxCompute accesses Tablestore data by using external tables. After you execute the `CREATE EXTERNAL TABLE` statement to introduce the description of Tablestore table data to the metadata of MaxCompute, you can process Tablestore data in the same way as you process data in a standard table.

Execute the following statements to create an external table:

```
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 storage handler that processes
Tablestore data. It defines the interaction between MaxCompute and Tablestore. The related logic is
implemented by MaxCompute.
WITH SERDEPROPERTIES (
-- SERDEPROPERITES is considered an interface that provides parameter options. Two options must be s
pecified for com.aliyun.odps.TableStoreStorageHandler: tablestore.columns.mapping and tablestore.tab
le.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 Tablestore table
s that are accessed by MaxCompute. The columns include primary key columns and attribute columns. Co
lumns whose names start with a colon (:) are primary key columns in Tablestore tables. In this examp
le, :o_orderkey and :o_orderdate are primary key columns. Other columns are attribute columns. A Tab
lestore table can have one to four primary keys of the BIGINT or STRING type. The first primary key
is the partition key. If you specify a table mapping, you must provide all primary key columns in th
e specified Tablestore table. You can specify only the attribute columns that are accessed by MaxCom
pute.
'tablestore.table.name'='ots_tpch_orders'
-- tablestore.table.name: This option is required. It describes the names of Tablestore tables that
are accessed by MaxCompute. If you specify an invalid (nonexistent) Tablestore table name, an error
is returned and MaxCompute does not create a Tablestore 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 Tablestore information, including the instance name and endpoin
t.
```

Note The preceding example maps a Tablestore table to an external table of MaxCompute. After the mapping is specified, you can perform subsequent operations on Tablestore data by using the external table.

1.17.2.2.2. Use an external table to access Tablestore data

This topic describes how to use an external table to access TableStore data and perform computations.

After you create an external table, Tablestore data is introduced into the MaxCompute ecosystem. Then, you can access Tablestore data by using the MaxCompute SQL syntax.

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 In this example, the MaxCompute SQL syntax that you are familiar with is directly used, and all the other implementations of accessing Tablestore data are performed by MaxCompute.

If you want to compute one piece of data multiple times, you can import the data from Tablestore into an internal table of MaxCompute. This way, you do not need to read the data from Tablestore each time you compute the data.

Example:

```
CREATE TABLE internal_orders AS
SELECT odps_orderkey, odps_orderdate, odps_custkey, odps_totalprice
FROM ots_table_external
WHERE odps_orderkey > 5000 ;
```

Note In this example, internal_orders is a MaxCompute table that you are familiar with. This table has all features of MaxCompute internal tables, such as efficient column compression for data storage and complete metadata. This table is stored in MaxCompute. Therefore, it can be accessed faster than an external table in Tablestore. This feature is particularly suitable for hotspot data that is used for multiple computations.

1.17.2.2.3. Write data from MaxCompute to Tablestore

This topic describes how to use external tables to write data from MaxCompute to Tablestore.

Data interaction between MaxCompute and Tablestore includes importing data from Tablestore to MaxCompute for batch processing and exporting the 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 proactively create external tables in Tablestore. Before you export data to a table in Tablestore, make sure that the table exists in Tablestore. If the table does not exist, an error is returned.

For example, you have created the external table ots_table_external 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 data record internal_orders and write the processing results to Tablestore, execute the `INSERT OVERWRITE TABLE` statement. Sample statement:

```
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 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.
- If you execute the `INSERT OVERWRITE` statement on the external table, MaxCompute inserts 200 data records into the table at a time 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

This topic describes how to access and process data of AnalyticDB for MySQL data sources in MaxCompute.

AnalyticDB for MySQL updates or processes data. If both the data processed by AnalyticDB for MySQL and the data in MaxCompute are used for data computations, the data from AnalyticDB for MySQL must be synchronized with the data from MaxCompute. Therefore, you must create an external table to access the data of AnalyticDB for MySQL data sources.

The following example shows how MaxCompute accesses and processes the data of AnalyticDB for MySQL data sources.

1.17.2.3.2. Write data to AnalyticDB

1.17.2.3.2.1. Create an external table

This topic describes how to create external tables for AnalyticDB for MySQL data sources and provides an example for reference.

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}'
```

1.17.2.3.2.2. Write and query data

This topic describes how to use external tables to write and query AnalyticDB for MySQL data.

After an external table is created, you can use it the same way that you use a standard table. You can use the INSERT OVERWRITE, INSERT INTO, and SELECT statements to write data to the external table and check whether the write operation succeeds. For more information about the usage of these statements, see Update the data of a table and SELECT.

1.17.2.3.3. Read data from AnalyticDB for MySQL

This topic describes how to read data from AnalyticDB for MySQL and provides an example for reference.

If you have compiled the ads_read_external script, execute the following statements to read data from AnalyticDB for MySQL:

```
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}'
-- Create an external table.
select * from ads_read_external;
-- Query and read data.
```

1.17.2.4. RDS data source

1.17.2.4.1. Overview

This topic describes how to access and process data of ApsaraDB RDS data sources in MaxCompute.

ApsaraDB RDS updates or processes data. If both the data processed by ApsaraDB RDS and the data in MaxCompute are used for computations, the data from ApsaraDB RDS must be synchronized with the data from MaxCompute. Therefore, you must create an external table to access the ApsaraDB RDS data source.

Some examples are provided to demonstrate how MaxCompute accesses and processes data of ApsaraDB RDS data sources.

> **Note** When you create an external table, you are not required to create the related table in ApsaraDB RDS. However, when you perform the SELECT or INSERT operation on external tables, you must first create the related tables in ApsaraDB RDS.

1.17.2.4.2. Write data to RDS

1.17.2.4.2.1. Create an external table

This topic describes how to create an external table for ApsaraDB RDS data sources. It also provides an example for reference.

Execute the following statements to create an external table:

> **Note** ApsaraDB RDS for MySQL versions earlier than V8.0 are supported. If the ApsaraDB RDS for MySQL version is 8.0 or later, the error `FAILED: Generating job conf failed, gen jobconf failed` is returned. In this case, the external table cannot be created.

```
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}'
```

1.17.2.4.2.2. Write and query data

This topic describes how to use external tables to write and query ApsaraDB RDS data.

After an external table is created, you can use it the same way you use a common data table. You can execute the INSERT OVERWRITE, INSERT INTO, and SELECT statements to write data and check whether the write operation succeeds. For more information about the usage of these statements, see Update the data of a table and SELECT.

1.17.2.4.3. Read data from ApsaraDB RDS

This topic describes how to read data from ApsaraDB RDS and provides an example for reference.

If you have compiled the rds_read_external script, run the following commands to read data from ApsaraDB 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}'
-- Create an external table.
select * from rds_read_external;
-- Query and read data from the external table.
```

1.17.2.5. HDFS data source (Alibaba Cloud)

1.17.2.5.1. Overview

This topic describes how to access and process Apsara File Storage for HDFS data sources in MaxCompute.

Apsara File Storage for HDFS is a distributed file system designed for Alibaba Cloud computing resources such as Elastic Compute Service (ECS) and Container Service.

Apsara File Storage for HDFS allows you to manage and access data in the same way as the open source Hadoop Distributed File System (HDFS). You can use a distributed file system without the need to modify existing big data applications. The distributed file system offers various features such as unlimited capacity, performance expansion, single namespace, multi-party sharing, high reliability, and high availability.

MaxCompute can interact with Apsara File Storage for HDFS for joint computations after you create external tables.

Apsara File Storage for HDFS supports multiple file formats, such as text file, sequence file, RC file, Parquet, and AVRO. The following example shows how MaxCompute accesses and processes data of Apsara File Storage for HDFS. The text file format is used in this example.

1.17.2.5.2. Data processing for common tables

1.17.2.5.2.1. Write data to HDFS

Create an external table

This topic describes how to create an external table for an Apsara File Storage for HDFS data source. It also provides an example for reference.

Run the following commands to create an external table after the textfile_external script is compiled:

```
set odps.sql.hive.compatible=true;
drop table if exists textfile_external;
CREATE external TABLE if not exists textfile_external
(
  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 to MountPointId.
-- /user/textfile is the file directory. Replace it with the actual file directory.
  TBLPROPERTIES(
    "mcfed.fs.dfs.impl"="com.alibaba.dfs.DistributedFileSystem"
);
```

Write and query data

This topic describes how to use external tables to write and query data of Apsara File Storage for HDFS.

After an external table is created, you can use it the same way you use a common data table. You can use INSERT OVERWRITE, INSERT INTO, and SELECT statements to write data and check whether the write operation succeeds. For more information about the usage of these statements, see Update the data of a table and SELECT.

1.17.2.5.2.2. Read data from Apsara File Storage for HDFS

This topic describes how to read data from Apsara File Storage for HDFS and provides an example for reference.

Run the following commands to read data from Apsara File Storage for HDFS after you compile the textfile_external_read script:

```
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 to MountPointId.
-- /user/textfile is the file directory. Replace it with the actual file directory.
  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_in
t=b.id;
-- Query and read data from the external table.
```

1.17.2.5.3. Data processing for partitioned tables

This topic describes how to use external tables to process data of open source Hadoop Distributed File System (HDFS) data sources in partitioned tables. It also provides an example for reference.

Execute the following statements to create an external table and add partitions to the table to process 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 in this command indicates MountPointId.
-- user/partition/textfile/ is the file directory. Replace it with the actual file directory.
  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 to the external table.
insert into table textfile_partition partition(date='20190218')select '1','cd' from (select count(*)
from textfile_partition)a;
insert into table textfile_partition partition(date='20190219')select '2','gh' from (select count(*)
from textfile_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 from the external table.
```

1.17.2.6. TDDL data source

1.17.2.6.1. Overview

This topic describes how to access and process data of Taobao Distributed Data Layer (TDDL) data sources in MaxCompute.

TDDL is a database middleware that is widely used within Alibaba Group. It supports data sharding, read and write splitting, and failovers by encapsulating MySQL databases. In most cases, TDDL can be directly used to access MySQL databases. TDDL also provides the Corona connection mode. Corona is a MySQL proxy that complies with the standard MySQL protocol and can use a Java Database Connectivity (JDBC) driver to establish a connection.

MaxCompute can access MySQL databases of TDDL. Built-in storage handlers encapsulate native APIs provided by Hadoop such as org, apache, hadoop, mapreduce, lib, and db. The JDBC driver of the MySQL database is used for data communication at the underlying layer.

Some examples are provided to demonstrate how MaxCompute accesses and processes data of TDDL data sources.

Notice In terms of create, read, update, and delete (CRUD) operations, only the following operations are supported:

- MaxCompute reads data from the external table that is created for a MySQL database.
- MaxCompute writes data to the external table in append mode.

1.17.2.6.2. Preparations

This topic describes the preparations that are required before you create an external table to process data of a Taobao Distributed Data Layer (TDDL) data source.

By default, many features are disabled in MaxCompute V2.0. Therefore, you must manually configure the following parameters to use a new framework to process unstructured data:

```
set odps.sql.hive.compatible=true;
-- Set this parameter for all DDL and DML statements that are used on the external tables for TDDL.
```

```
set odps.sql.udf.java.retain.legacy=false;
-- Set this parameter for all DDL and DML statements that are used on the external tables for TDDL.
```

```
set odps.sql.jdbc.splits.num=3;
-- Set the number of instance splits that are used by MaxCompute to read data from the MySQL databas
e. Maximum value: 256. Default value: 1. You must set this parameter for the SELECT operation on ext
ernal tables for TDDL.
```

```
set odps.sql.jdbc.reducer.num=3;
-- Set the number of concurrent instances that are used by MaxCompute to write data to the MySQL dat
abase. Maximum value: 256. Default value: 64. If the number of concurrent instances in the generated
execution plan is less than this value, you do not need to change this value. You must set this para
meter if you need to perform the INSERT operation on external tables for TDDL.
```

```
set odps.sql.hive.compatible=true;
-- Call an API operation defined by the open source community to obtain and parse the data types of
the MySQL database. You must set this parameter for all DDL and DML statements that are used on the
external tables for TDDL.
```

```
set odps.sql.type.system.odps2=true;
-- Set this parameter if new data types are involved in SQL statements, such as CREATE, SELECT, and
INSERT. The new data types are TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAMP, and BINARY.
```

1.17.2.6.3. Create an external table

1.17.2.6.3.1. Syntax

This topic describes the syntax that is used to create an external table for a Taobao Distributed Data Layer (TDDL) data source.

External tables can be used as interfaces between MaxCompute and databases. The method that is used to process unstructured data of MySQL databases in TDDL is similar to the method used to access and process OSS unstructured data. To create an external table, you must execute the CREATE EXTERNAL TABLE statement first. The following example shows the syntax:

```
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(
  ...
);
```

Parameters:

- column schema: supports the data types that are listed in the following table. Data type mapping

MySQL Type Name	MaxCompute Type Name
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
DECIMAL	DECIMAL(x, y) (The default precision is (10, 0). An error is returned when an overflow occurs.)

> **Notice** UNSIGNED data types are not supported in MaxCompute. Therefore, loss of precision may occur if UNSIGNED data types are specified. Handle this issue based on scenarios:
>
> - `setproject odps.sql.udf.strict.mode=true;` (strict mode, which is the default mode)
> - Read data from an external table: If loss of precision does not occur when data of the UNSIGNED data type is converted into the SIGNED data type, MaxCompute reads and converts data normally. If loss of precision occurs during data type conversion, the RuntimeException ("value out of range") error is returned.
> - Write data into an external table: MaxCompute does not check data types. You can enable the MySQL database to produce the expected behavior by setting SQL_Mode of the MySQL database. For more information about the setting and behavior of SQL_Mode, see Server SQL Modes.
> - `setproject odps.sql.udf.strict.mode=false;` (non-strict mode, which needs to be explicitly configured)
> - Read data from an external table: If loss of precision does not occur when data of the UNSIGNED data type is converted into the SIGNED data type, MaxCompute reads and converts data normally. If loss of precision occurs during data type conversion, NULL is returned.
> - Write data to an external table: MaxCompute does not check data types. You can enable the MySQL database to produce the expected behavior by setting SQL_Mode of the MySQL database.

- STORED BY: Only built-in storage handlers are supported. Table field types of TDDL must be the same as the data types that are specified by column schema.
- LOCATION: The following three methods can be used to connect to a location:

  Method 1: Connect to the MySQL database LOCATION by using a Java Database Connectivity (JDBC) connection string.

  ```
  jdbc:mysql://<user>:<password>@<host>/<databaseName>?useSSL=false&table=<tableName>
  ```

  user is the username of the JDBC connection string that is used to connect to the MySQL database. password is the password of the JDBC connection string used to connect to the MySQL database. host is the network address of the MySQL database instance. databaseName is the name of the MySQL database. tableName is the name of the MySQL database table that corresponds to the external table.

  Method 2: Connect to the database LOCATION of TDDL in Corona mode.

  ```
  jdbc:mysql://<user>:<password>@<host>/<databaseName>?useSSL=false&table=<tableName>
  ```

  Method 3: Connect to the database LOCATION of TDDL based on appname.

  ```
  jdbc:mysql://dummy_host?table=<tableName>
  ```

  tableName is the name of the MySQL table that corresponds to the external table. You must specify `odps.federation.jdbc.tddl.appname` in the TBLPROPERTIES clause.

> **Notice**
>
> In method 1, MaxCompute interacts with the database by using a direct JDBC connection. You must write the username and password in plaintext. This poses a security risk. Usernames and passwords are hidden when Logview or DESC EXTENDED TABLE is used in MaxCompute. For security concerns, we recommend that you use a separate DDL statement to create an external table before you use the external table.
>
> For example, a project member with higher permissions can create an external table in MaxCompute. Then, other project members can directly use the external table. This prevents project members with lower permissions from using the plaintext username and password. In this case, you do not need to include the plaintext password in SQL scripts.

- TBLPROPERTIES: includes the following items.
  - odps.federation.jdbc.condition: specifies the filter condition that is used when MaxCompute reads data from the MySQL database. Difference between `odps.federation.jdbc.condition` and `select * from text_test_jdbc_write_external where condition`:

    For example, the MySQL database table contains 100 rows of data and you want to obtain 10 data records that meet the specified condition. If you run `odps.federation.jdbc.condition`, data records of the MySQL database table is filtered and MaxCompute reads only 10 data records from the external table. If you run `select * from text_test_jdbc_write_external where condition`, MaxCompute reads 100 data records from the MySQL database table and then obtains 10 data records.
  - odps.federation.jdbc.colmapping: specifies the 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 to the MySQL database. Only the following types are supported: SimpleInsert, InsertOnDuplicateKeyUpdate, and ReplaceInto. If you do not set this parameter, the default value is SimpleInsert.

    The INSERT statement that is executed in MaxCompute is parsed into the following SQL statements to update data in 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 parameter, MaxCompute uses the application name to access the MySQL database by using the TDDL SDK.

1.17.2.6.3.2. Example

This topic provides an example of how to create an external table for a Taobao Distributed Data Layer (TDDL) data source.

The following example shows how to connect to the database of TDDL based on appname. In this example, the application name is `ODPS_TDDL_TEST_APP` and the table name is `odps_federation_localrun_write`.

Example:

```
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

This topic describes how to read data from an external table and provides an example for reference.

Before you perform complex operations such as GROUP JOIN, we recommend that you import data from external tables to MaxCompute tables. This improves the efficiency of data computations. 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:

```
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, a data channel is established between MaxCompute and a MySQL database. MaxCompute does not store data in the MySQL database. If data of the external table is missing from the MySQL database, the data is unavailable in MaxCompute.

If data of the MySQL database is imported to a MaxCompute table, the data is stored in MaxCompute. If data is missing from the MySQL database, you can retrieve the data from the MaxCompute table to which the data is imported.

1.17.2.6.5. Write data to an external table

This topic describes how to write data to an external table.

The column names and data types of the external table must be consistent with those of the database. This ensures that data is correctly written to the eternal table. For more information about data check actions when loss of precision occurs during data type conversions, see Syntax.

Run the following command to write data to an external table:

```
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`. Both statements are executed to append data to external tables. You can use `odps.federation.jdbc.insert.type` to specify the data insertion type. For more information, see Syntax.

1.17.3. External data sources

1.17.3.1. HDFS data source (open-source)

1.17.3.1.1. Overview

This topic describes how to access and process data of open source Hadoop Distributed File System (HDFS) data sources in MaxCompute.

HDFS is the most widely used storage service in the open source community. Most customers use HDFS at the underlying layer of their self-managed big data systems.

MaxCompute uses external tables to access open source HDFS data to facilitate data migration and interact with self-managed customer systems. This reduces customer efforts and costs.

HDFS supports multiple file formats, such as text file, sequence file, RC file, Parquet, and AVRO. Some examples are provided to demonstrate how MaxCompute accesses and processes data of open source HDFS data sources. The text file format is used in the examples.

1.17.3.1.2. Write data to HDFS

1.17.3.1.2.1. Create an external table

This topic describes how to create an external table for an open source Hadoop Distributed File System (HDFS) data source. It also provides an example for reference.

After you compile the textfiletest script, execute the following statements to create an external table:

```
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/';
-- The file directory /user/wbyy/ is for reference only. Replace it with the actual directory.
```

1.17.3.1.2.2. Write and query data

This topic describes how to use external tables to write and query data from the open source Hadoop Distributed File System (HDFS).

After an external table is created, you can use it the same way that you use a standard table. You can use the INSERT OVERWRITE, INSERT INTO, and SELECT statements to write data and check whether the write operation succeeds. For more information about how to use these statements, see Update the data of a table and SELECT.

1.17.3.1.3. Read data from HDFS

This topic describes how to read data from open source Hadoop Distributed File System (HDFS) and provides an example for reference.

After you compile the testfile_read script, execute the following statements to read data from open source HDFS:

```
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/';
-- The file directory /user/wbyy/ is for reference only. Replace it with the actual file directory.
-- Create an external table.
select * from testfile_read;
-- Query and read data from the external table.
```

1.17.3.2. MongoDB data source

1.17.3.2.1. Overview

This topic describes how to access and process data of ApsaraDB for MongoDB data sources in MaxCompute.

ApsaraDB for MongoDB is a stable, reliable, and auto-scaling database service that is fully compatible with MongoDB protocols. ApsaraDB for MongoDB offers a full range of database solutions, such as disaster recovery, backup, restoration, monitoring, and alerting.

MaxCompute can interact with ApsaraDB for MongoDB for joint computations after you create external tables.

Some examples are provided to demonstrate how MaxCompute accesses and processes data of ApsaraDB for MongoDB data sources.

1.17.3.2.2. Preparations

This topic describes the preparations that are required to create an external table and process data of an ApsaraDB for MongoDB data source.

Before you process data of an ApsaraDB for MongoDB data source, you must deploy ApsaraDB for MongoDB. Example:

1. Run the following command to activate ApsaraDB for MongoDB:

   ```
   bin/mongod --dbpath=./db
   ```

2. Run the following command to go to the ApsaraDB for 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 user is created. If 1 is returned, the user is created.

```
db.auth('${user}', '${password}')
```

1.17.3.2.3. Write data to MongoDB

1.17.3.2.3.1. Create an external table

This topic describes how to create external tables for ApsaraDB for MongoDB data sources and provides an example for reference.

Run the following command to create a collection in ApsaraDB for 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 actual va
lues.
```

After the collection is created, execute the following statements 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.${tablename}"
    TBLPROPERTIES(
        "mcfed.mongo.input.split_size"="2",
-- The value of input.split_size is for reference only. Replace it with the actual value.
        "mcfed.location"="mongodb://${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}"
);
```

1.17.3.2.3.2. Write and query data

This topic describes how to use external tables to write and query ApsaraDB for MongoDB data.

After an external table is created, you can use it the same way you use a standard table. You can execute the INSERT OVERWRITE, INSERT INTO, and SELECT statements to write data and check whether the write operation succeeded. For more information about the usage of these statements, see Update the data of a table and SELECT.

1.17.3.2.4. Read data from ApsaraDB for MongoDB

This topic describes how to read data from ApsaraDB for MongoDB and provides an example for reference.

After a row of data is inserted into an existing dataset, run the following command to read data from ApsaraDB for MongoDB:

```
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.${tablename}"
    TBLPROPERTIES(
        "mcfed.mongo.input.split_size"="2",
-- The value of input.split_size is for reference only. Replace it with the actual value.
        "mcfed.location"="mongodb://${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 from the external table.
```

1.17.3.3. HBase data source

1.17.3.3.1. Overview

This topic describes how to access and process data of ApsaraDB for HBase data sources in MaxCompute.

ApsaraDB for HBase is a distributed database based on Hadoop. It can store petabytes of big data and be used in scenarios that require high-throughput random read and write operations.

MaxCompute allows you to create external tables to interact with ApsaraDB for HBase for joint computations.

The following topics provide examples to show how MaxCompute accesses and processes data of ApsaraDB for HBase data sources.

1.17.3.3.2. Write data to HBase

1.17.3.3.2.1. Write data to ApsaraDB for HBase by using an online API

Create an external table

This topic describes how to create an external table of ApsaraDB for HBase data sources. It also provides an example for reference.

Run the following command to create an external table:

```
set odps.sql.hive.compatible=true;
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'='${table.name}','mcfed.hbase.columns.mapping'=':key,cf:a')
-- cf is the column family in the external table of ApsaraDB for HBase.
location 'hbase://host:port'
TBLPROPERTIES('hbase.table.name'='${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 actual values.
```

Write and query data

This topic describes how to use external tables to write and query ApsaraDB for HBase data.

After an external table is created, you can use it the same way that you use a standard table. You can use the INSERT OVERWRITE, INSERT INTO, and SELECT statements to write data and check whether the write operation succeeds. For more information about the usage of these statements, see Update the data of a table and SELECT.

1.17.3.3.2.2. Write data to ApsaraDB for HBase by using bulk load

This topic describes how to write data to ApsaraDB for HBase by using the bulk load method. It also provides an example for reference.

You can use bulk load to import large amounts of data to ApsaraDB for HBase. In the bulk load process, HFiles of ApsaraDB for HBase are generated and then moved to the directory of ApsaraDB for HBase. Compared with the method of calling the put method of ApsaraDB for HBase to write data, the bulk load method provides higher processing efficiency and has less impact on the running of ApsaraDB for HBase. This is because this method bypasses region servers and write-ahead logging (WAL).

MaxCompute performs the following steps to write data to an external table of ApsaraDB for HBase: 1. Sort data in the source table based on row keys. 2. Write data in the HFile format to Hadoop Distributed File System (HDFS). 3. Move HFiles to the directory of ApsaraDB for HBase and notify region servers to load the data.

Create an external table

Run the following command to create an external table:

```
set odps.sql.hive.compatible=true;
CREATE EXTERNAL TABLE if not exists test_hfile_hbase_external
(
id string,
cfa string,
cfb string
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ('mcfed.hbase.table.name'='${table.name}','mcfed.hbase.columns.mapping'=':key,cf:a,cf:b')
-- cf is the column family in the external table of ApsaraDB for HBase.
location 'hbase://host:port'
TBLPROPERTIES('hbase.table.name'='${table.name}','hbase.columns.mapping'=':key,cf:a,cf:b', 'hfile.temp.path'='hdfs://hbase-cluster/hfiletmp','hadoop.user.name'='admin','mcfed.fs.defaultFS'='hdfs://hbase-cluster','mcfed.dfs.nameservices'='hbase-cluster','mcfed.dfs.ha.namenodes.hbase-cluster'='nn1,nn2','mcfed.dfs.namenode.rpc-address.hbase-cluster.nn1'='j63g09343.sqa.eu95:8020','mcfed.dfs.namenode.rpc-address.hbase-cluster.nn2'='j63g09355.sqa.eu95:8020','mcfed.dfs.client.failover.proxy.provider.hbase-cluster'='org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider', '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 actual values.
```

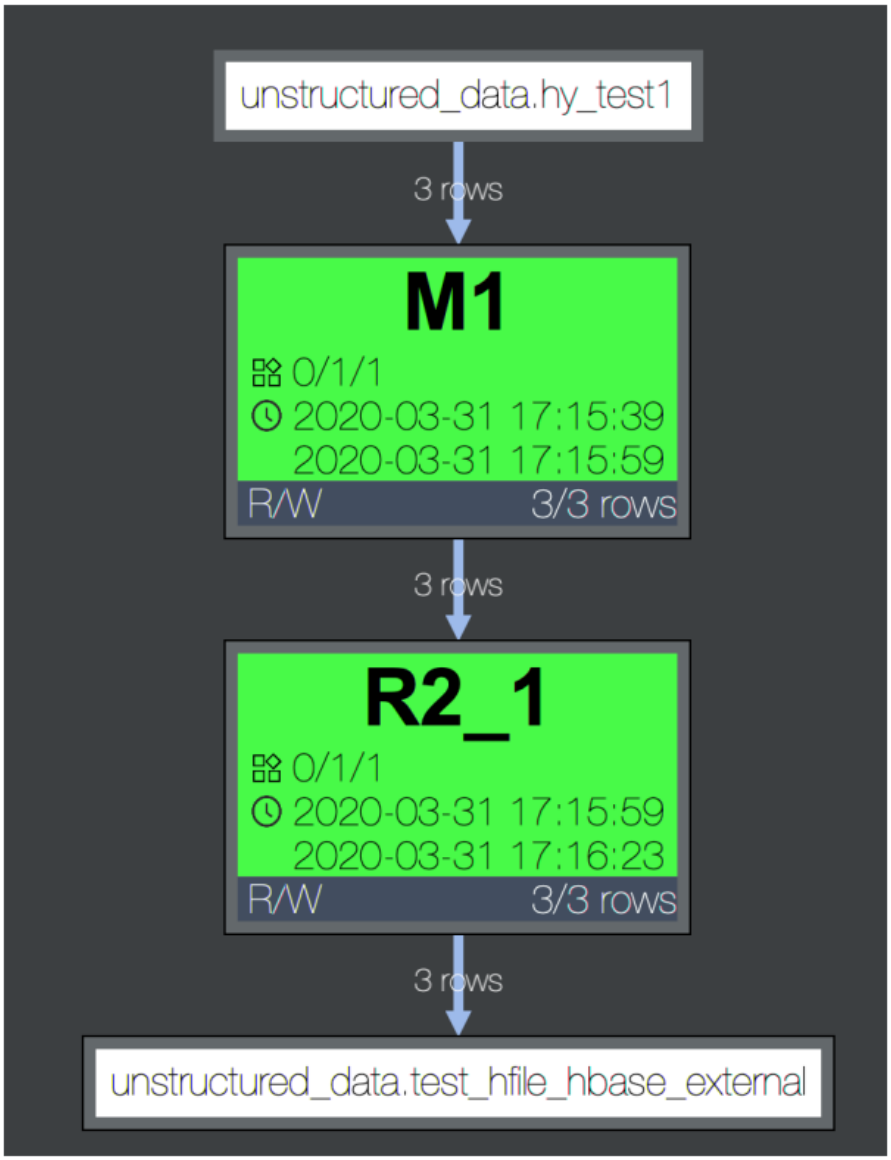
Sort data in the source table based on row keys

The range clustering mechanism of MaxCompute is used to sort data in the source table. If the optimizer generates an execution plan and finds that the table is an external table of ApsaraDB for HBase, the optimizer considers the table as a range-clustered table and sorts all data in the source table.

Example:

```
insert overwrite table test_hfile_hbase_external select uuid(), a_name from hy_test1;
```

The following figure shows the execution plan. Data in the source table hy_test1 is sorted based on row keys. This step is M1 in the execution plan, as shown in the following figure.



Write data in the HFile format to HDFS

The INSERT statement is executed to automatically write data in the HFile format to HDFS. Each Fuxi instance corresponds to a region for the external table of ApsaraDB for HBase. This step is R2_1 in the execution plan, as shown in the preceding figure. In this step, the sorted data is written in the HFile format to HDFS. You must configure a temporary directory of HDFS to store HFiles to which data is written.

Limits on writing data to HFiles:

- The first column must be a row key, which is :key in hbase.columns.mapping.
- The other columns must belong to the same column family. Each instance has only one HFile.

Move HFiles to the directory of ApsaraDB for HBase and notify region servers to load the data.

After data in the HFiles of all regions for the external table of ApsaraDB for HBase is written to HDFS, you can perform bulk load. In this step, MaxCompute automatically moves HFiles to the directory of ApsaraDB for HBase and notifies region servers to load the data .

You can run the following command to check the data import result:

```
select * from test_hfile_hbase_external;
```

1.17.3.3.3. Read data from ApsaraDB for HBase

This topic describes how to read data from ApsaraDB for HBase and provides an example for reference.

After you create a table on an ApsaraDB for HBase client and insert data into the table, run the following commands to read data from the ApsaraDB for HBase table:

```
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'='${table.name}','mcfed.hbase.columns.mapping'=':key,f
1:name,f1:a')
-- f1 is the column family in the ApsaraDB for HBase table.
location 'hbase://host:port'
TBLPROPERTIES('hbase.table.name'='${table.name}','hbase.columns.mapping'=':key,f1:name,f1:a', 'mcfed
.zookeeper.session.timeout'='30', 'mcfed.hbase.client.retries.number'='1', "mcfed.hbase.zookeeper.qu
orum"="${host}", "mcfed.hbase.zookeeper.property.clientPort"="${port}");
-- The values of the zookeeper.session.timeout and hbase.client.retries.number parameters are for re
ference only. Replace them with actual values.
-- 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 from the external table.
```

1.18. Unstructured data access and processing (inside MaxCompute)

1.18.1. Overview

This topic describes how to access and process unstructured volume data by using external tables.

If MaxCompute stores unstructured data as volumes, the data cannot be processed in MaxCompute. You must export the data to an external system for processing.

To address this issue, MaxCompute uses external tables to access various data types. MaxCompute uses external tables to read and write data volumes and processes unstructured data from external data sources, such as Object Storage Service (OSS).

The following topics describe how to create and access volume external tables.

1.18.2. Create a volume external table

1.18.2.1. Syntax

This topic describes the syntax that is used to create a volume external table.

You can first execute the CREATE EXTERNAL TABLE statement to create an external table. The following example shows the syntax:

```
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>']
;
```

Parameters:

- STORED BY: Two built-in storage handlers `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 storage handlers cannot be used, you can build a custom storage handler.
- WITH SERDEPROPERTIES: specifies table attributes such as delimiters for a custom storage handler.
- 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.
  > - The location of a partitioned table must point to the volume.
  > - 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 a storage handler. To use a custom storage handler, you must export the custom storage handler as a JAR package and then add it to MaxCompute as a JAR resource.

1.18.2.2. Use a built-in storage handler to create a table

This topic describes how to use a built-in storage handler to create a table.

You can use a built-in storage handler to create a partitioned table 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'--This is the built-in storage handler.
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'--This is the built-in storage handler.
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

This topic describes how to use a custom StorageHandler to create a table.

If the built-in StorageHandlers are unable to meet your business requirements, you can customize a StorageHandler by using Java and specify some attributes of the Volume external table through which you want to process data.

The following example shows how to create a table:

Assume that the data type of the table that you want to create is TXT and the column delimiter is |. You can perform the following steps to create an external table:

1. Use the MaxCompute Studio or MaxCompute Eclipse development tool to customize 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 process the volume data by using the external table.

1.18.3. Access a Volume external table

This topic describes how to access a Volume external table.

Volume external tables can be accessed in the same way that you access a MaxCompute table. Example:

```
select key,value from volume_ext_pt where ds="20190102";
```

1.19. MaxCompute multi-region deployment

1.19.1. Overview

MaxCompute supports multi-region deployment. This topic describes how to deploy control clusters and compute clusters across regions.

Centralized deployment of control clusters is designed to configure resources and manage computing tasks.

Independent deployment of compute clusters for each region is designed to create projects and deliver computing tasks.

1.19.2. Features

This topic describes the features of multi-region deployment on MaxCompute.

Multi-region deployment on MaxCompute has the following features:

- A MaxCompute system can manage multiple clusters in different regions.
- Data exchanges between clusters within MaxCompute, and data replication and synchronization between clusters are managed based on configured policies.
- Metadata is stored in a centralized manner. Therefore, the infrastructure requirements, such as the network connections of different data centers, are relatively high.
- A unified account system is used.
- The development systems for big data applications, such as DataWorks, are used for all clusters in all regions.
- MaxCompute must run in multi-cluster mode to support multi-region deployment.

> **Note** The conditions and limits on changing the cluster mode must meet the following requirements:
>
> - The network bandwidth must be sufficient to support multi-region data synchronization and link redundancy.
> - Control clusters in the central region have a high latency for basic services, such as Apsara Stack DNS and Tablestore. Therefore, we recommend that you deploy basic services in the same data center to limit the network latency within 5 ms.
> - The network latency between control clusters in the central region and compute clusters in other regions is within 20 ms.
> - Clocks must be synchronized between clusters in different regions and between servers in the same cluster.
> - The network bandwidth must be sufficient to support data replication between clusters.
> - Apsara Stack DNS is required.
> - Servers in different clusters can communicate with each other, and the clusters have similar network infrastructure (1-Gigabit or 10-Gigabit).

- The O&M and upgrades for multi-region deployment are different from those for single-cluster deployment. Multi-region deployment requires higher on-site O&M capabilities.

1.19.3. Instructions

This topic describes the multi-region deployment instructions for MaxCompute.

Instructions:

- Computing tasks on MaxCompute are implemented in a cluster based on data distribution within MaxCompute. Cross-cluster replication and direct reads are used for data exchanges between clusters based on the cluster configurations and distribution of data.
- Data replication and synchronization can be performed between clusters by table or partition based on the cluster configurations.
- You must plan the relationship between MaxCompute projects and clusters based on your business requirements. You can select one or more clusters for a single project.

  > **Note** If you select multiple clusters for a project, data replication is performed among these clusters.

- If cross-cluster computing cannot be implemented for business in a project, you can distribute the data of this business to different clusters and divide the computing tasks based on data distribution.

  > **Note** Cross-cluster data replication and management require management policies and maintenance.

- The bandwidth between clusters in different remote data centers must be sufficient for data replication or read-only operations.
- The two data centers that are involved in cross-region operations share the bandwidth. As a result, cross-cluster replication and read-only operations that involve large amounts of data may consume all of the bandwidth between the two data centers, which affects transmission of other data.
- A cluster failure in the primary data center affects the entire service because the primary data center stores information, including metadata and accounts.
- If the secondary data center fails, projects that have data stored on a cluster of this data center and the O&M of services are affected.

> **Note** If project data is separately stored in the primary data center, the failure of the secondary data center does not affect the computing tasks of the project.

- In common scenarios, data is distributed 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. Examples of multi-region deployment

1.19.4.1. Synchronize table data among multiple clusters

This topic provides an example of how to synchronize table data between multiple clusters in a multi-region deployment scenario.

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 endpoint of Apsara Stack MaxCompute AdminConsole is **http://{odps_ag}:9090**. 9090 indicates the port number of ODPS AG.
  > - To perform this step, choose **MaxCompute Configuration > Project Management > Create Project**.

- You have configured cross-cluster replication for the project. The following example shows 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 Steps 4 and 5, cross-cluster data synchronization is successful.

1.19.4.2. Query the status of data synchronization between primary and secondary clusters

This topic provides an example of 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 the clusters selected for the project. AT-MDU-TEST is set as the default cluster.
  > - The endpoint of Apsara Stack MaxCompute AdminConsole is **http://{odps_ag}:9090**. 9090 indicates the port number of ODPS AG.
  > - To perform this step, choose **MaxCompute Configuration > Project Management > Create Project**.

- You have configured cross-cluster replication for the project. Detailed operations:
  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 availableBandwith and totalBandwith to **2000**, and then 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 click **Start Replication**.

Procedure

Example:

1. Create 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 -con
tent=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 configured.

> **Note** If cross-cluster access is not configured and the data volume is large, the 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 the testmdu project, with AT-MDU-TEST as the default cluster. AT-5KN is selected for the testcross5kn project and serves as the default cluster.
   > - The URL of Apsara Stack MaxCompute AdminConsole is **http://{odps_ag}:9090**. 9090 indicates the port number of MaxCompute AG.
   > - The entry point to create the project is **MaxCompute Configuration > Project Management > Create Project**.

2. Create tables for the testmdu and testcross5kn projects.

   ```
   create table testrep(s string );
   create table tablecross5kn (s string );
   ```

3. Construct some data for the testrep and tablecross5kn tables.
4. Run the following commands to read the table data in the testmdu project directly from the testcross5kn project:

   ```
   use testcross5kn;
   select * from testmdu.testrep;
   ```

   > **Note** If the required data is returned, cross-region direct read 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 the testmdu project, with AT-MDU-TEST as the default cluster. AT-5KN is selected for the testcross5kn project and serves as the default cluster.
   > - The URL of Apsara Stack MaxCompute AdminConsole is **http://{odps_ag}:9090**. 9090 indicates the port number of MaxCompute AG.
   > - The entry point to create the project is **MaxCompute Configuration > Project Management > Create Project**.

2. Create tables for the testmdu and testcross5kn projects.

   ```
   create table testrep(s string );
   create table tablecross5kn (s string );
   ```

3. Construct some data for the testrep and tablecross5kn tables.
4. Configure cross-cluster replication for the testmdu project.

   > **Note** The entry point is **MaxCompute Configuration > Project Management**. Find testmdu and click **Cross-cluster Replication Configuration**.

5. Run the following commands to read the table data in the testmdu project directly from the testcross5kn project:

```
use testcross5kn;
select * from testmdu.testrep;
```

Note If the required data is returned, cross-region direct read 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 Apsara Stack MaxCompute AdminConsole, create two projects: crosssregion and crossregion02.

   Note
   - In this example, select the AT-MDU-TEST and AT-70N compute clusters for the crosssregion project. The default cluster for the crosssregion project is AT-MDU-TEST. Select only the default cluster AT-70N for the crossregion02 project.
   - The endpoint of Apsara Stack MaxCompute AdminConsole is **http://{odps_ag}:9090**. 9090 indicates the port number of ODPS AG.
   - To perform this step, choose **MaxCompute Configuration > Project Management > Create Project**.

2. Configure cross-cluster replication for the crosssregion project.

   Note Choose **MaxCompute Configuration > Project Management**. Find the crosssregion project and click **Cross-cluster Replication Configuration**.

3. Create the business table for the crosssregion project.

   ```
   create table business(bid bigint,name string,phone string,address string,region string);
   ```

4. Generate data for the business table.
5. Run the following command to import data into the business table:

   ```
   tunnel upload business business;
   ```

6. Create the product table for the crossregion02 project.

   ```
   create table if not exists product(pid bigint,name string,type string,color string,bid bigint);
   ```

7. Generate data for the product table.
8. Run the following command to import data into the product table:

   ```
   tunnel upload product product;
   ```

9. Run the following command in Apsara Stack MaxCompute AdminConsole to obtain specified data from the business and product tables:

   ```
   select pro.pid,pro.name,pro.type,pro.color,bus.name,bus.phone from crossregion02.product pro join 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 complete.

1.20. Security solution

1.20.1. Intended users

This section describes the users for which the MaxCompute security solution is intended.

This MaxCompute security solution is intended for all owners and administrators of MaxCompute projects, and users who are interested in the MaxCompute multi-tenant data security system. The MaxCompute multi-tenant data security system supports the following features:

- User authentication
- User and authorization management of projects
- Cross-project resource sharing
- Project data protection

1.20.2. Quick start

This topic describes the procedure of the MaxCompute security solution. This provides guidance for you to perform MaxCompute security operations.

Add a user and grant permissions to the user

Scenario: Jack is the administrator of the prj1 project. Alice, who has an Alibaba Cloud account (alice@aliyun.com), joins the prj1 project. Alice applies for the permissions to create and view tables and submit jobs.

Statements:

```
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.
```

> **Note** Only the project administrator can perform this operation.

Add a role and grant permissions to the role by using ACL-based authorization

Scenario: Jack is the administrator of the prj1 project. Three new members Alice, Bob, and Charlie join the project as data reviewers. The members apply for the permissions to view tables, submit jobs, and read the user profile table.

The project administrator can use ACL-based authorization for objects in this scenario. Statements:

```
use prj1
add user aliyun$alice@aliyun.com
-- Add users.
add user aliyun$bob@aliyun.com
add user aliyun$charlie@aliyun.com
create role tableviewer
-- Create the tableviewer role.
grant List, CreateInstance on project prj1 to role tableviewer
-- Grant permissions to the tableviewer role.
grant Describe, Select on table userprofile to role tableviewer
grant tableviewer to aliyun$alice@aliyun.com
-- Assign the tableviewer role to users.
grant tableviewer to aliyun$bob@aliyun.com
grant tableviewer to aliyun$charlie@aliyun.com
```

Package and share resources

Scenario: Jack is the administrator of the prj1 project. John is the administrator of the prj2 project. Jack wants to share some resources of the prj1 project, such as the datamining.jar file and the sampletable table to the prj2 project owned by John. If Bob in the prj2 project wants to access these resources, John can use ACL- or policy-based authorization to grant permissions to Bob, without the assistance of Jack.

Statements:

1. Jack, the administrator of the prj1 project, creates a resource package in the project.

   ```
   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 a table to the package.
   allow project prj2 to install package datamining
   -- Share the package to the prj2 project.
   ```

2. John installs the package in the prj2 project.

   ```
   use prj2
   install package prj1.datamining
   -- Install the package.
   describe package prj1.datamining
   -- View the resources of the package.
   ```

3. John grants Bob the permission to use the package.

   ```
   use prj2
   grant Read on package prj1.datamining to user aliyun$bob@aliyun.com
   -- Use ACL-based authorization to grant Bob the permission to use the package.
   ```

   > **Note** For more information about cross-project resource sharing, see Cross-project resource sharing.

Configure project data protection

Scenario: Jack is the administrator of the prj1 project. This project contains sensitive data, such as user IDs and shopping records. The project also stores a large number of data mining algorithms that are proprietary. Jack wants to protect the sensitive data and algorithms in the project and allows only users in the project to access the data. In this case, the data cannot be transferred to other projects.

Statements:

```
use prj1
set ProjectProtection=true
-- Enable project data protection.
```

If project data protection is enabled, data in the project can be shared only within the project. Data cannot be transferred to other projects. In some cases, for example, Alice wants to export data tables to other projects for business purposes. This operation is approved by the project administrator. MaxCompute provides two methods to export data from a project for which project data protection is enabled.

Method 1: Create an exception policy. For more information, see Data export methods when project protection is enabled.

1. Create a policy file. For example, create the */tmp/exception_policy.txt* file that only allows Alice to use SQL tasks to export the t1 table from the prj1 project. Content in the exception_policy.txt file:

   ```
   {
   "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. Method 1: Configure an exception policy.

   ```
   use prj1
   -- Enable project data protection and configure an exception policy.
   set ProjectProtection=true with exception /tmp/exception_policy.txt
   ```

   > **Note** To configure an exception policy, make sure that the principal is not granted the DROP and CREATE permissions or the UPDATE permission on resources. This prevents data leaks caused by time-of-check to time-of-use (TOC2TOU).

Method 2: Configure trusted projects. Configure the prj2 project as a trusted project of the prj1 project to enable data transfer from the prj1 project to the prj2 project. For more information, see Data export methods when project protection is enabled.

```
use prj1
add trustedproject prj2
```

> **Note** The package-based resource sharing and project data protection mechanisms are two orthogonal security mechanisms.

In MaxCompute, resource sharing has a higher priority than project data protection. If an object in a protected project is shared with other projects by using the package mechanism, cross-project access to this object is not subject to the rules for project data protection.

1.20.3. User authentication

This topic describes user authentication for MaxCompute security solutions.

User authentication is used to verify the identity of a request sender. Authentication typically includes:

- Verifying the 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

This topic describes user management, role management, and authorization for projects.

Projects are the foundation of the MaxCompute multi-tenant system and the basic unit of data management and computing. If you create a project, you are the owner of this project. You can manage all objects in the project, such as tables, instances, resources, and user-defined functions (UDFs). Users, except the project owner, can access objects in the project only after these users obtain approvals from the project owner.

For example, if Alice is the owner of the test_project project, and another user from the project team of Alice requests to access the resources in the test_project project, Alice can use the methods described in this topic to perform user management and authorization. If the user is not from the project team of Alice, we recommend that Alice implement cross-project resource sharing. For more information, see Cross-project resource sharing.

1.20.4.2. User management

This topic describes how to manage users in projects.

Add a user

If the project owner, Alice, decides to authorize another user, Alice must add the user to this project first. Alice can authorize only the users in the project.

Run the following command to add a user:

```
add user <full_username>
-- Add a user to a project.
```

Remove a user

If a user leaves the project team, Alice must remove the user from the project. After a user is removed from the project, the user no longer has permissions to access resources in the project.

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 permissions to access resources in the project.
> - 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-based authorization related to the user is retained. The policy-based authorization at the role level is revoked, but the policy-based authorization at the project level is retained. If the user is added to the project again, the historical permissions granted by using ACL-based authorization are activated again.
> - Removed users may be added to the original projects as different identities. This poses data security risks. If you are a project owner or use an account that has been assigned the Admin or Super_administrator role, you can run the `purge privs from user <username>;` command to permanently delete the permission information of removed users. The permission information includes the permissions granted by using ACL-, label-, and policy-based authorization information. If you do not remove the relevant users from the project, the following error is returned after you run the purge privs from user <username>; command: `Principal still exist in the project`

1.20.4.3. Role management

This topic describes the operations that you can perform to manage roles in projects.

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. We recommend that you use role-based authorization to authorize users.

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 user who is assigned 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.

Commands that are used for role management:

```
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 a role from a user.
```

> **Note** When you delete a role, MaxCompute checks whether users who are assigned this role exist. If these users exist, the role fails to be deleted. To delete the role, you must revoke this role from all users.

1.20.4.4. ACL-based authorization actions

This topic describes the actions of ACL-based authorization for projects.

Authorization usually involves three elements: subject, object, and action. In a MaxCompute project, a subject is a user and various types of objects are included. Different types of objects support different actions.

The following table describes the types of objects and actions supported by these objects in a MaxCompute object.

Types and supported actions of objects

Object	Action	Description
Project	**Read**	Views project information (excluding project objects), such as the creation time.
Project	**Write**	Updates project information (excluding project objects), such as comments.
Project	**List**	Views objects of all types in a project.
Project	**CreateTable**	Creates tables in a project.
Project	**CreateInstance**	Creates instances in a project.
Project	**CreateFunction**	Creates functions in a project.
Project	**CreateResource**	Creates resources in a project.
Project	**CreateJob**	Creates jobs in a project.
Project	**CreateVolume**	Creates volumes in a project.
Project	**All**	Supports all the preceding actions on projects.
Table	**Describe**	Reads the metadata of a table.
Table	**Select**	Reads data of a table.
Table	**Alter**	Modifies the metadata of a table. Adds or deletes partitions.
Table	**Update**	Overwrites or adds data of a table.
Table	**Drop**	Deletes a table.
Table	**All**	Supports all the preceding actions on tables.
Function	**Read**	Reads data and executes functions.
Function	**Write**	Updates data.
Function	**Delete**	Deletes data.
Function	**All**	Supports all the preceding actions on functions.
Resource,Instance,Job,Volume	**Read**	Reads data of resources, instances, jobs, and volumes.
Resource,Instance,Job,Volume	**Write**	Updates resources, instances, jobs, and volumes.
Resource,Instance,Job,Volume	**Delete**	Deletes resources, instances, jobs, and volumes.
Resource,Instance,Job,Volume	**All**	Supports all the preceding actions on resources, instances, jobs, and volumes.

> **Note** In the preceding table, the CREATE TABLE permission on projects and the SELECT, ALTER, UPDATE, and DROP permissions on tables must be used with the CREATE INSTANCE permission on projects. If these permissions are not used with the CREATE INSTANCE permission, these permissions may become ineffective.

After you add users or create roles, you must grant permissions to these users or roles. ACL-based authorization of MaxCompute is an object-based authorization. An access control list (ACL) that contains the authorization data is considered a resource of an object. Authorization can be performed only when the object exists. If the object is deleted, the ACL is automatically deleted.

ACL-based authorization of MaxCompute is performed by running the GRANT or REVOKE command defined in SQL-92. These commands are used to grant or revoke permissions to or from objects in a project.

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 | reso
urce res_name
subject ::= user full_username | role role_name
```

> **Note**
>
> The commands used for ACL-based authorization do not support the [WITH GRANT OPTION] clause. If 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 performed by one of the following types of users:
>
> - Project owner
> - Users who are assigned the admin role in a project
> - Object creators in a project

Example of ACL-based authorization:

Scenario: Users with Alibaba Cloud accounts alice@aliyun.com and bob@aliyun.com are new members of the test_project project. In the test_project project, they must be able to submit jobs, create data tables, and view existing objects of the project. The administrator performs the following authorization actions:

```
use test_project
-- Open a project.
security
add user aliyun$alice@aliyun.com
-- Add alice@aliyun.com to the project.
add user aliyun$bob@aliyun.com
-- Add bob@aliyun.com to the project.
create role worker
-- Create a role.
grant worker TO aliyun$alice@aliyun.com
-- Assign the role to alice@aliyun.com.
grant worker TO aliyun$bob@aliyun.com
-- Assign the role to bob@aliyun.com.
grant CreateInstance, CreateResource, CreateFunction, CreateTable, List ON PROJECT test_project TO R
OLE worke r
-- Grant permissions to the role.
```

1.20.4.5. View permissions

MaxCompute allows you to view various types of permissions. The permissions include the permissions of users or roles and the access control list (ACL) of specified objects.

View the permissions of a user

Syntax:

```
show grants
-- View the access permissions of a user.
show grants for <username>
-- View the access permissions of a specified user. Only project owners and administrators can perfo
rm this operation.
```

View the permissions of a role

Syntax:

```
describe role <rolename>
-- View the access permissions that are granted to a specified role.
```

View the ACL of a specified object

Syntax:

```
show acl for <objectName> [on type <objectType>]
-- View the ACL of a specified object.
```

> **Note** If [on type <objectType>] is not specified, the object type is table.

Show permissions

MaxCompute uses the following characters to indicate the permissions of users or roles:

- A: Allow. Access is allowed.
- D: Deny. Access is denied.
- C: with Condition. This is conditional authorization. This character appears only for policy-based authorization. For more information, see Condition block structure.
- G: with Grant Option. Permissions on objects can be granted.

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

This topic describes resource sharing across projects.

You are the owner or administrator (with the admin role assigned) of a project, and a user applies for accessing resources of your project. If the applicant is a member of your project, we recommend that you use the user and authorization management features for your project. For more information, see Project user and authorization management. Otherwise, you can share resources with the user across projects by using packages.

A package is used to share data and resources across projects. You can use a package to implement cross-project user authorization. The following example describes a scenario in which the package mechanism can be used.

Members of the Alifinance project need to access data of the Alipay project. The Alipay project administrator adds members of the Alifinance project to the Alipay project, and then grants the new members common permissions on the Alipay project. For security purposes, the Alipay project administrator does not want to authorize every user of the Alifinance project. A mechanism is required to allow the Alifinance project administrator to control access to the authorized objects.

If the package mechanism is used, 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 the package is installed, the Alifinance project administrator can determine whether to grant permissions on the package to members 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 permissions to access these resources, and grants the package user the permissions to install and use the package. The package user consumes the resources. After the package user installs 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

This topic describes the operations that package creators can perform.

Create a package

Run the following command to create a package:

```
create package <pkgname>
```

Delete a package

Run the following command to delete a package:

```
delete package <pkgname>
```

Add a resource that you want to share to a package

Run the following command to add a resource to a 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 object type cannot be project. You cannot use a package of a project to create objects in other projects.
> - In addition to the objects, the operation permissions on the objects are added to the package. If you do not specify permissions for an object by using [with privileges privileges], the object is read-only. You are granted only the READ, DESCRIBE, and SELECT permissions on the read-only objects. An object and its permissions are considered a whole. You can add or delete resources in a package. The permissions are revoked after resources are added or deleted.

Allow a project to use a package

Run the following command to allow a project to use a package:

```
allow project <prjname> to install package <pkgname> [using label <number>]
```

Revoke the permission for a project to use a package

Run the following command to revoke the permission for a project to use a package:

```
disallow project <prjname> to install package <pkgname>
```

View the packages that are already created and installed

Run the following command to view the packages that are already created and installed:

```
show packages
```

View details of a package

Run the following command to view details of a package:

```
describe package <pkgname>
```

1.20.5.2.2. Operations of package users

This topic describes the operations that are performed by package users.

The installed package is a type of independent object in MaxCompute. Resources in a package are those of other projects that are shared with you. To access resources in a package, you must have the permission to read the package. If you do not have this permission, request the project owner or user admin to grant the permission to you. The project owner or user admin can grant the permission by using ACL- or policy-based authorization.

Example of ACL-based authorization: If you are the project owner or user admin, you can run the following commands to allow a user with the Alibaba Cloud account 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
```

Example of policy-based authorization: If you are the project owner or user admin, you can run the following commands to allow all users in the prj2 project to access resources in the package.

```
use prj2
install package prj1.testpkg
put policy /tmp/policy.txt
```

The policy.txt file in the /tmp directory contains the following content:

```
{
"Version": "1", "Statement": [{
"Effect":"Allow",
"Principal":"*",
"Action":"odps:Read", "Resource":"acs:odps:*:projects/prj2/packages/prj1.testpkg"
}]
}
```

Install a package

Run the following command to install a package:

```
install package <pkgname>;
```

> **Note** In the preceding command, pkgname must be in the format of <projectName>.<packageName>.

Uninstall a package

Run the following command to uninstall a package:

```
uninstall package <pkgname>;
```

> **Note** In the preceding command, pkgname must be in the format of <projectName>.<packageName>.

View packages

Run the following commands to view packages:

```
show packages
-- View the packages that you have already created and installed.
describe package <pkgname>
-- View details of a package.
```

1.20.6. Project protection

1.20.6.1. Overview

This topic describes the project data protection mechanism.

Some enterprises, such as financial institutions and military enterprises, have high data security requirements. For example, their employees can perform their jobs only 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 leaks 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. In this case, you are required to perform configurations and operations for project data protection.

1.20.6.2. Project data protection

This topic describes the project data protection mechanism of MaxCompute.

MaxCompute provides a project data protection mechanism. This mechanism can forbid all operations that may cause data transfer to other projects. You need only to run the following command to enable the data protection mechanism for your project.

```
set security.ProjectProtection=true;
-- Set security.ProjectProtection to true. This ensures that data can be used within a project and c
annot be transferred to other projects. This setting does not take effect on data in external tables
.
-- You can also run the following commands to enable the project data protection mechanism.
set security.ProjectProtection=true;
set project IDENTIFY_EXTERNAL_TABLE_WRITE_AS_DATALEAK=true;
-- After you set security.ProjectProtection to true, set project IDENTIFY_EXTERNAL_TABLE_WRITE_AS_DA
TALEAK to true. This means that project data cannot be written to external storage sources by using
external tables .
```

After the project data protection mechanism is enabled, data of your project cannot be transferred to other projects.

By default, security.ProjectProtection is set to false, which indicates that the project data protection mechanism is disabled. Users who are authorized to access multiple projects can perform cross-project data access operations to transfer data. If a project stores highly-sensitive data, the project administrator must configure the project data protection mechanism.

1.20.6.3. Data export methods after project data protection is enabled

This topic describes the data export methods that you can use after project data protection is enabled for MaxCompute.

After data protection is enabled for your project, you may soon encounter this situation: Alice submits a request to export the data of a table from the project. You approve the request from Alice because this table does not contain sensitive data. To meet the business requirements of Alice, MaxCompute provides two methods to export data after project data protection is enabled.

Configure an exception policy

If a project owner enables project data protection, the owner can run the following command to configure an exception policy:

```
set ProjectProtection=true with exception <policyFile>
```

> **Note** The exception policy is different from the policy configured for policy-based authorization even though the two policies have the same syntax. This exception policy acts as an exception to the rules of project data protection. After the exception policy is configured, the access requests that meet the description of the exception policy are ignored by the project data protection rules.

Example of an exception policy:

The following policy allows Alice with the Alibaba Cloud account Alice@aliyun.com to export data to the alipay project when Alice uses SQL tasks to perform SELECT operations on the alipay.table_test table.

```
{
"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 exception policy is not a common authorization method. If Alice does not have the SELECT permission on the alipay.table_test table, Alice cannot export data even if the preceding exception policy is configured. Project data protection is a method to control data transfer but does not control data access. Project data protection is effective only if users can access their destination data.
> - After an exception policy is configured, data leaks may occur due to time-of-check to time-of-use (TOCTOU). TOCTOU is also considered a race condition. Symptom:
>
> i. [TOC stage] User A submits an application to the project owner to export the t1 table. After the owner verifies that the t1 table does not contain sensitive data, the owner configures an exception policy to authorize user A to export the t1 table.
>
> ii. A malicious user writes sensitive data to the t1 table.
>
> iii. [TOU stage] User A exports the t1 table. The t1 table exported by user A is not the table verified by the project owner.
>
> To prevent this issue, the project owner must make sure that all project members, including the users who are assigned the admin role, do not have the UPDATE permission or the DROP and CREATE TABLE permissions on the table that a user wants to export. In the preceding example, we recommend that the project owner create a snapshot of the t1 table in Step i, and then use this snapshot to configure an exception policy. In addition, the project owner does not assign the admin role to other project members.

Configure a trusted project

If data protection is enabled for a project, data of the project can be exported to its trusted project. This export does not violate the rules of project data protection. If multiple projects are configured as trusted projects for each other, they form a trusted project group. Data of a project can be exported to other projects in the group but cannot be exported to the projects out of the group.

You can run the following commands to manage the trusted projects:

```
ist trustedprojects
-- List all trusted projects that are configured for a project.
add trustedproject <projectname>
-- Add a project as a trusted project for the project.
remove trustedproject <projectname>
-- Remove a trusted project from the project.
```

1.20.6.4. Package-based resource sharing and project data protection

This topic describes the relationship between package-based resource sharing and data protection in MaxCompute projects.

In MaxCompute, package-based resource sharing and project data protection are mutually independent mechanisms that take effect at the same time, but their features are mutually exclusive.

In MaxCompute, package-based resource sharing takes precedence over project data protection. If a data object in a project is shared with users in other projects based on resource sharing, the project data protection rules will not apply to this object.

To prevent data transfer from a project to another, you must check the following items after you set ProjectProtection to True:

- Make sure that no trusted projects are added. If trusted projects are added, evaluate possible risks.
- Make sure that no exception policies are configured. If exception policies are configured, evaluate possible risks, especially the risks caused by time-of-check to time-of-use (TOC2TOU).
- Check whether package-based data sharing is not in use. If package-based data sharing is in use, make sure that the package contains no sensitive data.

1.20.7. Project security configurations

This topic describes security configurations for MaxCompute projects.

MaxCompute is a platform that allows multiple tenants to process data at the same time. When these tenants process data on MaxCompute, they may have different requirements for data security. To meet their requirements, MaxCompute allows tenants to configure security settings for projects. Project owners can specify an external account system that is supported by MaxCompute and an authentication model to suit their needs.

MaxCompute supports multiple orthogonal authorization mechanisms, such as ACL-based authorization, policy-based authorization, and implicit authorization. If implicit authorization is used, an object creator is automatically granted the permissions to access the object. However, not all users require these security mechanisms. You can configure an authentication model based on your business requirements and usage habits.

```
show SecurityConfiguration
-- View the security configuration of a project.
set security.CheckPermissionUsingACL=true/false
-- Enable or disable ACL-based authorization. The default value of security.CheckPermissionUsingACL
is true.
set security.CheckPermissionUsingPolicy=true/false
-- Enable or disable policy-based authorization. The default value of security.CheckPermissionUsingP
olicy is true.
set security.ObjectCreatorHasAccessPermission=true/false
-- Allow or disallow an object creator to be granted the object access permission by default. The de
fault value of security.ObjectCreatorHasAccessPermission is true.
set security.ObjectCreatorHasGrantPermission=true/false
-- Allow or disallow an object creator to be granted the authorization permission by default. The de
fault value of security.ObjectCreatorHasGrantPermission is true.
set security.LabelSecurity=true/false
-- Enable or disable the label security policy.
set security.ProjectProtection=true/false
-- Enable or disable project data protection to allow or disallow data transfer to other projects.
```

1.20.8. Authorization policies

1.20.8.1. Policy overview

This topic describes the policies that are used for policy-based authorization.

Policy-based authorization is based on principal. A principal can be a user or role. Policies are considered resources of a principal. You can perform policy-based authorization only on a principal that exists. If a principal is deleted, the policies of the principal are automatically deleted. Policy-based authorization uses a custom policy language defined by MaxCompute to perform authorization operations on principals. After the authorization is complete, principals are allowed or not allowed to access resources in projects.

Policy-based authorization is a new authorization mechanism. It is used to handle complicated scenarios in which ACL-based authorization struggles to deal with, such as:

- A principal is granted permissions to access a group of resources, such as all functions and all tables whose name starts with taobao, at a time.
- A principal is granted permissions with specified conditions, for example, the permission on a table is effective

at a specified time, for a request from a specified IP address, or for only SQL tasks (not supported for other types of tasks)

Command syntax:

```
GET POLICY;
-- Read the policies of a project.
PUT POLICY <policyFile>;
-- Set a policy for the project and use this policy to overwrite an existing policy.
GET POLICY ON ROLE <roleName>;
-- Read the policies of a role in the project.
PUT POLICY <policyFile> ON ROLE <roleName>;
-- Set a policy for a role in the project and use this policy to overwrite an existing policy.
```

Note MaxCompute supports project policies and role policies. A project policy applies to all users of the project, whereas a role policy applies only to users who are assigned roles. You must specify a user principal for project policies. You cannot specify a user principal for role policies because the principal has been assigned a role.

Example of configuring a project policy for policy-based authorization:

Scenario: A project policy needs to be configured to ensure that the user with the Alibaba Cloud account alice@aliyun.com can only submit a request from the CIDR block 10.32.180.0/23 before 2019-11-11 23:59:59. The user can perform only the operations to create instances and tables and view the operations that are performed on the test_project project. The user is not allowed to delete tables from the test_project project.

The following project policy is configured:

```
{
"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

- Only role policies and project policies are supported. User policies are not supported.
- Only one policy file can be configured for each type of policy. When the PUT POLICY command is executed to configure a policy, the configured policy overwrites an existing policy. To modify a policy, perform the following steps:
  i. Run the GET POLICY command.
  ii. Manually merge policy statements.
  iii. Run the PUT POLICY command.

1.20.8.2. Policy-related terminology

This topic describes the basic terms used in a policy.

Permission is a basic concept of access control. If 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. A policy refers to a set of statements.

An access policy consists of the following access control elements: principal, action, resource, access restriction, and effect. The following sections provide brief descriptions of these elements.

Principal

A principal of an object is a user or role 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, 2019. Michael is the principal of the object.

Action

An action is an activity that the principal has permissions to perform. For example, the access policy allows Michael to perform the CreateObject action on the resource SampleBucket before December 31, 2019. CreateObject is an action of the access policy.

Resource

A resource is the object that a principal requests access to. For example, the access policy allows Michael to perform the CreateObject action on the resource SampleBucket before December 31, 2019. SampleBucket is a resource of the access policy.

Access restriction

Access restriction is the condition that specifies whether a permission takes effect. For example, the access policy allows Michael to perform the CreateObject action on resource SampleBucket before December 31, 2019. The access restriction is before December 31, 2019.

Effect

Authorization effect has two options: allow or deny. Deny takes precedence over allow during permission checks.

> **Notice** Deny and revocation of authorization are different. Revocation of authorization involves revocation of allow and deny. For example, conventional databases support the Revoke and Revoke Deny actions.

1.20.8.3. Access policy structure

1.20.8.3.1. Overview

This topic describes the language structure of policies defined for policy-based authorization.

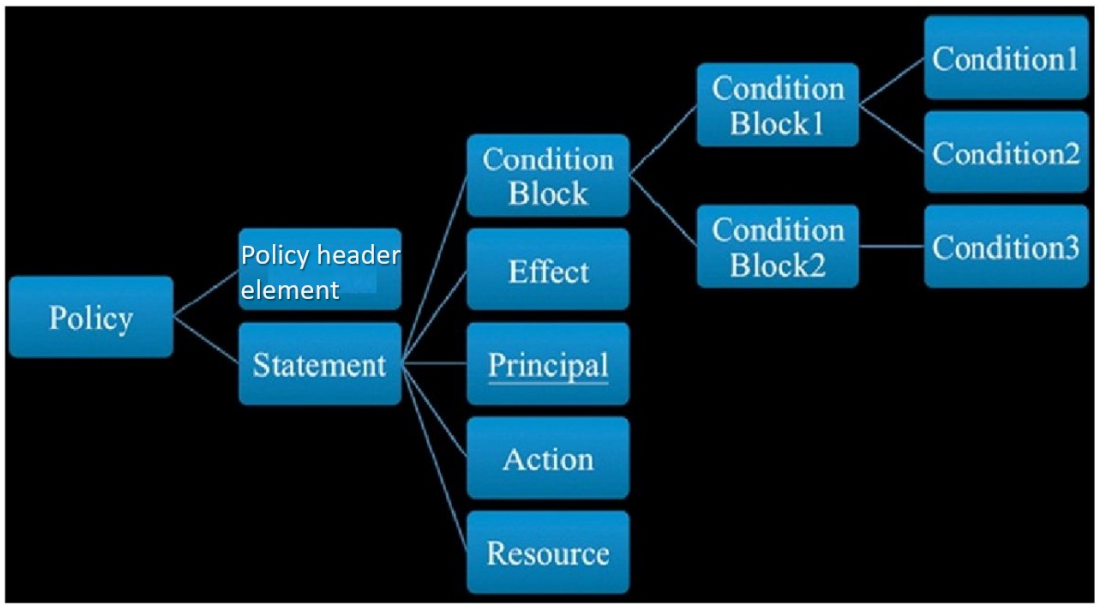
A policy consists of the following parts:

- An optional policy header
- One or more statements

> **Note** 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.

Structure of a policy



1.20.8.3.2. Authorization statement structure

This section describes the structure of the authorization statement in policies defined for policy-based authorization.

An authorization statement includes the following entries:

- Effect: indicates the permission type of this statement. The value can be 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 specify 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 multiple operation names and supports the asterisk (*) and question mark (?). For example, `Action = *` indicates all operations.
- Resource: indicates the authorization object. It can be one or multiple object names and supports the asterisk (*) and question mark (?). 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.

1.20.8.3.3. Structure of condition blocks

This topic describes the structure of condition blocks in the statements that are used for policy-based authorization.

A condition block consists of one or more condition clauses. A condition clause consists of a condition action type, condition key, and condition value. The condition action types and condition keys are described in detail in subsequent topics.

The following rules are used to check whether conditions in a condition block are met:

- A condition key can correspond to one or more values. If the value of the condition key is equal to one of the specified values, the condition is met.
- If multiple condition keys in a clause of a condition action type are met, this clause is met.
- If condition keys in all condition clauses of a condition block are met, this condition block is met.

1.20.8.3.4. Condition action types

This topic describes the types of condition actions in policy condition clauses.

The actions on the following data types are supported: STRING, BOOLEAN, and IP address as well as the numeric, date, and time types. Methods supported by each action type:

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. Condition keys

This topic describes the condition keys in policy condition clauses.

MaxCompute supports the condition keys reserved by Alibaba Cloud Service (ACS). The following table describes these condition keywords.

Condition keys

Condition key reserved by ACS	Data type	Description
acs:CurrentTime	Date and Time	The time when the web server receives a request. The time is based on the ISO 8601 standard, for example, 2019-11-11T23:59:59Z.

Condition key reserved by ACS	Data type	Description
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 sends a request.
acs:UserAgent	STRING	The User-Agent header in a request.
acs:Referer	STRING	The HTTP referer in a request.

Note acs:SourceIp indicates the remote IP address of an HTTP connection. This condition key does not indicate an IP address in the IP address list represented by x-forwarded- for in the HTTP header. For example, if 10.230.205.105 is a LAN IP address, acs:SourceIp indicates the egress gateway IP address of this LAN. If the network egress uses a proxy server, acs:SourceIp indicates the IP address of the proxy server. If the request traverses across multiple proxy servers, acs:SourceIp indicates the IP address of the final proxy server. The value of acs:SourceIp may vary based on the rules configured on proxy servers.

1.20.8.4. Access policy norm

1.20.8.4.1. Principal naming conventions

This topic describes the principal naming conventions in language specifications of policies defined for policy-based authorization.

The principal is the request sender. Only an Alibaba Cloud account, domain account, or Taobao account can be used as a principal. An Alibaba 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. Naming conventions of resources

This topic describes the naming conventions of resources in language specifications of policies defined for policy-based authorization.

MaxCompute resources are named in the following format:

```
acs:<service-name>:<namespace>:<relative-id>
```

The following table describes the parameters in this format.

Parameters

Parameter	Description
acs	The retained resource header.
service-name	The name of an Alibaba Cloud service (ACS), such as MaxCompute, Object Storage Service (OSS), and Tablestore.

Parameter	Description
namespace	The namespace that is used for resource isolation. If an Alibaba Cloud account is used for resource isolation, the value can be an Alibaba Cloud account. If this parameter is not required, you can use an asterisk (*) instead.
relative-id	The service-related resource. Its meaning is based on services. The value format of this parameter supports a tree structure that is similar to the file path. The following value format of relative-id is used for MaxCompute: `projects/<project_name>/<object_type>/<object_name>`

The following table provides examples of MaxCompute resource names.

Examples of MaxCompute resource names

Name	Description
*	Indicates all objects in the project.
projects/prj1/tables/t1	Indicates the t1 table of the prj1 project.
projects/prj1/instances/*	Indicates all instances in the prj1 project.
projects/prj1/tables/*	Indicates all tables in the prj1 project.
projects/prj1/tables/taobao*	Indicates all tables in the prj1 project. The table names are all prefixed by taobao.

1.20.8.4.3. Naming conventions of actions

This topic describes the naming conventions of actions in language specifications of policies defined for policy-based authorization.

Naming convention of actions:

```
<service-name>:<action-name>
```

Parameters:

- service-name: the name of an Alibaba Cloud service, for example, MaxCompute, Object Storage Service (OSS), and Tablestore.
- action-name: the name of the service-related action.

The following table provides examples of MaxCompute action names.

Examples of MaxCompute action names

Action name	Description
*	Indicates all actions.
odps:*	Indicates all MaxCompute actions.
odps:CreateTable	Indicates the CreateTable action of MaxCompute.
odps:Create*	Indicates all MaxCompute actions whose names start with Create.

1.20.8.4.4. Naming conventions of condition keys

This topic describes the naming conventions of condition keys in language specifications of policies defined for policy-based authorization.

Condition keys that are reserved for Alibaba Cloud Services (ACSs) are named in the following format:

```
acs:<condition-key>
```

Parameter: condition-key: the condition key, which is accessible to all ACSs. The following condition keys are supported: acs:CurrentTime, acs:SecureTransport, acs:SourceIp, acs:UserAgent, and acs:Referer.

Condition keys that are specific to an ACS are named in the following format:

```
<service-name>:<condition-key>
```

Parameter: condition-key: the condition key that is specific to an ACS.

1.20.8.4.5. Example of a policy for policy-based authorization

This topic provides an example of a policy for policy-based authorization.

Sample policy:

```
{
"Version": "1",
"Statement": [{
"Effect":"Allow",
"Principal":"ALIYUN$alice@aliyun.com",
"Action":["odps:CreateTable","odps:CreateInstance","odps:List"],
"Resource":"acs:odps:*:projects/prj1",
"Condition":{ "DateLessThan": {
"acs:CurrentTime":"2019-11-11T23:59:59Z"
},
"IpAddress": { "acs:SourceIp":"10.32.180.0/23"
}
}
},
{
"Effect":"Deny", "Principal":"ALIYUN$alice@aliyun.com",
"Action":"odps:Drop",
"Resource":"acs:odps:*:projects/prj1/tables/*"
}]
}
```

Note The user with the Alibaba Cloud account alice@aliyun.com is authorized to submit a request from CIDR block 10.32.180.0/23 only before 2019-11-11T23:59:59Z. The user has only the CREATE INSTANCE, CREATE TABLE, and LIST permissions on the prj1 project. The user is not allowed to delete tables from the prj1 project.

1.20.8.5. Differences between policy-based authorization and ACL-based authorization

This topic describes the differences between policy-based authorization and ACL-based authorization.

ACL-based authorization

- Before you grant or revoke permissions, make sure that the grantee, such as a user or role and an object such as a table exists. This also applies to authorization in Oracle and the purpose is to avoid the security risks that may occur after you delete and recreate an object with the same name.
- If you delete an object, all permissions related to the object are automatically revoked.
- Only allow (whitelist) authorization is supported. Deny (blacklist) authorization is not supported.
- The GRANT and REVOKE commands are used for ACL-based authorization. These commands are easy to use and not prone to mistakes. Conditional authorization is not supported.
- ACL-based authorization is suitable for simple scenarios where conditional authorization or a deny action is not required, and only the existing objects need to be authorized.

Policy-based authorization

- Before you grant or revoke permissions, you are not required to check whether the grantees or object exists. You can use double quotation marks "" to represent objects. If you use *projects/tbproj/tables/taobao* to represent an object, the object is all tables whose names start with taobao in the tbproj project. Similar to authorization in MySQL, policy-based authorization allows you to grant permissions to a non-existent object. The grantee must consider the security risks that may occur after you delete and recreate an object with the same name.
- If you delete an object, all permissions related to the object are not revoked.
- Both allow (whitelist) authorization and deny (blacklist) authorization are supported. If allow (whitelist) authorization and deny (blacklist) authorization are performed at the same time, the deny (blacklist) authorization takes precedence.
- Conditional authorization is supported. The grantee can enforce a maximum of 20 conditions on allow or deny authorization. For example, these conditions can be used to specify a valid range of IP addresses for requestors and the access time that must be earlier than the specified time.
- Policy-based authorization is suitable for relatively complicated scenarios where conditional authorization and deny actions are required for non-existent objects.
- The commands for policy-based authorization are used and these commands are complex.

1.20.8.6. Limits

This topic describes the limits of policy-based authorization.

Limits

Item	Upper Limit	Description
ACCESS_POLICY_SIZE_LIMIT	32 KB	The maximum size of the text of a policy.
USER_NUMBER_LIMIT_IN_ON E_PROJECT	1000	The maximum number of users that can be added to a project.
ROLE_NUMBER_LIMIT_IN_ON E_PROJECT	500	The maximum number of roles that can be created for a project.
ROLE_NAME_LENGTH_LIMIT	64 KB	The maximum character length of a role name.
SECURITY_COMMENT_SIZE_L IMIT	1 KB	The maximum character length of a comment.
PACKAGE_NAME_LENGTH_LI MIT	128 KB	The maximum character length of a package name.

Item	Upper Limit	Description
ALLOW_PROJECT_NUMBER_LIMIT_IN_ONE_PACKAGE	1024	The maximum number of projects for which a package can be installed.
RESOURCE_NUMBER_LIMIT_IN_ONE_PACKAGE	256	The maximum number of resources that can be included in a package.
PACKAGE_NUMBER_LIMIT_IN_ONE_PROJECT	512	The maximum number of packages that can be created for a project.
INSTALLED_PACKAGE_NUMBER_LIMIT_IN_ONE_PROJECT	64	The maximum number of packages that can be installed for a project.

1.20.9. Collection of security statements

1.20.9.1. Project security configurations

This topic describes the statements that are used for security configurations of projects.

Authentication configurations

Statements for authentication configurations

Statement	Description
show SecurityConfiguration	Allows you to view security configurations of a project.
set CheckPermissionUsingACL=true/false	Allows you to enable or disable ACL-based authorization.
set CheckPermissionUsingPolicy=true/false	Allows you to enable or disable policy-based authorization.
set ObjectCreatorHasAccessPermission=true/false	Allows you to grant or revoke default access permissions to or from object creators.
set ObjectCreatorHasGrantPermission=true/false	Allows or disallows an object creator to be granted the ACL-based authorization permission by default.

Project data protection

Statements for project data protection

Statement	Description
set ProjectProtection=false	Allows you to disable project data protection.
set ProjectProtection=true [with exception <policy>]	Allows you to enable project data protection.
list TrustedProjects	Allows you to view trusted projects.
add TrustedProject <projectName>	Allows you to add a project to trusted projects.
remove TrustedProject <projectName>	Allows you to remove a project from trusted projects.

1.20.9.2. Project permission management

This topic describes the statements used for permission management of projects.

User management

Statements

Statement	Description
list users	Allows you to view all users that are added to a project.
add user <username>	Allows you to add a user.
remove user <username>	Allows you to remove a user.

Role management

Statements

Statement	Description
list roles	Allows you to view all the existing roles.
create role <rolename>	Allows you to create a role.
drop role <rolename>	Allows you to delete a role.
grant <rolelist> to <username>	Allows you to revoke roles from a user.
revoke <rolelist> from <username>	Allows you to assign one or multiple roles to a user.

ACL-based authorization

Statements

Statement	Description
grant <privList> on <objType> <objName> to user <username>	Allows you to grant permissions to a user
grant <privList> on <objType> <objName> to role <rolename>	Allows you to grant permissions to a role.
revoke <privList> on <objType> <objName> from user <username>	Allows you to revoke permissions from a user.
revoke <privList> on <objType> <objName> from role <rolename>	Allows you to revoke permissions from a role.

Policy-based authorization

Statements

Statement	Description
get policy	Allows you to view policy settings of a project.
put policy <policyFile>	Allows you to configure a policy for a project
get policy on role <roleName>	Allows you to view the policy settings of a role.
put policy <policyFile> on role <roleName>	Allows you to configure a policy for a role.

Permission review

Statements

Statement	Description
whoami	Allows you to view the information about a user.
show grants [for <username>] [on type <objectType>]	Allows you to view the permissions and roles of a user.
show acl for <objectName> [on type <objectType>]	Allows you to view the authorization information of an object.
describe role <roleName>	Allows you to view the authorization and assignment information of a role.

1.20.9.3. Package-based resource sharing

This topic describes the statements that are used for package-based resource sharing.

Share resources

Statements for sharing resources

Statement	Description
create package <pkgName>	Allows you to create a package.
delete package <pkgName>	Allows you to delete a package.
add <objType> <objName> to package<pkgName> [with privileges privs]	Allows you to add resources that you want to share to a package.
remove <objType> <objName> from package <pkgName>	Allows you to remove shared resources from a package.
allow project <prjName> to install package <pkgName> [using label <num>]	Allows a project to use a package of a specified user.
disallow project <prjName> to install package <pkgName>	Disallows a project from using the package.

Use resources

Statements for using resources

Statement	Description
install package <pkgName>	Allows you to install a package.
uninstall package <pkgName>	Allows you to uninstall a package.

View packages

Statements for viewing packages

Statement	Description
show packages	Allows you to view all the packages that are created and installed.
describe package <pkgName>	Allows you to view the details of a package.

1.21. Hierarchical throttling

1.21.1. Overview

MaxCompute provides user-based job throttling.

If some users perform inappropriate operations, a series of issues may arise. For example, if a user submits a large number of jobs, the jobs may be queued for a long period of time and resources in the resource group may be exhausted. To address these issues, MaxCompute provides user-based job throttling in this version and later. After user-based job throttling is enabled, the project owner can specify an upper limit for the number of jobs that can be concurrently submitted based on the resources required for the business or teams.

1.21.2. Hierarchical throttling

MaxCompute supports hierarchical throttling.

MaxCompute uses global throttling for system-wide configurations. To properly control project resources, MaxCompute also allows project owners to use hierarchical throttling for a project.

> **Note** Project owners can use hierarchical throttling rules to specify the maximum number of instances that are allowed for a project.

1.21.3. Specify the maximum number of instances that are allowed for a project

This topic describes how to use hierarchical throttling rules to specify the maximum number of instances that are allowed for a project.

Formulate JSON rules for throttling

Assume that the following throttling rules are defined:

- test_rule1: indicates that the maximum number of instances that users except user1 can run in the information_schema_dev project at the same time is two.
- test_rule2: indicates that the maximum number of instances that user1 can run in the information_schema_dev project at the same time is four.

If the user field is set to a specified user other than an asterisk (*), the priority of the specified user is higher than the priorities of other users. After the two rules take effect, users except user1 can run a maximum of two instances in the information_schema_dev project at the same time. However, user1 can run a maximum of four instances in the information_schema_dev project at the same time.

In addition, JSON rules of a project must be written in the same list.

Sample rules:

```
[{
    "name": "test_rule1",
    "entity": {
        "project": "information_schema_dev",
        "user": "*"
     },
    "settings": [{
        "metric": "total_instances",
        "threshold": "2",
        "action": "deny"
    }]
},
{
    "name": "test_rule2",
    "entity": {
        "project": "information_schema_dev",
        "user": "user1"
    },
    "settings": [{
        "metric": "total_instances",
        "threshold": "4",
        "action": "deny"
    }]
}]
```

Note In the sample rules, you must specify name, project, user, and threshold. Other fields are reserved for future use and must retain their default values.

Configure and view throttling rules

Convert the preceding sample rules in the JSON format into a single row, and run the `setproject` command in the MaxCompute console to configure throttling rules.

```
information_schema_dev> setproject THROTTLING_RULES=[{"name":"test_rule1","entity":{"project":"information_schema_dev","user":"*"},"settings":[{"metric":"total_instances","threshold":"2","action":"deny"}]},{"name":"test_rule2","entity":{"project":"information_schema_dev","user":"19655xxxxxx48481"},"settings":[{"metric":"total_instances","threshold":"4","action":"deny"}]}];
```

Run the following command to view the throttling rules:

```
information_schema_dev> setproject;
```

View the output information for throttling

If a large number of instances that meet the throttling rules run in the information_schema_dev project, an error is returned. Example of error information:

```
FAILED: Request rejected by flow control. Break rule: test_rule2. Matched instance count: 10. Limit:
4
-- The request is rejected because test_rule 2 is violated. The number of instances that match this
rule is 10. However, this rule specifies that a maximum of 4 instances can run at the same time.
```

Delete throttling rules

You can leave THROTTLING_RULES empty to delete the throttling rules.

```
information_schema_dev> setproject THROTTLING_RULES=[];
```

Usage notes

For ease of description, rules can be written in the following format: `/project/user/empid? limit=x`. Wildcards are allowed in rules. If a rule with wildcards matches a throttling rule, a rule priority issue may occur.

In this case, the throttling rule has a higher priority than the rule with wildcards. The rule priority issue occurs only if a rule with wildcards matches a throttling rule. In other cases, the rule with wildcards and the throttling rule take effect at the same time.

Example 1: A throttling rule takes effect.

- Rule 1: /meta_dev/*? limit=10
- Rule 2: /meta_dev/123456789? limit=100

In this example, Rule 1 matches Rule 2. Rule 2 is a throttling rule and takes effect. Rule 1 is a rule with wildcards and does not take effect. The user with the ID of 123456789 in the meta_dev project can run a maximum of 100 instances at the same time. Other users can run a maximum of 10 instances at the same time.

Example 2: Two throttling rules take effect at the same time.

- Rule 3: /meta_dev? limit=10
- Rule 4: /meta_dev/123456789? limit=100

In this example, the two rules take effect at the same time. The two rules do not contain wildcards. Therefore, no rule priority issues occur. The user with the ID of 123456789 in the meta_dev project can run a maximum of 10 instances at the same time. This is because Rule 3 indicates that the total number of instances that all users in the meta_dev project can run at the same time is 10. This rule applies even if the user with the ID of 123456789 is allowed to run 100 instances at the same time.

Example 3: A throttling rule and a rule with wildcards take effect at the same time.

- Rule 5: /meta_dev/*? limit=10
- Rule 6: /meta_dev/123456789/111111? limit=100

In this example, the two rules take effect at the same time. Rule 5 is a rule with wildcards but does not match Rule 6. This is because a rule with wildcards matches only a throttling rule at the same directory level. Therefore, no rule priority issues occur.

1.22. Frequently-used tools

1.22.1. MaxCompute console

1.22.1.1. Usage notes

This topic provides usage notes for the MaxCompute client.

- Do not parse data based on the output format of the MaxCompute client. The client output format may not be forward compatible. The command syntax and behavior vary based on client versions.

- The MaxCompute client is a Java program. It requires JRE to run. You must download and install JRE 1.8 to run the MaxCompute client.
- Before you configure the client, make sure that a project is created by using an Alibaba Cloud account and the AccessKey ID and AccessKey secret of the account are obtained.
- For more information about how to use the MaxCompute client, see Configure the client.

1.22.1.2. Install the MaxCompute client

This topic describes how to install the MaxCompute client.

1. Download the client package to your computer.
2. Decompress the package to the folder where you want to store the client. The package contains the following folders:

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

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

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

   - Set access_id to the AccessKey ID of your Alibaba Cloud account and access_key to the AccessKey secret of your Alibaba Cloud account.
   - project_name=my_project specifies the project that you want to access. This is the default project that you access each time you log on to the client. If project_name 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.
4. After the modification, run *./bin/odpscmd* in the Linux operating system or *./bin/odpscmd.bat* in the Windows operating system to execute SQL statements. Example:

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

1.22.1.3. Configuration-related operations

This topic describes the configuration operations and related parameters of the MaxCompute client.

View the help information

Run the following command to view the help information about the client:

```
odps@ >./bin/odpscmd -h;
```

> **Note** You can also type `h;` or `help;` (not case-sensitive) in interactive mode.

Specify startup parameters

Run the following command to specify 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 begining 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 of specifying the -f parameter

1. Prepare the script.txt file that is stored in D:/. This file contains the following content:

```
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. View the output information. Sample output:

```
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&tokenRnlrSzJoL242YW43dFFIc1dmb1ZWZzFxQ1RFPSxPRFBTX09CTzoxMDcwMDI1NjI3ODA1 NjI5LDE0MzM0MjA2NzMseyJTdGF0ZW1lbnQiOlt7IkFjdGlvbiI6WyJvZHBzOlJlYWQiXSwiRWZmZWN0IjoiQWxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnB yb2plY3RzL29kcHNfcHVibGljX2Rld i9pbnN0YW5jZXMvMjAxNTA1MjgxMjI0MzI5MDZndXg3N2lvMyJdfV0sIlZlcnNpb24iOiIxIn0=
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&tokendSt0RXdlV0M5YjZET2I1MnJuUFkzWDN1aWpzPSxPRFBTX09CTzoxMDcwMDI1NjI3ODA1NjI5LDE0MzM0MjA2ODAseyJTdGF0ZW1lbnQiOlt7IkFjdGlvbiI6WyJvZHBzOlJlYWQiXSwiRWZmZWN0IjoiQWxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnB yb2plY3RzL29kcHNfcHVibGljX2Rld i9pbnN0YW5jZXMvMjAxNTA1MjgxMjI0MzkzMThnY21razZ1MSJdfV0sIlZlcnNpb24iOiIxIn0=
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&tokenNWlwL0EvQThxUXhzcTRERDc5NFg0b2IxZ3QwPSxPRFBTX09CTzoxMDcwMDI1NjI3OD A1NjI5LDE0MzM0MjA2ODAseyJTdGF0ZW1lbnQiOlt7IkFjdGlvbiI6WyJvZHBzOlJlYWQiXSwiRWZmZWN0IjoiQWxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnB yb2plY3RzL29kcHNfcHVibGljX2Rld i9pbnN0YW5jZXMvMjAxNTA1MjgxMjI0NDAzODlnOThjbWxtZiJdfV0sIlZlcnNpb24iOiIxIn0=
OK
```

Enter the interactive mode

You can enter the interactive mode immediately after you run the following command on the MaxCompute client.

```
[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 that is terminated by a semicolon (;) at the cursor, and press Enter to run the command.

Specify the output format

The output of an SQL statement is in the human-readable or machine-readable format. Human-readable is the default format. If you use the -M parameter when you run odpscmd, the output format is CSV.

Note This feature applies only to SELECT and READ statements and takes effect during data reading.

Specify the statements to execute

If you specify the -e or -f parameter and want to start with an intermediate statement among a few statements, you can specify the -k parameter. The -k parameter indicates that the execution starts from the specified statement and the preceding statements are skipped. If the value of the -k parameter is less than or equal to 0, the execution starts from the first statement. A statement terminated by a semicolon (;) is considered valid. At runtime, the output information indicates the specific statement that is executed or failed.

For example, the dual.sql file in the *tmp* folder contains the following 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 skip the first two statements:

```
odpscmd -k 3 -f dual.sql
```

Obtain information about the current logon user

Run a command to obtain the Alibaba Cloud account of the current logon user and the endpoint that is used. Command syntax:

```
whoami
```

Sample command:

```
odps@ hiveut>whoami; Name: odpstest@aliyun.com ID: 1090142773636588 End_Point: <Endpoint of MaxCompute> Project: lijunsecuritytest
```

Exit the MaxCompute client

Command syntax:

```
odps@ > quit;
```

You can also run the following command:

```
q;
```

Configure a job priority

Command syntax:

```
Admin@ > ./bin/odpscmd --instance-priority=<PRIORITY>;
```

Configuration file: odps_config.ini

```
instance_priority=<PRIORITY>
```

> **Notice**
>
> - <PRIORITY>
> - The priority setting in the configuration file applies to all instances submitted in the MaxCompute client.
> - If you do not configure the priority for a job, its priority is 9.

Dry run mode

In dry run mode, MaxCompute parses an SQL statement to check whether the syntax of this statement is correct and generates an execution plan. In this mode, MaxCompute does not submit a distributed job. Command syntax:

```
./bin/odpscmd -y
```

SQL reliability

If you execute statements, such as INSERT or CREATE TABLE AS, on the MaxCompute client and an exception occurs when you run a job, the MaxCompute client automatically restores data and metadata to the status before the SQL query is run based on the known information. The exceptions include:

- Data that is overwritten by using the INSERT OVERWRITE statement during queries is restored from the temporary backup directory to the original directory.
- Data that is generated by using the INSERT INTO statement during queries is deleted.
- Tables that are created during queries or the dynamically generated partition information that does not exist before queries are deleted.

If MaxCompute fails to restore data, the following error code is returned. The error code informs you that further attempts may cause some data unrecoverable. Error:

```
ODPS-
0110999: Critical! Internal error happened in commit operation and rollback failed, possible breach
of atomicity
```

1.22.2. Eclipse development plugin

1.22.2.1. Install the Eclipse development plug-in

This topic describes how to install the Eclipse development plug-in.

Context

MaxCompute provides the Eclipse development plug-in to help you use SDKs for Java to develop MapReduce programs or user-defined functions (UDFs). This plug-in can simulate the running process of MapReduce programs or UDFs. It provides local debugging methods and features to generate templates. You can click Eclipse to download the plug-in package.

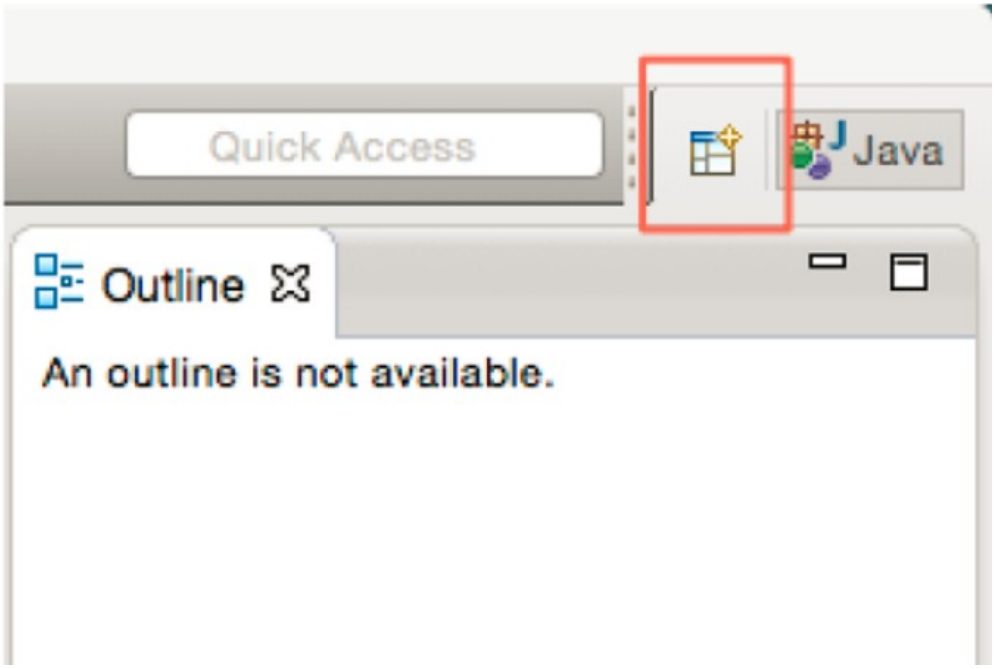
> **Note** Unlike the MapReduce program that runs in local mode, the Eclipse development plug-in cannot synchronize data with MaxCompute. You must manually copy the required data to the *warehouse* directory of the Eclipse development plug-in.

Procedure

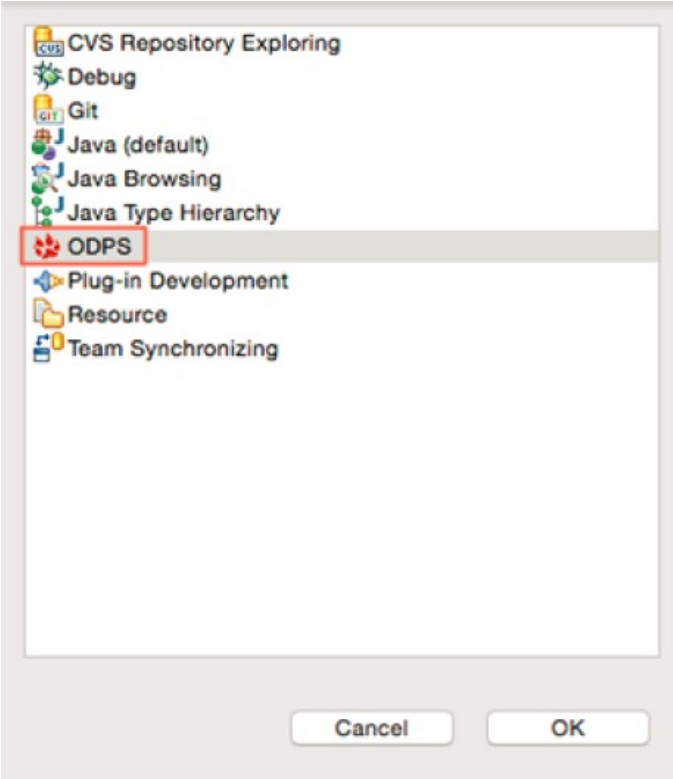
1. Decompress the Eclipse package to view the following JAR file in the package:

   odps-eclipse-plugin-bundle-0.16.0.jar

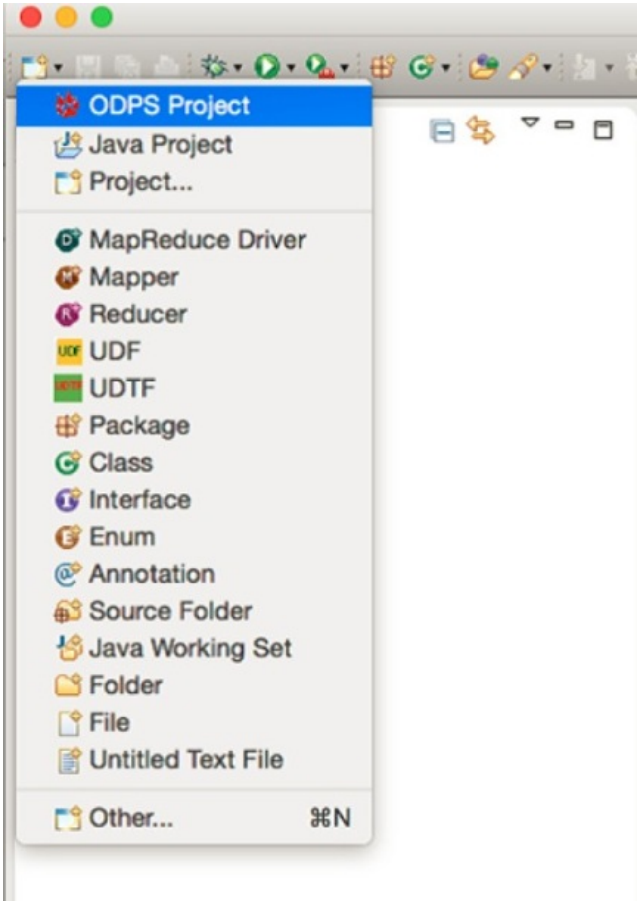
2. Place the JAR file in the *plugins* folder under the Eclipse installation directory.
3. Start the Eclipse development plug-in and click the **Open Perspective** icon in the upper-right corner, as shown in the following figure.



4. In the dialog box that appears, click **ODPS** and then **OK**.



5. In the navigation bar that appears, select the **ODPS project** and 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.



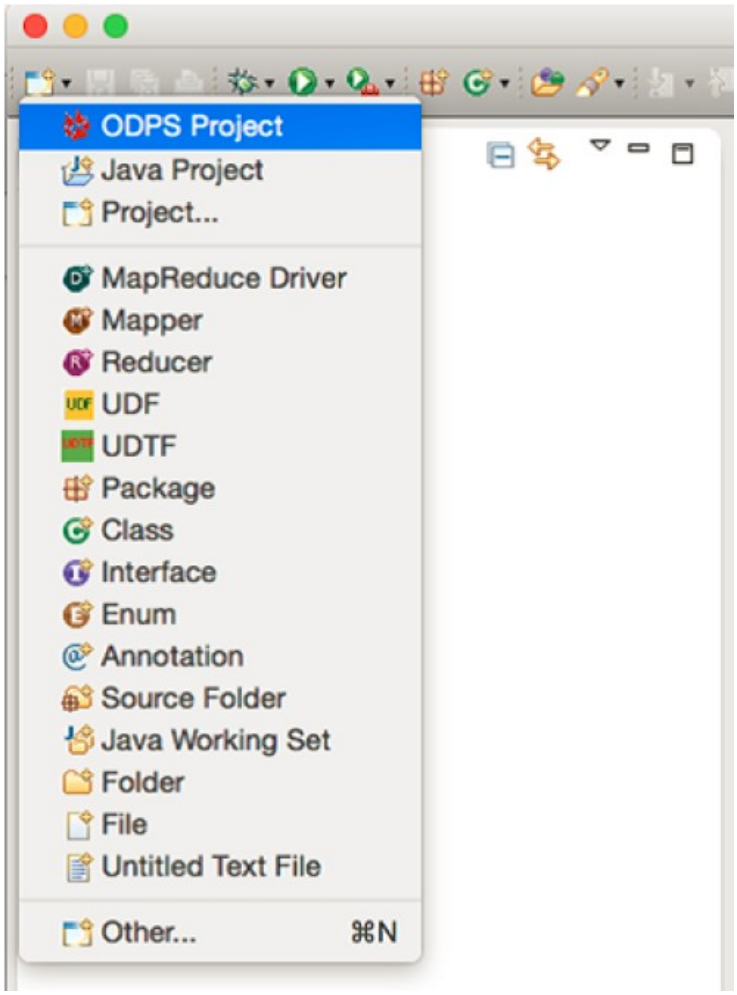
1.22.2.2. Create a project

1.22.2.2.1. Method 1

This topic describes Method 1. This method is used to create a project in the Eclipse development plug-in.

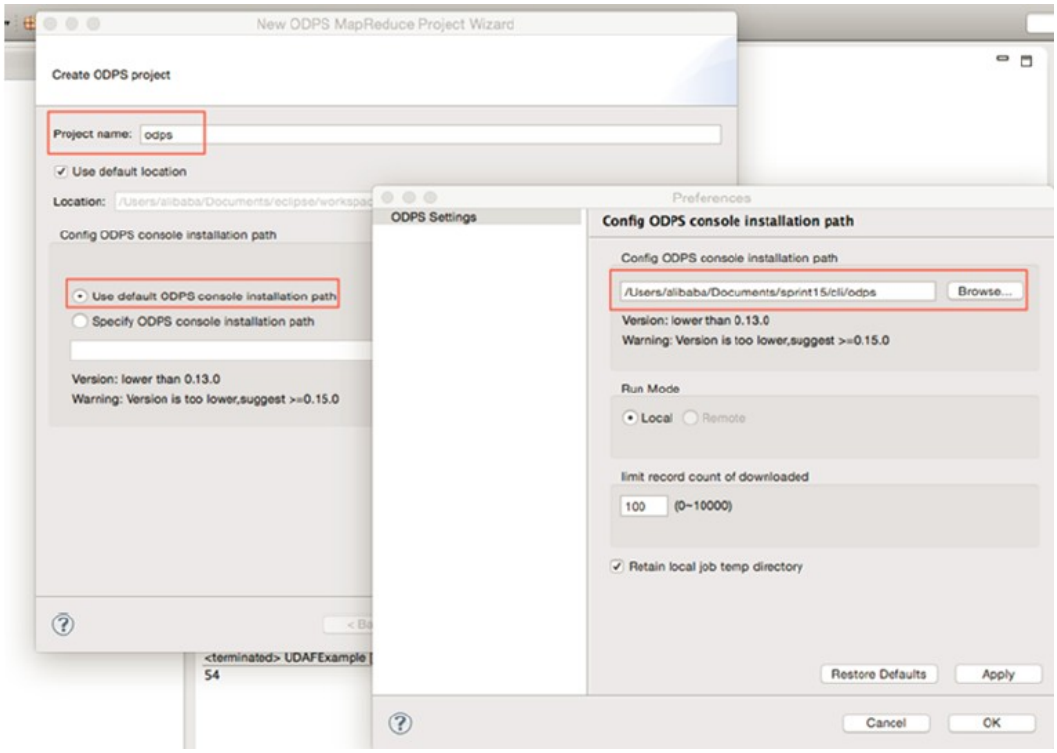
Procedure

1. In the upper-left corner of Eclipse, choose **File > New > Project > ODPS > ODPS Project** , as shown in the following figure.



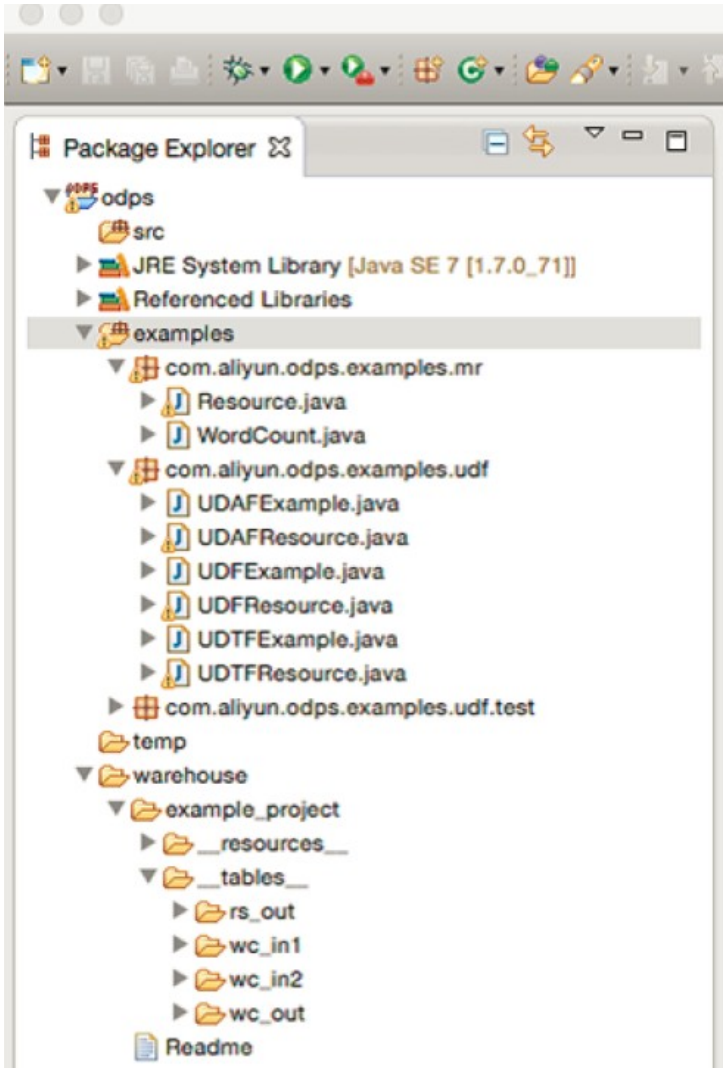
> **Note** In this example, the project name is ODPS.

2. In the dialog box shown in the following figure, enter the project name, select the installation path of the MaxCompute client, and then click **Finish**.



> **Note** The client must be installed in advance.

3. After you create a project, you can view the directory structure on the Package Explorer tab, as shown in the following figure.

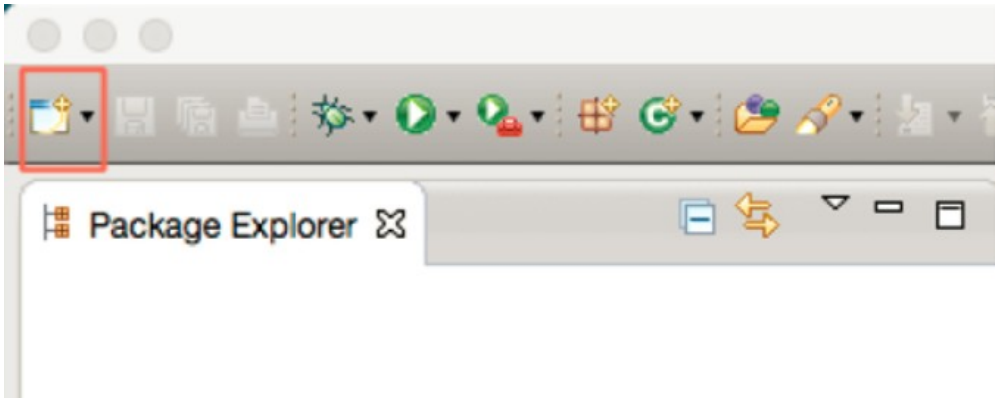


1.22.2.2.2. Method 2

This topic describes another method to create a project by using the Eclipse plug-in.

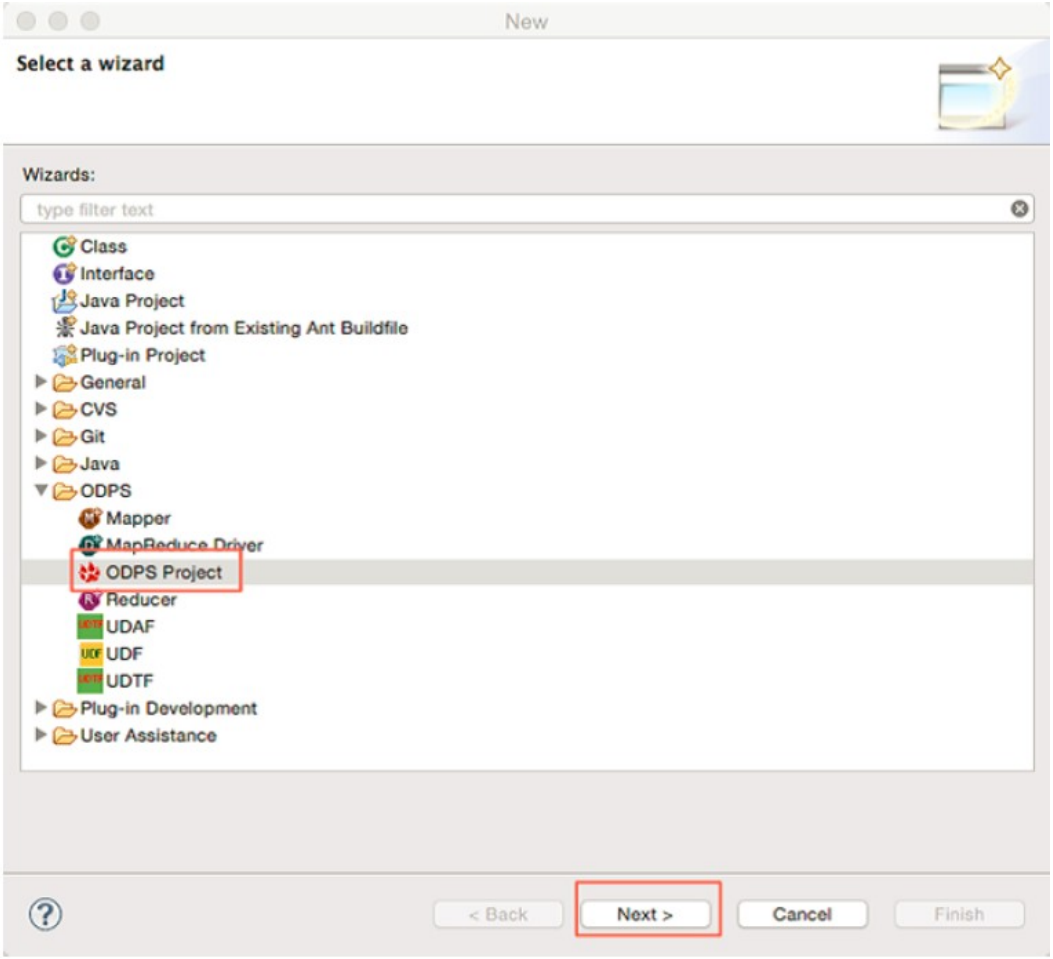
Procedure

1. In the toolbar of the Eclipse plug-in, click the **New** icon, as shown in the following figure.



2. In the dialog box that appears, unfold ODPS, click **ODPS Project**, and then click **Next**, as shown in the

following figure.



> **Note** In this example, the project name is ODPS.

3. The subsequent steps are the same as those in method 1. After you install the Eclipse plug-in, you can use it to compile a MapReduce program or user-defined function (UDF).

> **Note** For more information about how to run a MapReduce program by using the Eclipse plug-in, see MapReduce running example. For more information about how to develop and run a UDF, see UDF development and running example.

1.22.2.3. MapReduce running example

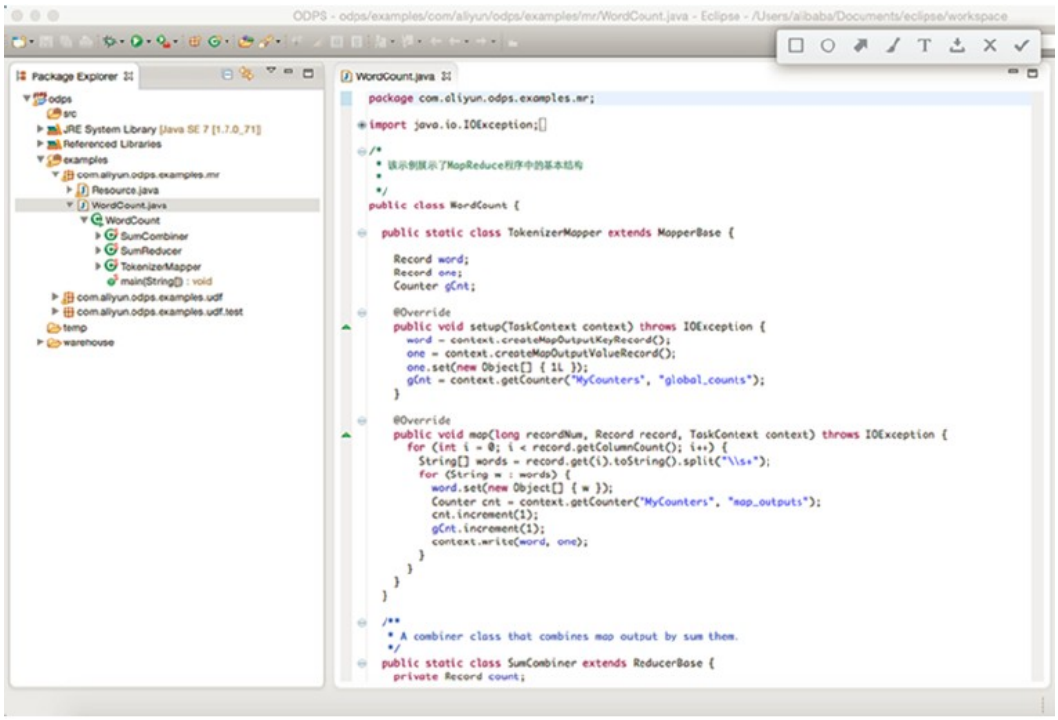
1.22.2.3.1. Run a WordCount program

This topic describes how to use MapReduce to run a WordCount program in the Eclipse development plug-in.

Procedure

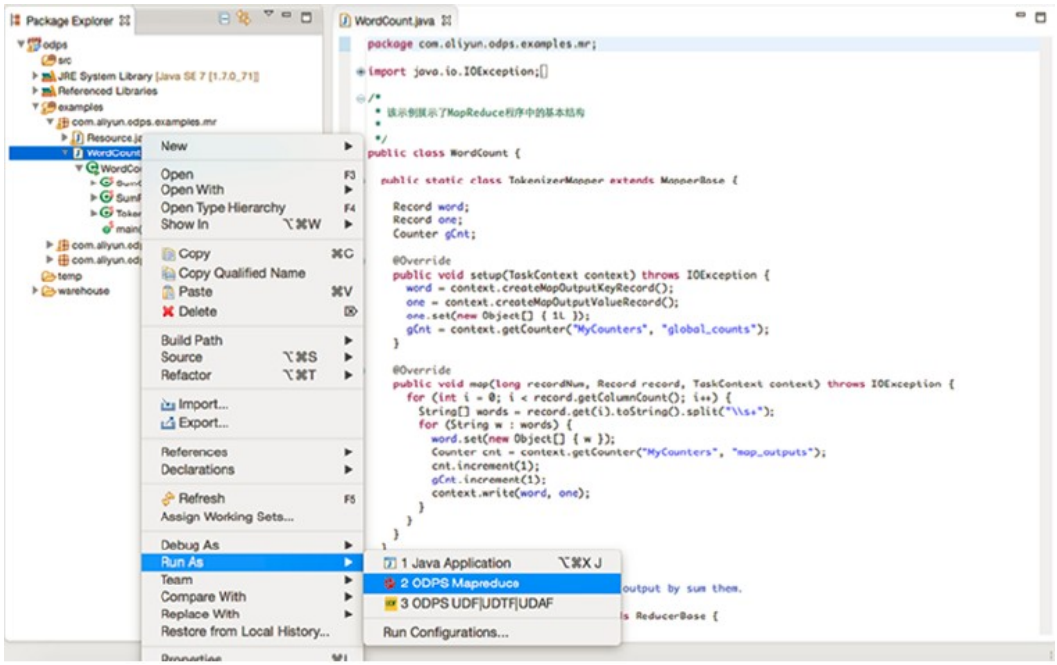
1. On the Package Explorer tab of the Eclipse development plug-in, choose Examples > com.aliyun.odps.examples.mr > WordCount.java from the navigation tree, as shown in the following figure.

   Choose the WordCount program



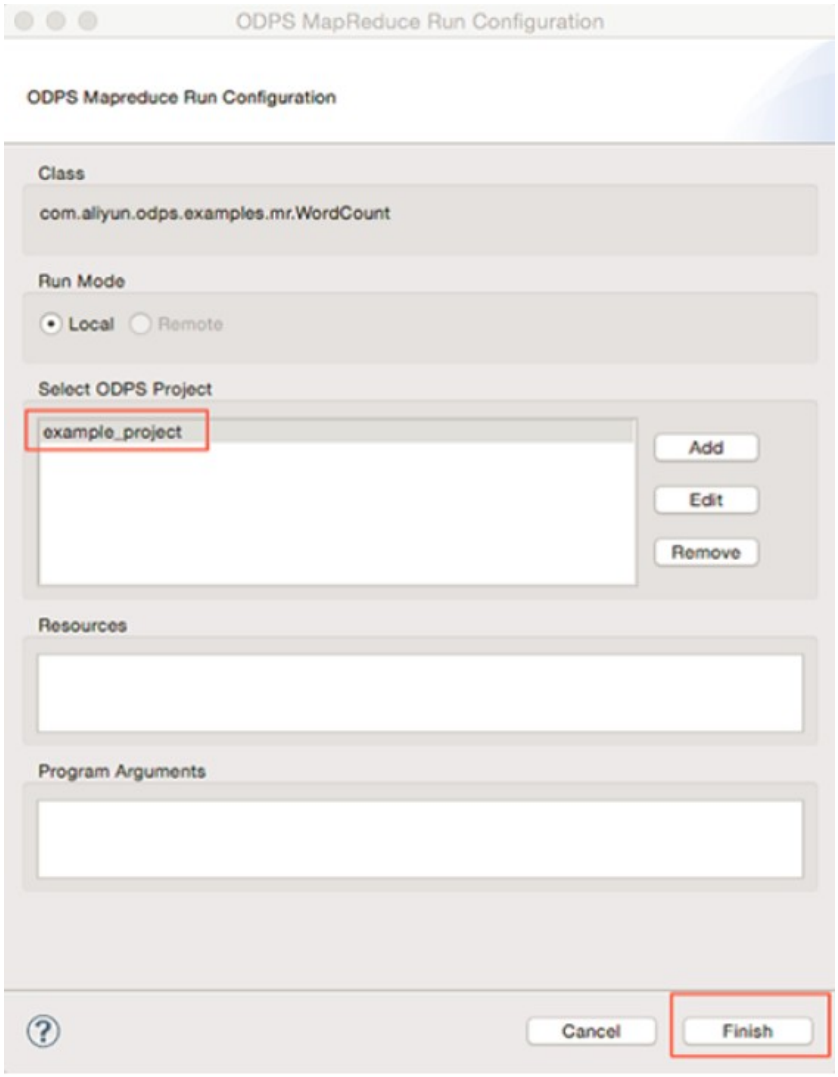
2. Right-click the **WordCount.java** program and choose **Run As2 ODPS Mapreduce**, as shown in the following figure.

   Run the WordCount program (1)



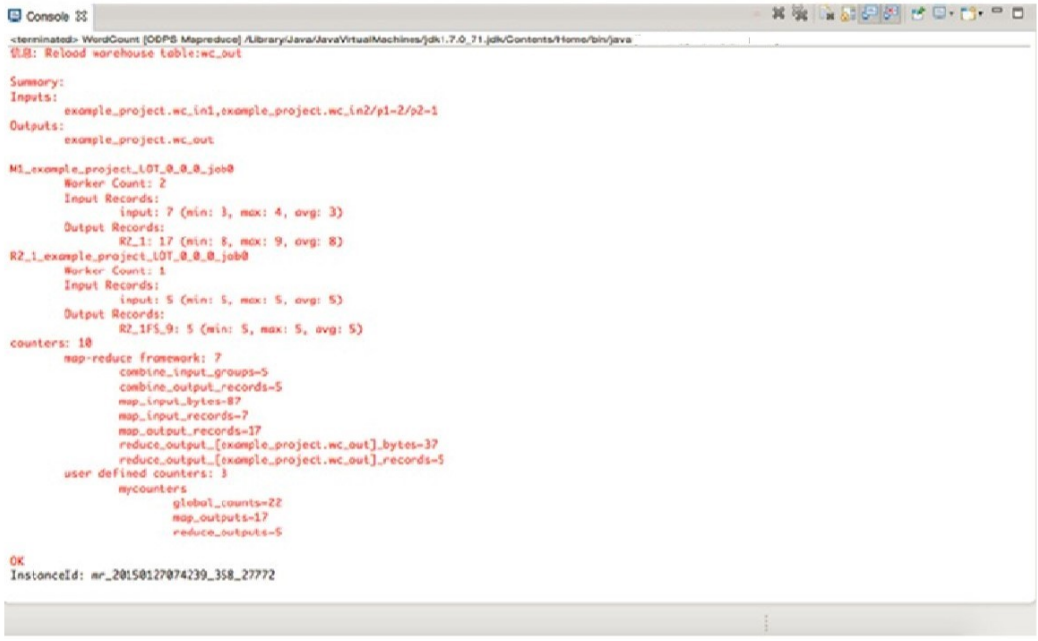
3. In the ODPS MapReduce Run Configuration dialog box, add **example_project** to the Select ODPS Project section and click **Finish**, as shown in the following figure.

   Run the WordCount program (2)



4. After you run the WordCount program, the result is displayed, as shown in the following figure.

   Execution result



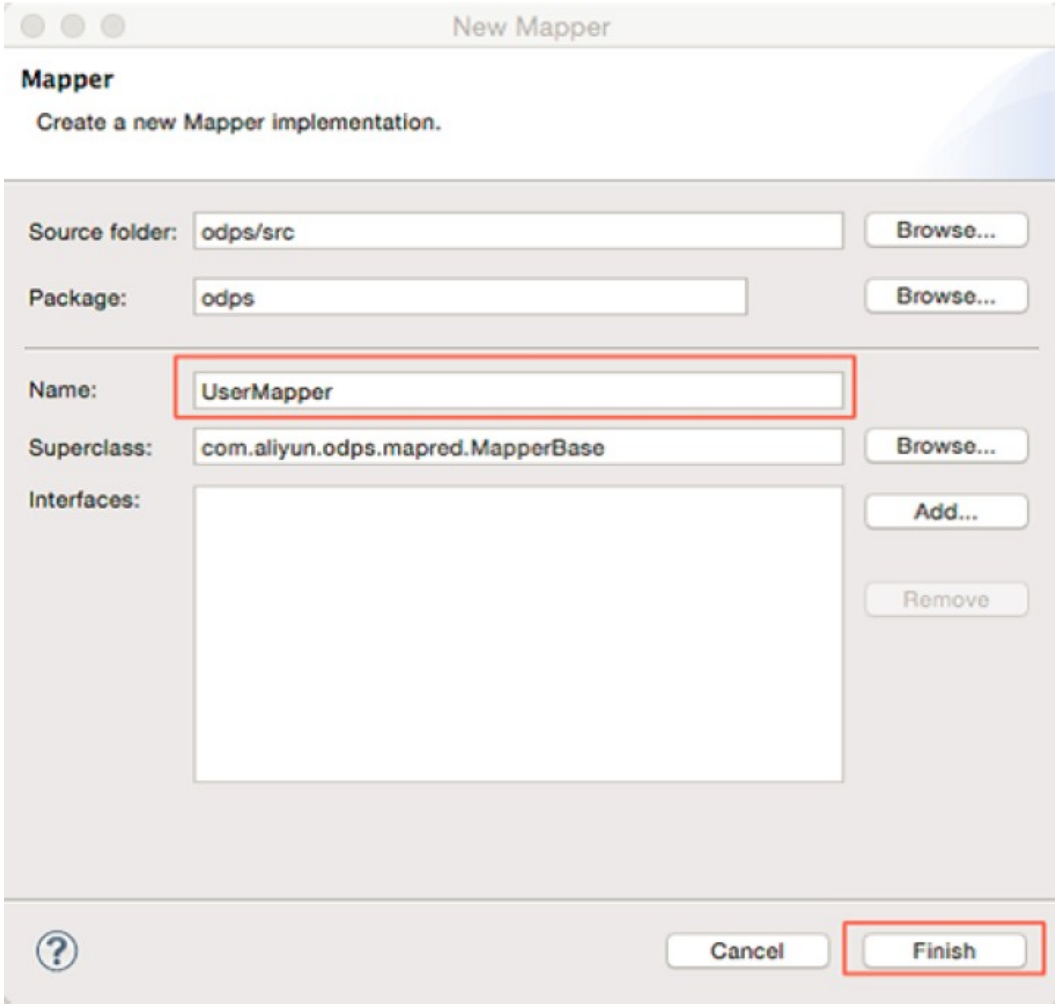
1.22.2.3.2. Run a custom MapReduce program

This topic provides an example on how to run a custom MapReduce program by using the Eclipse development plug-in.

Procedure

1. On the Package Explorer tab of the Eclipse development plug-in, choose odps > src from the navigation tree. Right-click **src** and choose **New > Mapper**.
2. In the New Mapper dialog box, enter the name of the Mapper class and click **Finish**, as shown in the following figure.

   New Mapper dialog box

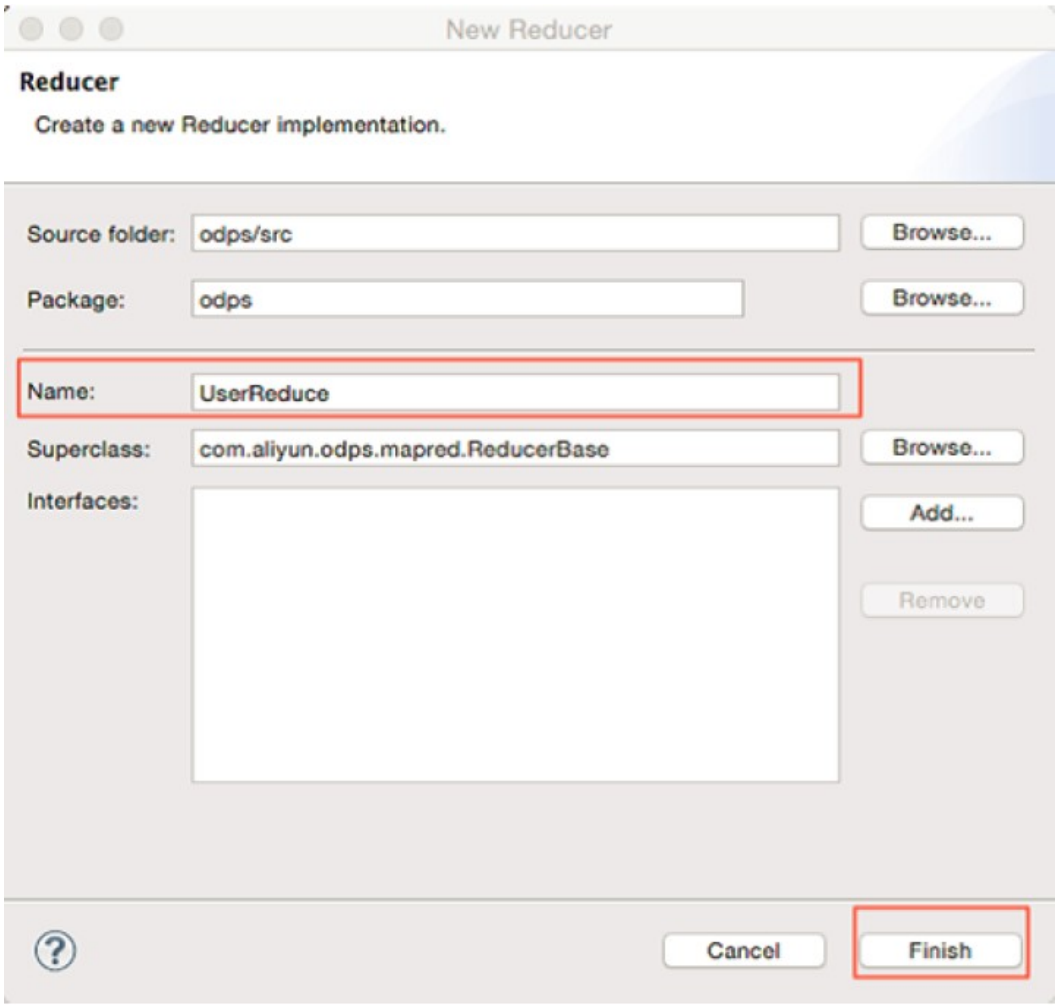


3. The UserMapper.java file is generated in the src folder shown in the navigation tree on the Package Explorer tab. The file is a Mapper class template. By default, the package name in the template is odps. The template contains the following information:

```
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(n
ew 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().sp
lit("\\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. On the Package Explorer tab of the Eclipse development plug-in, choose odps > src from the navigation tree. Right-click src and choose **New > Reduce**.
5. In the New Reducer dialog box, enter the name of the Reducer class and click **Finish**, as shown in the following figure.

   Reducer



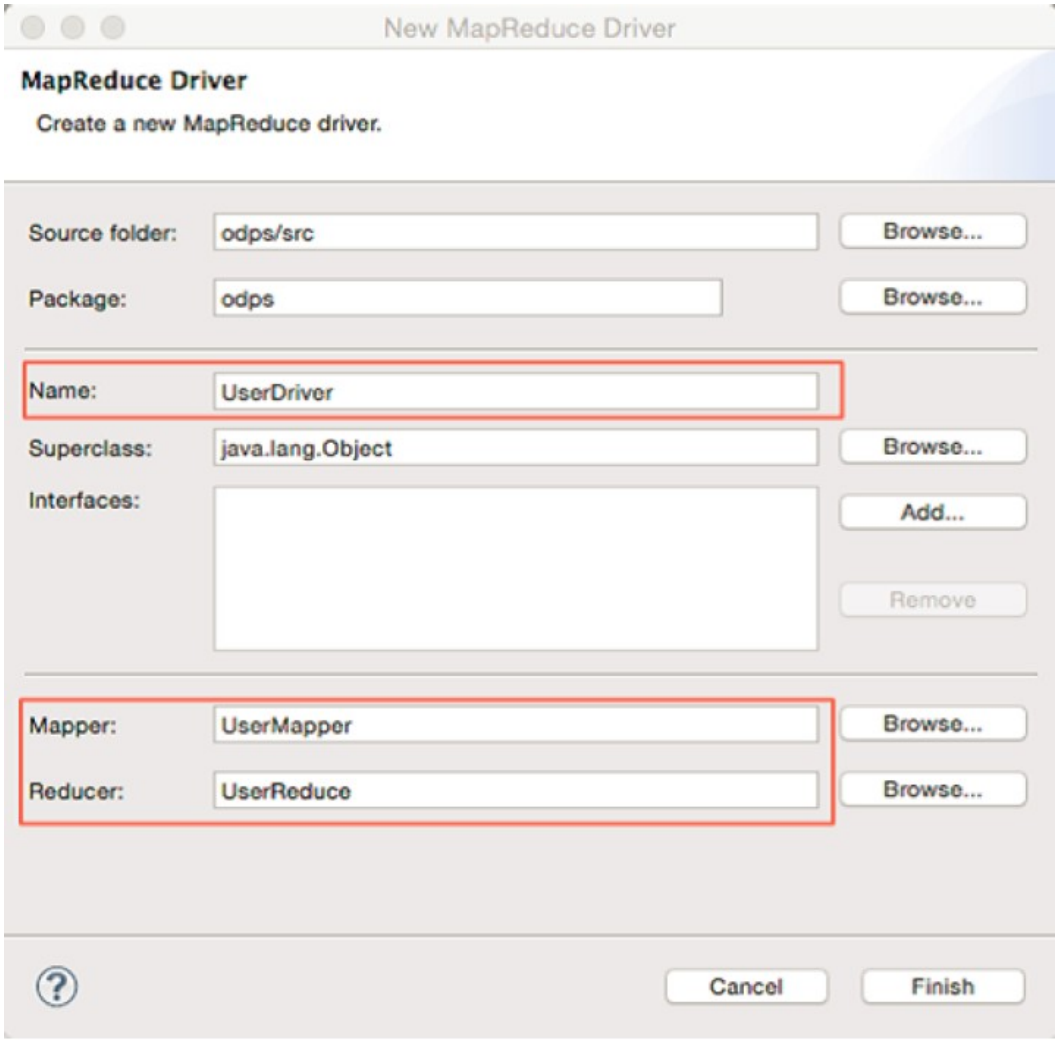
Note In this example, the name of the Reducer class is UserReducer.

6. On the Package Explorer tab, the UserReducer.java file is generated in the src folder. The file is a Reducer class template. By default, the package name in the template is odps. The template contains the following information:

```
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. On the Package Explorer tab of the Eclipse development plug-in, choose odps > src from the navigation tree. Right-click src and choose **New > MapReduce Driver**.
8. In the New MapReduce Driver dialog box, Specify Name, Mapper, and Reducer, and click **Finish**, as shown in the following figure.

   MapReduce Driver

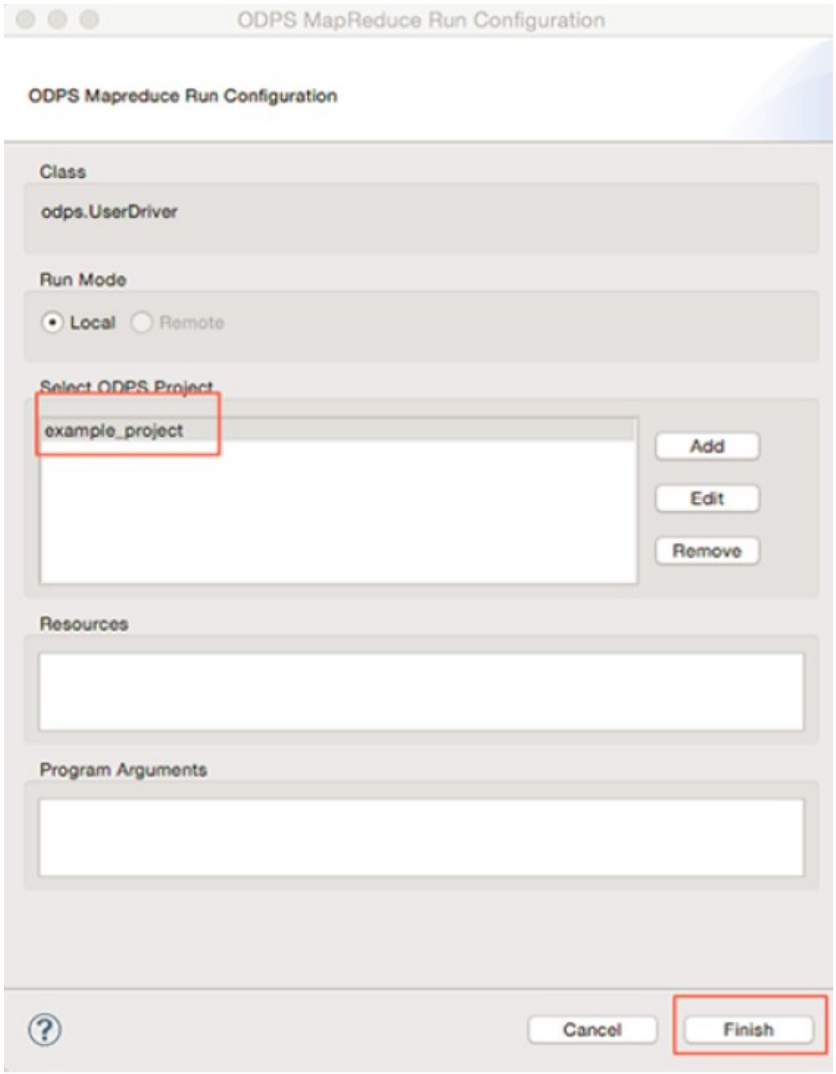


9. On the Package Explorer tab, the MyDriver.java file is generated in the src folder. The file is a MapReduce Driver class template. By default, the package name in the template is odps. The template contains the following information:

```
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();
}
}
```

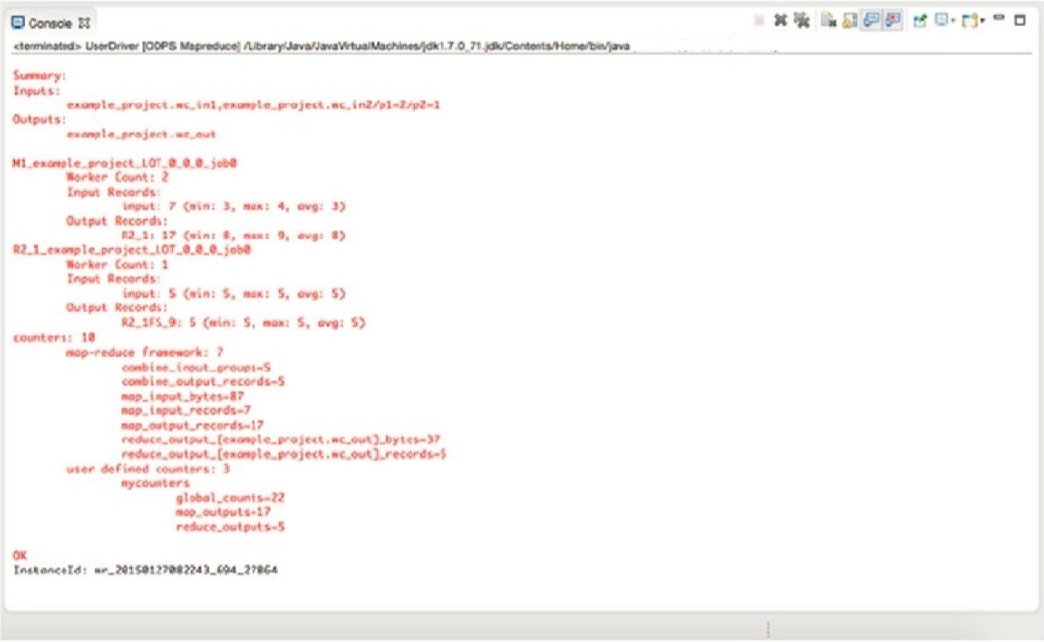
10. Run the MapReduce program. Right-click the UserDriver.java file, choose **Run As > ODPS MapReduce**, and then click **OK**. The ODPS MapReduce Run Configuration dialog box is displayed, as shown in the following figure.

    ODPS MapReduce Run Configuration



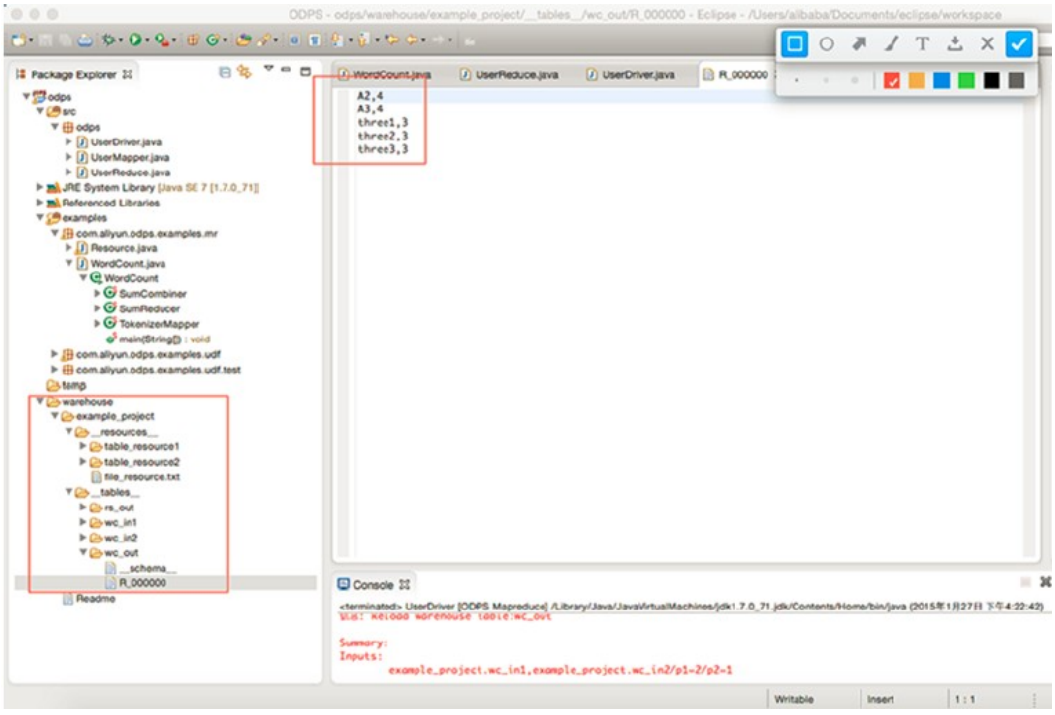
11. Add example_project in Select ODPS Project and click **Finish** to run the MapReduce program in local mode. If the following information is displayed, the MapReduce program properly runs in local mode.

    Console



12. Right-click the *warehouse* folder in the navigation tree on the Package Explorer tab and select Refresh to view the output result, as shown in the following figure.

    Output result

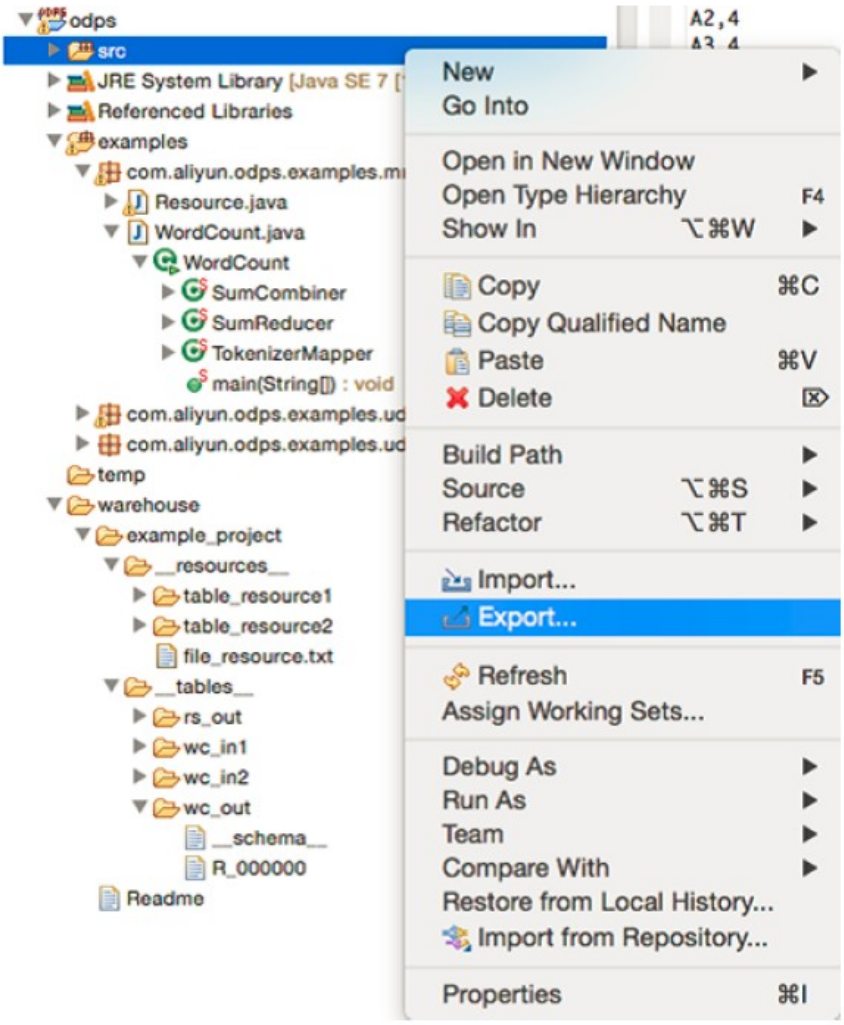


> **Note** wc_out is the output folder, and R_000000 is the result file. After you perform local debugging to verify that the output result is correct, you can use the export feature of the Eclipse development plug-in to package the MapReduce program for future use in a distributed environment.

13. Export the MapReduce program package.

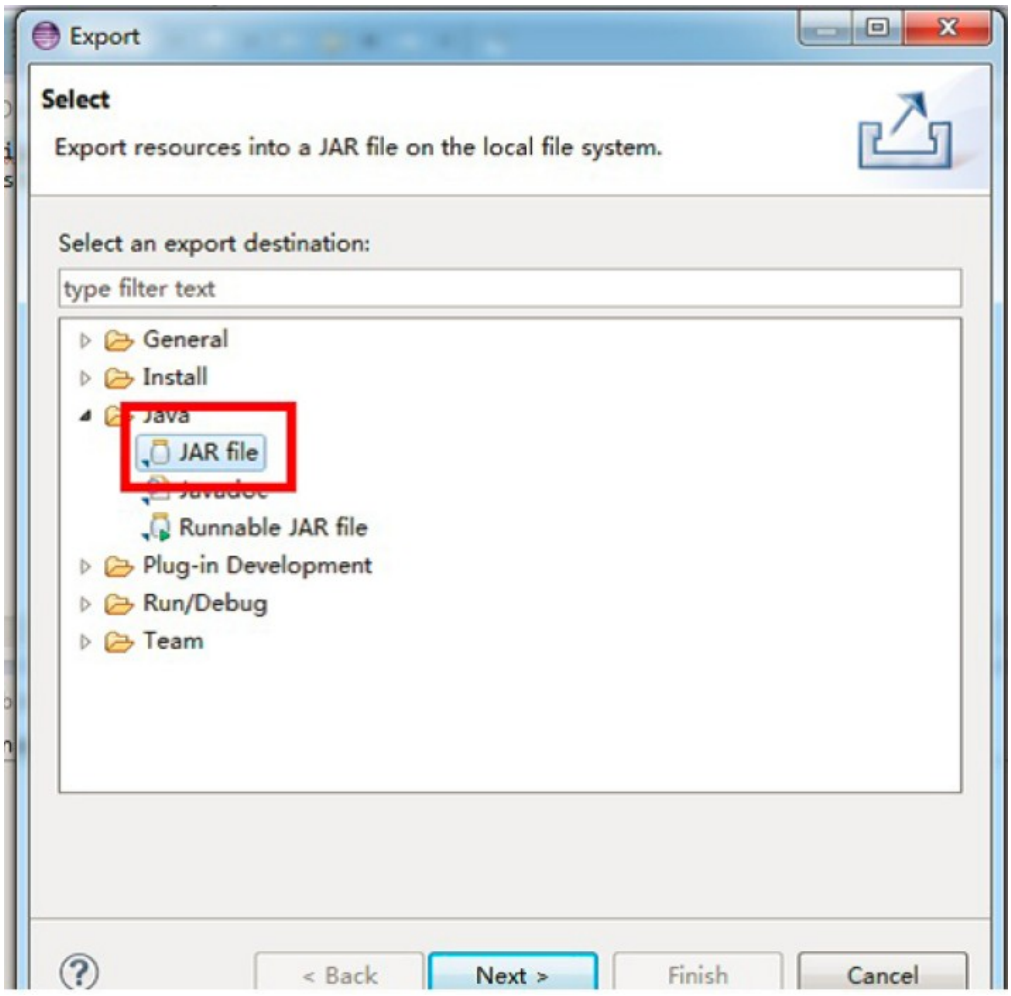
i. On the Package Explorer tab of the Eclipse development plug-in, choose odps > src from the navigation tree. Right-click **src** and select **Export**, as shown in the following figure.

Export dialog box



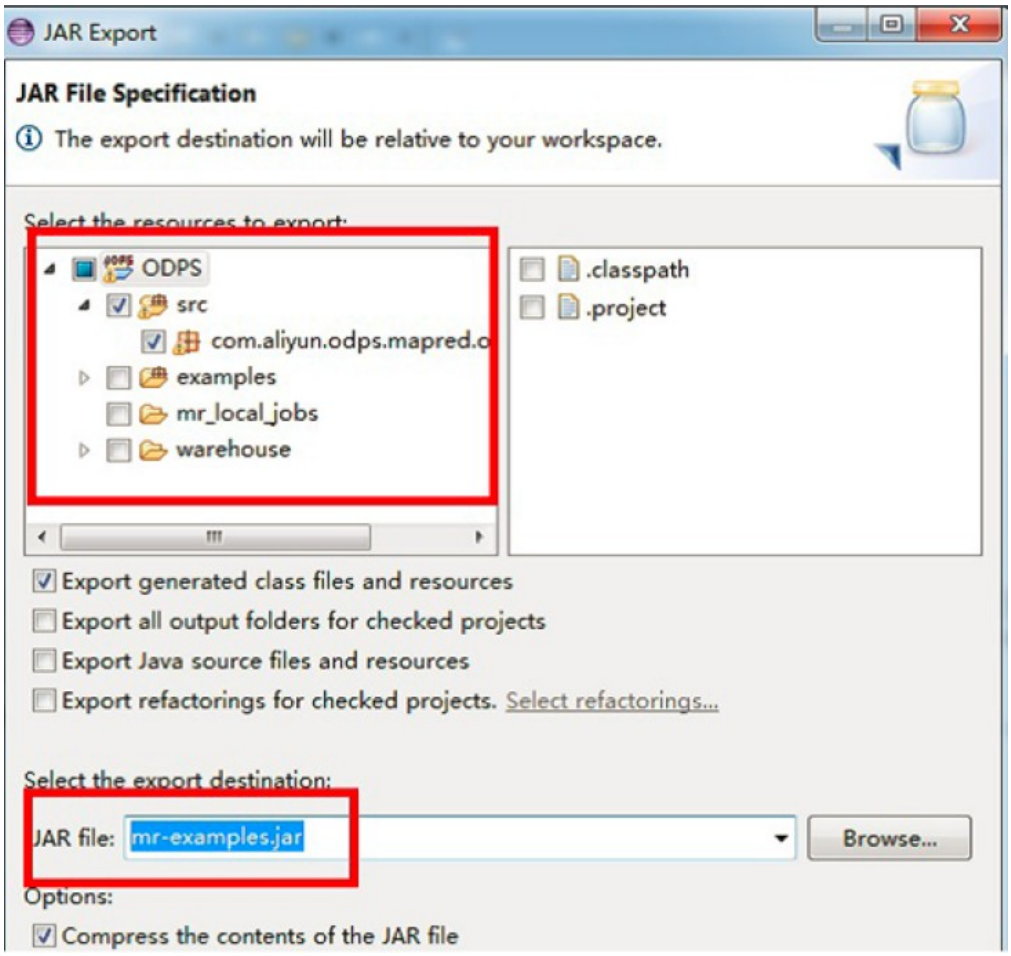
ii. In the Export dialog box, choose Java > JAR file in Select an export destination and click Next, as shown in the following figure.

JAR Export dialog box



iii. In the JAR Export dialog box, choose ODPS > src > com.aliyun.odps.mapred.open.example, and specify the mr-examples.jar file as the MapReduce program package, as shown in the following figure.

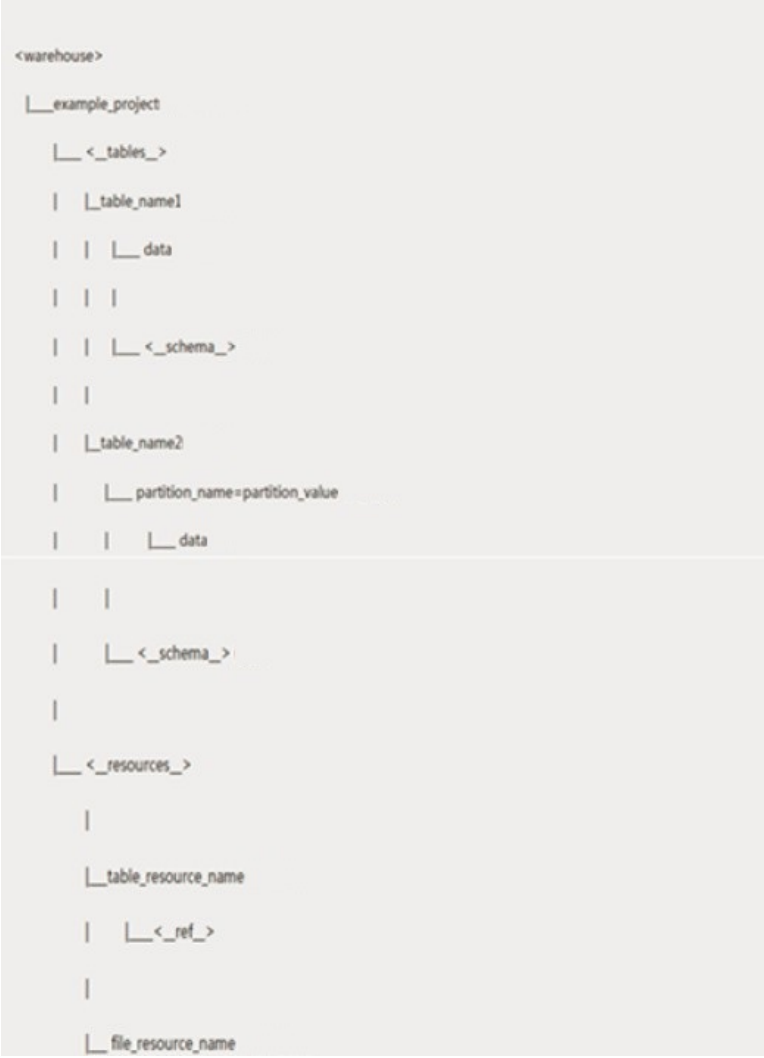
Select the source and destination packages



> **Note** In this example, the name of the MapReduce program package is mr-examples.jar. You can name the package based on your business requirements.

iv. Click Finish. The export process is complete.

14. If you want to create a project, you can create a folder at the same directory level as the example_project folder in the *warehouse* folder. The following figure shows the directory structure.



Examples of information in schema files:

```
project=project_name table=table_name
columns=col1:BIGINT,col2:DOUBLE,col3:BOOLEAN,col4:DATETIME,col5:STRING
-- Information in the schema file of a non-partitioned table:
project=project_name table=table_name
columns=col1:BIGINT,col2:DOUBLE,col3:BOOLEAN,col4:DATETIME,col5:STRING partitions=col1:BIGINT,co
l2:DOUBLE,col3:BOOLEAN,col4:DATETIME,col5:STRING
-- Information in the schema file of a partitioned table:
-- The following data types are supported: BIGINT, DOUBLE, BOOLEAN, DATETIME, and STRING. These
data types correspond to LONG, DOUBLE, BOOLEAN, Java.Util.Date, and Java.Lang.STRING in Java.
```

Example of a data file:

```
1,1.1,true,2015-06-04 11:22:42 896,hello world
\N,\N,\N,\N,\N
-- The time is accurate to milliseconds. \N represents NULL for all data types.
```

> Note
>
> - Before the MapReduce program runs in local mode, the program automatically checks for data tables or resources in the warehouse folder first. If the tables or resources are not found, the program downloads the tables or resources from a server to the *warehouse* folder.
> - After you run the MapReduce program, right-click the *warehouse* folder in the navigation tree on the Package Explorer tab and select Refresh to view the output result.

1.22.2.4. UDF development and running example

1.22.2.4.1. Local debug UDF programs

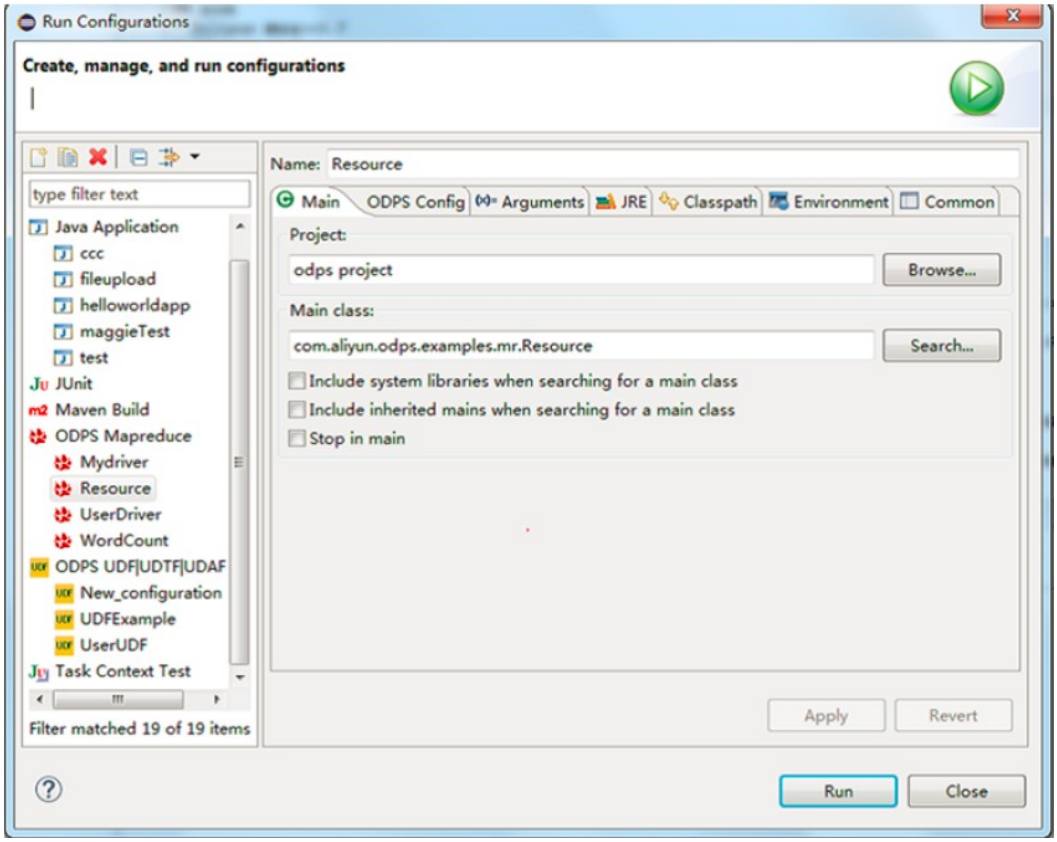
1.22.2.4.1.1. Run a UDF from the menu bar

This topic describes how to run a user-defined function (UDF) from the menu bar of the Eclipse development plug-in.

Procedure

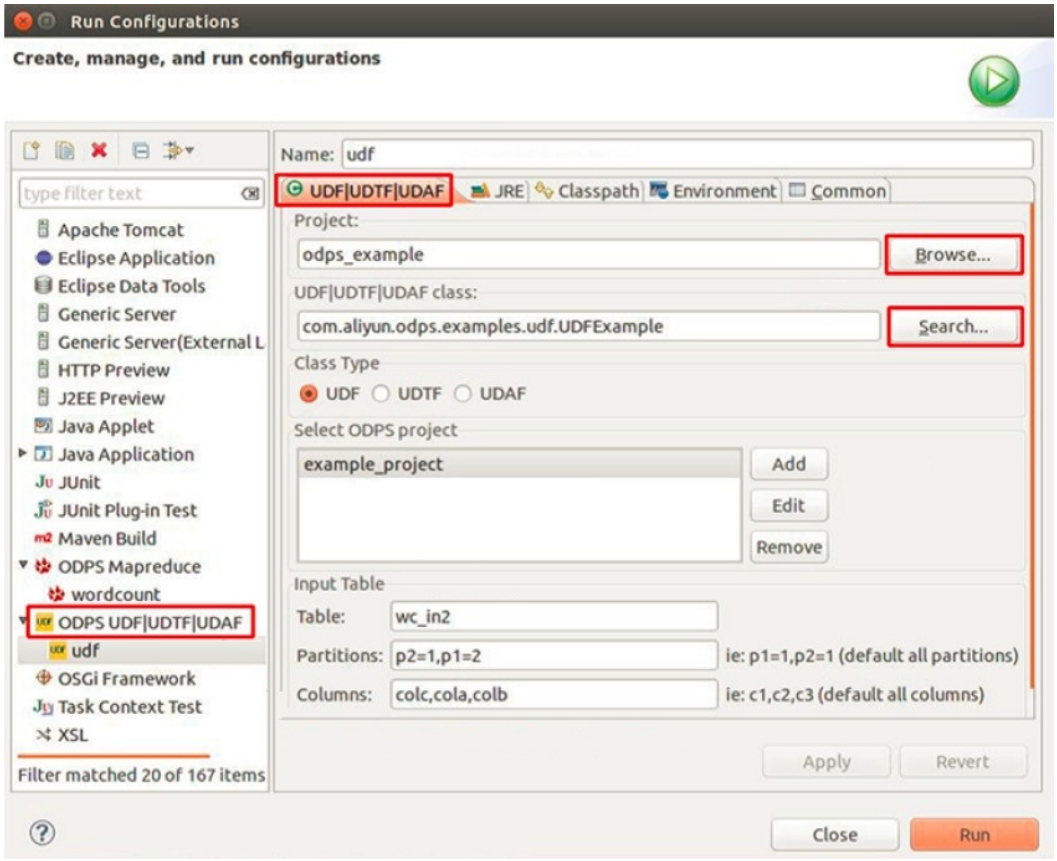
1. Choose **Run > Run Configurations** from the menu bar. The Run Configurations dialog box appears, as shown in the following figure.

   Run Configurations

2. In the left-side navigation tree, right-click ODPS UDF|UDTF|UDAF and select New. On the ODPS UDF|UDTF|UDAF tab, select the project, UDF class, class type, and ODPS project, and enter the table information, as shown in the following figure.

   Run Configurations 2



> **Note** In the preceding figure, the Table parameter indicates the input table of the UDF. The Partitions parameter indicates the partitions from which you want to read data. Partition names are separated by commas (,). The Columns parameter indicates the columns that are passed as parameters of the UDF. Column names are separated by commas (,).

3. Click **Run**. The running result is displayed in the console, as shown in the following figure.

   Console

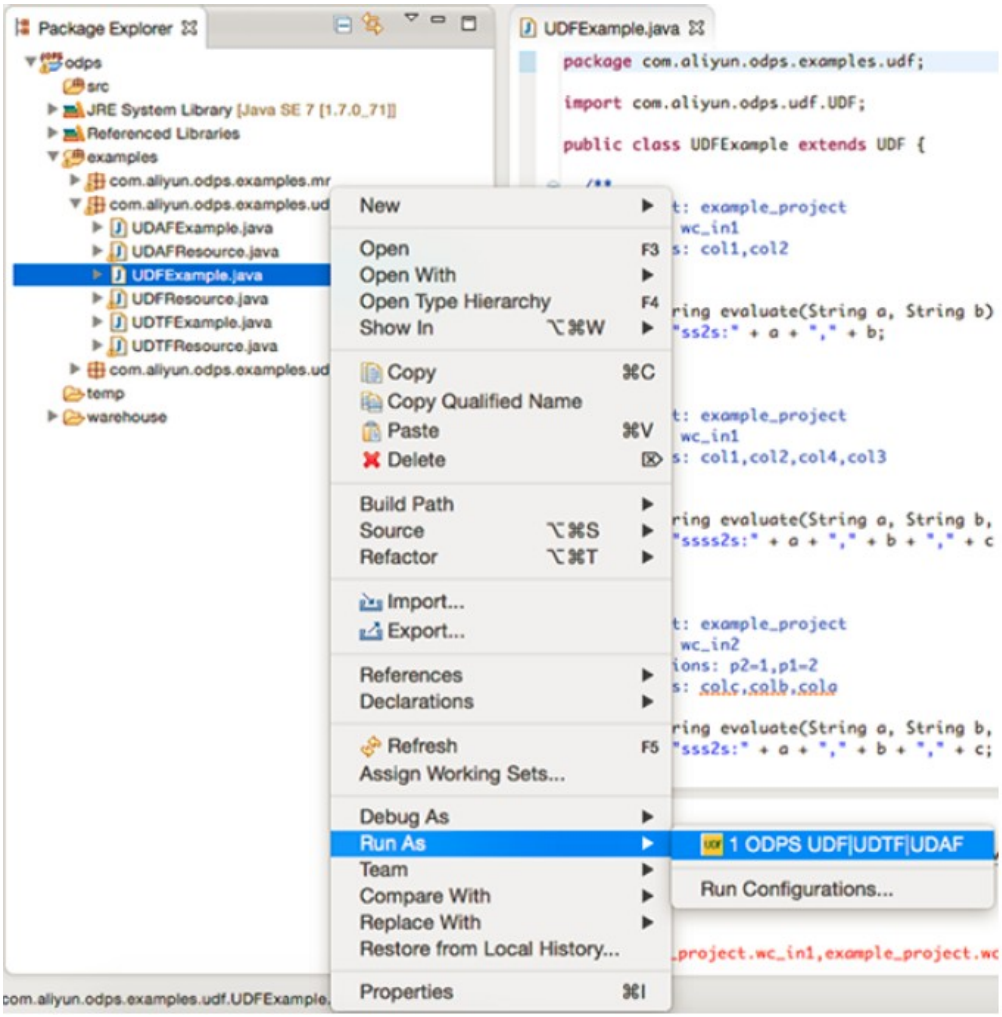
1.22.2.4.1.2. Use the right-click shortcut menu to run a UDF

This topic describes how to use the right-click shortcut menu to run a user-defined function (UDF) in the Eclipse development plug-in.

Procedure

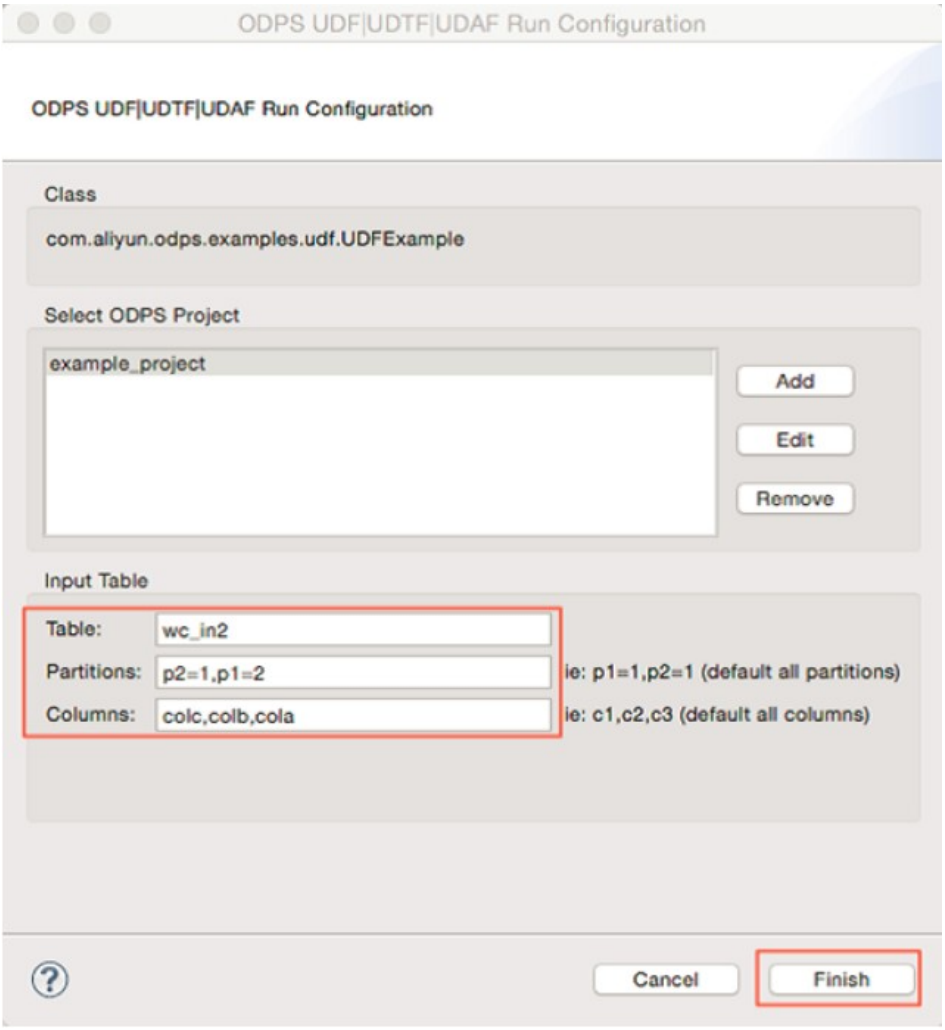
1. On the Package Explorer tab of the Eclipse development plug-in, right-click a Java file in the navigation tree, such as UDFExample.java, and choose **Run As > Run UDF|UDAF|UDTF**, as shown in the following figure.

   Shortcut menu



2. In the ODPS UDF|UDTF|UDAF Run Configuration dialog box, configure related parameters, as shown in the following figure.

   ODPS UDF|UDTF|UDAF Run Configuration dialog box



> **Note** In the ODPS UDF|UDTF|UDAF Run Configuration dialog box, the Table parameter indicates the input table of the UDF. The Partitions parameter indicates the partitions from which you want to read data. Separate multiple partitions with commas (,). The Columns parameter indicates the columns that are passed as the parameters of the UDF. Separate multiple columns with commas (,).

3. Click **Finish** to run the UDF and obtain the output result.

1.22.2.4.2. Run a UDF

This topic describes how to run a user-defined function (UDF) in the Eclipse development plug-in.

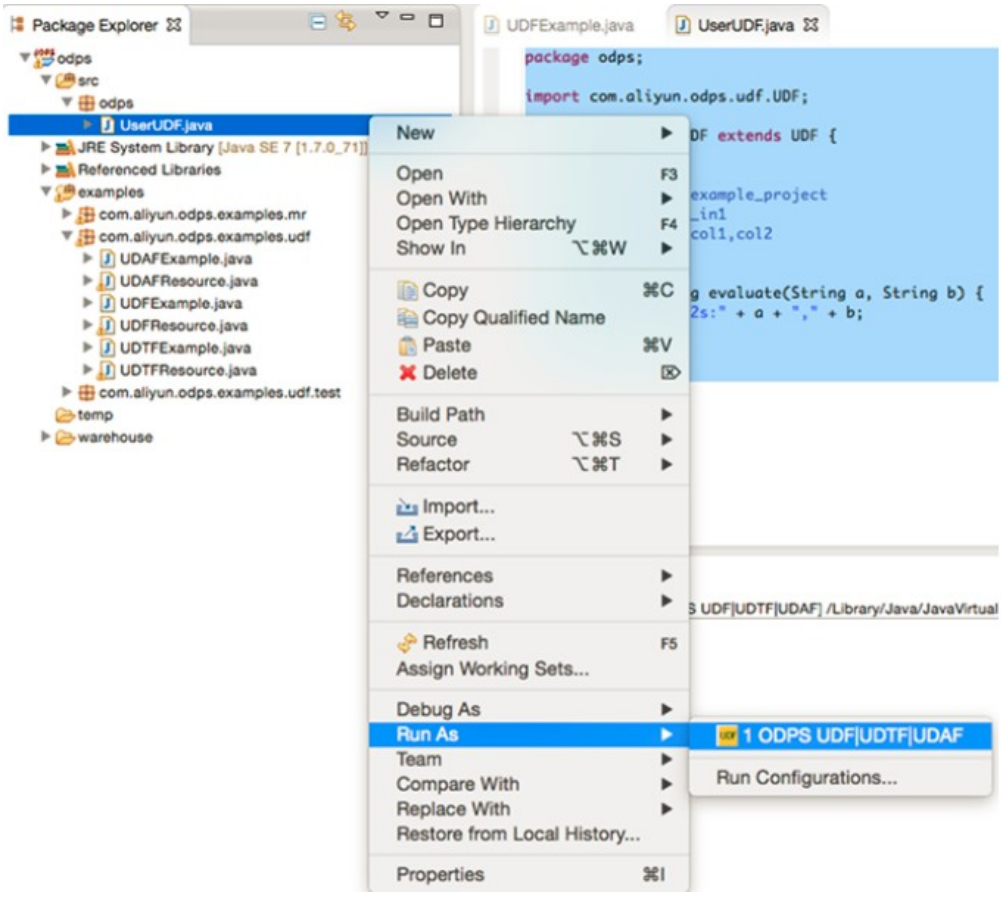
Procedure

1. Right-click a project and choose **New > UDF**. You can also choose **File > New > UDF** from the menu bar. Then, 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. Sample content in the Java file:

```
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;
}
}
```

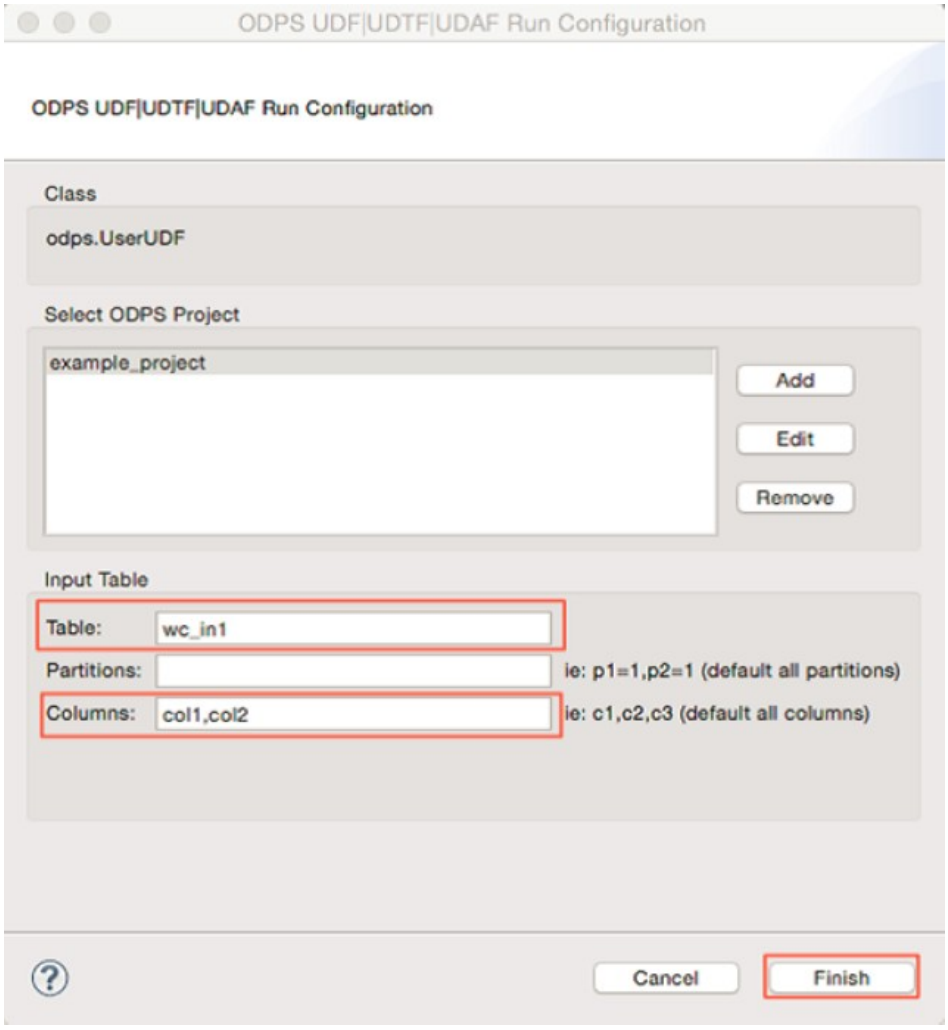
2. Right-click the Java file, such as the UserUDF.java file, and choose **Run As > ODPS UDF|UDTF|UDAF**, as shown in the following figure.

   Run a UDF (1)



3. In the dialog box that appears, configure the parameters shown in the following figure.

   Run a UDF (2)



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. You can use the same method to run a user-defined table-valued function (UDTF).

1.22.2.5. Graph operation example

After you create a MaxCompute project, you can compile your own Graph program and perform the following operations to complete local debugging: This topic provides an example about how to run a Graph.

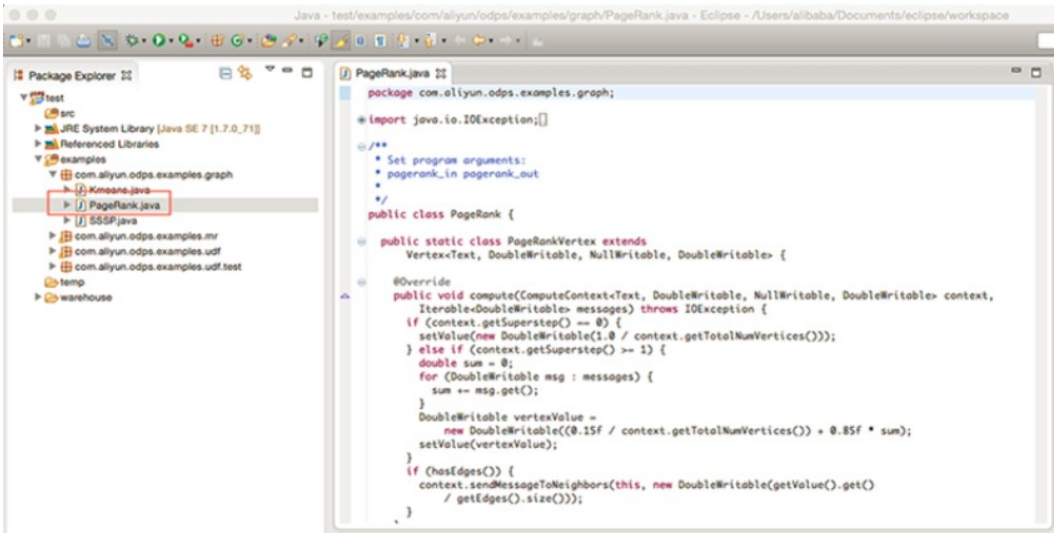
Context

In this example, you can use the PageRank.java file provided by a plug-in to perform local debugging.

Procedure

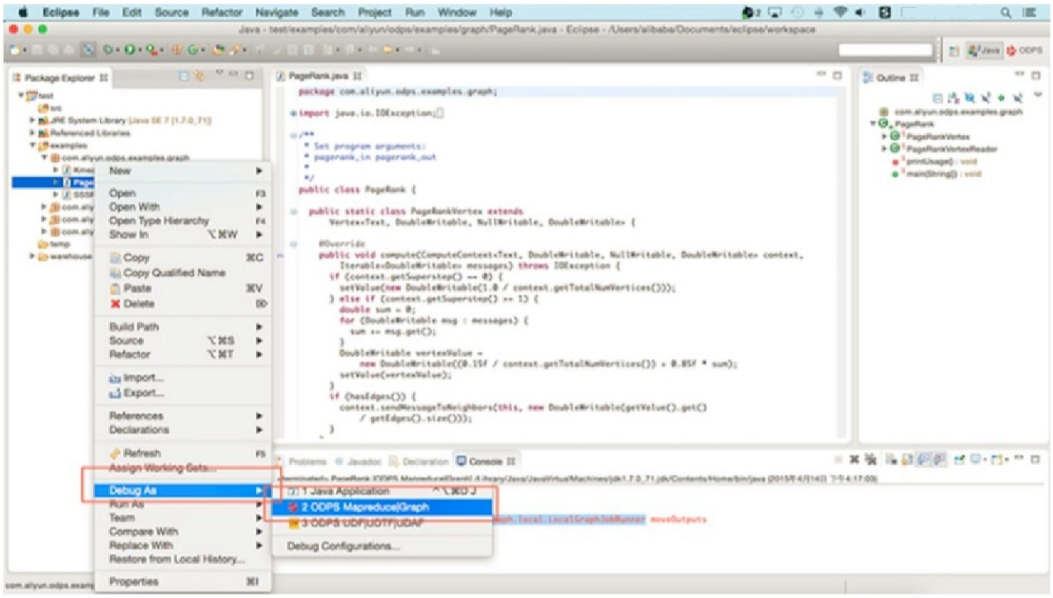
1. On the Package Explorer tab, choose example > com.aliyun.odps.examples.graph > PageRank.java, as shown in the following figure.

Choose PageRank.java



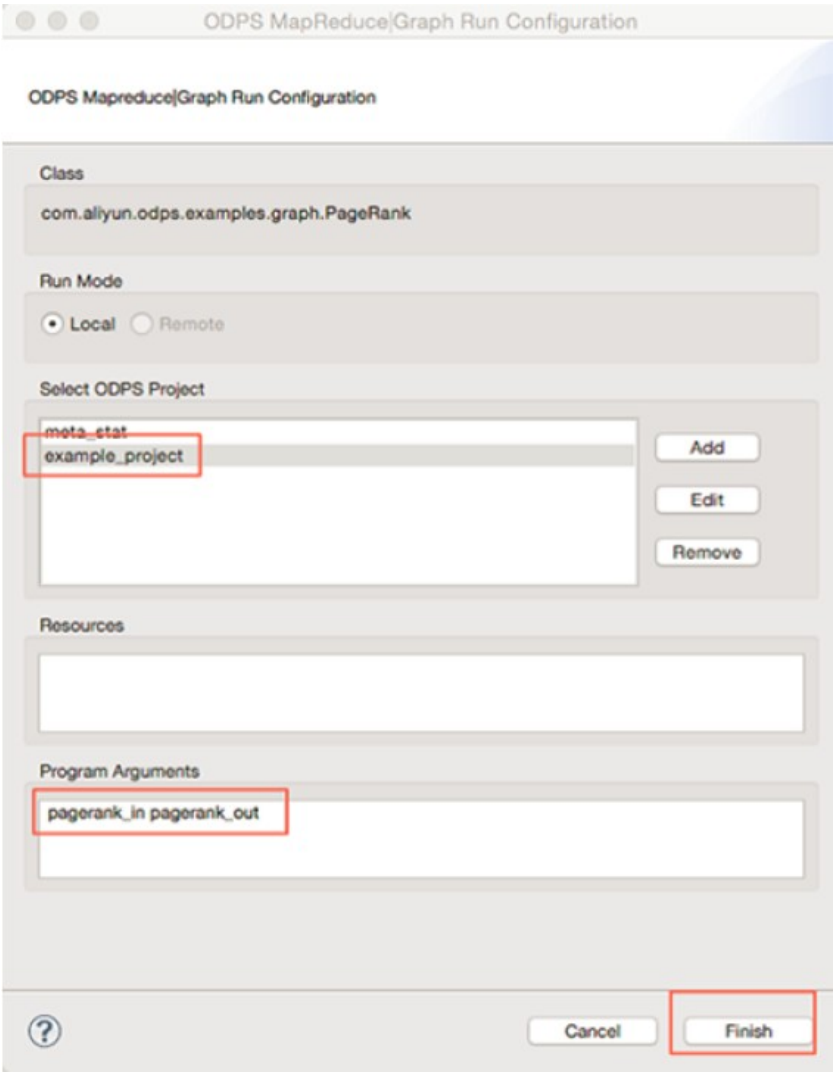
2. Right-click the PageRank.java file and choose **Debug As > ODPS MapReduce|Graph**, as shown in the following figure.

Choose Debug As > ODPS MapReduce|Graph



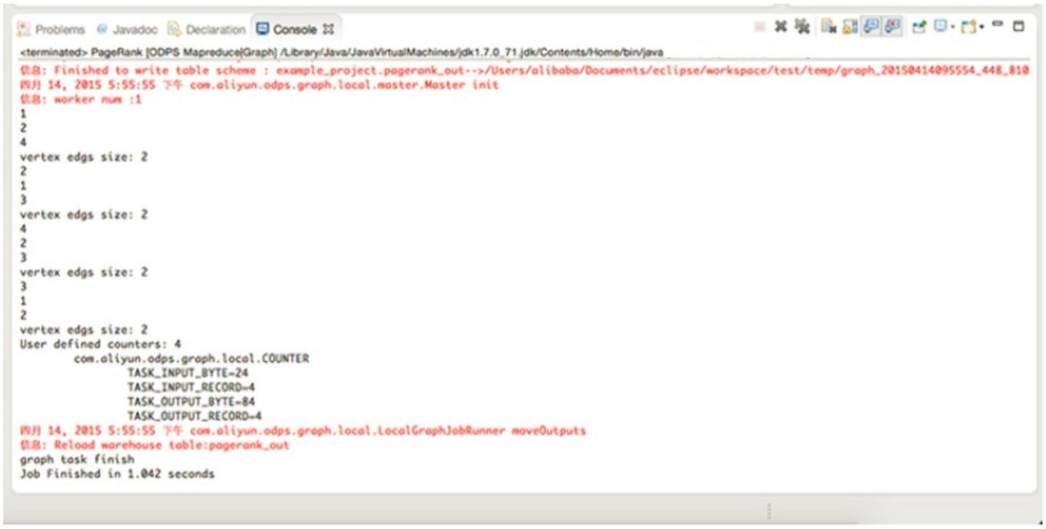
3. In the ODPS MapReduce/Graph Run Configuration dialog box, set the parameters to the values shown in the following figure.

ODPS MapReduce/Graph Run Configuration dialog box



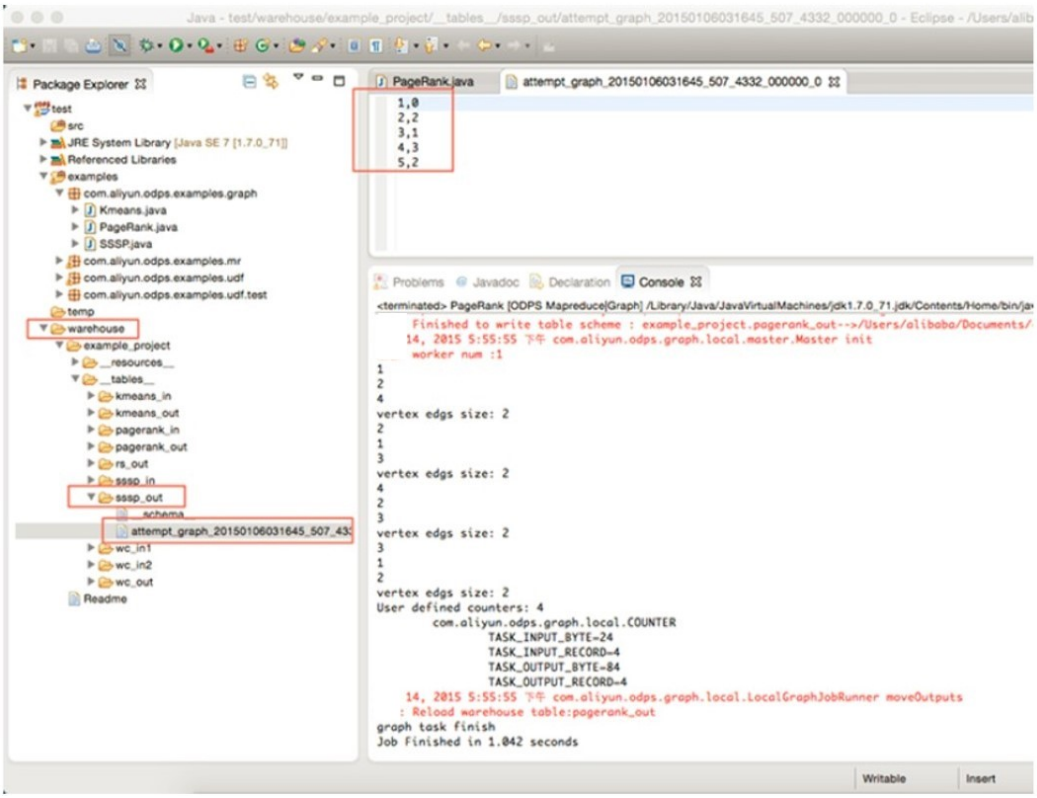
4. Click **Finish** to view the job running result, as shown in the following figure.

   Running result



View the local computing result, as shown in the following figure.

Local computing result



After local debugging succeeds, you can package the program, upload the package to MaxCompute as a JAR resource, and submit a 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 how to upload JAR resources, see Compile and run a Graph job.

1.23. MaxCompute feature enhancement packages

1.23.1. Content moderation

This topic describes the content moderation enhancement package.

Purpose

The content moderation enhancement package provides an end-to-end general-purpose or industrial solution for content moderation in big data fields. This package is developed for Apsara Stack customers who have big data and content moderation requirements. In this solution, you can use MaxCompute to capture data, including structured data and unstructured data. You can also identify and review the content related to politics, pornography, and violence based on the content moderation engine.

Description

- Content moderation

Content Moderation is a service that intelligently moderates the content of multimedia objects, such as images, videos, text, and audios in various scenarios. This service helps you effectively reduce the risk of content violations.

Common detection scenarios include intelligent pornography detection, terrorism recognition, graphic advertising recognition, logo recognition, facial recognition, QR code recognition, OCR image-text recognition, text-based anti-spam, voice-based anti-spam, and file content-based anti-spam.

Content Moderation provides the website detection feature. This feature automatically detects risky and illicit content on your website on a regular basis. The OSS violation detection feature is provided to perform pornography- and violence-related content detection on images and videos in your specified Object Storage Service (OSS) buckets. You can also directly call the Content Moderation API to submit machine identification tasks for specific scenarios.

- Types of content moderation

  Content moderation is classified into the following types based on the data types and API definitions:

  - Image moderation.
  - Video moderation.
  - Text moderation.
  - Audio moderation.
  - File moderation. After you submit a file moderation task, file content is automatically parsed to moderate images and text in the file.

- Unstructured data processing in MaxCompute

  In addition to structured data, MaxCompute also allows you to process unstructured data. You can use external tables to process unstructured data stored in data storage services, such as OSS and Tablestore. You can also use volumes to store and process unstructured data in MaxCompute.

  In content moderation scenarios, you can directly use Content Moderation to process unstructured data in OSS. In volume or table scenarios, we recommend that you perform content moderation on data in MaxCompute.

Examples

Raw data:

```
localtest_byok>read table_iacs_demo;
```

The following result is returned:

```
+------------+
| data       |
+------------+
| aaaa       |
| Obama and Trump met at White House yesterday. |
| Hi, a booty call? |
| You are such a rubbish, really heartless. |
| If you have free time and want to find a part-time job, you can call me. |
| bbbb       |
+------------+
```

The following content is provided after content moderation is performed:

```
localtest_byok>select IAcs_demo(data, "demo") as (code,msg,content,filteredContent,suggestion,label,
results) from table_iacs_demo;
```

The following result is returned:

```
+------+-----+---------+-----------------+------------+-------+---------+
| code | msg | content | filteredcontent | suggestion | label | results |
+------+-----+---------+-----------------+------------+-------+---------+
| 200  | OK  | aaaa    | NULL            | pass       | normal | [{"rate":99.91,"suggestion":"pass",
"label":"normal","scene":"antispam"}] |
| 200  | OK  | Obama and Trump met at White House yesterday. | Obama and *** met at White House yest
erday. | block      | customized | [{"rate":99.91,"suggestion":"block","details":[{"contexts":[{"lib
Code":"4386012","libName":"demo","context":"Trump"}],"label":"customized"}],"label":"customized","sc
ene":"antispam"}] |
| 200  | OK  | Hi, a booty call? | NULL            | block      | porn  | [{"rate":99.91,"suggestion
":"block","details":[{"contexts":[{"context":"Hi, a booty call?"}],"label":"porn"}],"label":"porn","
scene":"antispam"}] |
| 200| OK | You are such a rubbish, really heartless. | NULL            | block      | abuse | [{"ra
te":89.1,"suggestion":"block","details":[{"label":"abuse"}],"label":"abuse","scene":"antispam"}] |
| 200 | OK | If you have free time and want to find a part-time job, you can call me. | NULL
| pass       | normal | [{"rate":99.91,"suggestion":"pass","label":"normal","scene":"antispam"}] |
| 200  | OK  | bbbb    | NULL            | pass       | normal | [{"rate":99.91,"suggestion":"pass",
"label":"normal","scene":"antispam"}] |
+------+-----+---------+-----------------+------------+-------+---------+
```

1.23.2. MCQA

1.23.2.1. Overview

This topic describes the architecture, key features, scenarios, and limits of MaxCompute Query Acceleration (MCQA).

Description

MCQA of MaxCompute accelerates the execution of small- and medium-sized query jobs and reduces the execution time from minutes to seconds. MCQA is compatible with other query features of MaxCompute.

MCQA allows you to perform ad hoc queries or business intelligence (BI) analysis by connecting mainstream BI tools or SQL clients to MaxCompute projects.

MCQA uses an independent resource pool that does not use quota groups. MCQA automatically identifies query jobs and shortens the job queue length, which improves user experience.

Key features

MCQA provides the following key features:

- Low-latency resource scheduling

  Uses an efficient and low-latency resource scheduling policy and an independent resource pool.

- Automatic identification

  Automatically identifies the size of query jobs. MaxCompute can use MCQA to accelerate the queries or process multiple query jobs at the same time and then return the query results. This allows you to analyze query jobs of different sizes or complexities.

- Syntax compatibility

  Uses the same SQL syntax as MaxCompute.

- Selection of query methods

  Allows the MaxCompute client to use MCQA or the offline mode to run query jobs. You can also configure to forcibly use MCQA in latency-sensitive scenarios.

Scenarios

The following table describes the scenarios for which MCQA is suitable.

Scenario	Description	Applicable scope
Ad hoc query	You can use MCQA to optimize the query performance of small- and medium-sized datasets (less than 100 GB) and perform low-latency queries on MaxCompute tables. This helps develop and analyze data.	You can specify query criteria based on your requirements, obtain query results, and adjust the query logic. In this scenario, the query latency must be within dozens of seconds. Users are data developers or data analysts who have mastered SQL skills and prefer to use the client tools that they are familiar with to analyze queries.
BI analysis	If you use MaxCompute to build an enterprise-class data warehouse, MaxCompute performs extract, transform, load (ETL) operations to process data into business-oriented and consumable aggregate data. MCQA features low latency and supports elastic concurrency and data caching. You can use MCQA with partitions and buckets in MaxCompute tables to run concurrent jobs, generate reports, analyze statistics, and analyze fixed reports at a low cost.	In most cases, the query object is aggregate data. This scenario is suitable for multidimensional queries, fixed queries, or high-frequency queries that contain small amounts of data. In this scenario, queries are latency-sensitive, and the results are returned in seconds. For example, the latency for most queries is less than 5 seconds. The time elapsed for each query varies based on the data size and query complexity.
Detailed queries and analysis of large amounts of data	MCQA automatically identifies the size of query jobs. MCQA can respond to and process small-sized jobs in a timely manner, and can allocate the resources required for large-sized jobs. This helps analysts run query jobs of different sizes and complexities.	In this scenario, large amounts of historical data is queried. The size of valid data that is queried is small, and the requirement for latency is low. Users are business analysts who need to gain business insights from data, explore potential business opportunities, and validate business assumptions.

Limits

MCQA supports only SELECT statements and does not support user-defined functions (UDFs). If you submit a statement or feature that MCQA does not support from the MaxCompute client, the MaxCompute client automatically rolls back to the common offline mode to execute the statement or feature. Other tools cannot roll back to the common offline mode to execute the submitted statement or feature that MCQA does not support.

The following table describes the limits of MCQA.

Item	Limit

Item	Limit
Query	• If more than 1,000 MCQA query jobs are executed at the same time, the system automatically rolls back these jobs to SQL query jobs. • If you submit an MCQA query job from the MaxCompute client, the default timeout period is 30 seconds. If you submit an MCQA query job from the ad hoc query module of DataWorks, the default timeout period is 20 seconds. If the MCQA query job times out, the system automatically rolls back the MCQA query job to an SQL query job. • MCQA can cache only data that is stored in ALIORC tables into memory to accelerate queries. • You cannot use MCQA to query data from external tables.
Query concurrency	You can run a maximum of 120 concurrent MCQA query jobs in each MaxCompute project.

1.23.2.2. Usage notes

MaxCompute Query Acceleration (MCQA) is a built-in feature of MaxCompute. MCQA uses the native MaxCompute SQL syntax and supports MaxCompute built-in functions and permission systems. This topic describes how to use MCQA.

Context

You can use one of the following methods to enable the MCQA feature:

- Use the MaxCompute client. For more information, see Enable MCQA on the MaxCompute client.
- Use the ad hoc queries or data analytics feature of DataWorks. By default, the MCQA feature is enabled for ad hoc queries or data analytics of DataWorks. For more information, see Enable MCQA on the Ad-Hoc Query or Manually Triggered Workflow page.
- Use the MaxCompute Java Database Connectivity (JDBC) driver. For more information, see JDBC.
- Use a MaxCompute SDK. This method requires dependencies in a pom.xml file. For more information, see Enable MCQA by using MaxCompute SDK for Java.

Enable MCQA on the MaxCompute client

To enable the MCQA feature on the MaxCompute client, perform the following steps:

1. Download the client package of the latest version.
2. Decompress the package to a folder that contains the following subfolders:

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

3. Install and configure the client. For more information, see Configure the client.
4. Append the following command line to the odps_config.ini file in the conf folder.

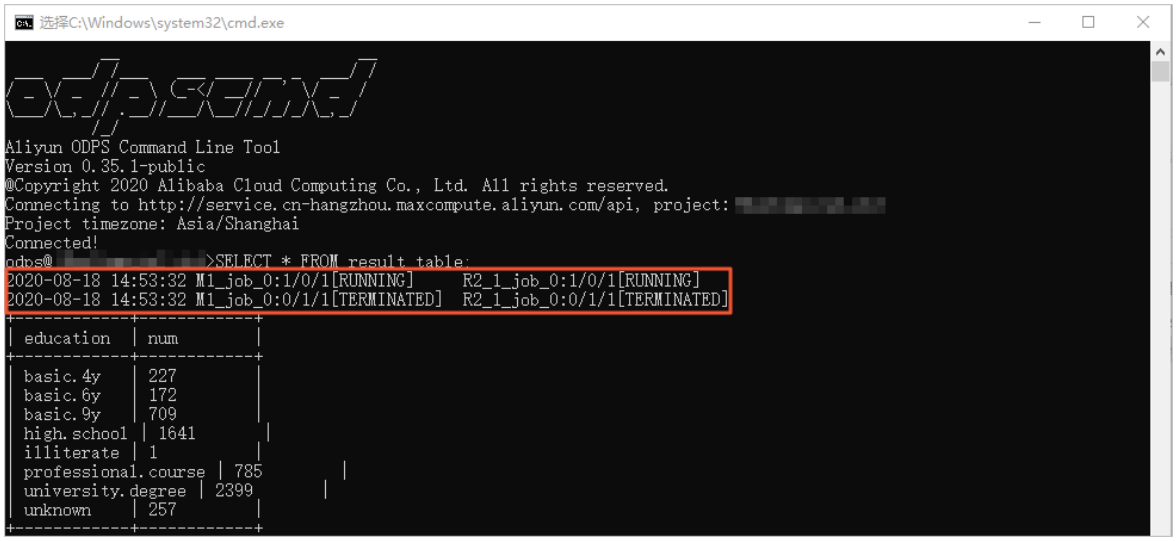
   ```
   enable_interactive_mode=true
   ```

5. Run the odpscmd file in the bin directory in the Linux operating system or run the odpscmd.bat file in the bin directory in the Windows operating system. If the following information appears, the MaxCompute client

runs properly.

6. Run a query job to verify that the MCQA feature is enabled.

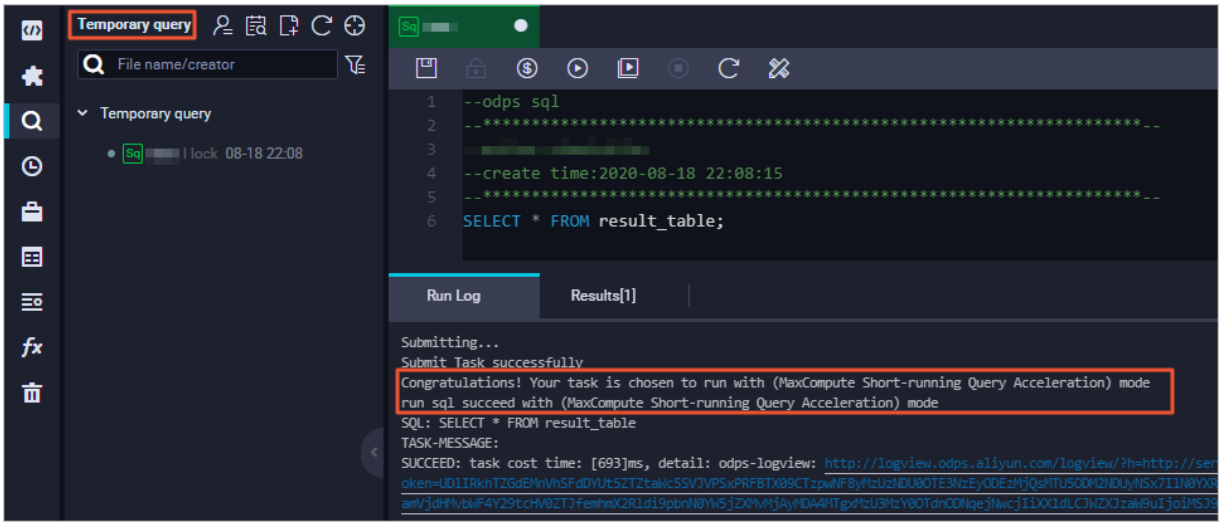
   If the returned results contain the following information after you run the query job, the MCQA feature is enabled.



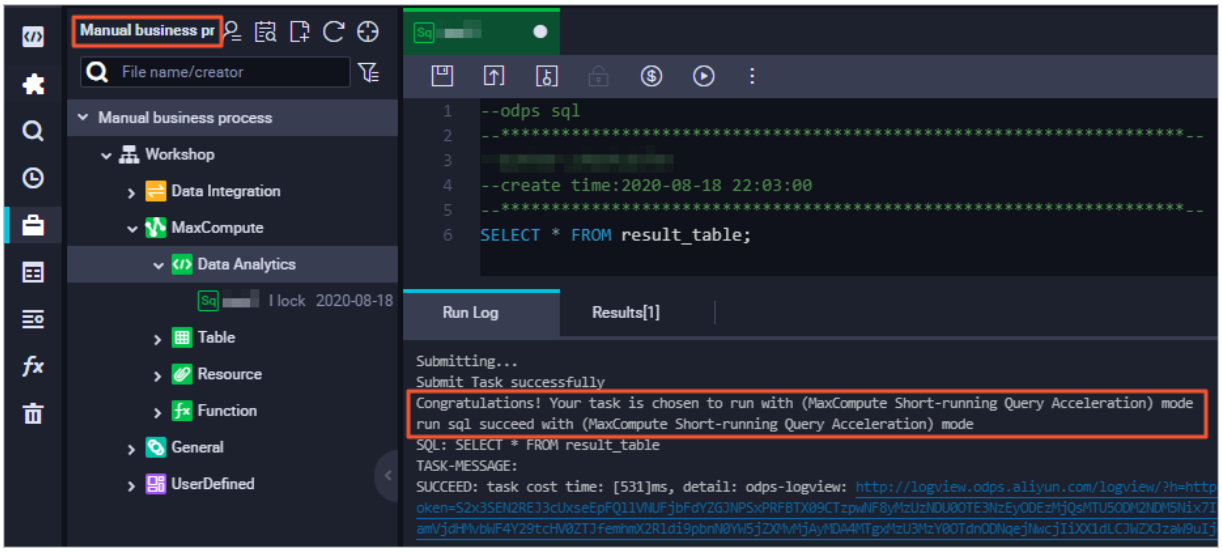
Enable MCQA on the Ad-Hoc Query or Manually Triggered Workflow page

By default, the MCQA feature is enabled on the **Ad-Hoc Query** or **Manually Triggered Workflow** page of DataWorks. Manual operations are not required. If you want to disable the MCQA feature, submit a ticket.

Run a query job on the **Ad-Hoc Query** page. If the returned results contain the following information, the MCQA feature is enabled.



Run a query job on the **Manually Triggered Workflow** page. If the returned results contain the following information, the MCQA feature is enabled.



JDBC

You can use the MaxCompute JDBC driver in the following scenarios:

- Use the MaxCompute JDBC driver to connect to MaxCompute. To enable the MCQA feature in this scenario, you must modify related configurations. For more information, see Enable MCQA for the MaxCompute JDBC driver.
- Use the MaxCompute JDBC driver to connect to Tableau. Then, you can use Tableau to analyze data in MaxCompute in a visualized manner. To enable the MCQA feature in this scenario, you must modify related configurations. For more information, see Enable MCQA on Tableau Server based on the MaxCompute JDBC driver.
- Use the MaxCompute JDBC driver to connect to SQL Workbench/J. Then, you can use SQL Workbench/J to execute SQL statements on data in MaxCompute. To enable the MCQA feature in this scenario, you must modify related configurations. For more information, see Enable MCQA on SQL Workbench/J based on the MaxCompute JDBC driver.

Enable MCQA for the MaxCompute JDBC driver

If you use the JDBC driver to connect to MaxCompute, perform the following steps to enable the MCQA feature.

1. Download the JDBC JAR file that supports the MCQA feature or download the source code that can be compiled.
2. Add the following dependency to the pom.xml file in the Maven repository:

```
<dependency>
  <groupId>com.aliyun.odps</groupId>
  <artifactId>odps-jdbc</artifactId>
  <version>3.2.0</version>
  <classifier>jar-with-dependencies</classifier>
</dependency>
```

> **Note** The version must be V3.2.0 or later.

3. Create a Java program based on the source code and configure the required information. For more information, see ODPS JDBC.

   The source code contains the following information:

```
String accessId = "your_access_id";
String accessKey = "your_access_key";
String conn = "jdbc:odps:http://service.odps.aliyun.com/api?project=<your_project_name>"&accessId&accessKey&charset=UTF-8&interactiveMode=true";
Statement stmt = conn.createStatement();
-- Replace your_access_id with the AccessKey ID and your_access_key with the AccessKey secret of your Alibaba Cloud account. Replace your_project_name with the name of the project for which the MCQA feature is enabled.
Connection conn = DriverManager.getConnection(conn, accessId, accessKey);
Statement stmt = conn.createStatement();
String tableName = "testOdpsDriverTable";
stmt.execute("drop table if exists " + tableName);
stmt.execute("create table " + tableName + " (key int, value string)");
```

To enable the MCQA feature for the MaxCompute JDBC driver, you need only to modify `String conn` or the code.

- `String conn` : Add `interactiveMode=true` .
- Code: Add `interactiveMode=true` .

4. (Optional)Configure the following parameters in the connection string to optimize the processing logic.
   - `enableOdpsLogger` : is used to display logs. If you do not configure Simple Logging Facade for Java (SLF4J), we recommend that you set this parameter to True.
   - `fallbackForUnknownError` : The default value of this parameter is False. If this parameter is set to True, the system rolls back to the offline mode when an unknown error occurs.
   - `fallbackForResourceNotEnough` : The default value of this parameter is False. If this parameter is set to True, the system rolls back to the offline mode when resources are insufficient.
   - `fallbackForUpgrading` : The default value of this parameter is False. If this parameter is set to True, the system rolls back to the offline mode during an upgrade.
   - `fallbackForRunningTimeout` : The default value of this parameter is False. If this parameter is set to True, the system rolls back to the offline mode when an operation times out.
   - `fallbackForUnsupportedFeature` : The default value of this parameter is False. If this parameter is set to True, the system rolls back to the offline mode when the MCQA feature is not supported.
   - `fallbackForAll` : The default value of this parameter is False. If this parameter is set to True, the system rolls back to the offline mode in the preceding scenarios.

Enable MCQA on Tableau Server based on the MaxCompute JDBC driver

Add `interactiveMode=true` to the URL of **Tableau Server**. We recommend that you also add `enableOdpsLogger=true` to display logs.

Sample URL of **Tableau Server**:

```
http://service.cn-beijing.maxcompute.aliyun.com/api?project=****_beijing&interactiveMode=true&enableOdpsLogger=true
```

To enable MCQA for some tables in a MaxCompute project, add the `table_list=table_name1, table_name2` property to the URL of Tableau Server. Then, use this property to specify the tables for which you want to enable MCQA. Separate table names with commas (,). If you specify an excessive number of tables, access to the URL of Tableau Server is time-consuming. We recommend that you specify only the required tables in the URL of Tableau Server. Sample URL:

```
http://service.cn-beijing.maxcompute.aliyun.com/api?project=****_beijing&interactiveMode=true&enableOdpsLogger=true&table_list=orders,customers
```

If a table contains a large number of partitions, we recommend that you do not use data from all partitions as the data source. You can filter the required partitions or run custom SQL queries to obtain the required data.

Enable MCQA on SQL Workbench/J based on the MaxCompute JDBC driver

After you configure the MaxCompute JDBC driver, you can use the MCQA feature on SQL Workbench/J by modifying the JDBC URL that you specified on the profile configuration page.

Specify the JDBC URL in the following format: `jdbc:odps:<MaxCompute_endpoint>? project=<MaxCompute_project_name>&accessId=<AccessKey ID>&accessKey=<AccessKey Secret>&charset=UTF-8&interactiveMode=true`. Parameters:

- maxcompute_endpoint: the endpoint of your MaxCompute.
- maxcompute_project_name: the name of your MaxCompute project.
- AccessKey ID: the AccessKey ID that is used to access the specified project.
- AccessKey Secret: the AccessKey secret that corresponds to the AccessKey ID.
- charset=UTF-8: the character set encoding format.
- interactiveMode: specifies whether to enable the MCQA feature. To enable the MCQA feature, set this parameter to `true`.

Enable MCQA by using MaxCompute SDK for Java

You must add a specified dependency to the pom.xml file in the Maven repository. Sample dependency:

```
<dependency>
  <groupId>com.aliyun.odps</groupId>
  <artifactId>odps-sdk-core</artifactId>
  <version>0.36.2-public</version>
</dependency>
```

Run commands to create a Java program. Sample commands:

```
import com.aliyun.odps.Odps;
import com.aliyun.odps.OdpsException;
import com.aliyun.odps.OdpsType;
import com.aliyun.odps.account.Account;
import com.aliyun.odps.account.AliyunAccount;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.ResultSet;
import com.aliyun.odps.sqa.*;
import java.io.IOException;
import java.util.*;
public class SQLExecutorExample {
    public static void SimpleExample() {
        // Specify the account and project information.
        Account account = new AliyunAccount("your_access_id", "your_access_key");
        Odps odps = new Odps(account);
        odps.setDefaultProject("your_project_name");
        odps.setEndpoint("http://service.odps.aliyun.com/api");
        // Prepare to build an SQLExecutor.
        SQLExecutorBuilder builder = SQLExecutorBuilder.builder();
        SQLExecutor sqlExecutor = null;
        try {
            // run in offline mode or run in interactive mode
            if (false) {
                // Create an SQLExecutor that runs offline SQL queries by default.
                sqlExecutor = builder.odps(odps).executeMode(ExecuteMode.OFFLINE).build();
            } else {
```

```
                // Create an SQLExecutor that runs SQL queries with MCQA enabled by default.
                sqlExecutor = builder.odps(odps).executeMode(ExecuteMode.INTERACTIVE).build();
            }
            // Pass special query settings if required.
            Map<String, String> queryHint = new HashMap<>();
            queryHint.put("odps.sql.mapper.split.size", "128");
            // Submit a query job. You can pass hints.
            sqlExecutor.run("select count(1) from test_table;", queryHint);
            // List the System.out.println() statements that can be used to query common information
.
            // UUID
            System.out.println("ExecutorId:" + sqlExecutor.getId());
            // Query the Logview URL of the current query job.
            System.out.println("Logview:" + sqlExecutor.getLogView());
            // Query the instance on which the current query job is run. In interactive mode, multip
le query jobs may be run on the same instance.
            System.out.println("InstanceId:" + sqlExecutor.getInstance().getId());
            // Query the progress of the current query job. You can check the progress bar in the co
nsole.
            System.out.println("QueryStageProgress:" + sqlExecutor.getProgress());
            // Query the changelogs about the execution status for the current query job, such as ro
llback messages.
            System.out.println("QueryExecutionLog:" + sqlExecutor.getExecutionLog());
            // Obtain results of query jobs by calling one of the following API operations:
            if(false) {
                // Query the results of all query jobs. The API operation that you called is a synch
ronous operation and may occupy a thread until the query succeeds or fails.
                List<Record> records = sqlExecutor.getResult();
                printRecords(records);
            } else {
                // Query the ResultSet iterator of the query results. The API operation that you cal
led is a synchronous operation and may occupy a thread until the query succeeds or fails.
                ResultSet resultSet = sqlExecutor.getResultSet();
                while (resultSet.hasNext()) {
                    printRecord(resultSet.next());
                }
            }
            // run another query
            sqlExecutor.run("select * from test_table;", new HashMap<>());
            if(false) {
                // Query the results of all query jobs. The API operation that you called is a synch
ronous operation and may occupy a thread until the query succeeds or fails.
                List<Record> records = sqlExecutor.getResult();
                printRecords(records);
            } else {
                // Query the ResultSet iterator of the query results. The API operation that you cal
led is a synchronous operation and may occupy a thread until the query succeeds or fails.
                ResultSet resultSet = sqlExecutor.getResultSet();
                while (resultSet.hasNext()) {
                    printRecord(resultSet.next());
                }
            }
        } catch (OdpsException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (sqlExecutor ! = null) {
                // Close the SQLExecutor and release related resources.
```

```
                sqlExecutor.close();
            }
        }
    }
    // SQLExecutor can be reused by pool mode
    public static void ExampleWithPool() {
        // Specify the account and project information.
        Account account = new AliyunAccount("your_access_id", "your_access_key");
        Odps odps = new Odps(account);
        odps.setDefaultProject("your_project_name");
        odps.setEndpoint("http://service.odps.aliyun.com/api");
        // Run queries by using a connection pool.
        SQLExecutorPool sqlExecutorPool = null;
        SQLExecutor sqlExecutor = null;
        try {
            // Create a connection pool. Specify the connection pool size and the default execution
mode.
            SQLExecutorPoolBuilder builder = SQLExecutorPoolBuilder.builder();
            builder.odps(odps)
                    .initPoolSize(1) // init pool executor number
                    .maxPoolSize(5)  // max executors in pool
                    .executeMode(ExecuteMode.INTERACTIVE); // run in interactive mode
            sqlExecutorPool = builder.build();
            // Obtain an SQLExecutor from the connection pool. If no SQLExecutor can be obtained fro
m the connection pool, you can add an SQLExecutor. Make sure that the total number of SQLExecutors d
oes not exceed the upper limit.
            sqlExecutor = sqlExecutorPool.getExecutor();
            // Use the SQLExecutor in the same way it is used in the preceding example.
            sqlExecutor.run("select count(1) from test_table;", new HashMap<>());
            System.out.println("InstanceId:" + sqlExecutor.getId());
            System.out.println("Logview:" + sqlExecutor.getLogView());
            List<Record> records = sqlExecutor.getResult();
            printRecords(records);
        } catch (OdpsException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            sqlExecutor.close();
        }
        sqlExecutorPool.close();
    }
    private static void printRecord(Record record) {
        for (int k = 0; k < record.getColumnCount(); k++) {
            if (k ! = 0) {
                System.out.print("\t");
            }
            if (record.getColumns()[k].getType().equals(OdpsType.STRING)) {
                System.out.print(record.getString(k));
            } else if (record.getColumns()[k].getType().equals(OdpsType.BIGINT)) {
                System.out.print(record.getBigint(k));
            } else {
                System.out.print(record.get(k));
            }
        }
    }
    private static void printRecords(List<Record> records) {
        for (Record record : records) {
            printRecord(record);
```

```
            System.out.println();
        }
    }
    public static void main(String args[]) {
        SimpleExample();
        ExampleWithPool();
    }
}
```

1.23.3. VVP On MaxCompute

1.23.3.1. Overview

This topic describes VVP On MaxCompute in MaxCompute feature enhancement packages.

VVP On MaxCompute encapsulates the features of Realtime Compute for Apache Flink that is developed on the Ververica Platform (VVP) based on MaxCompute resources. After you enable the VVP On MaxCompute feature, you can use the Cupid joint computing platform to complete the operations related to real-time computing by using the underlying storage and computing resources of MaxCompute on the VVP UI.

1.23.3.2. Benefits

This topic describes the benefits of VVP On MaxCompute.

- Seamless integration with MaxCompute
  - MaxCompute is interconnected with Realtime Compute for Apache Flink to share data. This way, Realtime Compute for Apache Flink can benefit from the scheduling and storage capabilities of MaxCompute.
  - Computing and storage resources are managed in a unified manner to ensure high elasticity. You do not need to be concerned about the issues, such as storage space expansion difficulties and time-consuming data computations that are caused by the increased data volume. The storage and retrieval capabilities of a MaxCompute cluster are automatically extended with your data volume. This enables you to focus on your business and data value.
  - Hardware integration is used to eliminate the differences in machine configuration requirements, time consumption, and peak loads of stream and batch processing. This way, the number of machines in all clusters is reduced for lower hardware costs and the model is unified for lower maintenance costs.
  - Intelligent cluster deployment and O&M of Realtime Compute for Apache Flink: You can deploy Realtime Compute for Apache Flink along with MaxCompute to achieve real-time data processing and harness all the capabilities provided by MaxCompute and its related ecosystem components, such as Spark, Elasticsearch, advanced data analysis, and AI.
  - Enterprise-class system security: Security isolation and control are implemented based on the multi-tenancy capability and security control of MaxCompute.
  - APIs: SDK programming interfaces for Apache Flink are used to support features provided by Apache Flink and user-defined features.
- End-to-end platform development: Jobs can be created, configured, and run on MaxCompute.
- End-to-end platform O&M: Comprehensive monitoring of job running status and optimization and troubleshooting of MaxCompute are implemented. Enterprise-class services are provided, and service issues can be troubleshot and optimized.
- Excellent platform ecosystem: Real-time data computing and batch processing of data warehouses are supported on the same resource platform to integrate batch and stream processing.
- Full compatibility with enterprise-class computing engines of Apache Flink

- Business GeminiStateBackend is provided, which brings the following benefits:
  - Uses a new data structure to accelerate ad hoc queries and reduce frequent disk I/O operations.
  - Optimizes the cache policy. If memory is sufficient, hot data is not stored in disks and cache entries do not expire after compaction.
  - Uses Java to implement GeminiStateBackend, which eliminates Java Native Interface (JNI) overheads that are caused by RocksDB.
  - Uses off-heap memory and implements an efficient memory allocator based on GeminiDB to eliminate the impact of garbage collection for Java Virtual Machines (JVMs).
  - Supports asynchronous incremental checkpoints. This ensures that only memory indexes are copied during data synchronization. Compared with RocksDB, GeminiStateBackend avoids jitters that are caused by I/O operations.
  - Supports local recovery and storage of the timer.
- Support for various connectors: Connectors for services, such as Log Service, DataHub, Message Queue for Apache Kafka, and Tablestore, are supported. The supported connectors are continuously updated.
- Full compatibility with Apache Flink: VVP On MaxCompute is continuously updated along with release updates of Apache Flink.

1.23.3.3. Activate VVP On MaxCompute

This topic describes how to activate VVP On MaxCompute.

> **Note** You must perform the following operations as user admin, unless otherwise specified.

1. Run the following command on ODPS AG to start the Webrm service:

   ```
   cd /home/admin/vvp/web_rm_server
   sudo sh webrm.sh restart
   ```

   After you run the command, run `netstat -nlp |grep 9088`. If the following information is displayed, the Webrm service is started.

   ```
   $netstat -nlp|grep 9088
   (Not all processes could be identified, non-owned process info
    will not be shown, you would have to be root to see it all.)
   tcp        0      0 10.5.10.203:9088            0.0.0.0:*                   LISTEN      -
   ```

2. Run the following command on ODPS AG to check the running status of Docker containers.

   ```
   ps -ef|grep vvp
   ```

   ```
   $docker ps|grep vvp
   4d6e2222fa99        39c894ad404b        "tail -f /dev/null"      2 days ago          Up 2 days
    odps-service-regress.VVPUI__.vvp_ui.1606281917
   73e611f482b8        8d34715b6d78        "tail -f /dev/null"      2 days ago          Up 2 days
    odps-service-regress.VVPAppmanager__.vvp_appmanager.1606281916
   9b3b7ed738a6        d2aa039a3150        "tail -f /dev/null"      2 days ago          Up 2 days
    odps-service-regress.VVPGateway__.vvp_gateway.1606281914
   ```

3. Log on to the vvp_ui container based on the container ID obtained in Step 2, go to the */home/admin* directory, and then run the `./start` command to run the script. If you run `netstat -nlp|grep 4200` in a new window on ODPS AG and the following information is displayed, the container starts.

   ```
   $netstat -nlp|grep 4200
   (Not all processes could be identified, non-owned process info
    will not be shown, you would have to be root to see it all.)
   tcp        0      0 0.0.0.0:4200            0.0.0.0:*                   LISTEN      -
   ```

4. Log on to the vvp_gateway container based on the container ID obtained in Step 2, go to the */home/admin*

directory, and then run the `./start` command to run the script. If you run `netstat -nlp|grep 8989` in a new window on ODPS AG and the following information is displayed, the container starts.

```
$netstat -nlp|grep 8989
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:8989                0.0.0.0:*                   LISTEN      -
```

5. Log on to the vvp_appmanager container based on the container ID obtained in Step 2, go to the */home/admin* directory, and then run the `./start` command to run the script. If you run `netstat -nlp|grep 9100` in a new window on ODPS AG and the following information is displayed, the container starts.

```
$netstat -nlp|grep 9100
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:9100                0.0.0.0:*                   LISTEN      -
```

> **Note** Perform Step 3 to Step 5 in sequence. The next step can be performed only after the preceding step is complete. Random order is not allowed.

VVP On MaxCompute is activated. You can complete relevant operations on the VVP UI.

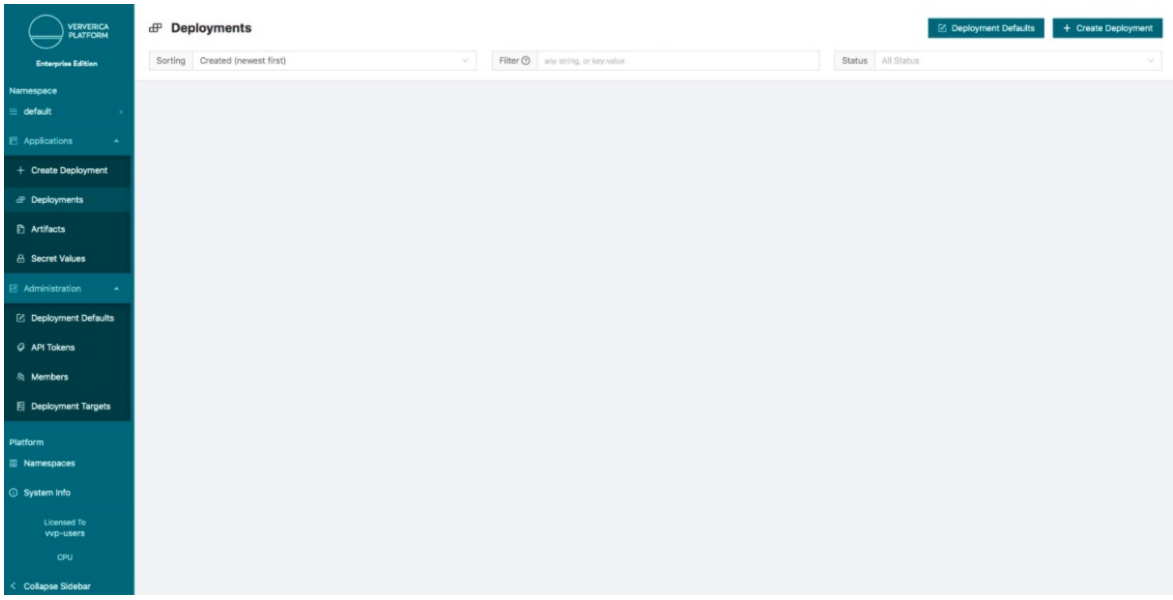
1.23.3.4. Usage notes

This topic describes how to perform operations on the VVP UI.

1. Open your browser, enter the URL of the VVP UI in the address bar, and then press Enter to log on to the VVP UI.

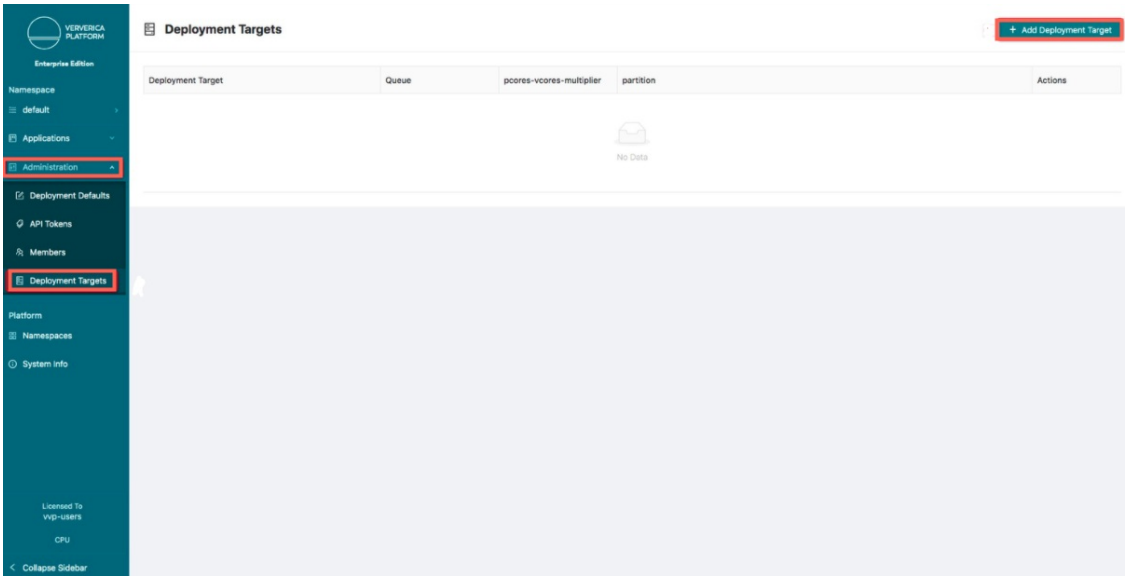
   The URL of the VVP UI is in the format of *xx.xx.xx.xx:8989*. xx.xx.xx.xx indicates the IP address of ODPS AG and 8989 is the port number.

   > **Note** We recommend that you use the Google Chrome browser.



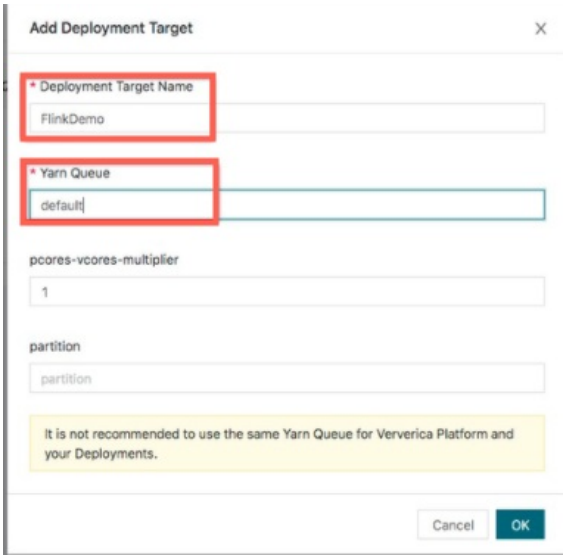
2. Create a deployment target.

i. In the left-side navigation pane, choose **Administration > Deployment Targets**. On the Deployment Targets page, click **+ Add Deployment Target**. The **Add Deployment Target** dialog box appears.

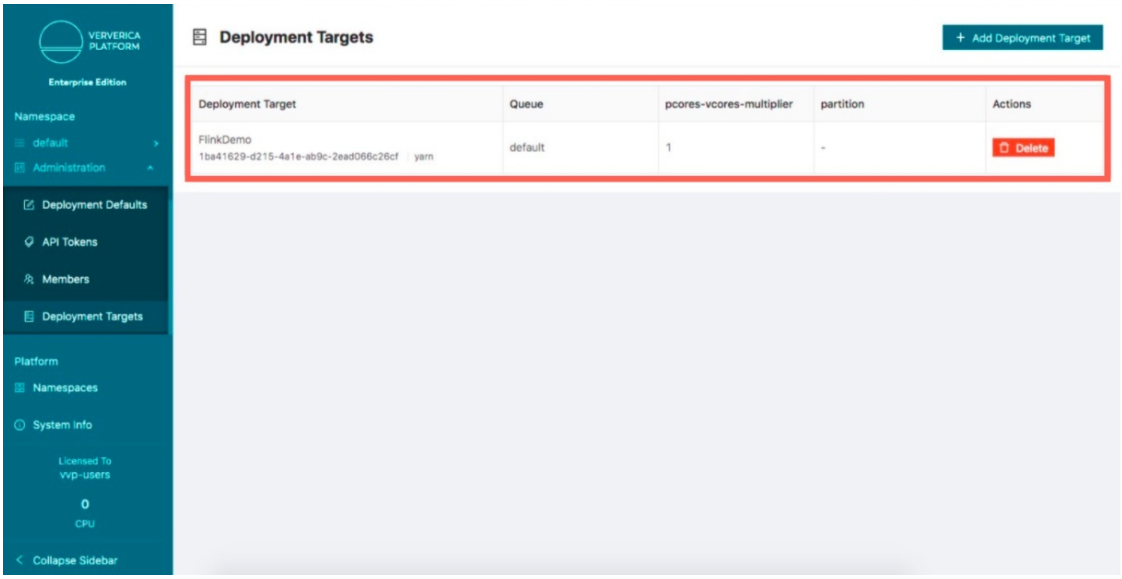


ii. In the Add Deployment Target dialog box, specify Deployment Target Name and Yarn Queue, and click **OK**.

> **Note** Deployment Target Name allows you to customize the settings. Yarn Queue must be set to default.



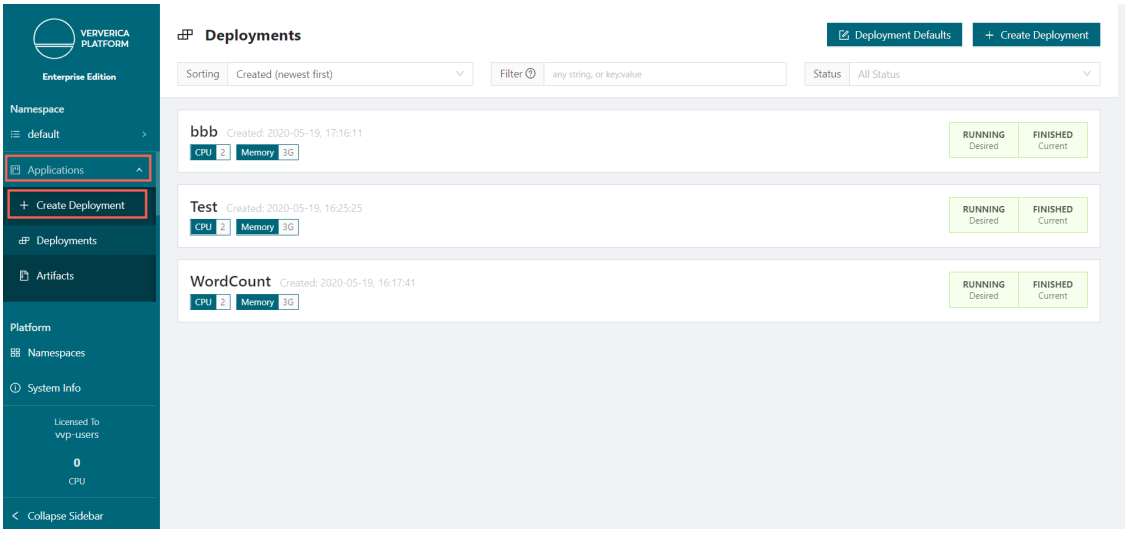
iii. View the information of the deployment target that you created on the VVP UI.



3. Create a deployment.

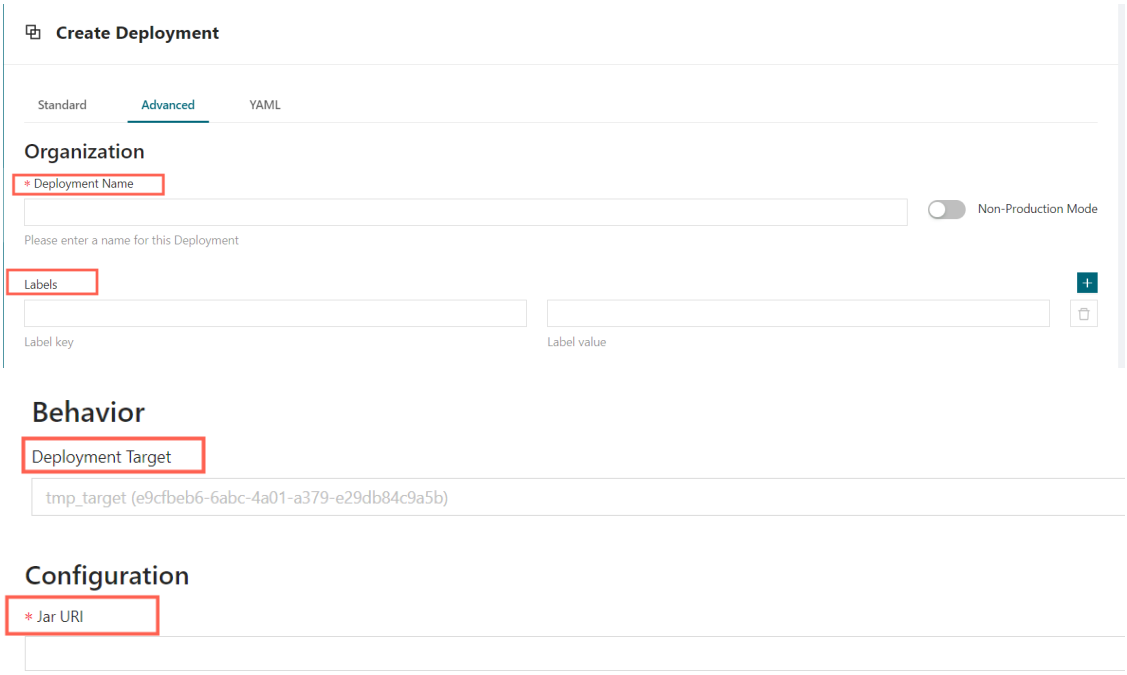
> **Note** Before you create a deployment, make sure that the deployment target already exists.

i. In the left-side navigation pane, choose **Applications > + Create Deployment**. The **Create Deployment** page appears.

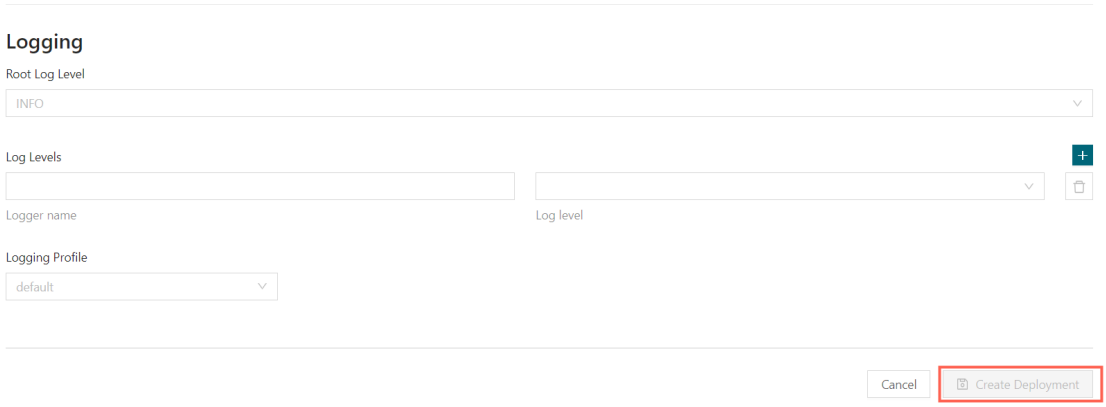


ii. Click the **Advanced** tab and configure parameters.

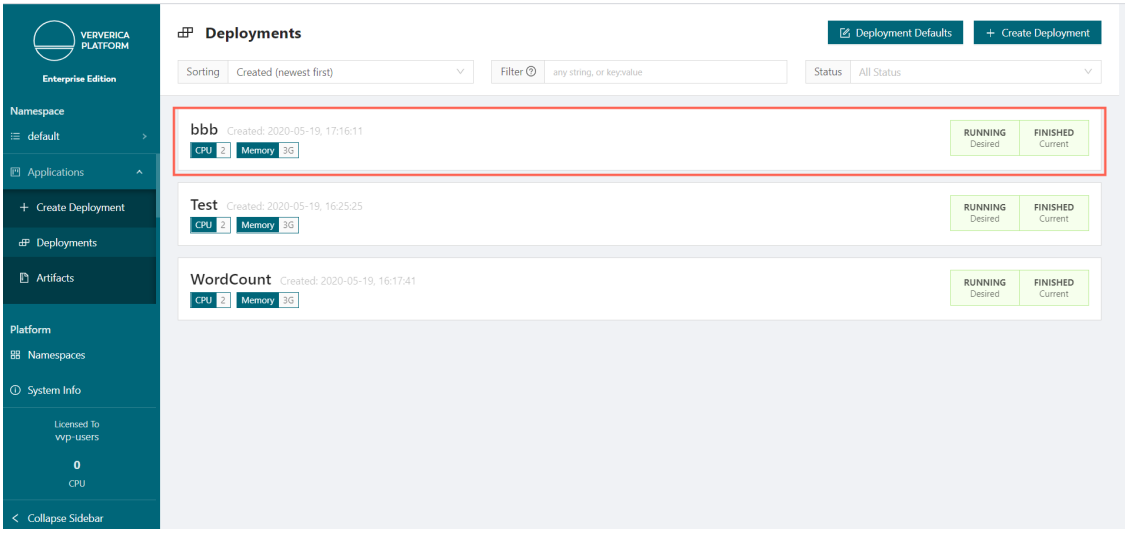
> **Note** Deployment Name and Labels allow you to customize settings. The value of Labels is in the format of key-value pairs. Select an existing deployment target from the Deployment Target drop-down list, enter the address for downloading the JAR file in Jar URI, and then retain default values for other parameters.



iii. Click **Create Deployment** to create the deployment.

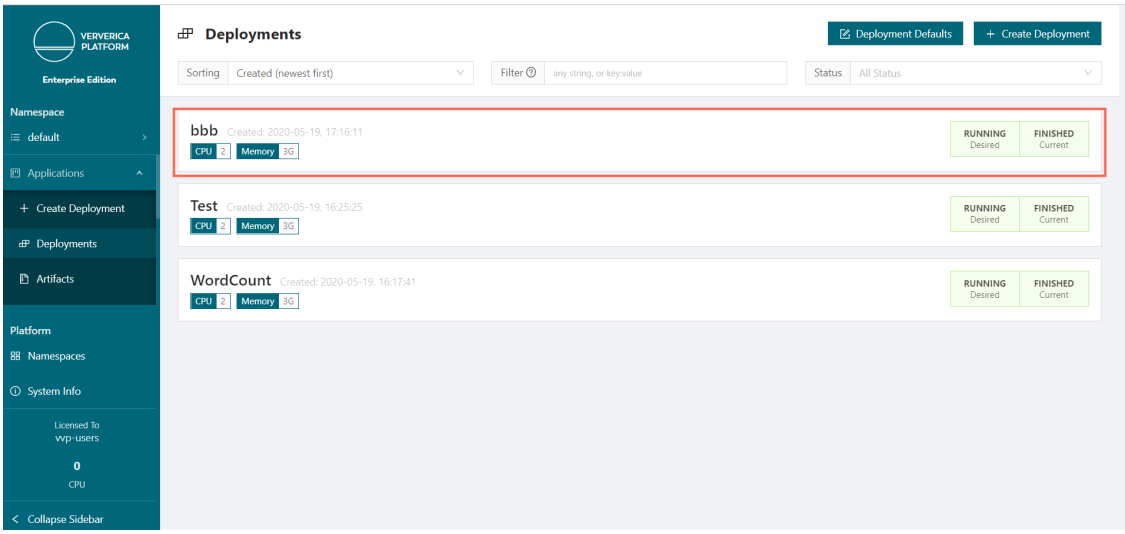


iv. View the information of the new deployment on the VVP UI.

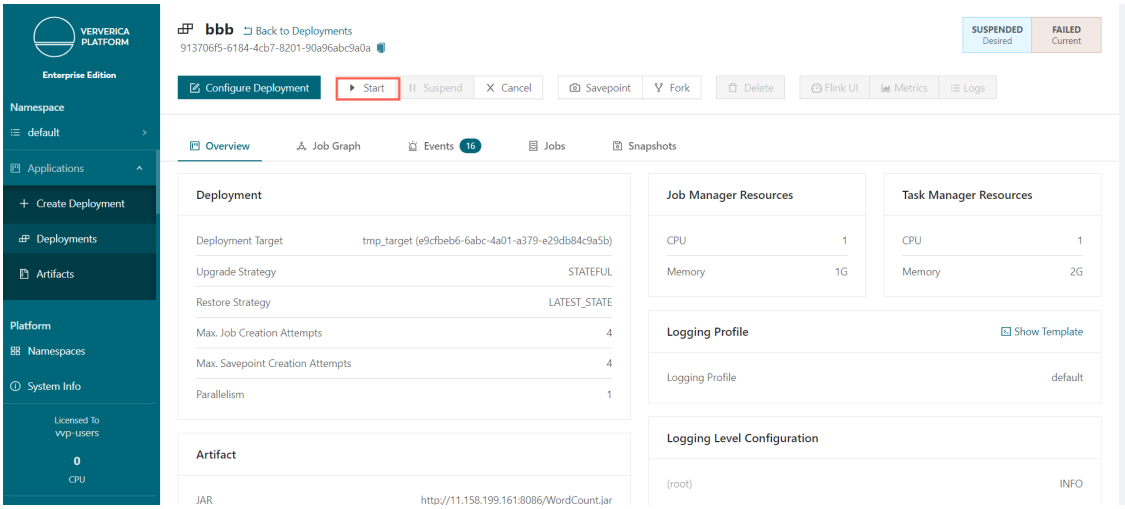


4. Start the deployment.

i. In the left-side navigation pane, choose **Applications > Deployments**. On the Deployments page, click the deployment that you created. The deployment details page appears.

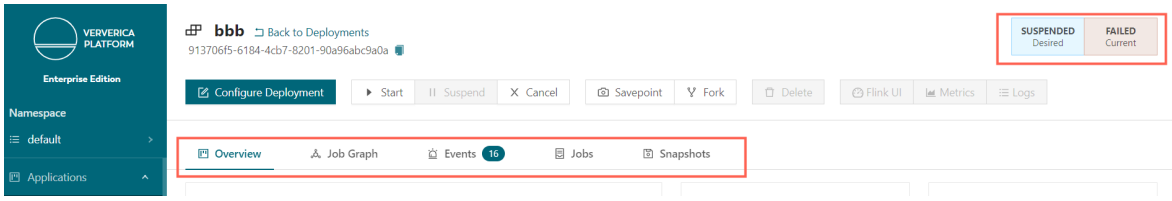


ii. On the deployment details page, click **Start** to start the deployment.

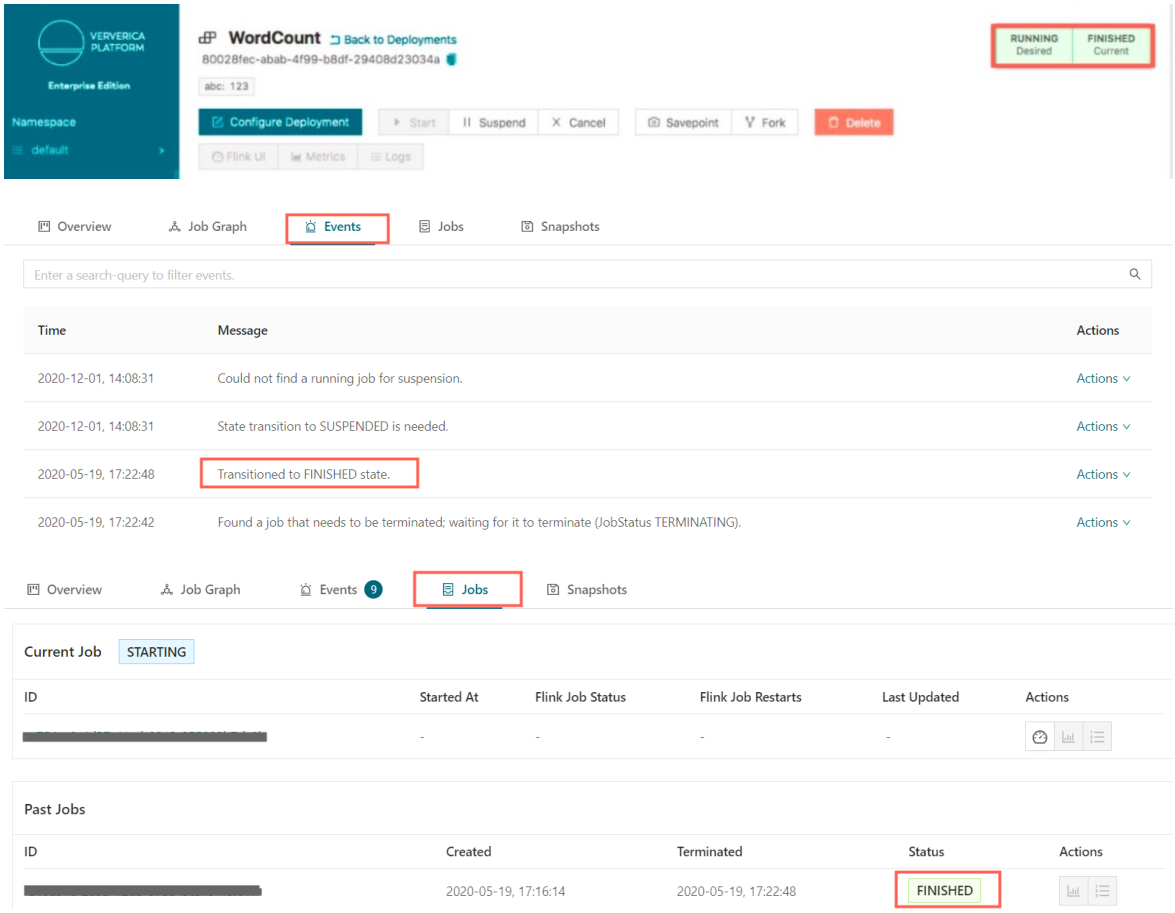


5. View the information of the deployment.

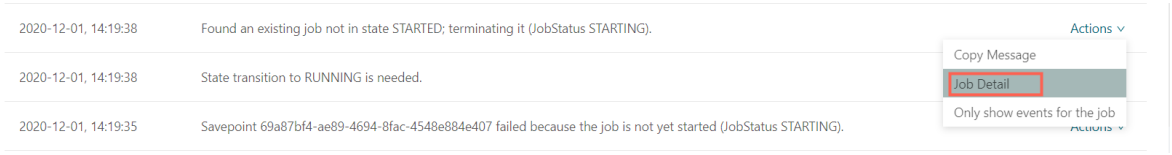
   When you run the deployment, you can view the information shown on the deployment details page. The information provides the running status of the deployment.



   After the deployment is complete, its status changes. You can view the deployment completion information on the **Events** and **Jobs** tabs.



   If an error occurs on a running deployment, you can view further details about error information on the **Events** tab.



1.23.4. Mars

1.23.4.1. Overview

This topic describes the features of Mars, the differences between Mars and PyODPS, and the scenarios of using Mars and the PyODPS DataFrame API.

Scenarios

Use Mars and the PyODPS DataFrame API in the following scenarios:

- Mars
  - You often call the to_pandas() method of the PyODPS DataFrame API to convert a PyODPS DataFrame into a pandas DataFrame.
  - You are familiar with the pandas API, but do not want to learn the PyODPS DataFrame API.
  - You want to use indexes.
  - You want to maintain the data order after you create a DataFrame.

    The Mars DataFrame API provides the iloc method to retrieve rows and obtain data from specific rows. For example, `df.iloc[10]` is used to obtain data in the tenth row. The Mars DataFrame API also provides the `df.shift()` and `df.ffill()` methods, both of which can be used only in scenarios where the data order is maintained.
  - You want to run NumPy or scikit-learn in a parallel and distributed manner, or run TensorFlow, PyTorch, and XGBoost in a distributed manner.
  - You want to process data whose volume is less than 1 TB.
- PyODPS DataFrame
  - You want to use MaxCompute to schedule jobs. The PyODPS DataFrame API compiles operations on DataFrames to MaxCompute SQL statements. If you want to schedule jobs by using MaxCompute, we recommend that you use the PyODPS DataFrame API.
  - You want to schedule jobs in a more stable environment. The PyODPS DataFrame API compiles operations to MaxCompute SQL statements for execution. MaxCompute is stable, which means that PyODPS is stable. Mars is new and less stable. Therefore, we recommend that you use the PyODPS DataFrame API if you require high stability.
  - If you want to process data whose volume is larger than 1 TB, we recommend that you use the PyODPS DataFrame API.

Differences between Mars and PyODPS

- API
  - Mars

    The Mars DataFrame API is fully compatible with pandas. The Mars tensor API is compatible with NumPy. The Mars learn API is compatible with scikit-learn.
  - PyODPS

    PyODPS provides only the DataFrame API, which is different from the pandas API.
- Indexing

- Mars

  The Mars DataFrame API supports operations based on indexes, including row indexes and column indexes. Example:

```
In [1]: import mars.dataframe as md
In [5]: import mars.tensor as mt
In [7]: df = md.DataFrame(mt.random.rand(10, 3), index=md.date_range('2020-5-1', periods=10))
In [9]: df.loc['2020-5'].execute()
Out[9]:
                   0         1         2
2020-05-01  0.061912  0.507101  0.372242
2020-05-02  0.833663  0.818519  0.943887
2020-05-03  0.579214  0.573056  0.319786
2020-05-04  0.476143  0.245831  0.434038
2020-05-05  0.444866  0.465851  0.445263
2020-05-06  0.654311  0.972639  0.443985
2020-05-07  0.276574  0.096421  0.264799
2020-05-08  0.106188  0.921479  0.202131
2020-05-09  0.281736  0.465473  0.003585
2020-05-10  0.400000  0.451150  0.956905
```

- PyODPS

  PyODPS does not support index-based operations.

- Data order
  - Mars

    After a Mars DataFrame is created, it maintains the data order. The Mars DataFrame API provides time series methods such as `shift`, and missing value handling methods such as `ffill` and `bfill`.

```
In [3]: df = md.DataFrame([[1, None], [None, 1]])
In [4]: df.execute()
Out[4]:
     0    1
0  1.0  NaN
1  NaN  1.0
In [5]: df.ffill().execute() # Fill the missing value with the value in the previous row.
Out[5]:
     0    1
0  1.0  NaN
1  1.0  1.0
```

  - PyODPS

    PyODPS processes and stores data by using MaxCompute, which does not maintain the data order. Therefore, PyODPS does not ensure the data order or support time series methods.

- Execution
  - Mars

    Mars consists of a client and a distributed execution layer. You can call the `o.create_mars_cluster` method to create a Mars cluster in MaxCompute and submit computing jobs to the Mars cluster. This process greatly reduces the costs for scheduling. Mars outperforms PyODPS in processing smaller amounts of data.

- PyODPS

  PyODPS is a client and does not contain any servers. When you use the PyODPS DataFrame API, the system compiles the operations to MaxCompute SQL statements for execution. Therefore, the operations supported by the PyODPS DataFrame API depend on MaxCompute SQL. Every time you call the `execute` method, a MaxCompute job is submitted for the cluster to schedule.

Usage notes

Mars is a unified distributed computing framework based on tensors. Mars can use parallel and distributed computing technologies to accelerate data processing for Python data science libraries such as NumPy, pandas and scikit-learn.

Mars provides the following common APIs:

- Mars tensor

  The Mars tensor API mimics the NumPy API and can process large multidimensional arrays, which are also called tensors. The following code shows an example of how to use the Mars tensor API:

```
import mars.tensor as mt
a = mt.random.rand(10000, 50)
b = mt.random.rand(50, 5000)
a.dot(b).execute()
```

- Mars DataFrame

  The Mars DataFrame API mimics the pandas API and can process and analyze a large amount of data. The following code shows an example of how to use the Mars DataFrame API:

```
import mars.dataframe as md
ratings = md.read_csv('Downloads/ml-20m/ratings.csv')
movies = md.read_csv('Downloads/ml-20m/movies.csv')
movie_rating = ratings.groupby('movieId', as_index=False).agg({'rating': 'mean'})
result = movie_rating.merge(movies[['movieId', 'title']], on='movieId')
result.sort_values(by='rating', ascending=False).execute()
```

- Mars learn

  The Mars learn API mimics the scikit-learn API. The Mars learn API can be integrated with TensorFlow, PyTorch, and XGBoost. The following code shows an example of how to use the Mars learn API:

```
import mars.dataframe as md
from mars.learn.neighbors import NearestNeighbors
df = md.read_csv('data.csv')
nn = NearestNeighbors(n_neighbors=10)
nn.fit(df)
neighbors = nn.kneighbors(df).fetch()
```

Reference

- Mars on GitHub
- Mars documentation
- Mars column

1.23.4.2. Preparations

This topic describes how to prepare the Mars runtime environment.

To run Mars in MaxCompute, you must prepare the Mars runtime environment by using one of the following methods:

- DataWorks
  i. Create a DataWorks PyODPS 3 node, which provides features of PyODPS and Mars.

     You can run the following commands in the new PyODPS 3 node to check the versions of PyODPS and Mars. Make sure that the versions meet the requirements.

     ```
     from odps import __version__ as odps_version
     from mars import __version__ as mars_version
     print(odps_version)
     print(mars_version)
     ```

     odps_version: the version of PyODPS. Make sure that the PyODPS version is V0.9.3.1 or later. mars_version: the version of Mars. Make sure that the Mars version is V0.4.4 or later.

  ii. Initialize a MaxCompute entry point.

     You can use the MaxCmpute entry point provided by the DataWorks PyODPS 3 node.

- Other environments
  i. Install pip. After pip is installed, install PyODPS and Mars by running the pip install command in CLI, such as the Command Prompt in Windows. For more information about how to install pip, see Installation in the pip documentation. The following commands show an example of how to use the pip install command:

     ```
     pip install -U pip  # Optional. Make sure that pip is in the latest version.
     pip install pyodps -i https://mirrors.aliyun.com/pypi/simple/ # Install the latest version of
     PyODPS. In the command, https://mirrors.aliyun.com/pypi/simple/ is the URL of the Python Packa
     ge Index (PyPI) mirror that Alibaba Cloud provides to accelerate package download.
     pip install pymars -i https://mirrors.aliyun.com/pypi/simple/ # Install the latest version of
     Mars.
     pip install protobuf -i https://mirrors.aliyun.com/pypi/simple/ # Install the latest version o
     f protocol buffers.
     pip install pyarrow -i https://mirrors.aliyun.com/pypi/simple/ # Optional. Install the latest
     version of PyArrow to accelerate job execution in Mars.
     ```

  ii. Initialize a MaxCompute entry point.

     You must use your AccessKey ID and AccessKey secret to initialize the MaxCompute entry point. For more information about how to initialize a MaxCompute entry point, see PyODPS: ODPS Python SDK and data analysis framework.

1.23.4.3. Usage notes

This topic describes how to perform operations on Mars clusters, read and write MaxCompute tables, and obtain the URLs of Mars UI, Logview, and Jupyter Notebook.

For more information about how to develop Mars jobs, see Mars.

Mars cluster operations

- Create a Mars cluster

  Run the following commands to create a Mars cluster. This process takes a while to complete.

  ```
  from odps import options
  options.verbose = True
  # If the preceding commands have been configured on the DataWorks PyODPS 3 node, you do not need t
  o run the two commands.
  client = o.create_mars_cluster(5, 4, 16, min_worker_num=3)
  ```

  where:

  - 5: the number of worker nodes in the cluster. In this example, the cluster consists of five worker nodes.

- 4: the number of CPU cores for each worker node. In this example, each worker node has four CPU cores.
- 16: the memory size of each worker node. In this example, each worker node has 16 GB of memory.

  > **Note**
  > - The memory size that you request for each worker node must be greater than 1 GB. The optimal ratio of CPU cores to the memory size is 1:4. For example, configure a worker node with 4 CPU cores and 16 GB of memory.
  > - You can create a maximum of 30 worker nodes. If the number of worker nodes exceeds the upper limit, the image server may be overloaded. If more than 30 workers are required, submit a ticket.

- min_worker_num: the minimum number of worker nodes that must be started for the system to return a client object. If this parameter is set to 3, the system returns a client object after the three worker nodes are started.

If you set `options.verbose` to True when you create a Mars cluster, the URLs of Logview, Mars UI, and Jupyter Notebook of the MaxCompute instance are displayed in the command output. You can use the Mars UI to connect to Mars clusters and query the status of clusters and jobs.

- Job

  When you create a Mars cluster, the cluster creates a default session that connects to the cluster. You can call the `.execute()` method to submit a job to the cluster and run the job in the default session.

```
import mars.dataframe as md
import mars.tensor as mt
md.DataFrame(mt.random.rand(10, 3)).execute()  # Call the .execute() method to submit the job to t
he created cluster.
```

- Stop and release a cluster

  A Mars cluster is automatically released three days after it is created. If you no longer require a Mars cluster, you can call the `client.stop_server()` method to release the cluster.

```
client.stop_server()
```

Read and write operations on MaxCompute tables

Mars can directly read and write MaxCompute tables.

- Read MaxCompute tables

  Mars uses the `o.to_mars_dataframe` method to read a MaxCompute table and return a Mars DataFrame.

```
In [1]: df = o.to_mars_dataframe('test_mars')
In [2]: df.head(6).execute()
Out[2]:
       col1  col2
0         0     0
1         0     1
2         0     2
3         1     0
4         1     1
5         1     2
```

- Write MaxCompute tables

  Mars calls the `o.persist_mars_dataframe(df, 'table_name')` method to save a Mars DataFrame as a MaxCompute table.

```
In [3]: df = o.to_mars_dataframe('test_mars')
In [4]: df2 = df + 1
In [5]: o.persist_mars_dataframe(df2, 'test_mars_persist')  # Save the Mars DataFrame as a MaxComp
ute table.
In [6]: o.get_table('test_mars_persist').to_df().head(6)  # Call the PyODPS DataFrame API operatio
n to query data.
       col1  col2
0         1     1
1         1     2
2         1     3
3         2     1
4         2     2
5         2     3
```

- Use the Jupyter Notebook of a Mars cluster

  > **Note** The Jupyter Notebook can be used only if `with_notebook=True` is specified in `create_mars_cluster`.

  When you create a Jupyter Notebook document, a session is automatically created to submit jobs to the Mars cluster. Therefore, session creation does not need to be shown in the Jupyter Notebook document.

```
import mars.dataframe as md
md.DataFrame(mt.random.rand(10, 3)).sum().execute() # Call the .execute() method in the Jupyter No
tebook to submit the job to the current cluster. Therefore, session creation does not need to be s
hown in the Jupyter Notebook document.
```

  > **Note**
  > - The Jupyter Notebook document is not automatically saved. We recommend that you manually save the Jupyter Notebook document as required.
  > - You can connect your Jupyter Notebook to an existing Mars cluster. For more information, see Use an existing Mars cluster.

Other operations

- Use an existing Mars cluster
  - Recreate an existing Mars cluster based on the instance ID.

```
client = o.create_mars_cluster(instance_id=**instance-id**)
```

  - To use an existing Mars cluster, create a Mars session to visit the URL of the Mars UI.

```
from mars.session import new_session
new_session('**URL of the Mars UI**').as_default() # Set the created session as the default sess
ion.
```

- Obtain the URL of the Mars UI

  If you set `options.verbose` to True when you create a Mars cluster, the URL of the Mars UI is automatically displayed in the command output. You can use `client.endpoint` to obtain the URL of the Mars UI.

```
print(client.endpoint)
```

- Obtain the Logview URL of an instance

If you set `options.verbose` to True when you create a Mars cluster, the Logview URL is automatically displayed in the command output. You can also use `client.get_logview_address()` to obtain the Logview URL.

```
print(client.get_logview_address())
```

- Obtain the Jupyter Notebook URL

  If you set `options.verbose` to True when you create a Mars cluster, the Jupyter Notebook URL is automatically displayed in the command output. You can also use `client.get_notebook_endpoint()` to obtain the Jupyter Notebook URL.

  ```
  print(client.get_notebook_endpoint())
  ```

1.24. MaxCompute FAQ

This topic provides answers to some frequently asked questions about MaxCompute.

How do I check MaxCompute resource usage when SQL statements are slowly executed?

Log on to the host where MaxCompute AG is deployed as user admin and perform the following steps:

1. Run the following command to sort the hosts in ascending order of the number of remaining resources on each host:

   ```
   r tfrl|sed 's/,//g'|sort -t "|" -k2 -n
   ```

2. Run the following command to view resource details of each host and total cluster resources in MaxCompute:

   ```
   r ttrl|sed 's/,//g'
   ```

3. Calculate the percentage of remaining resources to all resources in the cluster based on the statistics obtained in the preceding steps. Then, obtain the resource usage of MaxCompute.

Some jobs submitted by a project are slowly executed even if remaining resources in a MaxCompute cluster are sufficient. What do I do?

A possible cause is that the resources for the quota group where the project resides 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 host where MaxCompute AG is deployed as user admin and run the following command to check the resource usage of the quota group:

   ```
   r quota
   ```

2. If resources in the quota group are exhausted, you can use Apsara Big Data Manager (ABM) to modify the settings of the quota group.

How do I modify the settings of a quota group?

1. Log on to the host where MaxCompute AG is deployed as user admin and run the following command to create or modify quotas for the quota group.

   ```
   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 exists, the quotas are modified. Otherwise, quotas are created.

2. Log on to ABM to configure related settings.

How do I perform simple operations on the metadata warehouse?

1. Log on to the host where MaxCompute AG is deployed as user admin.
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?

You must install the metadata warehouse enhancement package package+view first. Before you install the package, make sure that you are the owner of a project that is granted the package installation permission. After you install the package, you can use the smart metadata warehouse. For more information, see Operations for package creators.

- **Installation**

  package+view is a metadata warehouse enhancement package. You can run the `odpscmd --config=odps_config.ini -f init.sql` command in the *system metadata warehouse* directory to complete the installation.

  > **Note**
  > - If the system asks you to re-install the package, comment out create package from the first line of odpscmd --config=odps_config.ini -f init.sql, and then run the command again.
  > - odps_config.ini is a configuration file that contains the configuration of the account. The account you configured is also the project owner of the metadata warehouse and can access the metadata warehouse.

- **Permission assignment**

  Run the following command to allow you to install the package+view package for a project, such as the p1 project:

```
odpscmd --config=odps_config.ini -e "allow project p1 to install package systables;"
```

  Run the following command to allow you to install the package+view package for all projects:

```
odpscmd --config=odps_config.ini -e "allow project * to install package systables;"
```

- **User operations**

  You can run the following command in odpscmd to install the package. Before you perform this step, make sure 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 query the description of views in the package:

```
desc package meta.systables;
```

> **Note** After the preceding operations are complete, you can use the smart metadata warehouse.

- **Views**

  To learn more about the definition of the table schema, see the related content in the view description, which is displayed 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	Authorization information of tables, UDFs, and resources
package_resources	Object information contained in the package
partitions	Partition information of a partitioned table
policies	Policies that define user or role permissions
resources	Resource information
roles	Role information
table_label_grants	Label authorization information of a table
table_labels	Label information of a table
tables	Table or view information
tasks	Job execution records
tunnels	Data upload and download records
udf_resources	Information of resources used in UDFs
udfs	Information of UDFs
user_roles	Information of associations between users and roles
users	User information

- **Notes:**

- By default, the package+view package only allows you to query metadata in the last 180 days.
- If you do not run a job on a day, the metadata warehouse does not store the metadata of the day.
- We recommend that you specify a query range to prevent a full data scan in the last 180 days.
- The metadata of the previous day can be queried only after the metadata is generated by the metadata warehouse in the early morning of the next day. In the current configuration, metadata can be queried immediately after it is generated by the metadata warehouse.
- The metadata warehouse does not provide metadata on the day when a project is created. The purpose is to obtain only the metadata of the projects with unique names.

How do I grant Java sandbox permissions?

1. Log on to AdminConsole and choose **MaxCompute Configuration > Project Management**. Select the project to which you want to grant Java sandbox permissions and double-click the project to open the property dialog box.
2. In the **ODPS Sandbox Setting** section, enter the method or class that you want to use in Sandbox Java Permissions.

   > **Note** Make sure that the content you entered is in a valid format. The following example is for reference only. Enter each item in a single line and end it with a semicolon (;).

   ```
   permission java.lang.RuntimePermission "readSystemProperty"; permission java.lang.RuntimePermiss
   ion "modifyThreadGroup"; permission java.security.AllPermission;
   ```

3. Click **Finish Modification**.

What do I do if disk space is insufficient?

In most cases, you can delete scripts to release the disk space. The most possible cause is that the root directory of MaxCompute AG or the /apsara directory occupies too much disk space. Therefore, you must delete scripts in the two directories.

How do I add a MaxCompute host to a blacklist?

1. Log on to Apsara AG as user admin. Run the following command to enable the Fuxi blacklisting 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 the host to the Fuxi blacklist:

   ```
   /apsara/deploy/rpc_wrapper/rpc.shblacklist cluster add $hostname
   ```

4. Run the following command to check that the host has been added to the Fuxi blacklist:

   ```
   /apsara/deploy/rpc_wrapper/rpc.shblacklist cluster get
   ```

How do I export data from MaxCompute?

You can use a Tunnel command to export data. You can also configure synchronization tasks in DataWorks to export data from MaxCompute to other destinations.

How do I view the MaxCompute version?

Run the following commands to obtain the version of MaxCompute:

```
cat /apsara/odps_info/version|grep odps
```

```
cat /apsara/version
```

How do I restart services in MaxCompute?

1. Run the following commands to save the configurations of resident services of MaxCompute 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 command that is not required.
# You can run 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 services in MaxCompute:

```
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 services in MaxCompute:

```
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 off MaxCompute and then power it on?

1. Run the following commands to save the configurations of resident services of MaxCompute 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 command that is not required.
# You can run 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 services in MaxCompute:

```
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 distributed operating 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 distributed operating system:

```
/home/admin/dayu/bin/allapsara start
```

7. Run the following commands to start services in MaxCompute:

```
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 a large number of resources.
2. If the check result shows that task processes occupy a large number of resources, terminate the tasks after these tasks are complete or after communication with users.

1.25. Open source features of MaxCompute

This topic describes open source features of MaxCompute.

SDK

MaxCompute provides APIs for SDK for Java and SDK for Python to create, view, and delete MaxCompute tables. You can use SDKs to edit code and perform required operations on MaxCompute

Technical support: View the official documentation or submit a ticket.

MaxCompute RODPS

MaxCompute RODPS is an R plug-in for MaxCompute. For more information, see aliyun-odps-r-plugin on GitHub.

Technical support: Leave a message or create an issue in aliyun-odps-r-plugin on GitHub.

MaxCompute JDBC

MaxCompute JDBC is an official JDBC driver provided by MaxCompute. It provides a set of interfaces to run SQL tasks for Java programs. The project is hosted in aliyun-odps-jdbc on GitHub.

Technical support: Leave a message or create an issue in aliyun-odps-jdbc 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.

Flume and OGG plug-ins are implemented based on the DataHub SDK, whereas Sqoop, 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, Kettle plug-in, and Hive Data Transfer UDTF are used to transfer data 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.

Technical support: Leave a message or create an issue in aliyun-maxcompute-data-collectors on GitHub.

2.DataWorks

2.1. User Guide

2.1.1. Log on to the DataWorks console

This topic describes how to log on to the DataWorks console.

Prerequisites

- The URL of the Apsara Uni-manager Management Console is obtained from the deployment personnel before you log on to the Apsara Uni-manager Management Console.
- We recommend that you use the Google Chrome browser.

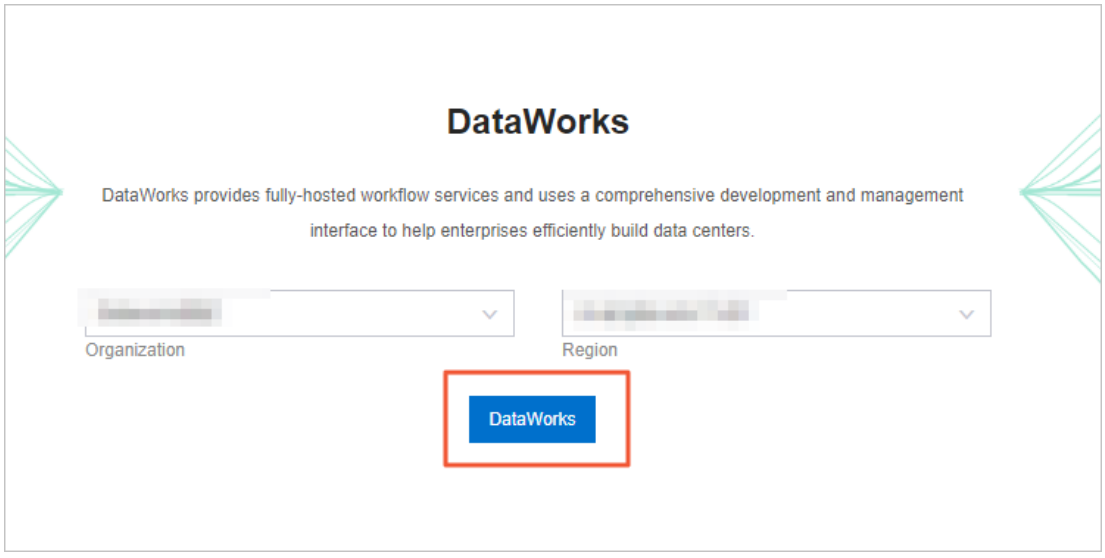
Procedure

1. In the address bar, enter the URL of the Apsara Uni-manager Management Console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password that you can use to log on to the console from the operations administrator.

   > **Note** When you log on to the Apsara Uni-manager Management Console for the first time, you must change the password of your username. Your password must meet complexity requirements. The password must be 10 to 32 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters
   > - Digits
   > - Special characters, which include ! @ # $ %

3. Click **Log On**.
4. In the top navigation bar, choose **Products > Big Data > DataWorks**.
5. On the page that appears, specify **Organization** and **Region** and click **Access as Administrator**.



> **Notice** You are not allowed to log on to the DataWorks console by using the root organization.

2.1.2. 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**.

2.1.3. Quick Start

2.1.3.1. Overview

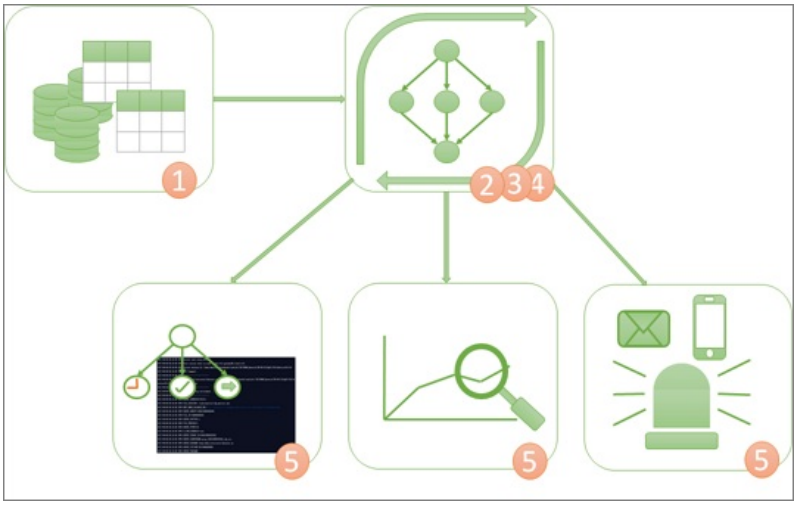
Quick Start guides you through a complete process of data analytics and O&M.

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.



2.1.3.2. 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.

2.1.3.3. 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.

2.1.3.4. Create a synchronization node

This topic describes how to create a synchronization node to synchronize data from MaxCompute to MySQL.

Background information

In DataWorks, Data Integration can be used to periodically synchronize the business data generated in a business system to a workspace. After the data is computed in SQL nodes, Data Integration periodically synchronizes the computing results to your specified data source for further display or use.

Add a data source

> **Note** Only the workspace administrator can add data sources. Members of other roles can only view data sources.

1. Log on to the DataWorks console. On the DataSudio page, click the icon in the upper-left corner and choose **All Products > Data Integration**.
2. In the left-side navigation pane, click **Connection** to go to the **Data Source** page. On the Data Source page, click **New data source** in the upper-right corner.
3. In the **Add data source** dialog box, select **MySQL** in the Relational Databases section.
4. In the **Add MySQL data source** dialog box, configure the parameters. In this example, a MySQL data source is added by using the **connection string mode**.

Parameter	Description
Data Source Type	The type of the data source. In this example, set the value to **Connection string mode**.
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_) and must start with a letter.
Data source description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
JDBC URL	The Java Database Connectivity (JDBC) URL of the database, in the format of `jdbc:mysql://ServerIP:Port/Database`.

Parameter	Description
User name	The username that you use to connect to the database. **Note** You must enter the information of your MySQL database.
Password	The password that you use to connect to the database.

5. Click **Test connectivity**.
6. If the connectivity test is successful, click **Complete**.

Create a table 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
);
```

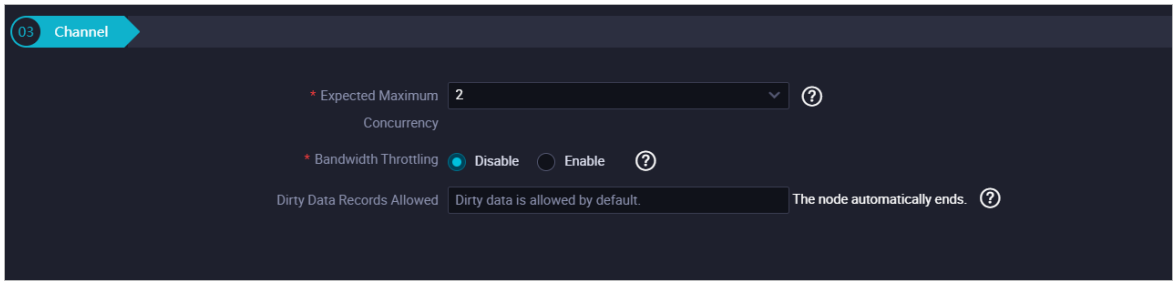
After the table is created, execute the `desc odps_result;` statement to view the table details.

Create and configure a synchronization node

This section describes how to create and configure the synchronization node write_result to synchronize data in the result_table table to your MySQL database. To synchronize the data, perform the following steps:

1. Go to the **DataStudio** page and create the synchronization node write_result.
2. Configure the insert_data node as the ancestor node of the write_result node.
3. Select **odps_first** of the **ODPS** type as the source and select the result_table as the source table.
4. Select the odps_result table of the newly added MySQL data source as the destination table.
5. Configure field mappings between the source and destination tables in the Mappings section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination 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.
6. In the Channel section, configure the parameters.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization node on the codeless user interface (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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

7. Preview and save the configuration.

   After the node is configured, you can scroll up and down to view the node configuration. Verify that the configuration is correct and click the **Save** icon in the top toolbar.

Commit the synchronization node

Return to the workflow after you save the synchronization node. Click the **Submit** icon in the top toolbar to commit the synchronization node. The scheduling system automatically and periodically runs the node from the next day based on the properties configured for the node.

What to do next

You have learned how to create a synchronization node to synchronize data to a specific data source. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure properties and dependencies for a synchronization node. For more information, see Configure recurrence and dependencies for a node.

2.1.3.5. 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

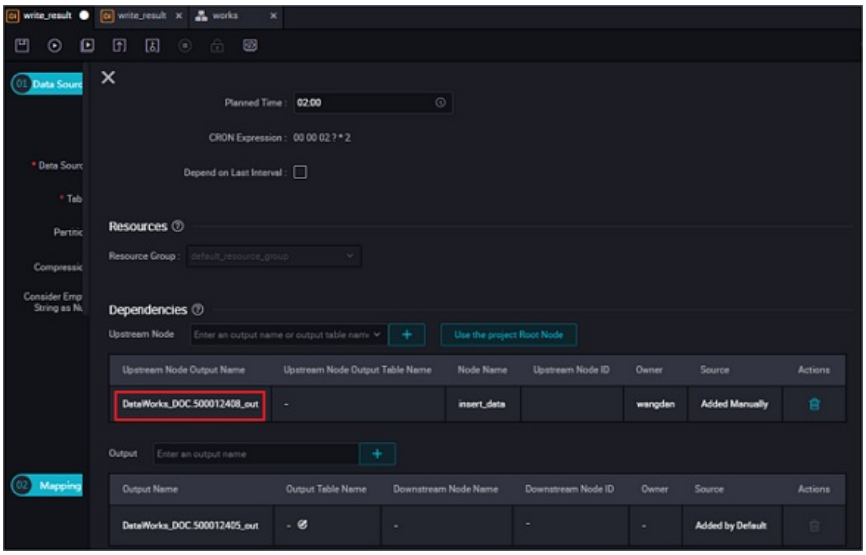
1. After the sync node write_result is created, double-click the sync node to configure it.
2. Click the **Properties** tab in the right-side navigation pane to configure recurrence for the sync node.

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.

Parameter	Description
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.

2.1.3.6. 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.

2.1.4. Data Integration

2.1.4.1. Overview

Data Integration is a stable, efficient, and scalable data synchronization service. It is designed to rapidly and stably migrate and synchronize data between a wide range of heterogeneous data sources in complex network environments.

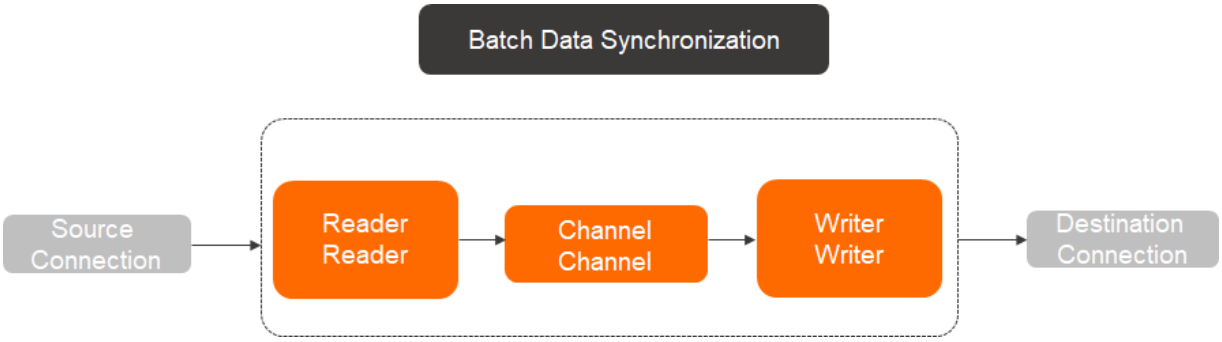
Limits

- Data Integration can synchronize structured, semi-structured, and unstructured data. Structured data sources include ApsaraDB RDS and PolarDB-X. Unstructured data, such as Object Storage Service (OSS) objects and text files, must be converted to structured data. Data Integration can synchronize only the data that can be abstracted to two-dimensional logical tables to MaxCompute. It cannot synchronize the unstructured data that cannot be converted to structured data, such as MP3 files that are stored in OSS, to MaxCompute.
- Data Integration supports data synchronization and exchange in the same region or across regions.

  Data can be transmitted between some regions by using the classic network, but the network connectivity is not ensured. If the transmission fails over the classic network, we recommend that you transmit data by using the Internet.
- Data Integration supports only data synchronization but not data consumption.

Batch data synchronization

Data Integration can be used to synchronize large amounts of offline data. Data Integration facilitates data transmission between diverse structured and semi-structured data sources. It provides readers and writers for the supported data sources, defines a transmission channel between the sources and destinations, and defines datasets, based on simplified data types.



Supported data sources

- Relational databases: MySQL, SQL Server, PostgreSQL, Oracle, Dameng, PolarDB-X, PolarDB, HybridDB for MySQL, AnalyticDB for PostgreSQL, AnalyticDB for MySQL 2.0, and AnalyticDB for MySQL 3.0
- Big data storage: MaxCompute and DataHub
- Semi-structured storage: OSS, Hadoop Distributed File System (HDFS), and File Transfer Protocol (FTP)
- NoSQL: MongoDB, Memcache, Redis, and Tablestore
- Message queue: LogHub
- Graph compute engine: Graph Compute

For more information, see Supported data sources.

> **Note** The configurations for data sources vary greatly. You can view the specific parameters that need to be configured when you configure data sources and synchronization nodes.

Development modes of synchronization nodes

You can develop synchronization nodes in the following modes:

- **Codeless UI**: Data Integration provides step-by-step instructions to help you configure a synchronization node. This mode is easy to use but provides only limited features.
- **Code editor**: You can write a JSON script to create a synchronization node. This mode provides advanced features to facilitate flexible configuration.

> **Note**
> - The code generated for a synchronization node on the codeless UI can be converted to a script. This conversion is irreversible.
> - Before you write code, you must configure data sources and create a destination table.

Network types

A data source can reside on the classic network or in a virtual private cloud (VPC). The support for a data source in a data center has been planned and will be available soon.

- Classic network: a network that is deployed by Alibaba Cloud, which is shared with other tenants. This network is easy to use.
- VPC: a network that is created on Alibaba Cloud, which can be used by only one Apsara Stack tenant account. You have full control over your VPC. For example, you can customize the IP address range, divide the VPC into multiple subnets, and configure route 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. Data Integration provides the feature to automatically detect the reverse proxy for the following data sources based on the wide deployment of VPCs: ApsaraDB RDS for MySQL, ApsaraDB RDS for PostgreSQL, ApsaraDB RDS for SQL Server, PolarDB, PolarDB-X, HybridDB for MySQL, AnalyticDB for PostgreSQL, and AnalyticDB for MySQL 3.0. This feature frees you from purchasing an Elastic Compute Service (ECS) instance in your VPC to configure synchronization nodes for these data sources. Instead, Data Integration automatically uses this feature to provide network connectivity to these data sources.

When you configure synchronization nodes for other Alibaba Cloud data sources in a VPC, such as ApsaraDB RDS for 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 connect to the data sources.

- Data center: a data center deployed by yourself, which is isolated from the Alibaba Cloud network.

> **Note** You can access data sources 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 the Internet.

Terms

- Parallelism

  Parallelism indicates the maximum number of parallel threads that the synchronization node uses to read data from or write data to data sources.

- Bandwidth throttling

  Bandwidth throttling indicates that a maximum transmission rate is specified for a synchronization 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 VARCHAR-type data 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 data.

- Data source

  A data source is a source from which data is processed by DataWorks. A data source can be a database or a data warehouse. DataWorks supports various types of data sources and data type conversion during synchronization.

2.1.4.2. 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 *View the statistics on the Overview page*.

2.1.4.3. 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 ReadPreferenceSer
verSelector{readPreference=primary}. Client view of cluster state is {type=UNKNOWN, servers=[(xx
xxxxxxxx), type=UNKNOWN, state=CONNECTING, exception={com.mongodb.MongoSocketReadException: Prem
aturely 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.

2.1.4.4. Data sources

2.1.4.4.1. 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	Configure Hive Reader	Hive Writer
Hologres	Supported	Supported
HybridDB for MySQL	Supported	Supported
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

Data store	Reader	Writer
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	Configure Vertica Writer

2.1.4.4.2. 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.

2.1.4.4.3. Synchronization data monitoring

The Sync Data Monitoring page displays the total number of synchronization node instances for different data sources and the instance details based on the selected workspace and time range.

The cut-off time of the data to display is Current hour:00:00. 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.
2. Click the 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 synchronization node instances for different data sources and the instance details.

- View summary data by data source type

  The Sync Instances: Source section displays the summary data of sources, and the Sync Instances: Target section displays the summary data of destinations. In the Sync Instances: Source section, the value of MaxCompute is 1. This indicates that one synchronization node instance whose source is MaxCompute is run in the selected time range.

- View instance details

  The Sync Instances section displays the details of all the synchronization 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 type, and destination type. Then, sort search results based on the number of synchronized data entries or the size of synchronized data.

2.1.4.4.4. 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.

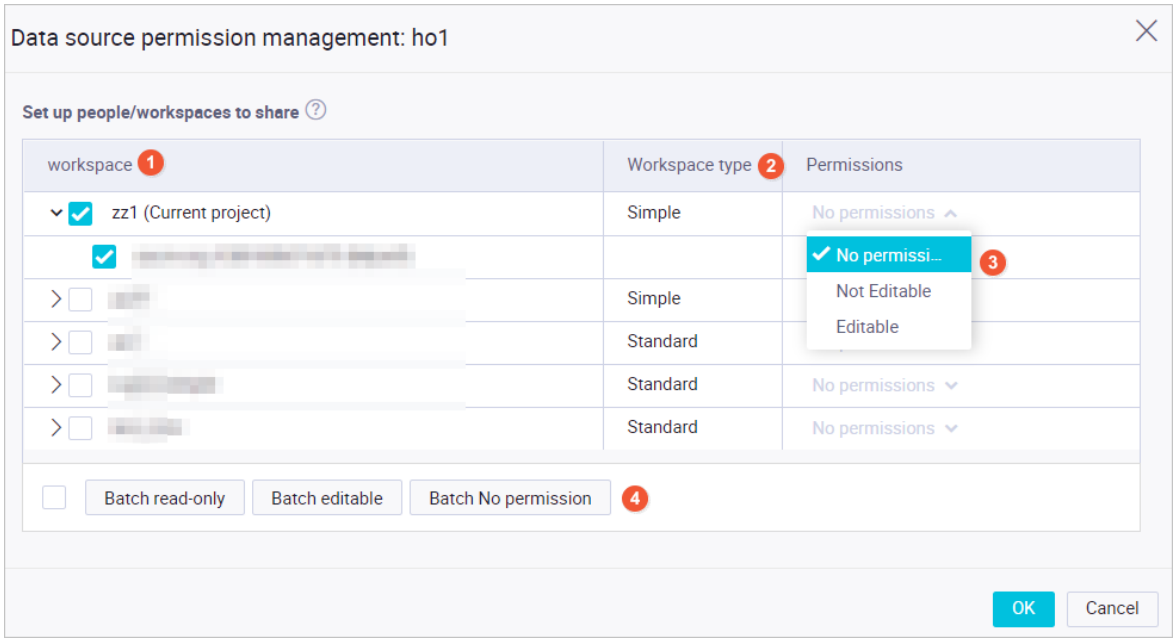
Context

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
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**.

No.	Parameter	Description
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.

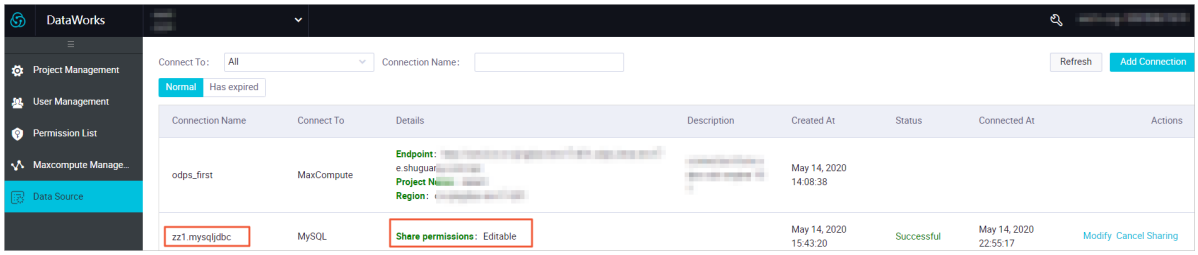
View shared connections

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.

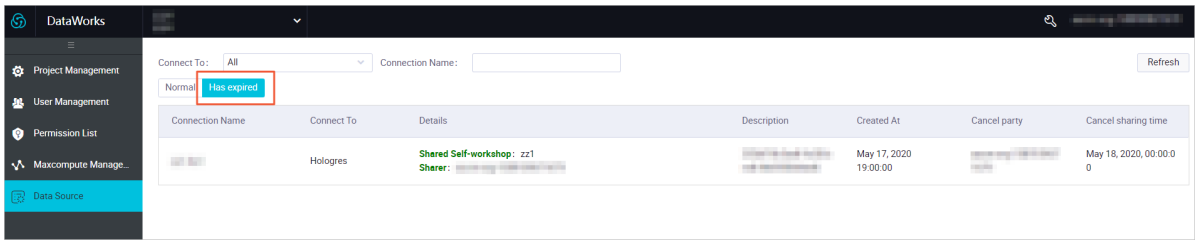


If the current user has the Editable permission on the connection, **Modify** appears in the Actions column.

- **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.



2.1.4.4.5. Configure a MySQL data source

DataWorks provides MySQL Reader and MySQL Writer for you to read data from and write data to MySQL data sources. You can use the codeless user interface (UI) or code editor to configure synchronization nodes for MySQL data sources.

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 **New data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **MySQL** in the Relational Databases section.
4. In the **Add MySQL data source** dialog box, configure the parameters.

   You can set the Data source type parameter to **Alibaba Cloud instance mode** or **Connection string mode** for a MySQL data source.

   - The following table describes the parameters that appear after you set **Data source type** to **Alibaba Cloud instance mode**.

Parameter	Description
Data source type	The type of the data source. Set the parameter to **Alibaba Cloud instance mode**.

Parameter	Description
Data Source Name	The name of the data source. The name must contain letters, digits, and underscores (_) and start with a letter.
Description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
RDS instance ID	The ID of your ApsaraDB RDS for MySQL instance. You can view the ID in the ApsaraDB 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 ApsaraDB RDS for MySQL database.
User name	The username that is used to connect to the database.
Password	The password that is used to connect to the database.

- The following table describes the parameters that appear after you set **Data source type** to **Connection string mode**.

Parameter	Description
Data source type	The type of the data source. Set the parameter to **Connection string mode**.
Data Source Name	The name of the data source. The name must contain letters, digits, and underscores (_) and start with a letter.
Description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
JDBC URL	The Java Database Connectivity (JDBC) URL of the database, in the format of `jdbc:mysql://ServerIP:Port/Database`.
User name	The username that is used to connect to the database.
Password	The password that is used to connect to the database.
Whether the data source is in a VPC	Specifies whether to connect to the data source by using a VPC. If you cannot connect to the ECS instance where the data source is located but can connect to the VPC to which the data source belongs, select the check box.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

What's next

You have learned how to configure a MySQL data source. You can proceed to subsequent tutorials. The subsequent tutorials describe how to configure MySQL Reader or MySQL Writer. For more information, see Configure the MySQL reader or Configure MySQL Writer.

2.1.4.4.6. Configure an SQL Server data source

DataWorks provides SQL Server Reader and SQL Server Writer for you to read data from and write data to SQL Server data sources. You can use the codeless user interface (UI) or code editor to configure synchronization nodes for SQL Server data sources.

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 **New data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **SQLServer** in the Relational Databases section.
4. In the **Add SQLServer data source** dialog box, configure the parameters.

   You can set the Data source type parameter to **Alibaba Cloud instance mode** or **Connection string mode** for an SQL Server data source.

   - The following table describes the parameters that appear after you set **Data source type** to **Alibaba Cloud instance mode**.

Parameter	Description
Data source type	The type of the data source. Set the parameter to **Alibaba Cloud instance mode**.
Data Source Name	The name of the data source. The name must contain letters, digits, and underscores (_) and start with a letter.
Description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
RDS instance ID	The ID of the ApsaraDB RDS for SQL Server instance. You can view the ID in the ApsaraDB 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.

Parameter	Description
Database name	The name of the ApsaraDB RDS for SQL Server database.
User name	The username that is used to connect to the database.
Password	The password that is used to connect to the database.

- The following table describes the parameters that appear after you set **Data source type** to **Connection string mode**.

Parameter	Description
Data source type	The type of the data source. Set the parameter to **Connection string mode**.
Data Source Name	The name of the data source. The name must contain letters, digits, and underscores (_) and start with a letter.
Description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
JDBC URL	The Java Database Connectivity (JDBC) URL of the database, in the format of `jdbc:sqlserver://ServerIP:Port;DatabaseName=Database`.
User name	The username that is used to connect to the database.
Password	The password that is used to connect to the database.
Whether the data source is in a VPC	Specifies whether to connect to the data source by using a VPC. If you cannot connect to the ECS instance where the data source is located but can connect to the VPC to which the data source belongs, select the check box.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

What's next

You have learned how to configure an SQL Server data source. You can proceed to subsequent tutorials. The subsequent tutorials describe how to configure SQL Server Reader or SQL Server Writer. For more information, see Configure SQL Server Reader or Configure SQL Server Writer.

2.1.4.4.7. Configure a PostgreSQL data source

DataWorks provides PostgreSQL Reader and PostgreSQL Writer for you to read data from and write data to PostgreSQL data sources. You can use the codeless user interface (UI) or code editor to configure synchronization nodes for PostgreSQL data sources.

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 **New data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **PostgreSQL** in the Relational Databases section.
4. In the **Add PostgreSQL data source** dialog box, configure the parameters.

   You can set the Data source type parameter to **Alibaba Cloud instance mode** or **Connection string mode** for a PostgreSQL data source.

   - The following table describes the parameters that appear after you set **Data source type** to **Alibaba Cloud instance mode**.

Parameter	Description
Data source type	The type of the data source. Set the parameter to **Alibaba Cloud instance mode**.
Data Source Name	The name of the data source. The name must contain letters, digits, and underscores (_) and start with a letter.
Description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
RDS instance ID	The ID of your ApsaraDB RDS for PostgreSQL instance. You can view the ID in the ApsaraDB RDS console.
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 the ID on the security settings page in the ApsaraDB RDS console.
Database name	The name of the ApsaraDB RDS for PostgreSQL database.
User name	The username that is used to connect to the database.
Password	The password that is used to connect to the database.

   - The following table describes the parameters that appear after you set **Data source type** to **Connection string mode**.

Parameter	Description
Data source type	The type of the data source. Set the parameter to **Connection string mode**.
Data Source Name	The name of the data source. The name must contain letters, digits, and underscores (_) and start with a letter.

Parameter	Description
Description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
JDBC URL	The Java Database Connectivity (JDBC) URL of the database, in the format of `jdbc:postgresql://ServerIP:Port/Database`.
User name	The username that is used to connect to the database.
Password	The password that is used to connect to the database.
Whether the data source is in a VPC	Specifies whether to connect to the data source by using a VPC. If you cannot connect to the ECS instance where the data source is located but can connect to the VPC to which the data source belongs, select the check box.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.8. 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 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 **New data source** 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.

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.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:oracle:thin:@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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.9. 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 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 **New data source** 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.

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.
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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.10. 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 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 **New data source** 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.

Parameter	Description
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 connectivity**.
6. After the data source 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.

2.1.4.4.11. 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 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 **New data source** 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://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.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.12. 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 **New data source** 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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.13. 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 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 **New data source** 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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.14. 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 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 **New data source** 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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.15. 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 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 **New data source** 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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.16. 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 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 **New data source** 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.

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.
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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.17. 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 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 **New data source** 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
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.
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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.18. 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 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 **New data source** 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
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
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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.19. 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 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 **New data source** 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.

Parameter	Description
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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.20. 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 **New data source** 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.

Parameter	Description
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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.21. 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 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 **New data source** 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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.22. Configure a Hologres data source

DataWorks provides Hologres Reader and Hologres Writer for you to read data from and write data to Hologres data sources. You can use the codeless user interface (UI) or code editor to configure synchronization nodes for Hologres data sources.

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 **New data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **Hologres** in the Big Data Storage section.
4. In the **Add Hologres data source** dialog box, configure the parameters.

Parameter	Description
Data source type	The type of the data source. The value of this parameter can be only **Alibaba Cloud instance mode**.

Parameter	Description
Data Source Name	The name of the data source. The name must contain letters, digits, and underscores (_) and start with a letter.
Description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
Instance ID	The ID of the Hologres instance.
Database name	The name of the Hologres database.
AccessKey ID	The AccessKey ID of the account that is used to access the Hologres database.
AccessKey Secret	The AccessKey secret that corresponds to the AccessKey ID. The AccessKey secret is equivalent to a logon password.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.23. Configure a Hive data source

DataWorks provides Hive Reader and Hive Writer for you to read data from and write data to Hive data sources. You can use the codeless user interface (UI) or code editor to configure synchronization nodes for Hive data sources.

Limits

DataWorks supports only Hive 2.3.3 and Hive 2.3.5.

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 **New data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **Hive** in the Big Data Storage Systems section.
4. In the **Add Hive data source** dialog box, configure the parameters.

Parameter	Description
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_) and must start with a letter.

Parameter	Description
Data source description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only if the workspace is in standard mode.
HIVE JDBC URL	The Java Database Connectivity (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.
HIVE Login	The mode that is used to connect to the Hive database. Valid values: **Login with username and password** and **Anonymous**. If you select **Login with username and password**, you must specify the **HIVE username** and **HIVE password** parameters.
metastoreUris	The Uniform Resource Identifiers (URIs) of the Hive metadatabase, in the format of `thrift://ip1:port1,thrift://ip2:por2`.
defaultFS	The address of the NameNode node in the Active state in the Hadoop Distributed File System (HDFS), in the format of `hdfs://ip:port`.

Extended parameters	The advanced parameters of Hive, such as parameters related to high availability. The following code provides 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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.24. Add an OSS data source

Alibaba Cloud Object Storage Service (OSS) is a secure, highly reliable cloud storage service that allows you to store 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 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 **New data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **OSS** in the Semi-Structured Storage Systems section.
4. In the **Add OSS data source** dialog box, set the parameters.

Parameter	Description
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_). It must start with a letter.
Data source description	The description of the data source. The description can be up to 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only for workspaces in standard mode.
Endpoint	The OSS endpoint, which varies by region. Enter different endpoints if you want to access OSS buckets that reside in different regions. ○ If you want to read data from OSS, the endpoint is in the following format: `http://my_bucket_name.aliyuncs.com`. ○ If you want to write data to OSS, the endpoint is in the following format: `http://aliyuncs.com`. **Note** You must add a bucket name to the endpoint if you want to test connectivity, such as `http://my_bucket_name.aliyuncs.com`.
Bucket	The name of the OSS bucket. A bucket is a container for storing objects. You can specify one or more buckets and add one or more objects to each bucket. You can search for objects in a bucket in a data synchronization node only after the bucket is specified here.
AccessKey ID	The AccessKey ID.
AceessKey Secret	The AccessKey secret.

Notice If data in OSS is stored in a CSV file, the file must be a standard CSV file. For example, if the data in a column is enclosed in two single quotation marks ('), you must replace single quotation marks with double quotation marks ("). Otherwise, the file may be incorrectly split.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.25. 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 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 **New data source** 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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.26. 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 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 **New data source** 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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.27. Configure a MongoDB data source

MongoDB is a document-oriented database that is second only to Oracle and MySQL. DataWorks provides MongoDB Reader and MongoDB Writer for you to read data from and write data to MongoDB data sources. You can use the code editor to configure synchronization nodes for MongoDB data sources.

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 **New data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **MongoDB** in the NoSQL section.
4. In the **Add MongoDB data source** dialog box, configure the parameters.

   You can set the Data source type parameter to **Alibaba Cloud instance mode** or **Connection string mode**.

   - **Alibaba Cloud instance mode**: In most cases, the classic network is used to access MongoDB data sources of this type. You can use the classic network to access MongoDB data sources that are in the same region as the classic network. However, access to MongoDB data sources that are in different regions from the classic network is not ensured.

Parameter	Description
Data source type	The type of the data source. Set this parameter to **Alibaba Cloud instance mode**. **Note** If you have not assigned the default role to Data Integration, log on to the Resource Access Management (RAM) console by using your Apsara Stack tenant account and perform authorization. Then, refresh this configuration page.
Data Source Name	The name of the data source. The name must contain letters, digits, and underscores (_) and start with a letter.
Description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
Region	The region where your ApsaraDB for MongoDB instance resides.
Instance ID	The ID of your 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 create a database and specify a username and a password for the database in this console.
User name	The username that is used to connect to the database.
Password	The password that is used to connect to the database.

   - **Connection string mode**: In most cases, the Internet is used to access MongoDB data sources of this type, which may cost you fees.

Parameter	Description

Parameter	Description
Data source type	The type of the data source. Set this parameter to **Connection string mode**.
Data Source Name	The name of the data source. The name must contain letters, digits, and underscores (_) and start with a letter.
Description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
Access address	The endpoint in the `host:port` format. To add an endpoint, click **Add access address** and specify the endpoint to add. To add more endpoints, repeat the preceding operation. **Note** You can add public or internal endpoints. However, the added endpoints must be of the same type.
Database name	The name of the database.
User name	The username that is used to connect to the database.
Password	The password that is used to connect to the database.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.28. 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 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 **New data source** 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 connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.29. Configure a Redis data source

DataWorks provides Redis Reader and Redis Writer for you to read data from and write data to Redis data sources. You can use the code editor to configure synchronization nodes for Redis data sources.

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 **New data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **Redis** in the NoSQL section.
4. In the **Add Redis data source** dialog box, configure the parameters.

   You can set the Data source type parameter to **Alibaba Cloud instance mode** or **Connection string mode** for a Redis data source.

   - The following table describes the parameters that appear after you set **Data source type** to **Alibaba Cloud instance mode**.

Parameter	Description
Data source type	The type of the data source. Set the parameter to **Alibaba Cloud instance mode**.

Parameter	Description
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_) and must start with a letter.
Data source description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
Redis instance ID	The ID of the ApsaraDB for Redis instance. You can view the ID in the ApsaraDB for Redis console.
Redis access password	The password that is used to connect to the ApsaraDB for Redis instance. Leave it empty if no password is required.

- The following table describes the parameters that appear after you set **Data source type** to **Connection string mode**.

Parameter	Description
Data source type	The type of the data source. Set the parameter to **Connection string mode**.
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_) and must start with a letter.
Data source description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
Server address	The server address, in the format of `host:port`. Click **Add server address** to add a server address in the format of `host:port`.
Redis access password	The password that is used to connect to the Redis data source.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.30. Configure a Tablestore data source

Tablestore is a NoSQL database service that is built on the Apsara distributed operating system. The service allows you to store and access large volumes 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 **New data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **OTS** in the NoSQL section.
4. In the **Add OTS data source** dialog box, configure the parameters.

Parameter	Description
Data Source Name	The name of the data source. The name must contain letters, digits, and underscores (_) and start with a letter.
Description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
Endpoint	The endpoint of Tablestore.
Table Store instance name	The name of the Tablestore instance.
AccessKey ID	The AccessKey ID of the account that is used to access the Tablestore instance. You can view the AccessKey ID on the **User Information** page.
AccessKey Secret	The AccessKey secret that corresponds to the AccessKey ID. The AccessKey secret is equivalent to a logon password.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.31. 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 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. On the **Data Source** page, click **Add data source** 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.
Password	The password that you can use to connect to the database.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.4.32. 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 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 **New data source** 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.

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.
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.
AceessKey Secret	The AccessKey secret for connecting to the LogHub project.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

2.1.4.5. Configure data synchronization tasks

2.1.4.5.1. Configure a synchronization node by using the codeless UI

This topic describes how to configure a synchronization node by using the codeless user interface (UI).

Procedure

1. Add data sources.
2. Create a batch synchronization node.
3. Configure a source.
4. Configure a destination.
5. Configure field mappings.
6. Configure channel control policies, such as the maximum transmission rate and the maximum number of dirty data records allowed.
7. Configure properties for the node.

Add data sources

Synchronization nodes can synchronize data between various homogeneous or heterogeneous data sources. Log on to the DataWorks console, click the icon in the upper-right corner, and then choose **All Products > Data Integration**. On the Data Integration page, click Data Source in the left-side navigation pane. On the page that appears, click New data source in the upper-right corner. In the Add data source dialog box, add a data source. For more information, see Configure a data source.

After you add a data source, you can select it when you configure a synchronization node on the DataStudio page.

> **Note**
>
> - Data Integration does not support connectivity testing for some types of data sources. For more information, see Connectivity testing.
> - If an on-premises data source does not have a public IP address or is not accessible from a network, the connectivity testing fails when you configure the data source. To resolve the connection failure, you can use a custom resource group to connect to the data source. For more information about how to create a custom resource group, see Create a custom resource group for Data Integration. If a data source is not accessible from a network, Data Integration cannot obtain the table schema of the data source. In this case, you can configure a synchronization node for this data source only by using the code editor.

Create a workflow

1. Log on to the DataWorks console.
2. On the **DataStudio** page, move the pointer over the icon and select **Workflow**.
3. In the **Create Workflow** dialog box, set the **Workflow Name** and **Description** parameters.

   > **Notice** The workflow name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Create**.

Create a batch synchronization node

1. Click the newly created workflow and right-click **Data Integration**.
2. Choose **Create > Batch Synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Notice** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.

Configure a source

After you create a batch synchronization node, you must select a data source and a table in the Source section.

> **Note**
>
> - For more information about how to set the parameters in the Source section, see Configure a reader.
> - By default, a maximum of 25 tables in the selected data source are displayed in the Table drop-down list. If the selected data source contains more than 25 tables and the table that you want to select is not displayed in the Table drop-down list, enter the name of the table in the Table field. Alternatively, configure the batch synchronization node in the code editor.
> - Some synchronization nodes may need to synchronize incremental data. In this case, you can use the scheduling parameters of DataWorks to specify the date and time for incremental data synchronization. For more information, see Scheduling parameters.

Configure a destination

After you configure a source, you must select a data source and a table in the Target section.

> **Note**
> - For more information about how to set the parameters in the Target section, see Configure a writer.
> - You can specify the write mode, such as overwriting or appending, for most synchronization nodes. The write mode that you can specify for a synchronization node varies based on the data source type that you selected.

Configure field mappings

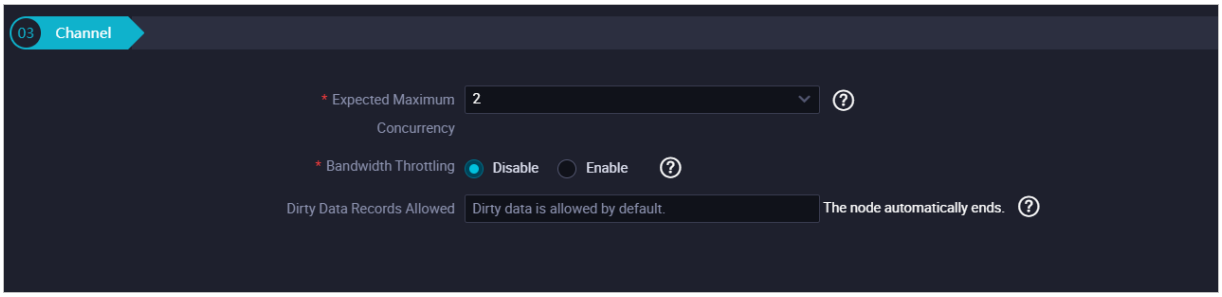
After you specify the source and destination tables, you must specify the mappings between fields in the source and destination tables. You can click **Map Fields with the Same Name**, **Map Fields in the Same Line**, **Delete All Mappings**, or **Auto Layout** to perform the related operation.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specific rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit the 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 that are 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 each source field is the same as or compatible with that of the mapped destination field.

Configure channel control policies

After you complete the preceding steps, you can configure channel control policies for the synchronization node.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure properties for the synchronization node

In most cases, synchronization nodes use scheduling parameters to filter data. This section describes how to configure scheduling parameters for a synchronization node.

On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane.

You can specify scheduling parameters by using ${Variable name}. After a variable is specified, 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 or subtract the date, see Scheduling parameters.

On the Properties tab, you can configure the properties of the synchronization node, such as the recurrence, time when the node is run, and dependencies. Batch synchronization nodes do not have ancestor nodes because they are run before extract, transform, and load (ETL) nodes. We recommend that you specify the root node of the workspace as their ancestor node.

2.1.4.5.2. Create a synchronization node by using the code editor

This topic describes how to create a synchronization node by using the code editor.

Procedure

To create a synchronization node by using the code editor, perform the following steps:

1. Add a data source.
2. Create a batch synchronization node.
3. Apply a template.
4. Configure a reader for the synchronization node.
5. Configure a writer for the synchronization node.
6. Configure field mappings.
7. Configure channel control policies, such as the maximum transmission rate and the maximum number of dirty data records allowed.
8. Configure properties for the node.

Add a data source

A synchronization node can synchronize data between various homogeneous or heterogeneous data sources. On the DataStudio page of the DataWorks console, click the **Workspace Manage** icon in the upper-right corner. On the page that appears, click **Data Source** in the left-side navigation pane and add a data source. For more information, see Add a data source.

After you add a data source, you can select it when you configure a synchronization node on the DataStudio page.

> **Note**
>
> - Data Integration does not support connectivity testing for some data source types. For more information, see Connectivity testing.
> - If an on-premises data source does not have a public IP address or is not accessible from a network, the connectivity testing fails when you configure the data source. You can use a custom resource group to resolve the connection failure. For more information, see Create a custom resource group for Data Integration.
>
>   If a data source cannot be directly connected over a network, Data Integration cannot obtain the table schema. In this case, you can create a synchronization node for this data source only by using the code editor.

Create a workflow

1. Log on to the DataWorks console.
2. On the **DataStudio** page, move the pointer over the icon and select **Workflow**.
3. In the **Create Workflow** dialog box, set the **Workflow Name** and **Description** parameters.

   > **Notice** The workflow name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Create**.

Create a batch synchronization node

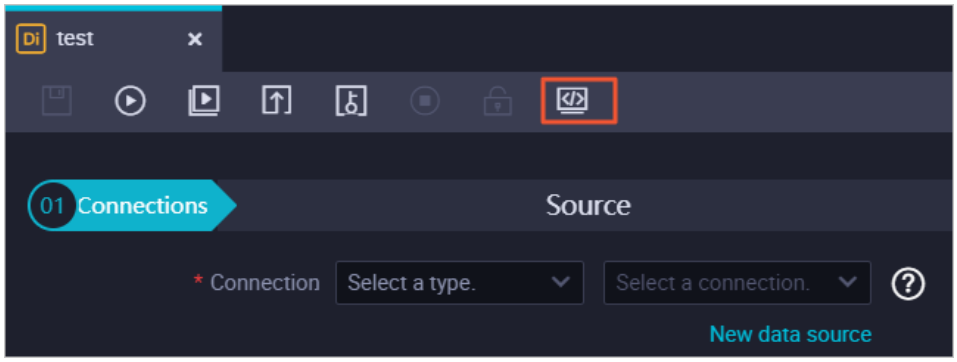
1. Click the newly created workflow and right-click **Data Integration**.
2. Choose **Create > Batch Synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Notice** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.

Apply a template

1. On the node configuration tab that appears, click the **Switch to Code Editor** icon in the top toolbar.



2. In the **Confirm** dialog box, click **OK**.

> **Note** The code editor supports more features than the codeless user interface (UI). For example, you can configure synchronization nodes in the code editor even when the connectivity test fails.

3. Click the icon in the top toolbar.
4. In the **Apply Template** dialog box, configure the following parameters: **Source Connection Type**, **Connection**, **Target Connection Type**, and **Connection**.
5. Click **OK**.

Configure a reader for the synchronization node

After the template is applied, the basic settings of the reader are generated. You can configure the source and source table based on your actual requirements.

```
{"type": "job",
    "version": "2.0",
    "steps": [// Do not modify the preceding lines. They indicate the header code of the synchroniza
tion node.
        {
            "stepType": "mysql",
            "parameter": {
                "datasource": "MySQL",
                "column": [
                    "id",
                    "value",
                    "table"
                ],
                "socketTimeout": 3600000,
                "connection": [
                    {
                        "datasource": "MySQL",
                        "table": [
                            "`case`"
                        ]
                    }
                ],
                "where": "",
                "splitPk": "",
                "encoding": "UTF-8"
            },
            "name": "Reader",
            "category": "reader"// Specifies that these settings are related to the reader.
        },
```

Parameters:

- type: the type of the synchronization node. You must set the value to job.
- version: the version number of the synchronization node. You can set the value to 1.0 or 2.0.

> **Note**
> - For more information about how to configure the source, see Configure MaxCompute Reader.
> - Some synchronization nodes may need to synchronize incremental data. In this case, you can use the scheduling parameters of DataWorks to specify the date and time for incremental data synchronization. For more information, see Scheduling parameters.

Configure a writer for the synchronization node

After the reader is configured, you can configure the destination and destination table based on your actual requirements.

```
{
  "stepType": "odps",
  "parameter": {
      "postSql":[],// The SQL statement that you want to execute after the synchronization node is r
un.
      "partition": "",
      "truncate": true,
      "compress": false,
      "datasource": "odps_first",
      "column": [
          "*"
       ],
       "emptyAsNull": false,
       "table": "",
       "preSql":[
               "delete from XXX;"// The SQL statement that you want to execute before the synchroniz
ation node is run. Separate multiple statements with semicolons (;).
             ]
     },
     "name": "Writer",
     "category": "writer"// Specifies that these settings are related to the writer.
   }
 ],
```

> **Note**
> - For more information about how to configure the destination, see Configure DataHub Writer.
> - You can select the writing method for most nodes. For example, the writing method can be overwriting or appending. Supported writing methods vary based on the data source type.

Map the fields in the source and destination tables

The code editor supports only the mappings of fields in the same row. The data types of the fields must match.

> **Note** Make sure that the data type of a source field is the same as that of the mapped destination field or the data type conversion is feasible.

Configure channel control policies

After the preceding steps are performed, you can configure the channel control policies for the synchronization node. The setting parameter specifies node efficiency parameters, including the number of parallel threads, 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 parallel threads.
        }
    },
```

Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure properties for the synchronization node

In most cases, synchronization nodes use scheduling parameters to filter data. This section describes how to configure scheduling parameters for a synchronization node.

On the **DataStudio** page, double-click the batch synchronization node in the related workflow. On the node configuration tab, click the **Properties** panel in the right-side navigation pane to configure properties for the node.

In the Properties panel, you can configure the properties of the synchronization node, such as the recurrence, time when the synchronization node is run, and dependencies. Batch synchronization nodes do not have ancestor nodes because they are run before extract, transform, and load (ETL) nodes. We recommend that you specify the root node of the workspace as their ancestor node.

After the synchronization node is configured, save and commit the node. For more information, see Scheduling parameters.

2.1.4.5.3. Configure the reader

2.1.4.5.3.1. Configure PolarDB-X Reader

PolarDB-X Reader can read data from PolarDB-X. PolarDB-X Reader connects to a remote PolarDB-X database by using Java Database Connectivity (JDBC) and executes a SELECT statement to read data from the database.

PolarDB-X Reader supports only MySQL engines. PolarDB-X is a distributed MySQL database service that complies with the MySQL protocols in most cases.

Specifically, PolarDB-X Reader connects to a remote PolarDB-X database by using JDBC, generates a SELECT statement based on your configurations, and then sends the statement to the database. PolarDB-X executes the statement and returns results. Then, PolarDB-X Reader assembles the returned data into abstract datasets of custom data types supported by Data Integration, and passes the datasets to a writer.

PolarDB-X 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 PolarDB-X database. PolarDB-X does not support all MySQL protocols that contain statements such as JOIN.

PolarDB-X Reader supports most PolarDB-X data types. Make sure that your data types are supported.

The following table lists the data types supported by PolarDB-X Reader.

Category	PolarDB-X 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 name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table from which you want to read data.	Yes	No default value
column	The names of the columns from which you want to read data. The columns are described in a JSON array. The default value is [*], which indicates all columns. • You can also select specific columns to synchronize. • The column order can be changed. You can configure PolarDB-X Reader to synchronize 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 compliance with the SQL syntax supported by MySQL, such as `["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 all the columns that you want to synchronize. This parameter cannot be left empty.	Yes	No Default value

Parameter	Description	Required	Default value
where	The WHERE clause. PolarDB-X Reader generates a SELECT statement based on the table, column, and where parameters that you configured and uses the generated SELECT statement to read data. For example, if you 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)` for a test in an actual business scenario, data generated on the day is synchronized. • You can use the WHERE clause to synchronize incremental data. • If the where parameter is left empty, all data is synchronized.	No	No default value

Configure PolarDB-X Reader by using the codeless UI

1. Configure data sources.

   Configure **Source** and **Target** for the synchronization node.

Parameter	Description
Connection	This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
Table	This parameter corresponds to the table parameter that is described in the preceding section.
Filter	The condition used to filter the data that you want to synchronize. PolarDB-X Reader cannot filter data based on the limit keyword. The SQL syntax is determined by the selected data source.
Shard Key	The shard key. 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. Only integer fields are supported. If you specify this parameter, data sharding is performed based on the value of this parameter, and parallel threads can be used to read data. This improves data synchronization efficiency. **Note** The Shard Key parameter is displayed only after you configure the source for the synchronization node.

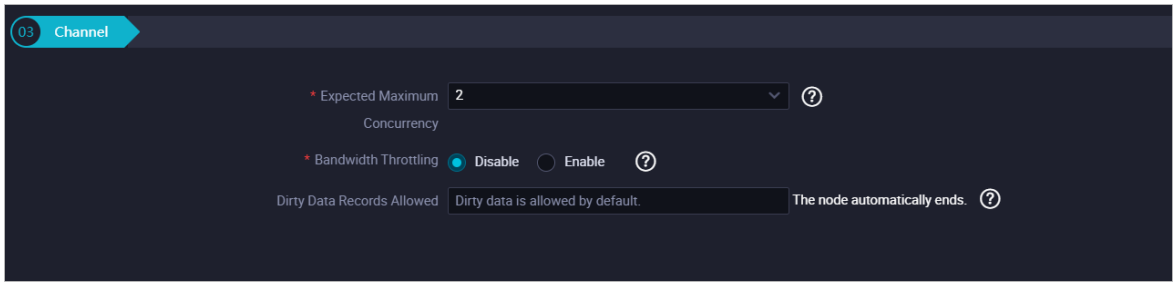
2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination 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.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.

Operation	Description
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit the 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. Take note of the following rules when you 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 as Unidentified.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure PolarDB-X Reader by using the code editor

In the following code, a synchronization node is configured to read data from a PolarDB-X database:

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"drds",// The reader type.
            "parameter":{
                "datasource":"",// The name of the data source.
                "column":[// The names of the columns from which you want to write data.
                    "id",
                    "name"
                ],
                "where":"",// The WHERE clause.
                "table":"",// The name of the table from which you want to read data.
                "splitPk": ""// The shard key.
            },
            "name":"Reader",
            "category":"reader"
        },
        {// The following code is used to configure Stream Writer. For more information about how to configure other writers, see the related 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 parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}:"Writer"
            }
        ]
    }
}
```

Additional information

- Consistency

  PolarDB-X cannot provide a consistent view of multiple tables in multiple databases because it is a distributed database service. PolarDB-X Reader extracts different snapshots from different shards but cannot extract the snapshot of database and table shards at the same time slice. As a result, PolarDB-X Reader cannot ensure strong consistency for data queries.

- Character encoding

  PolarDB-X 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.

  PolarDB-X Reader uses JDBC, which can automatically convert the encoding of characters. Therefore, you do not need to specify the encoding format.

  If you specify an encoding format for a PolarDB-X database but data is written to the PolarDB-X database in a different encoding format, PolarDB-X Reader cannot recognize this inconsistency and may export garbled characters.

- Incremental data synchronization.

  PolarDB-X Reader connects to a database by using JDBC and uses a SELECT statement with a WHERE clause to read incremental data in the following ways:

  - For batch data, incremental additions, updates, and deletion 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 greater than the maximum ID in the last synchronization.

  If incremental data cannot be distinguished, PolarDB-X Reader can synchronize only full data but cannot synchronize incremental data.

- Syntax validation

  PolarDB-X Reader allows you to specify custom SELECT statements by using the querySql parameter but does not verify the syntax of the custom SELECT statements.

2.1.4.5.3.2. 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
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

Parameter	Description	Required	Default value

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:
	`"column": [ { "name": "rowkey", "type": "string" }, { "name": "info:age", "type": "string" } ]`
		Yes	None

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

Parameter	Description	Required	Default value
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 co
nnection.
                "scanBatchSize": 256, // The number of columns read by an HBase client with each RPC
connection.
                "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 en
d rowkeys into the byte[] format. Default value: false. If the value is true, Bytes.toBytesBinary(ro
wkey) is used. If the value is false, Bytes.toBytes(rowkey) is used.
                },
                "maxVersion": , // 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 vers
ions are read.
                "encoding": "UTF-8",
                "table": "test" // The name of the HBase table from which data is read. The name is
case-sensitive.
            },
            "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 na
me is case-sensitive.
                    "encoding": "utf-8", // The encoding format, by using which binary data stored i
n byte[] format 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
versions are available. Valid values: -1 and integers greater than 1. A value of -1 indicates that a
ll versions are read.
                    "column": [ // The HBase columns from which data is read. The name parameter spe
```

```
cifies 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 th
e date 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"
            }
        ]
    }
}
```

2.1.4.5.3.3. 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

Category	HDFS data type
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
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
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
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

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

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 co
rresponding 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 throt
tled. The maximum transmission rate takes effect only if you set this parameter to true.
        }
    },
    "order": {
        "hops": [
            {
```

```
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

2.1.4.5.3.4. Configure MaxCompute Reader

This topic describes the data types and parameters supported by MaxCompute Reader and how to configure it by using the codeless user interface (UI) and code editor.

MaxCompute Reader can read data from MaxCompute by using Tunnel based on the source project, table, partition, and table fields that you configured.

MaxCompute Reader cannot read views. It can read only partitioned and non-partitioned tables. To allow MaxCompute Reader to read partitioned tables, you must specify the partition information. For example, to read data from the t0 table, set pt to 1 and ds to hangzhou. The partition information is not required for non-partitioned tables. In addition, 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. 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
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table from which you want to read data. The name is not case-sensitive.	Yes	No default value

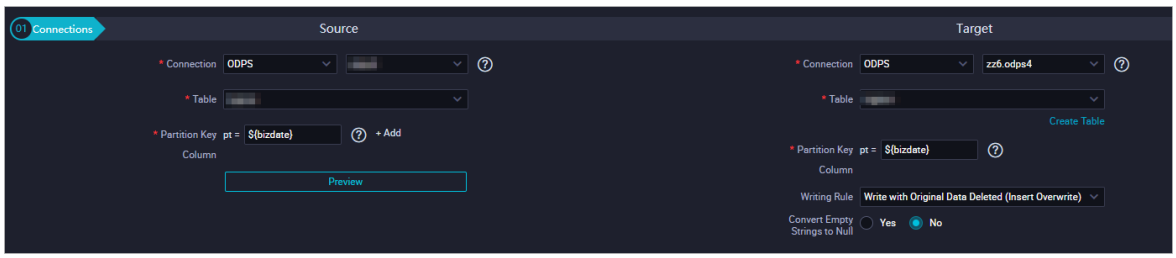
Parameter	Description	Required	Default value
partition	The partitions from which you want to read data. You can use Linux Shell wildcards to specify the partitions. An asterisk (*) indicates multiple numbers of characters, and a question mark (?) indicates a single character. For example, the partitioned table 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 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 partitioned tables	No default value
column	The names of the columns from which you want to read data in the source table. For example, the table test contains the id, name, and age fields. • To read the fields in turn, enter `"column":["id","name","age"]` or `"column":["*"]`. **Note** We recommend that you do not set this parameter to ["*"], which indicates that MaxCompute Reader reads all the fields in turn. If the source table changes in the column order, data type, or number of columns, the columns in the source and destination tables are not consistent. As a result, the data synchronization may fail. • To read the name and id fields in turn, enter `"column":["name","id"]`. • You can add constant fields to the source table to establish mappings between the source table columns and destination table columns. Each constant must be enclosed in single quotation marks (`'`). For example, if you enter `"column":["age","name","'1988-08-08 08:08:08'","id"]`, the data that is extracted from the source table contains an age column, a name column, a constant "1988-08-08 08:08:08", and an id column in sequence. 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 all the columns that you want to synchronize. This parameter cannot be left empty.	Yes	No default value

Configure MaxCompute Reader by using the codeless UI

On the **DataStudio** page, create a synchronization node under a workflow and configure the node.

1. Configure data sources.

   Configure **Source** and **Target** for the synchronization node.



Parameter	Description
Connection	This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
Table	This parameter corresponds to the table parameter that is described in the preceding section.
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 allows you to use wildcards and specify one or more partitions.
>
> - `"partition":"pt=20140501/ds=*"` indicates that all ds partitions with pt=20140501 will be synchronized.
> - `"partition":"pt=top?"` indicates that the partitions with pt=top and pt=to will be synchronized.
>
> You can specify the partition key columns that you want to synchronize, such as a partition key column named pt. For example, the partition key column of a MaxCompute table is pt=${bdp.system.bizdate}. You can add the pt column to the source table in the Mappings section. If the column is marked as unidentified, ignore the mark and proceed to the next step. To synchronize all partitions, enter pt=*. To synchronize specific partitions, specify the required dates.

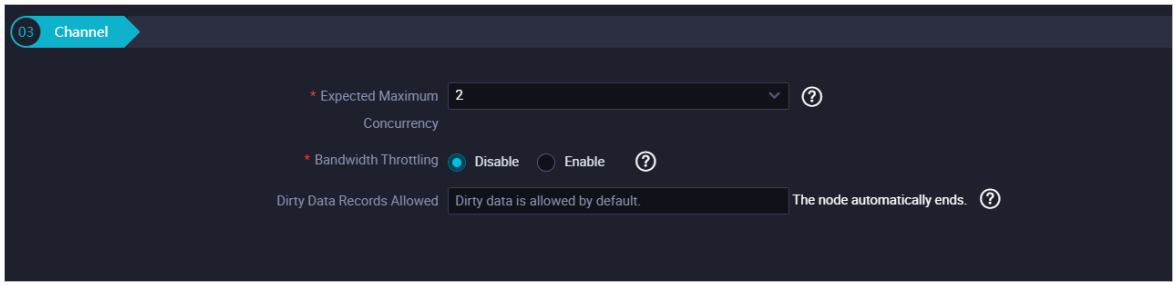
2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination 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.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specified rules.

Operation	Description
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit the 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 that are supported by relational databases, such as now() and count(1). ○ Fields that cannot be parsed are indicated as Unidentified.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure MaxCompute Reader by using the code editor

In the following code, a synchronization 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 synchronization node.
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"odps",// The reader type.
            "parameter":{
                "partition":[],// The partitions from which you want to read data.
                "isCompress":false,// Specifies whether to enable compression.
                "datasource":"",// The name of the data source.
                "column":[// The names of the columns from which you want to read data.
                    "id"
                ],
                "emptyAsNull":true,
                "table":""// The name of the table from which you want to read data.
            },
            "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. The value false ind
icates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling
is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1,// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",// The source of the synchronization node.
                "to":"Writer"// The destination of the synchronization node.
            }
        ]
    }
}
```

2.1.4.5.3.5. 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

Parameter	Description	Required	Default value
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 i
ndicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is thrott
led. 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"
            }
        ]
    }
}
```

Note You cannot retrieve data elements from arrays.

2.1.4.5.3.6. Configure Db2 Reader

Db2 Reader reads data from Db2 databases. Db2 Reader connects to a remote Db2 database by using Java Database Connectivity (JDBC) and executes a SELECT statement to read data from the database.

Specifically, Db2 Reader connects to a remote Db2 database by using JDBC, generates a SELECT statement based on your configurations, and then sends the statement to the database. Db2 Reader executes the statement and returns results. Then, Db2 Reader assembles the returned data into abstract datasets of 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 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 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

Category	Db2 data type
Binary	Blob

Parameters

Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
jdbcUrl	The JDBC URL of the Db2 database. The URL must be in the jdbc:db2://ip:port/database format in accordance with official Db2 specifications. You can also specify the information of the attachment facility.	Yes	No default value
username	The username that you use to connect to the database.	Yes	No default value
password	The password that you use to connect to the database.	Yes	No default value
table	The name of the table from which you want to read data. You can select only one source table for each synchronization node.	Yes	No default value

column	The names of the columns from which you want to read data. The columns are described in a JSON array. The default value is [*], which indicates all columns.
	• You can also select specific columns to synchronize.
	• The column order can be changed. You can configure Db2 Reader to synchronize 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 compliance with the SQL syntax supported by Db2, such as `["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 a pair of single quotation marks (').
	  ◦ null: a null pointer.
	  ◦ upper('abc_lower'): a function expression.
	  ◦ 2.3: a floating-point number.
	  ◦ true: a Boolean value.
	• The column parameter must explicitly specify all the columns that you want to synchronize. This parameter cannot be left empty.
		Yes	No default value

Parameter	Description	Required	Default value
splitPk	The field that is used for data sharding when Db2 Reader reads data. If you specify the splitPk parameter, the table is sharded based on the shard key that is specified by this parameter. Data Integration then runs parallel threads to synchronize data. This way, data can be more efficiently synchronized. • We recommend that you set the splitPk parameter to the primary key of the table. This way, data can be well distributed to different shards based on the primary key, 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, FLOAT, and DATE. If you specify this parameter to an unsupported data type, Db2 Reader returns an error.	No	N/A
where	The WHERE clause. Db2 Reader generates a SELECT statement based on the column, table, and where parameters that you have configured, and uses the generated SELECT statement to read data. For example, set this parameter to gmt_create>$bizdate in an actual business scenario to synchronize the data on the day. You can use the WHERE clause to synchronize incremental data. If this parameter is not specified, all data is synchronized.	No	No default value
querySql	The SQL statement used for refined data filtering. If you specify this parameter, the system filters data based on the value of 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 configured.	No	No default value
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 an out of memory (OOM) error during data synchronization.	No	1024

Configure Db2 Reader by using the codeless UI

This method is not supported.

Configure Db2 Reader by using the code editor

In the following code, a synchronization 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 that you use to connect to the Db2 database.
                "jdbcUrl":"",// The JDBC URL of the Db2 database.
                "column":[
                    "id"
                ],
                "where":"",// The WHERE clause.
                "splitPk":"",// The field that is used for data sharding when Db2 Reader reads data.
                "table":"",// The name of the table from which you want to read data.
                "username":""// The username that you use to connect to the Db2 database.
            },
            "name":"Reader",
            "category":"reader"
        },
        {// The following code provides an example of how to configure Stream Writer. For more information about how to configure other writers, see the related 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. The value false indicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1,// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

Additional information

- 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 binary logs. Latency exists in the synchronization of data from the primary database to the secondary database. In particular, if network conditions are unfavorable, data latency between the primary and secondary databases is unavoidable, which can lead to data inconsistency.

- Parallelism control

  Db2 is a relational database management system (RDBMS), which ensures strong consistency for data queries. A database snapshot is created before a synchronization 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 ensured when you enable Db2 Reader to use parallel threads to synchronize data. Db2 Reader shards the table based on the splitPk parameter and uses parallel threads to synchronize data. These parallel threads belong to different transactions, and they read data at different time points. Therefore, the parallel threads provide different snapshots.

  Theoretically, the preceding data inconsistency issue is unavoidable. The following workarounds can be used:

  - Do not enable parallel threads for a single synchronization node. Essentially, do not specify the splitPk parameter. In this way, data consistency is ensured, but data is synchronized at low efficiency.
  - Do not write data to the source table to ensure that the data remains unchanged during data synchronization. For example, lock the table and disable data synchronization between primary and secondary databases. This way, data is efficiently synchronized, but your ongoing services may be interrupted.

- Character encoding

  Db2 Reader uses JDBC which can automatically convert the encoding format of characters. Therefore, you do not need to specify the encoding format.

- Incremental data synchronization

  Db2 Reader connects to a database by using JDBC and uses a SELECT statement with a WHERE clause to read incremental data in the following ways:

  - For batch data, incremental additions, updates, and deletion 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 greater than the maximum ID that is involved in the last synchronization.

  If incremental data cannot be distinguished, Db2 Reader can synchronize only full data but not incremental data.

- 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.

2.1.4.5.3.7. 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 by using Java Database Connectivity (JDBC), generates a SELECT statement based on your configurations, and then sends the statement to the database. MySQL Reader executes the statement and returns results. Then, MySQL Reader assembles the returned data into abstract datasets of custom data types supported by Data Integration, and sends the datasets to a writer.

Specifically, MySQL Reader connects to a remote MySQL database by using JDBC and executes a SELECT statement to read data from the database.

MySQL Reader can read tables and views. For table fields, you can specify all or specific columns in sequence, change 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

- MySQL Reader supports only the data types that are listed in the preceding table.
- MySQL Reader considers TINYINT(1) as an integer type.
- MySQL Reader does not support MySQL 8.0 or later.

Parameters

Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table from which you want to read data. You can select only one table for each synchronization node.	Yes	No default value

Parameter	Description	Required	Default value

column	The names of the columns from which you want to read data. The columns are described in a JSON array. The default value is [*], which indicates all the columns in the source table.
	• You can select specific columns to synchronize.
	• The column order can be changed. You can configure MySQL Reader to synchronize 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 compliance with the SQL syntax supported by MySQL, such as `["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 the 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 all the columns that you want to synchronize. This parameter cannot be left empty.
		Yes	No default value

splitPk	The field that is used for data sharding when MySQL Reader reads data. If you specify the splitPk parameter, the table is sharded based on the shard key that is specified by this parameter. The system then runs parallel threads to synchronize data. This way, data can be more efficiently synchronized. • We recommend that you set the splitPk parameter to the primary key of the table. Data can be well distributed to different shards based on the primary key, 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, FLOAT, and DATE. If you set this parameter to a field of an unsupported data type, MySQL Reader ignores the splitPk parameter and synchronizes data by using a single thread. • If the splitPk parameter is not provided or is left empty, the system synchronizes data by using a single thread.	No	No default value

Parameter	Description	Required	Default value
where	The WHERE clause. For example, set this parameter to `gmt_create>$bizdate` in an actual business scenario to synchronize the data on the current day. • You can use the WHERE clause to synchronize incremental data. If the where parameter is not provided or is left empty, the system synchronizes all data. • Do not set the where parameter to limit 10, which does not conform to the constraints of MySQL on the SQL WHERE clause.	No	No default value
querySql (available only in the code editor)	The SELECT statement used for refined data filtering. If you specify this parameter, the system directly filters data based on the value of 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 the priorities 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 configured. The system parses information, such as the username and password, that is required by the data source specified by the datasource parameter from the querySql parameter.	No	No default value
singleOrMulti (suitable only for sharding)	Specifies whether to shard the database or table. After you switch from the codeless UI to the code editor, the `"singleOrMulti":"multi"` configuration is automatically generated. However, if you use the code editor at the beginning, the configuration is not automatically generated, and you must manually specify this parameter. If you do not specify this parameter, MySQL Reader can read data only from the first shard. The singleOrMulti parameter is used only by the frontend.	Yes	*multi*

Configure MySQL Reader by using the codeless UI

1. Configure data sources.

   Configure **Source** and **Target** for the synchronization node.

Parameter	Description
Connection	This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
Table	This parameter corresponds to the table parameter that is described in the preceding section.
Filter	The condition for filtering the data that you want to synchronize. MySQL Reader cannot filter data based on the limit keyword. The SQL syntax is determined by the selected data source.

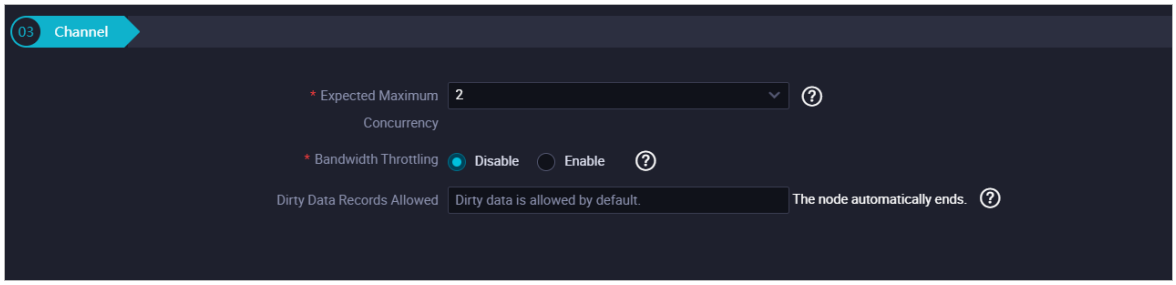
Parameter	Description
Shard Key	The shard key. You can specify a column in the source table as the shard key. We recommend that you use the primary key or an indexed column as the shard key. Only integer fields are supported. If you specify this parameter, data sharding is performed based on the value of this parameter, and parallel threads can be used to read data. This improves data synchronization efficiency. **Note** The Shard Key parameter is displayed only after you select the data source for the synchronization node.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination 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.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit the 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 as Unidentified.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure MySQL Reader by using the code editor

The following example shows how to configure MySQL Reader to read data from a database or table that is not sharded by using the code editor. 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 names of the columns from which you want to read data.
                    "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 name of the data source.
                        "table":[
                            "xxx"// The name of the table from which you want to read data.
                        ]
                    }
                ],
                "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. The value false indicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1,// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

The following example shows how to configure MySQL Reader to read data from a database or table that is sharded by using the code editor. For more information about the parameters, see the preceding parameter description.

Note In the case of 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": {
        }
    }
}
```

2.1.4.5.3.8. Oracle Reader

This topic describes the data types and parameters that are supported by Oracle Reader and how to configure Oracle Reader by using the codeless user interface (UI) and code editor.

Oracle Reader can read data from Oracle.

Oracle Reader connects to a remote Oracle database by using Java Database Connectivity (JDBC), generates an SQL statement based on your configurations, and then sends the statement to the database. The system executes the statement on the database and returns data. Then, Oracle Reader assembles the returned data into abstract datasets of the data types supported by Data Integration and sends the datasets to a writer.

- Oracle Reader generates the SQL statement based on the settings of the table, column, and where parameters and sends the generated statement to the Oracle database.
- If you specify the querySql parameter, Oracle Reader sends the value of this parameter to the Oracle database.

Data types

Oracle Reader supports most Oracle data types. Make sure that the data types of your database are supported.

The following table lists the data types that are 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 name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table from which you want to read data.	Yes	No default value
column	The names of the columns from which you want to read data. Specify the columns in a JSON array. The default value is ["*"], which indicates all columns in the source table. • You can also select specific columns to read. • The column order can be changed. You can configure Oracle Reader to read 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 the JSON format. Example: `["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	No default value

Parameter	Description	Required	Default value
splitPk	The field that is used for data sharding when Oracle Reader reads data. If you specify this parameter, the source table is sharded based on the value of this parameter. Data Integration then runs parallel threads to read data. This improves data synchronization efficiency. • We recommend that you set the splitPk parameter to the primary key column of the table. Data can be evenly distributed to different shards based on the primary key, but not intensively distributed only to specific shards. • The splitPk parameter supports sharding for data only of numeric, string, floating point, and date data types. • If you do not specify the splitPk parameter, Oracle Reader uses a single thread to read all data in the table.	No	No default value
where	The WHERE clause. Oracle Reader generates the SQL statement based on the settings of the column, table, and where parameters and uses the generated statement to read data. For example, you can set this parameter to row_number() for a test or `id>2 and sex=1` in an actual business scenario to synchronize data that is generated on the current day. • You can use the WHERE clause to read incremental data. • If the where parameter is not provided or is left empty, Oracle Reader reads all data in the table.	No	No default value
querySql (available only in the code editor)	The SQL statement that is used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on the value of this parameter. For example, if you want to join multiple tables for data synchronization, you can 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 settings of the table, column, and where parameters.	No	No default value
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** If you set this parameter to a value greater than 2048, an out of memory (OOM) error may occur during data synchronization.	No	1024

Parameter	Description	Required	Default value

driverVersion	The version of the driver that can be used to access the database. Valid values: ojdbc5, ojdbc6, ojdbc7, ojdbc8, and ojdbc14.
	If you want to use a driver of a specific version that is compatible with the database, add the following configurations by using the code editor:
	`"driverVersion": "ojdbc5"// Driver version of ojdbc5-11.2.0.3.jar`
	`"driverVersion": "ojdbc6"// Driver version of ojdbc6-12.1.1.jar`
	`"driverVersion": "ojdbc7"// Driver version of ojdbc7-12.1.0.2.jar`
	`"driverVersion": "ojdbc8"// Driver version of ojdbc8-12.2.0.1.jar`
	`"driverVersion": "ojdbc14"// Driver version of ojdbc14-10.2.0.3.0.jar`
		No	No default value

Configure Oracle Reader by using the codeless UI

1. Configure data sources.

   Log on to the DataWorks console. The **DataStudio** page appears. On the DataStudio page, move the pointer over the icon and choose **Data Integration > Batch Synchronization**. In the **Create Node** dialog box, configure the parameters to create a batch synchronization node.

   On the configuration tab of the batch synchronization node, configure **Source** and **Target** for the node.

Parameter	Description
Connection	The name of the data source from which you want to read data. This parameter is equivalent to the datasource parameter that is described in the preceding section.
Table	The name of the table from which you want to read data. This parameter is equivalent to the table parameter that is described in the preceding section.
Filter	The condition that is used to filter the data you want to read. Filtering based on the LIMIT keyword is not supported. The SQL syntax is determined by the selected data source.
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 column or an indexed column. Only integer columns are supported. If you specify this parameter, data sharding is performed based on the value of this parameter, and parallel threads can be used to read data. This improves data synchronization efficiency. **Note** The Shard Key parameter is displayed only after you select the data source for the synchronization node.

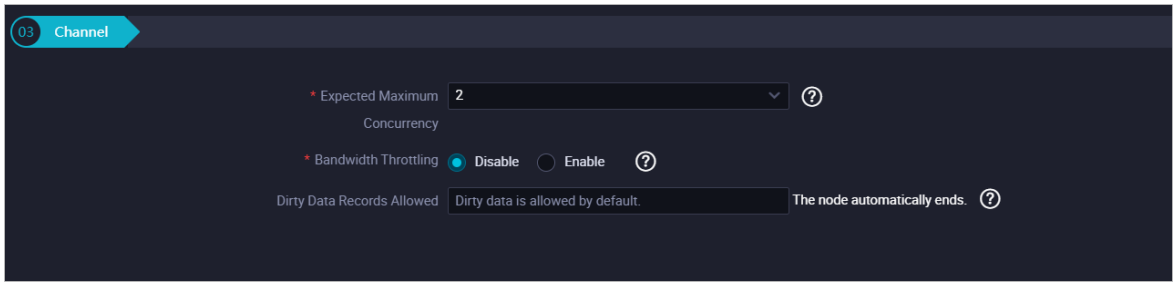
2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in

the preceding section.

Fields in the source on the left have a one-to-one mapping with fields in the destination on the right. You can click **Add** to add a field. To remove an added field, move the pointer over the field and click the **Remove** icon.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit the 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	○ 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 field that you entered cannot be parsed, the value of Type for the field is Unidentified.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and specify a maximum transmission rate to avoid heavy read workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure Oracle Reader by using the code editor

In the following code, a synchronization 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 name of the data source.
                "column":[// The names of the columns from which you want to read data.
                    "id",
                    "name"
                ],
                "where":"",// The WHERE clause.
                "splitPk":"",// The shard key.
                "table":""// The name of the table from which you want to read data.
            },
            "name":"Reader",
            "category":"reader"
        },
        {// The following code is used to configure Stream Writer. If you want to configure other wr
iters, see the related topic.
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":true,// Specifies whether to enable bandwidth throttling. The value false ind
icates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling
is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1,// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
} "to":"Writer"
            }
        ]
    }
}
```

Additional information

- 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 binary logs. Data latency between the primary and secondary databases is unavoidable especially when network conditions are unfavorable. This may result in data inconsistency.

- Data consistency control

  Oracle is a relational database management system (RDBMS), which supports strong consistency for data queries. A database snapshot is created before a synchronization 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 ensured when you enable Oracle Reader to use parallel threads to read data in a synchronization node.

  Oracle Reader shards the source table based on the value of the splitPk parameter and uses parallel threads to read data. These parallel threads belong to different transactions. They read data at different points in time. Therefore, the parallel threads observe different snapshots.

  Theoretically, data inconsistencies are unavoidable if parallel threads are used for a synchronization node. The following workarounds can be used:

  - Enable Oracle Reader to use a single thread to read data in a synchronization node. This indicates that you do not specify a shard key for Oracle Reader. This way, data consistency is ensured, but data is synchronized at low efficiency.
  - Make sure that no data is written to the source table during data synchronization. This ensures that the data in the source table remains unchanged during data synchronization. For example, you can lock the source table or disable data synchronization between primary and secondary databases. This way, data is efficiently synchronized, but your ongoing services may be interrupted.

- Character encoding

  Oracle Reader uses JDBC to read data. This enables Oracle Reader to automatically convert the encoding formats of characters. Therefore, you do not need to specify the encoding format.

- Incremental data synchronization

  Oracle Reader uses JDBC to connect to a database and uses a SELECT statement with a `WHERE` clause to read incremental data.

  - For batch data, incremental add, update, and delete operations (including logically delete operations) are distinguished by timestamps. Specify the WHERE clause based on a specific timestamp. The time indicated by the timestamp must be later than the time indicated by the latest timestamp in the last synchronization.
  - For streaming data, specify the WHERE clause based on the ID of a specific record. The ID must be greater than the maximum ID involved in the last synchronization.

  If incremental data cannot be distinguished, Oracle Reader cannot read incremental data but can read only full data.

- Syntax validation

  Oracle Reader allows you to specify custom SELECT statements by using the querySql parameter but does not verify the syntax of these statements.

2.1.4.5.3.9. Configure OSS Reader

This topic describes the data types and parameters supported by OSS Reader and how to configure it by using the codeless user interface (UI) and code editor.

OSS Reader can read data stored in OSS. Specifically, OSS Reader connects to Object Storage Service (OSS) by using Alibaba Cloud OSS SDK for Java and reads data from OSS. Then, OSS Reader converts the data into a format that is readable to Data Integration and sends the converted data to a writer.

OSS stores only unstructured data. OSS Reader supports the following features:

- Reads TXT objects that store two-dimensional tables. OSS Reader can read only TXT objects.
- Reads data stored in CSV-like objects with custom delimiters.
- Reads various types of data as strings and supports constants and column pruning.
- Supports recursive data read and object name-based filtering.
- Supports the following object compression formats: GZIP, BZIP2, and ZIP.

  > **Note** You cannot compress multiple objects into one package.

- Uses parallel threads to read multiple objects.

OSS Reader does not support the following features:

- Uses parallel threads to read an uncompressed object.
- Uses parallel 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 name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value

Parameter	Description	Required	Default value
Object	The name of the OSS object from which you want to read data. You can specify multiple object names. For example, a bucket has a directory that is named yunshi, and this directory contains an object that is named ll.txt. In this case, you can set this parameter to yunshi/ll.txt. • If you specify a single OSS object name, OSS Reader uses only one thread to read the object. Parallel thread reading of a single uncompressed object will be available in the future. • If you specify multiple OSS object names, OSS Reader uses multiple threads to read these objects. The actual number of threads is determined by the number of channels. • If a name contains a wildcard, OSS Reader reads all objects that match the name. For example, you set this parameter to abc[0-9]. In this case, OSS Reader reads data from objects abc0 to abc9. We recommend that you do not use wildcards because wildcards may cause out of memory (OOM) errors. Note • Data Integration considers all the objects on a synchronization node as a single table. Make sure that all the objects on each synchronization 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	No default value

Parameter	Description	Required	Default value

column	The columns from which you want to read data. The type parameter specifies the source data type. The index parameter specifies the ID of the column in the source object, 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 code, the value is the constant "alibaba".`
	`}`
	Note For the column parameter, you must specify the type parameter and specify either the index or value parameter.
		Yes	Asterisks (Not recommended)

fieldDelimiter	The column delimiter that is used in the OSS object from which you want to read data. **Note** You must specify the column delimiter for OSS Reader. The default delimiter is commas (,). If you do not specify the column delimiter, the default column delimiter is also used on the codeless UI.	Yes	,
compress	The compression format of the object from which you want to read data. By default, this parameter is left empty. In this case, objects are not compressed. OSS Reader supports the following object compression formats: GZIP, BZIP2, and ZIP.	No	*No default value*
encoding	The encoding format of the object from which you want to read data.	No	*utf-8*
nullFormat	The string that represents a null pointer. No standard strings can represent a null pointer 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	No default value

Parameter	Description	Required	Default value
skipHeader	Specifies whether to skip the header of a CSV-like object if the object has a header. 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. The CSV reader supports multiple configurations.	No	No default value

Configure OSS Reader by using the codeless UI

1. Configure data sources.

   Configure **Source** and **Target** for the synchronization node.

Parameter	Description
Connection	This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
Object Name Prefix	This parameter corresponds to the object parameter that is described in the preceding section. **Note** If an OSS object is named based on the date, such as aaa/20171024abc.txt, you can set this parameter to aaa/${bdp.system.bizdate}abc.txt.
Field Delimiter	This parameter corresponds to the fieldDelimiter parameter that is described in the preceding section. The default delimiter is commas (,).
Encoding	This parameter corresponds to the encoding parameter that is described in the preceding section. The default encoding format is UTF-8.
Null String	This parameter corresponds to the nullFormat parameter that is described in the preceding section. Enter a string that represents a null pointer. If the source contains the string, the string is replaced with a null pointer.
Compression Format	This parameter corresponds to the compress parameter that is described in the preceding section. By default, objects are not compressed.
Include Header	This parameter corresponds to the skipHeader parameter that is described in the preceding section. The default value is No.

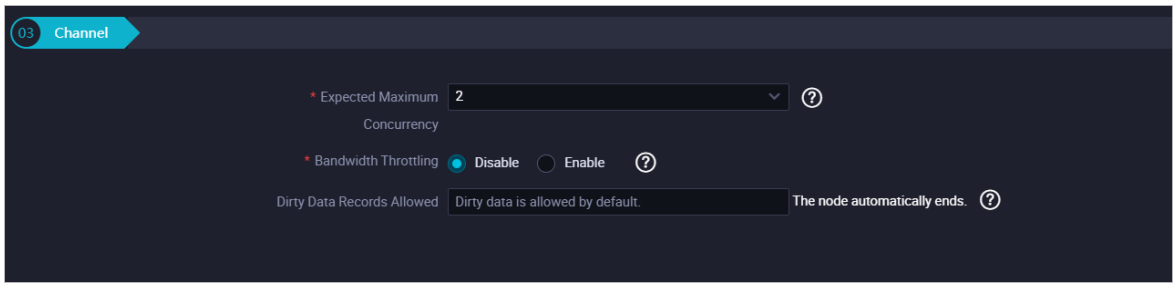
2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination 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.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.

Operation	Description
Delete All Mappings	Click **Delete All Mappings** to remove mappings that are established.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure OSS Reader by using the code editor

In the following code, a synchronization 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 a null pointer.
                "compress":"",// The compression format.
                "datasource":"",// The name of the data source.
                "column":[// The names of the columns from which you want to read data.
                    {
                        "index":0,// The IDs of the columns from which you want to read data.
                        "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 time format.
                        "index":4,
                        "type":"date"
                    }
                ],
                "skipHeader":"",// Specifies whether to skip the header of a CSV-like object if the
object has a header.
                "encoding":"",// The encoding format.
                "fieldDelimiter":",",// The column delimiter.
                "fileFormat": "",// The format in which OSS Reader stores the object from which you
want to read data.
                "object":[]// The name of the object from which you want to read data.
            },
            "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. The value false in
dicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttlin
g is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1,// The number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.1.4.5.3.10. Configure FTP Reader

This topic describes the data types and parameters supported by FTP Reader and how to configure it by using the codeless user interface (UI) and code editor.

FTP Reader reads data from a remote File Transfer Protocol (FTP) server. Specifically, FTP Reader connects to a remote FTP server, reads data from the server, converts the data to a format that is readable to Data Integration, and then sends the converted data to a writer.

FTP Reader can read only FTP files that store logical two-dimensional tables, such as text information in the CSV format.

FTP servers store only unstructured data. FTP Reader provides the following features:

- Reads TXT files that store logical two-dimensional tables. FTP Reader can read only TXT files.
- Reads data stored in CSV-like files with custom delimiters.
- Reads data of various types as strings and 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.
- Uses parallel threads to read multiple files.

FTP Reader does not support the following features:

- Uses parallel threads to read an uncompressed file.
- Uses parallel threads to read a compressed file.

A remote FTP file does not distinguish between data types. The data types 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 name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value

Parameter	Description	Required	Default value
path	The path of the FTP file from which you want to read data. You can specify multiple paths. • If you specify only one path, FTP Reader uses only one thread to read the related file. Parallel thread reading of a single uncompressed file is available soon. • If you specify multiple paths, FTP Reader uses parallel threads to read the related files. The actual number of threads is determined by the number of channels. • If a path contains a wildcard, FTP Reader attempts to read all files that match the path. If a path is ended with a forward slash (/), FTP Reader reads all files in the specified path. For example, if you specify the /bazhen/ path, FTP Reader reads all the files in this path. FTP Reader supports only asterisks (*) as file path wildcards. Note • We recommend that you do not use asterisks (*) because asterisks may cause Java virtual machine (JVM) memory overflow. • Data Integration considers all the files in a data synchronization node as a single table. Make sure that all the files in each synchronization node can adapt to the same schema and Data Integration has permissions 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	No default value

Parameter	Description	Required	Default value
column	The names of the columns from which you want to read data. 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. In this case, set the value to `["*"]`. You can also specify the column parameter in the following way: `{ "type": "long", "index": 0// The first INT-type column of the file from which you want to read data. }, { "type": "string", "value": "alibaba"// The value of the current column. In this code, the value is a constant "alibaba". }` For the column parameter, you must specify the type parameter and specify either the index or value parameter.	Yes	Asterisks (Not recommended)
fieldDelimiter	The column delimiter that is used in the file from which you want to read data. **Note** You must specify the column delimiter for FTP Reader. The default delimiter is commas (,). If this parameter is not specified, the system uses the default delimiter on the codeless UI.	Yes	,
skipHeader	Specifies whether to skip the header of a CSV-like file if the file has a header. The header of a CSV-like file is not skipped by default. The skipHeader parameter is not supported for compressed files.	No	false
encoding	The encoding format of the files from which you want to read data.	No	*utf-8*
nullFormat	The string that represents a null pointer. No standard strings can represent a null pointer 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	No default value
markDoneFileName	The name of the file that is used to indicate that the synchronization node can start. Data Integration detects whether the file exists before data synchronization. If the file does not exist, Data Integration performs the detection again later. Data Integration starts the synchronization node only after the file is detected.	No	No default value
maxRetryTime	The maximum number of retries for the detection of the file if no file is detected. By default, 60 retries are allowed. Data Integration detects the file every 1 minute. The whole process lasts 60 minutes.	No	*60*

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. The CSV reader supports multiple configurations.	No	No default value
fileFormat	The format of the file that is stored by FTP Reader. By default, FTP Reader converts the data to 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. In most cases, you must 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	No default value

Configure FTP Reader by using the codeless UI

1. Configure data sources.

   Configure the source and destination for the synchronization node.

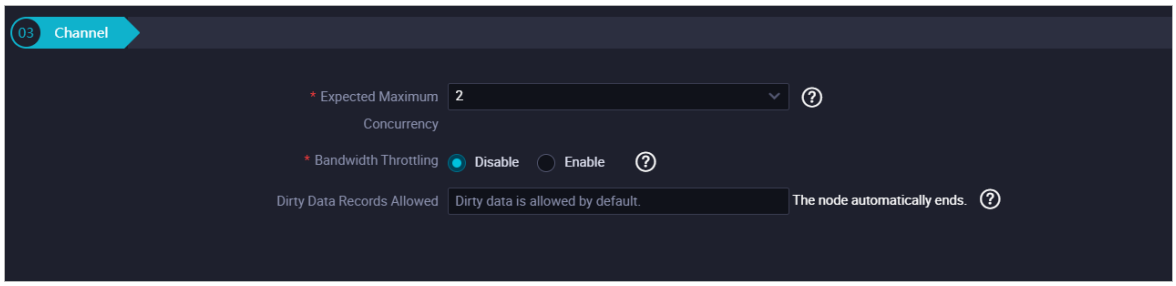
Parameter	Description
Connection	This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
File Path	This parameter corresponds to the path parameter that is described in the preceding section.
File Type	The format of the files from which you want to read data. The default format is CSV.
Field Delimiter	This parameter corresponds to the fieldDelimiter parameter that is described in the preceding section. The default delimiter is commas (,).
Encoding	This parameter corresponds to the encoding parameter that is described in the preceding section. The default encoding format is *UTF-8*.
Null String	This parameter corresponds to the nullFormat parameter that is described in the preceding section. This parameter defines a string that represents the null value.
Compression Format	This parameter corresponds to the compress parameter that is described in the preceding section. The default value is None.
Include Header	This parameter corresponds to the skipHeader parameter that is described in the preceding section. The default value is *No*.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination 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.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that are established.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure FTP Reader by using the code editor

In the following code, a synchronization 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 the null value.
                "compress":"",// The compression format.
                "datasource":"", // The name of the data source.
                "column":[// The names of the columns from which you want to read data.
                    {
                        "index":0,// The IDs of the columns from which you want to read data.
                        "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 that is stored by FTP Reader.
            },
            "name":"Reader",
            "category":"reader"
        },
        {// The following code is used to configure Stream Writer. If you want to configure other wr
iters, see the related 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. The value false ind
icates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling
is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1,// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.1.4.5.3.11. Configure Tablestore Reader

This topic describes the data types and parameters supported by Tablestore Reader and how to configure it by using the code editor.

Tablestore Reader can read incremental data from Tablestore based on the specified range. Tablestore 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.

Tablestore is a NoSQL database service that is built on the Apsara distributed operating system. The service allows you to store and access large volumes of structured data in real time. Tablestore organizes data into instances and tables. It uses data sharding and load balancing technologies to seamlessly expand the data scale.

Tablestore Reader connects to the Tablestore server by using Tablestore SDK for Java and reads data from the server. Then, Tablestore Reader converts the data into a format that is readable to Data Integration based on the official data synchronization protocols, and sends the converted data to a writer.

Tablestore Reader splits a synchronization node into multiple concurrent tasks based on the table range to synchronize data in a Tablestore table. Each Tablestore Reader thread runs a task.

Tablestore Reader supports all Tablestore data types. The following table lists the data types.

Category	Tablestore data type
Integer	INTEGER
Floating point	DOUBLE
String	STRING
Boolean	BOOLEAN
Binary	BINARY

Note Tablestore does not support DATE-type data. Applications use the LONG-type UNIX timestamp to indicate time.

Parameters

Parameter	Description	Required	Default value
endpoint	The endpoint of the Tablestore server.	Yes	None
accessId	The AccessKey ID of the account that is used to access Tablestore.	Yes	None
accessKey	The AccessKey secret of the account that is used to access Tablestore.	Yes	None
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 Tablestore console before you can create and manage tables. Instances are the basic units that you can use to manage Tablestore resources. Access control and resource measurement for applications are implemented at the instance level.	Yes	None

Parameter	Description	Required	Default value
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
column	The columns that you want to synchronize from the source table. The columns are described in a JSON array. Tablestore is a NoSQL database service. You must specify column names for Tablestore Reader to read data. • You can specify common columns. For example, you can specify {"name":"col1"} for Tablestore Reader to read data from column 1. • You can specify partial columns. Tablestore Reader reads data only from the specified columns. • You can specify constant columns. For example, you can specify {"type":"STRING", "value":"DataX"} for Tablestore Reader to read data from 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 Tablestore, and INF_MAX indicates the maximum value specified by Tablestore. If you set the type to INF_MIN or INF_MAX, do not set the value. If you set the value, errors may occur. • You cannot specify a function or custom expression. This is because Tablestore does not provide functions or expressions that are similar to those of SQL. Tablestore Reader cannot read data from columns that contain functions or expressions.	Yes	None

Parameter	Description	Required	Default value
begin and end	The Tablestore table range from which you want to read data. 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 Tablestore 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 data from a Tablestore table with the primary key of [DeviceID, SellerID], specify the begin and end parameters in the following way: `"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.] }`	Yes	None

Parameter	Description	Required	Default value
split	The custom rule for data sharding. This parameter is an advanced configuration item. We recommend that you do not set this parameter. If data is unevenly distributed in a Tablestore table and the automatic sharding feature of Tablestore Reader fails to work, you can customize a sharding rule. The sharding rule that is specified by the split parameter must fall in the range that is specified by the begin and end parameters and must be the values of the partition key. This means that you specify only the values of the partition key instead of the values of primary key columns in the split parameter. To read data from a Tablestore table with the primary key of [DeviceID, SellerID], specify the following parameters: `"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 synchronization node is split into concurrent tasks based on the values of the begin, end, and split parameters. Data is sharded based only on the partition key, which 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 Tablestore Reader by using the codeless UI

This method is not supported.

Configure Tablestore Reader by using the code editor

In the following code, a node is configured to read data from a Tablestore table:

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"ots",// The reader type.
            "parameter":{
                "datasource":"",// The data source.
                "column":[// The columns from which data is read.
                    {
                        "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 source table.
            },
            "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. The value false in
```

```
dicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttlin
g is enabled. The concurrent parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1,// The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.1.4.5.3.12. Configure PostgreSQL Reader

This topic describes the data types and parameters supported by PostgreSQL Reader and how to configure it by using the codeless user interface (UI) and code editor.

PostgreSQL Reader connects to a remote PostgreSQL database and executes an SQL statement to select and read data from the database. ApsaraDB RDS provides the PostgreSQL storage engine.

Specifically, PostgreSQL Reader connects to a remote PostgreSQL database by using Java Database Connectivity (JDBC), generates an SQL statement based on your configurations, and then sends the statement to the database. The system executes the statement and returns results. Then, PostgreSQL Reader assembles the returned data to abstract datasets of custom data types supported by Data Integration, and passes the datasets to a writer.

- PostgreSQL Reader generates the SQL statement based on the table, column, and where parameters that you configured and sends the generated 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. Make sure 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
Floating point	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**
>
> - Data types that are not listed in the preceding table are not supported.
> - You must use the syntax such as `a_inet::varchar` to convert data of the MONEY, INET, and BIT data types.

Parameters

Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table from which you want to read data.	Yes	No default value

column	The columns from which you want to read data in the table. The columns are described in a JSON array. The default value is [*], which indicates all the columns in the table.
	• You can also select specific columns to synchronize.
	• The column order can be changed. You can configure PostgreSQL Reader to synchronize 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 compliance with the SQL syntax that is supported by PostgreSQL, such as `["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': 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 all the columns from which you want to read data. The parameter cannot be left empty.
		Yes	No default value

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 parallel threads to synchronize data, which improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Data can be well distributed to different shards based on the primary key, but not intensively distributed to specific shards. • The splitPk parameter supports data sharding only for integers but not for other types of data, such as strings, floating points, and dates. If you set this parameter to a column of a data type that is not supported by PostgreSQL Reader, PostgreSQL Reader ignores the splitPk parameter and uses a single thread to synchronize data. • If the splitPk parameter is not provided or is left empty, Data Integration uses a single thread to synchronize data	No	No default value
where	The WHERE clause. PostgreSQL Reader generates an SQL statement based on the table, column, and where parameters that are specified, and uses the generated statement to select and read data. For example, set this parameter to `id>2 and sex=1`. • You can use the WHERE clause to synchronize incremental data. • If the where parameter is not provided or is left empty, all data is synchronized.	No	No default value
querySql (advanced parameter, which cannot be set on the codeless UI)	The SQL statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on the value of 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, PostgreSQL Reader ignores the table, column, and where parameters that you configured.	No	No default value
fetchSize	The number of data records to read at a time. This parameter determines the number of interactions between Data Integration and the database on the network and improves read efficiency. **Note** If you set this parameter to a value greater than 2048, an out of memory (OOM) error may occur during data synchronization.	No	*512*

Configure PostgreSQL Reader by using the codeless UI

1. Configure data sources.

   Configure **Source** and **Target** for the synchronization node.

Parameter	Description
Connection	This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
Table	This parameter corresponds to the table parameter that is described in the preceding section.

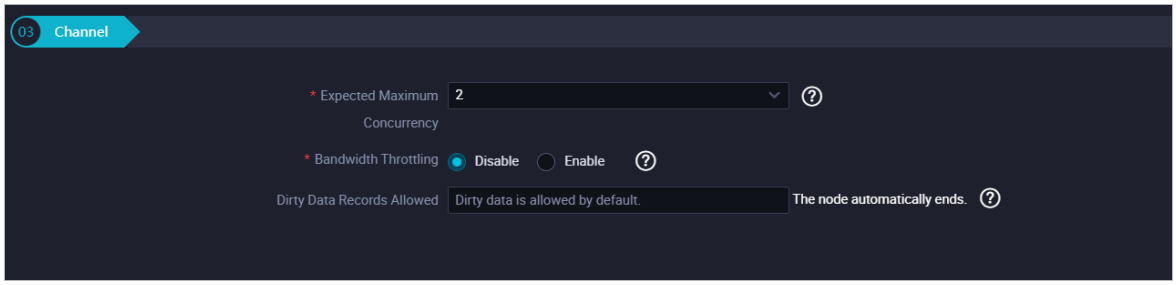
Parameter	Description
Filter	The condition for filtering the data that you want to synchronize. PostgreSQL Reader cannot filter data based on the limit keyword. The SQL syntax is determined by the selected data source.
Shard Key	The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key or an indexed column. Only integer fields are supported. If you specify this parameter, data sharding is performed based on the value of this parameter, and parallel threads can be used to read data. This improves data synchronization efficiency. **Note** The Shard Key parameter is displayed only after you select the data source for the synchronization node.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination 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.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specific rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit the 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 parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure PostgreSQL Reader by using the code editor

In the following code, a synchronization node is configured to read data from a PostgreSQL database:

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"postgresql",// The reader type.
            "parameter":{
                "datasource":"",// The data source.
                "column":[// The columns that you want to synchronize.
                    "col1",
                    "col2"
                ],
                "where":"",// The WHERE clause.
                "splitPk":"",// The shard key based on which the table is sharded. Data Integration initiates parallel threads to synchronize data.
                "table":""// The name of the source table.
            },
            "name":"Reader",
            "category":"reader"
        },
        { // The following template is used to configure Stream Writer. For more information, see the related 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. The value false indicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1 // The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

Additional information

- 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 binary logs. Especially when network conditions are unfavorable, data latency between the primary and secondary databases is unavoidable, which can result in data inconsistencies.

- Concurrency control

  PostgreSQL is a relational database management system (RDBMS), which supports strong consistency for data queries. A database snapshot is created before a 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 parallel threads in a single synchronization node.

  PostgreSQL Reader shards the source table based on the value of the splitPk parameter and runs parallel threads to synchronize data. These parallel threads belong to different transactions. They read data at different points in time. Therefore, the parallel threads observe different snapshots.

  Theoretically, data inconsistencies are unavoidable if a single data synchronization node includes multiple threads. The following workarounds can be used:

  - Do not enable parallel threads for a single synchronization node. Specifically, do not specify the splitPk parameter. This way, data consistency is ensured, but data is synchronized at low efficiency.
  - Do not write data to the source table during data synchronization. This ensures that the data remains unchanged during data synchronization. For example, lock the table or disable data synchronization between primary and secondary databases. This way, data can be efficiently synchronized, but your ongoing services may be interrupted.

- Character encoding

  A PostgreSQL database supports only the EUC_CN and UTF-8 encoding formats for simplified Chinese characters. PostgreSQL Reader uses JDBC to extract data. This enables PostgreSQL Reader to automatically convert the encoding formats of characters. Therefore, you do not need to specify the encoding format.

  If you specify the encoding format for a PostgreSQL database but data is written to the PostgreSQL database in a different encoding format, PostgreSQL Reader cannot recognize this inconsistency and may export garbled characters.

- Incremental data synchronization

  PostgreSQL Reader uses JDBC to connect to a database and uses a SELECT statement with a `WHERE` clause to read incremental datain the following ways:

  - For batch data, 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, PostgreSQL Reader cannot synchronize incremental data but can synchronize only full data.

- Syntax validation

  PostgreSQL Reader allows you to use the querySql parameter to specify a custom SELECT statement but does not verify the syntax of the statement.

2.1.4.5.3.13. 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 reads data from SQL Server. SQL Server Reader connects to a remote SQL Server database by using Java Database Connectivity (JDBC) and executes a SELECT statement to read data from the database.

Specifically, SQL Server Reader connects to a remote SQL Server database by using JDBC, generates a SELECT statement based on your configurations, and then sends the statement to the database. SQL Server executes the statement to read data from the database and returns the result. Then, SQL Server Reader assembles the returned data into abstract datasets of the custom data types supported by Data Integration and sends the datasets to a writer.

- SQL Server Reader generates the SELECT statement based on the table, column, and where parameters that you 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 name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table from which you want to read data. You can select only one source table for each synchronization node.	Yes	No default value

Parameter	Description	Required	Default value

column	The names of the columns from which you want to read data. The columns are described in a JSON array. The default value is [*], which indicates all columns in the source table.
	• You can also select specific columns to synchronize.
	• The column order can be changed. You can configure SQL Server Reader to synchronize 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 compliance with the SQL syntax supported by SQL Server, such as `["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': 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 all the columns that you want to synchronize. This parameter cannot be left empty.
		Yes	No default value

splitPk	The field used for data sharding when SQL Server 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 parallel threads to synchronize data. This way, data can be more efficiently synchronized. • We recommend that you set the splitPk parameter to the primary key of the table. Data can be well distributed to different shards based on the primary key, 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, FLOAT, and DATE. If you set this parameter to a column of an unsupported data type, SQL Server Reader returns an error.	No	No default value
where	The WHERE clause. SQL Server Reader generates a SELECT statement based on the table, column, and where parameters that you configured, and uses the generated SELECT statement to read data. For example, you can set this parameter to limit 10 for a test or `gmt_create > $bizdate` in an actual business scenario to synchronize data on the day. • You can use the WHERE clause to synchronize incremental data. • If you do not specify the where parameter, all data is synchronized.	No	No default value

Parameter	Description	Required	Default value
querySql	The SELECT statement that is used for refined data filtering. Specify this parameter in the following format: `"querysql" : "SELECT statement",` . If you specify this parameter, Data Integration filters data based on this parameter. For example, 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 configured.	No	No default value
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 read efficiency. **Note** A value greater than 2048 may result in an out of memory (OOM) error during data synchronization.	No	*1024*

Configure SQL Server Reader by using the codeless UI

1. Configure data sources.

   Configure the source and destination for the synchronization node.

Parameter	Description
Connection	This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
Table	This parameter corresponds to the table parameter that is described in the preceding section.
Filter	The condition for filtering the data that you want to synchronize. SQL Server Reader cannot filter data based on the limit keyword. The SQL syntax is determined by the selected data source.
Shard Key	The shard key. 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.

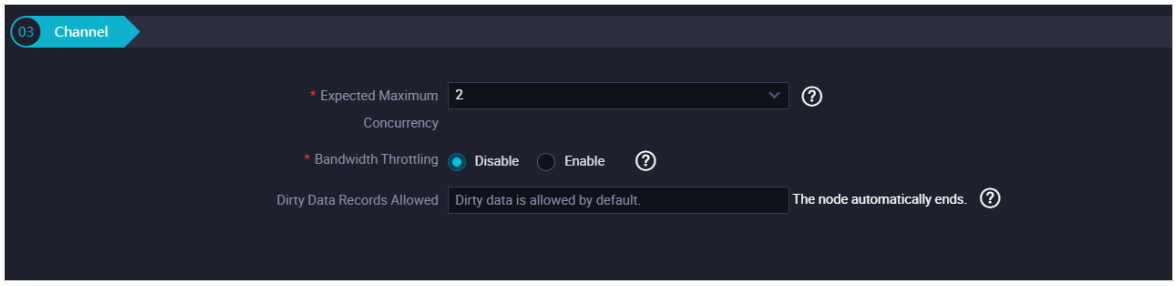
2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination 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.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that are established.

Operation	Description
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit the 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 as Unidentified.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node can use to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure SQL Server Reader by using the code editor

In the following code, a synchronization 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 name of the data source.
                "column":[// The names of the columns from which you want to read data.
                    "id",
                    "name"
                ],
                "where":"",// The WHERE clause.
                "splitPk":"",// The shard key based on which the table is sharded.
                "table":""// The name of the table from which you want to read data.
            },
            "name":"Reader",
            "category":"reader"
        },
        {// The following code is used to configure Stream Writer. For more information about how to configure other writers, see the related 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. The value false indicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1 // The maximum number of parallel 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. For example, the SQL Server data source is sql_server_source, the table that you want to query is dbo.test_table, and the column that you want to query 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 information

- 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 binary logs. Data latency between the primary and secondary databases is unavoidable especially when network conditions are unfavorable, which can result in data inconsistency.

- Parallelism control

  SQL Server is a relational database management system (RDBMS), which supports strong consistency for data queries. A database snapshot is created before a synchronization node starts. SQL Server Reader reads data from the snapshots of the database. Therefore, if new data is written to the database during data synchronization, SQL Server Reader cannot obtain the new data.

  Data consistency cannot be ensured when you enable SQL Server Reader to use parallel threads to synchronize data.

  SQL Server Reader shards the table based on the splitPk parameter and uses parallel threads to synchronize data. These parallel threads belong to different transactions. They read data at different time points. Therefore, the parallel threads observe different snapshots.

  Theoretically, the preceding data inconsistency issue is unavoidable. The following workarounds can be used:

  - Do not enable parallel threads for a single synchronization node. Essentially, do not specify the splitPk parameter. This way, data consistency is ensured, but data is synchronized at low efficiency.
  - Do not write data to the source table to ensure that the data remains unchanged during data synchronization. For example, lock the table and disable data synchronization between the primary and secondary databases. This way, data is efficiently synchronized, but your ongoing services may be interrupted.

- Character encoding

  SQL Server Reader uses JDBC, which can automatically convert the encoding format of characters. Therefore, you do not need to specify the encoding format.

- Incremental data synchronization

  SQL Server Reader connects to a database by using JDBC and uses a SELECT statement with a WHERE clause to read incremental data in the following ways:

  - For batch data, incremental additions, updates, and deletion 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 greater than the maximum ID that is involved in the last synchronization.

If incremental data cannot be distinguished, SQL Server Reader can synchronize only full data but not incremental data.

- 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.

2.1.4.5.3.14. Configure LogHub Reader

This topic describes the data types and parameters supported by LogHub Reader and how to configure it by using the codeless user interface (UI) and code editor.

Log Service is an end-to-end, real-time data logging service and allows you to collect, consume, deliver, query, and analyze log data. Log Service can comprehensively improve the capabilities to process and analyze numerous logs. LogHub Reader consumes real-time log data in LogHub by using Log Service SDK for Java, converts the data to a format that can be read by Data Integration, and then sends the converted data to a writer.

How it works

LogHub Reader consumes real-time log data in LogHub by using Log Service SDK for Java. The following code provides an example of the SDK:

```
<dependency>
    <groupId>com.aliyun.openservices</groupId>
    <artifactId>aliyun-log</artifactId>
    <version>0.6.7</version>
</dependency>
```

In Log Service, a Logstore is a basic unit that you can use to collect, store, and query 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, right-open interval of MD5 values so that intervals do not overlap each other. The range indicated by all intervals covers all the allowed MD5 values. Each shard can independently provide services.

- Write: 5 MB/s, 2,000 times/s
- Read: 10 MB/s, 100 times/s

LogHub Reader consumes log data in shards based on the following process in which the GetCursor and BatchGetLog API operations are called:

- Obtains a cursor based on a time range.
- Reads logs based on the cursor and step parameters and returns the next cursor.
- Keeps moving the cursor to consume logs.
- Splits the node based on shards and uses parallel threads to run the node.

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

Parameter	Description	Required	Default value
endpoint	The endpoint of Log Service, which is a URL that you can use to access a project and log data. It varies based on the Alibaba Cloud region where the project resides and the project name.	Yes	No default value
accessId	The AccessKey ID that is used to access Log Service.	Yes	No default value
accessKey	The AccessKey secret that is used to access Log Service.	Yes	No default value
project	The name of the project. A project is a basic unit for managing resources in Log Service. Projects are used to isolate resources and control access to the resources.	Yes	No default value
logstore	The name of the Logstore from which you want to read data. A Logstore is a basic unit that you can use to collect, store, and query log data in Log Service.	Yes	No default value
batchSize	The number of entries that are queried from Log Service at a time.	No	*128*
column	The name of the columns that you want to synchronize. You can set this parameter to the metadata in Log Service. Supported metadata includes the log topic, unique identifier of the host, hostname, path, and log time. **Note** The column name is case-sensitive.	Yes	No default value
beginDateTime	The start time of data consumption. The value is the time when log data reaches LogHub. This parameter specifies the left boundary of a left-closed, right-open interval in the format of yyyyMMddHHmmss, such as 20180111013000. The parameter can work with the scheduling time parameters in DataWorks. **Note** The beginDateTime and endDateTime parameters must be used in pairs.	You can specify either beginDateTime or beginTimestampMillis.	No default value
endDateTime	The end time of data consumption. This parameter specifies the right boundary of a left-closed, right-open interval in the format of yyyyMMddHHmmss, such as 20180111013010. The parameter can work with the scheduling time parameters in DataWorks. **Note** The end time of the previous interval must be the same as or later than the end time of the current interval. Otherwise, data may not be pulled in some regions.	You can specify either endDateTime or endTimestampMillis.	No default value

Parameter	Description	Required	Default value
beginTimestampMillis	The start time of data consumption. This parameter specifies the left boundary of the left-closed, right-open interval, measured in milliseconds. **Note** The beginTimestampMillis and endTimestampMillis parameters must be used in pairs. The value -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 can specify either beginTimestampMillis or beginDateTime.	No default value
endTimestampMillis	The end time of data consumption. This parameter specifies the right boundary of the left-closed, right-open interval, measured in milliseconds. **Note** The beginTimestampMillis and endTimestampMillis parameters must be used in pairs. The value -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 can specify either endTimestampMillis or endDateTime.	No default value

Configure LogHub Reader by using the codeless UI

1. Configure data sources.

   Configure **Source** and **Target** for the synchronization node.

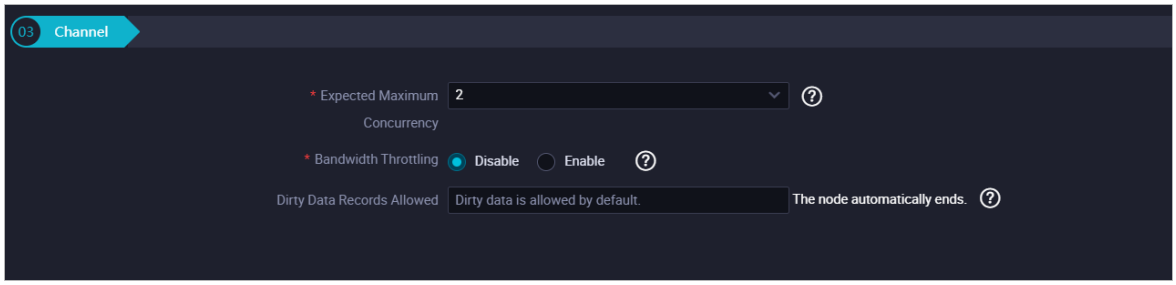
Parameter	Description
Connection	This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
Logstore	The name of the Logstore from which you want to read data.
Start Timestamp	The start time of data consumption. The value is the time when log data reaches LogHub. This parameter specifies the left boundary of a left-closed, right-open interval in the format of yyyyMMddHHmmss, such as 20180111013000. The parameter can work with the scheduling time parameters in DataWorks.
End Timestamp	The end time of data consumption. This parameter specifies the right boundary of a left-closed, right-open interval in the format of yyyyMMddHHmmss, such as 20180111013010. The parameter can work with the scheduling time parameters in DataWorks.
Records per Batch	The number of entries that are queried from Log Service at a time.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination on the right. You can click **Add** to add a field. To delete a field, move the pointer over the field and click the **Delete** icon.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specific rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit the 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 parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure LogHub Reader by using the code editor

In the following code, a synchronization node is configured to read data from Log Service. 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 data source.
             "column":[// The columns that you want to synchronize from the source table.
                 "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 Logstore from which you want to read 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. The value false indic
ates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling i
s enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
         "concurrent":1 // The maximum number of parallel 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__`.

2.1.4.5.3.15. 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

Parameter	Description	Required	Default value
accessKey	The AccessKey secret for connecting to Tablestore.	Yes	None
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.
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.

Parameter	Description	Required	Default value
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.
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.

Parameter	Description	Required	Default value
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
timeRange: {"end"} (applicable only to the multi-version mode)	The end time for reading data. Valid values: 0 to LONG_MAX.	No	*LONG_MAX (9223372036854775806L)*
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": {}
}
```

2.1.4.5.3.16. 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 to synchronize the incremental data of Tablestore. Incremental data can be considered as operation logs that include data and operation information.

Unlike plug-ins used to synchronize full data, OTSStream Reader supports only the multi-version mode. When you use OTSStream Reader to synchronize incremental data, you cannot synchronize the data of specified columns, which is related to the principle of synchronizing incremental data. The following section describes the implementation process.

Before you use OTSStream Reader, make sure that the Stream feature is enabled for your table. You can enable this feature when you create the table, or you can use the UpdateTable operation in the SDK to enable this feature.

The following example describes how to enable the Stream feature:

```
SyncClient client = new SyncClient("", "", "", "");
Enable this feature when you create a table.
CreateTableRequest createTableRequest = new CreateTableRequest(tableMeta);
createTableRequest.setStreamSpecification(new StreamSpecification(true, 24));// The value 24 indicat
es that the incremental data is retained for 24 hours.
client.createTable(createTableRequest);
If this feature is not enabled when you create a table, enable it by using the UpdateTable operation
.
UpdateTableRequest updateTableRequest = new UpdateTableRequest("tableName");
updateTableRequest.setStreamSpecification(new StreamSpecification(true, 24));
client.updateTable(updateTableRequest);
```

How it works

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 Tablestore server additionally saves your operation logs. Each partition has a sequential operation log queue. Each operation log is recycled after your specified expiration time.

The Tablestore SDK provides several Stream APIs that are used to read these operation logs. OTSStream Reader obtains incremental data by using these APIs, transforms the incremental data into multiple six-tuples (pk, colName, version, colValue, opType, and sequenceInfo), and then synchronizes them into MaxCompute.

Format of the synchronized data

In the multi-version model of Tablestore, table data is organized in a three-level mode: row, column, and version. One row can have multiple columns, and the column name is not fixed. Each column can have multiple versions, and each version has a specific timestamp (the version number).

You can perform read or write operations by using Tablestore APIs. Tablestore records the incremental data by recording the recent write and modification operations on table data. Therefore, incremental data can be considered as a set of operation records.

Tablestore supports the following three types of modification operations:

- PutRow: writes a row. If the row already exists, it is overwritten.
- UpdateRow: updates a row without the need to change other data of the original row. You can add column values, overwrite column values if the related version of the column already exists, delete all the versions of a column, or delete a version of a column.
- DeleteRow: deletes a row.

Tablestore generates incremental data records based on each type of operation. OTSStream Reader reads these records and synchronizes the data in the format supported by DataX.

Tablestore 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 Tablestore. A row in Tablestore may correspond to multiple synchronized rows. Each synchronized 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 format supported by DataX, 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 column based on the primary key and the column name.
- DR (DELETE_ROW): deletes a row. Delete all the data of the row based on 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 synchronized, which corresponds to three rows in the Tablestore 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 all the 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

OTSStream Reader supports all Tablestore 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 name of the data source. It must be the same as the added data source. You can add data sources by using the code editor.	Yes	No default value
dataTable	The name of the table from which incremental data is synchronized. You must enable the Stream feature for a table when you create the table, or you can call the UpdateTable operation to enable this feature.	Yes	No default value

Parameter	Description	Required	Default value
statusTable	The name of the table that OTSStream Reader uses to store status records. These records help find the data that is not required and improve synchronization efficiency. A status table is used to store status records. If no such table exists, OTSStream Reader automatically creates one. After the running of an offline export task 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 need only to provide a table name rather than manually creating a status table. OTSStreamReader attempts to create a status table under your instance. If no such table exists, OTSStream Reader automatically creates one. If the table already exists, OTSStream Reader determines whether the metadata of the table meets the expectation. If not, an error is reported. • After the running of an export task is completed, you do not need to delete the table. The status records in the table can be used for the next export task. • The table enables Time To Live (TTL), and data automatically expires, which indicates that the data volume is small. • You can use the same status table to store the status records of the multiple tables that are specified by the dataTable parameter and managed by the same instance. The status records are independent of each other. In conclusion, you can configure a name similar to TableStoreStreamReaderStatusTable. You must make sure that the name is inconsistent with that of a business-related table.	Yes	No default value
startTimestampMillis	The start time (included) of the incremental data, in milliseconds. • OTSStream Reader finds a point that corresponds to the time specified by the startTimestampMillis parameter from the status table, and starts to read and synchronize data from this point. • If OTSStream Reader cannot find the required point, it starts to read incremental data retained by the system from the first entry, and skips the data which is written later than the time specified by startTimestampMillis.	No	No default value
endTimestampMillis	The end time (excluded) of the incremental data, in milliseconds. • OTSStream Reader exports data from the time specified by the startTimestampMillis parameter and stops exporting data when the timestamp of a data record is later than or equal to the time specified by the endTimestampMillis parameter. • After all the incremental data is read, OTSStream Reader stops reading data even before the time specified by the endTimestampMillis parameter.	No	No default value
date	The date on which data is synchronized. Specify this parameter in the yyyyMMdd format, such as 20151111. You must specify either the date 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	No default value

Parameter	Description	Required	Default value
isExportSequenceInfo	Specifies whether to synchronize time series information which includes the time when data is written. The default value is *false*, which indicates that time series information is not synchronized.	No	No default value
maxRetries	The maximum number of retries for each request of reading incremental data from Tablestore. The default value is 30. Retries are performed at specific intervals. The total time of 30 retries is about 5 minutes. You can keep the default settings.	No	No default value
startTimeString	The start time (included) of the incremental data, in milliseconds. Specify this parameter in the `yyyymmddhh24miss` format.	No	No default value
endTimeString	The end time (excluded) of the incremental data, in milliseconds. Specify this parameter in the `yyyymmddhh24miss` format.	No	No default value
mode	The synchronization mode. If this parameter is set to single_version_and_update_only, data is exported by row. By default, this parameter is not specified, and data is not synchronized by column.	No	No default value

Configure OTSStream Reader by using the codeless UI

This method is not supported.

Configure OTSStream Reader by using the code editor

The following example shows how to configure a synchronization node to read the incremental data of Tablestore. 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 OTSStream Reader uses to store status records.
                "maxRetries":30,// The maximum number of retries on each request of reading incremental data from Tablestore. It is set to 30 by default.
                "isExportSequenceInfo":false,// Specifies whether to synchronize the time series information.
                "datasource":"$srcDatasource",// The name of the data source.
                "startTimeString":"${startTime}",// The start time (included) of the incremental data.
                "table":"",// The name of the table from which you want to read data.
                "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. The value false indicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.1.4.5.3.17. 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.		

Parameter	Description	Required	Default value
jdbcUrl	{ "name": "rdbmsreader", "class": "com.alibaba.datax.plugin.reader.rdbmsreader.RdbmsReader", "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
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

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 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
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

Parameter	Description	Required	Default value
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"
        }
    ],
    "type": "job",
    "version": "2.0"
}
```

2.1.4.5.3.18. Configure Stream Reader

This topic describes the data types and parameters supported by Stream Reader and how to configure it by using the codeless user interface (UI) and code editor.

Stream Reader automatically generates data from the memory. It is mainly used to test the basic features and performance of data synchronization.

The following table lists the data types supported by Stream Reader.

Stream Reader Data type	Category
string	String
long	Long integer
date	Date and time
bool	Boolean
bytes	Bytes

Parameters

Parameter	Description	Required	Default value
column	The data and types of columns in the source table. You can specify multiple columns. You can set this parameter to generate random strings of a specific length. The following code provides an example: `"column" : [ { "random": "8,15" }, { "random": "10,10" } ]` Configuration items: • "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	No default value
sliceRecordCount	The number of columns that are repeatedly generated.	Yes	No default value

Configure Stream Reader by using the codeless UI

This method is not supported.

Configure Stream Reader by using the code editor

In the following example, a synchronization node is configured to read data from the memory:

```
{
    "type":"job",
```

```
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",// The reader type.
            "parameter":{
                "column":[// The columns from which you want to read data.
                    {
                        "type":"string",// The data type.
                        "value":"field"// The value.
                    },
                    {
                        "type":"long",
                        "value":100
                    },
                    {
                        "dateFormat":"yyyy-MM-dd HH:mm:ss",// The time format.
                        "type":"date",
                        "value":"2014-12-12 12:12:12"
                    },
                    {
                        "type":"bool",
                        "value":true
                    },
                    {
                        "type":"bytes",
                        "value":"byte string"
                    }
                ],
                "sliceRecordCount":"100000"// The number of columns that are repeatedly generated.
            },
            "name":"Reader",
            "category":"reader"
        },
        {// The following template is used to configure Stream Writer. For more information about ho
w to configure other writers, see the related 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. The value false in
dicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttlin
g is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1// The number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
```

```
        }
}
```

2.1.4.5.3.19. Configure Hive Reader

Hive Reader can read data from Hive. This topic describes how Hive Reader works, the parameters supported by Hive Reader, and how to configure it by using the 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 Hadoop Distributed File System (HDFS).
- Hive uses MapReduce programs to analyze data at the underlying layer.
- Hive runs MapReduce programs on Yarn.

How it works

Hive Reader connects to a Hive metadatabase and parses the configuration to obtain the metadata information of the HDFS file that corresponds to the Hive table, such as the storage path, format, and column delimiter. Then, Hive Reader reads data from the HDFS file.

Hive Reader connects to the HiveMetastore service to obtain the metadata information of the Hive table. Hive Reader can read data based on the following items:

- HDFS files

  Hive Reader connects to Hive Metastore and parses the configuration to obtain the metadata information of the HDFS file that corresponds to the Hive table, such as the storage path, format, and column delimiter. Then, Hive Reader reads data from the HDFS file.

  The underlying logic of Hive Reader is the same as that of HDFS Reader. After Hive Reader reads data, it synchronizes data from the HDFS file to the destination table by using Hive Java Database Connectivity (JDBC). You can configure the parameters of HDFS Reader in the parameters of Hive Reader, and the configured parameters are transparently transmitted to HDFS Reader.

- Hive JDBC

  Hive Reader connects to HiveServer2 by using Hive JDBC to read data. Hive Reader allows you to specify the where parameter to filter data and execute SQL statements to read data.

Parameters

Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source.	Yes	No default value
table	The name of the Hive table from which you want to read data. **Note** The name is case-sensitive.	Yes	No default value

Parameter	Description	Required	Default value
readMode	The mode in which Hive Reader reads data. • If Hive Reader reads data based on HDFS files, set this parameter to `"readMode":"hdfs"`. • If Hive Reader reads data based on Hive JDBC, set this parameter to `"readMode":"jdbc"`.	No	No default value
partition	The partitions from which you want to read data in the Hive table. • This parameter is not required if Hive Reader reads data based on Hive JDBC. • If the Hive table that you want to synchronize is a partitioned table, you must configure the partition parameter. The synchronization node reads data from partitions that are specified by the partition parameter. Hive Reader allows you to use asterisks (*) as wildcards to specify partitions in the format of `pt1=a,pt2=b,...`. • If the Hive table is a non-partitioned table, the partition parameter is not required.	No	No default value
column	The names of the columns from which you want to read data in the Hive table. Example: `"column": ["id", "name"]`. • You can select specific columns to synchronize. • The column order can be changed. You can configure Hive Reader to synchronize the specified columns in an order different from that specified in the schema of the table. • Partition columns are supported. • Constants are supported. • The column parameter must explicitly specify all the columns that you want to synchronize. This parameter cannot be left empty.	Yes	No default value
querySql	If Hive Reader reads data based on Hive JDBC, you can configure the querySql parameter to read data.	No	No default value
where	If Hive Reader reads data based on Hive JDBC, you can specify the where parameter to filter data.	No	No default value

Configure Hive Reader by using the code editor

Hive Reader can read data based on HDFS files or Hive JDBC:

- Read data based on HDFS files

```
{
    "type": "job",
    "steps": [
        {
            "stepType": "hive",
            "parameter": {
                "partition": "pt1=a,pt2=b,pt3=c",// The partitions in the Hive table.
                "datasource": "hive_not_ha_****",// The name of the data source.
                "column": [// The names of the columns from which you want to read data.
                    "id",
                    "pt2",
                    "pt1"
                ],
                "readMode": "hdfs",// The mode in which Hive Reader reads data.
                "table": "part_table_1"
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "hive",
            "parameter": {
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "version": "2.0",
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": ""// The maximum number of dirty data records allowed.
        },
        "speed": {
            "concurrent": 2,// The maximum number of parallel threads.
            "throttle": false
        }
    }
}
```

- Read data based on Hive JDBC

```
{
    "type": "job",
    "steps": [
        {
            "stepType": "hive",
            "parameter": {
                "querySql": "select id,name,age from part_table_1 where pt2='B'",
                "datasource": "hive_not_ha_****",// The name of the data source.
                "column": [// The names of the columns from which you want to read data.
                    "id",
                    "name",
                    "age"
                ],
                "where": "",
                "table": "part_table_1",
                "readMode": "jdbc"// The mode in which Hive Reader reads data.
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "hive",
            "parameter": {
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "version": "2.0",
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": ""
        },
        "speed": {
            "throttle": false,
            "concurrent": 2// The maximum number of parallel threads.
        }
    }
}
```

2.1.4.5.3.20. 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 que
ry that uses 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.
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*

Parameter	Description	Required	Default value
search	The query parameter of Elasticsearch.	Yes	None
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[*]"
        }
```

2.1.4.5.3.21. 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
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

Parameter	Description	Required	Default value
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
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

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. 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"
        }
    ],
    "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 throt
tled. The maximum transmission rate takes effect only if you set this parameter to true.
             "concurrent": 1 // The maximum number of concurrent threads.
         }
    }
}
```

2.1.4.5.3.22. 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
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

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

Parameter	Description	Required	Default value
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;"], // Spec
ify the querySql parameter 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 i
ndicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is thrott
led. 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"
            }
        ]
    }
}
```

2.1.4.5.3.23. KingbaseES Reader

This topic describes the data types and parameters that are supported by KingbaseES Reader and how to configure KingbaseES Reader by using the codeless user interface (UI) and code editor. Before you create a Data Integration node, you can refer to this topic to familiarize yourself with the data types and parameters that you must configure for KingbaseES Reader to read data from data sources.

Context

KingbaseES Reader connects to a remote KingbaseES database by using Java Database Connectivity (JDBC), generates an SQL statement based on your configurations, and then sends the statement to the database. The system executes the statement on the database and returns data. Then, KingbaseES Reader assembles the returned data into abstract datasets of the data types supported by Data Integration and sends the datasets to a writer.

Data types

The following table lists the data types that are supported by KingbaseES Reader.

Data type	The data type of SAP HANA
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 BOOL
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

Notice

- Data types that are not listed in the preceding table are not supported.
- KingbaseES Reader processes TINYINT(1) as an integer data type.

Parameters

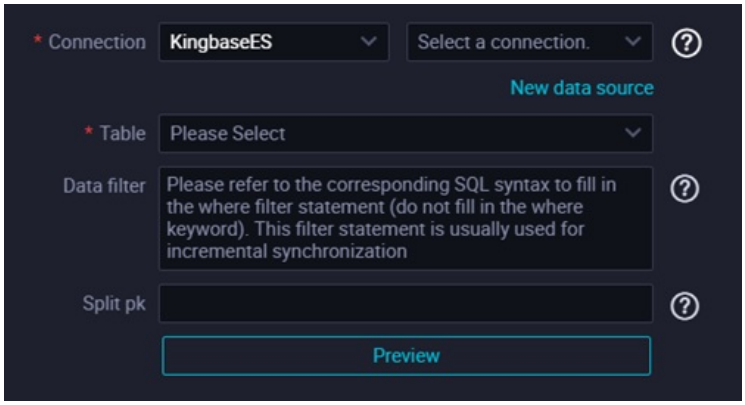
Parameter	Description
datasource	The name of the data source.
column	The names of the columns from which you want to read data. If you want to read data from all the columns in the source table, set this parameter to an asterisk (*).
table	The name of the source table.
splitPk	The field that is used for data sharding when KingbaseES Reader reads data. If you specify this parameter, the source table is sharded based on the value of this parameter. Data Integration then runs parallel threads to read data. You can specify a field of an integer data type for the splitPk parameter. If the source table does not contain fields of integer data types, you can leave this parameter empty.

Configure KingbaseES Reader by using the codeless UI

1. Configure data sources.

   Log on to the DataWorks console. The **DataStudio** page appears. On the DataStudio page, move the pointer over the icon and choose **Data Integration > Batch Synchronization**. In the **Create Node** dialog box, configure the parameters to create a batch synchronization node.

   Configure **Source** and Target for the synchronization node.



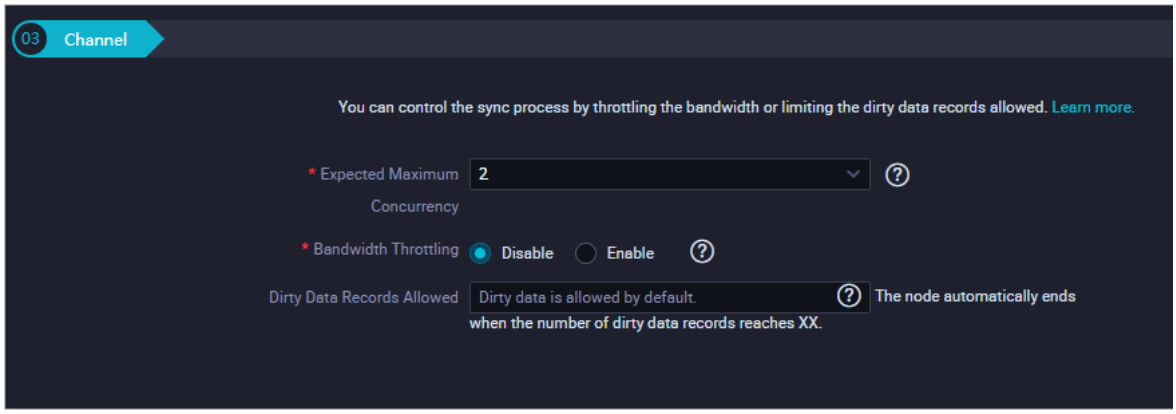
2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination on the right. You can click **Add** to add a field. To remove an added field, move the pointer over the field and click the **Remove** icon.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specific rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit the 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.

Operation	Description
Add	Click **Add** to add a field. Take note of the following rules when you 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 that are supported by relational databases, such as now() and count(1). ○ If the field that you entered cannot be parsed, the value of Type for the field is Unidentified.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure KingbaseES Reader by using the code editor

The following examples show how to configure KingbaseES Reader to read data from a database or table that is not sharded and how to configure KingbaseES Reader to read data from a database or table that is sharded.

- Configure KingbaseES Reader to read data from a database or table that is not sharded

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"kingbasees",// The reader type.
            "parameter":{
                "column":[// The names of the columns from which you want to read data.
                    "id"
                ],
                "connection":[
                    {   "querySql":["select a,b from join1 c join join2 d on c.id = d.id;"], // Th
e SQL statement that is used to read data from the source table.
                        "datasource":"",// The name of the data source.
                        "table":[// The name of the source table. The table name must be enclosed
in brackets [].
                            "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. The value false
indicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throt
tling is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.

            "concurrent":1,// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

- Configure KingbaseES Reader to read data from a database or table that is sharded

> **Note** When you configure a synchronization node to read data from a sharded database or table, you can select multiple KingbaseES tables with the same schema.

```
{
    "type": "job",
    "version": "1.0",
    "configuration": {
        "reader": {
            "plugin": "kingbasees",
            "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": {
        }
    }
}
```

2.1.4.5.3.24. SAP HANA Reader

This topic describes the data types and parameters that are supported by SAP HANA Reader and how to configure SAP HANA Reader by using the codeless user interface (UI) and code editor. Before you create a Data Integration node, you can refer to this topic to familiarize yourself with the data types and parameters that you must configure for SAP HANA Reader to read data from data sources.

Context

SAP HANA Reader connects to a remote SAP HANA database by using Java Database Connectivity (JDBC), generates an SQL statement based on your configurations, and then sends the statement to the database. The system executes the statement on the database and returns data. Then, SAP HANA Reader assembles the returned data into abstract datasets of the data types supported by Data Integration and sends the datasets to a writer.

Data types

The following table lists the data types that are supported by SAP HANA Reader.

Category	SAP HANA 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

Notice

- Data types that are not listed in the preceding table are not supported.
- SAP HANA Reader processes TINYINT(1) as an integer data type.

Parameters

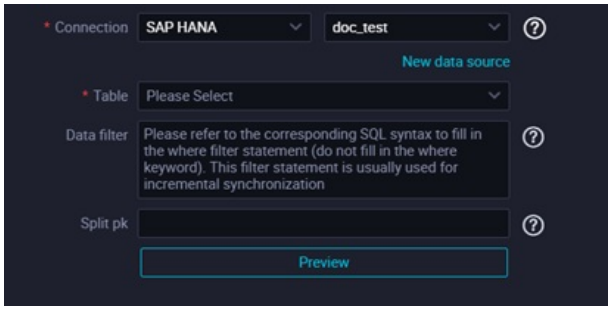
Parameter	Description
datasource	The name of the data source from which you want to read data. If no data source is available, click **Add data source** to add a data source.
column	The names of the columns from which you want to read data. If you want to read data from all the columns in the source table, set this parameter to an asterisk (*).
table	The name of the source table.
splitPk	The field that is used for data sharding when SAP HANA Reader reads data. If you specify this parameter, the source table is sharded based on the value of this parameter. Data Integration then runs parallel threads to read data. You can specify a field of an integer data type for the splitPk parameter. If the source table does not contain fields of integer data types, you can leave this parameter empty.

Configure SAP HANA Reader by using the codeless UI

1. Configure data sources.

   Log on to the DataWorks console. The **DataStudio** page appears. On the DataStudio page, move the pointer over the icon and choose **Data Integration > Batch Synchronization**. In the **Create Node** dialog box, configure the parameters to create a batch synchronization node.

   Configure **Source** and Target for the synchronization node.



Parameter	Description
Connection	The name of the data source from which you want to read data. This parameter corresponds to the datasource parameter that is described in the preceding section.
Table	The name of the table from which you want to read data. This parameter corresponds to the table parameter that is described in the preceding section.
Data filter	The condition that is used to filter the data you want to read. Filtering based on the LIMIT keyword is not supported. The SQL syntax is determined by the selected data source.
Split pk	The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key column or an indexed column. Only integer columns are supported. If you specify this parameter, data sharding is performed based on the value of this parameter, and parallel threads can be used to read data. This improves data synchronization efficiency. **Note** The Split pk parameter is displayed only after you select the data source for the synchronization node.

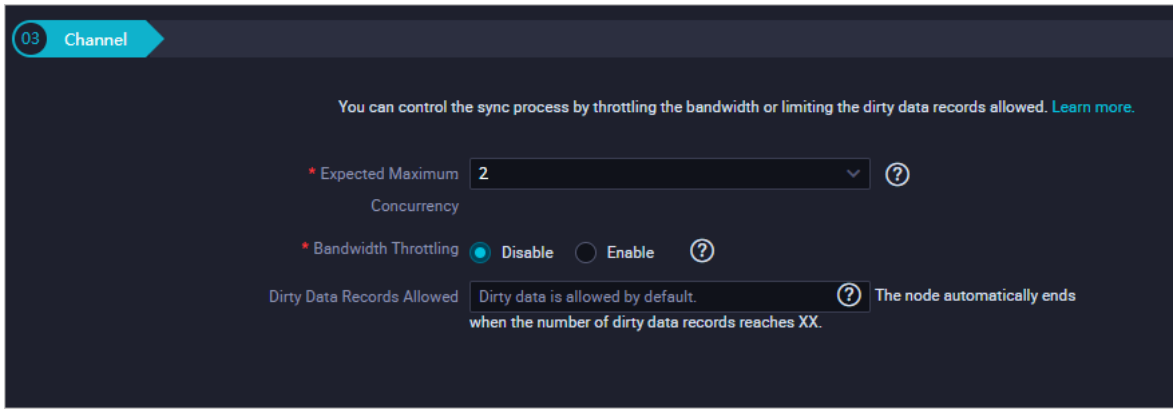
2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination on the right. You can click **Add** to add a field. To remove an added field, move the pointer over the field and click the **Remove** icon.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.

Operation	Description
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specific rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit the 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. Take note of the following rules when you 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 that are supported by relational databases, such as now() and count(1). ○ If the field that you entered cannot be parsed, the value of Type for the field is Unidentified.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure SAP HANA Reader by using the code editor

The following examples show how to configure SAP HANA Reader to read data from a database or table that is not sharded and how to configure SAP HANA Reader to read data from a database or table that is sharded.

- Configure SAP HANA Reader to read data from a database or table that is not sharded

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"saphana",// The reader type.
            "parameter":{
                "column":[// The names of the columns from which you want to read data.
                    "id"
                ],
                "connection":[
                    {   "querySql":["select a,b from join1 c join join2 d on c.id = d.id;"], // Th
e SQL statement that is used to read data from the source table.
                        "datasource":"",// The name of the data source.
                        "table":[// The name of the source table. The table name must be enclosed
in brackets [].
                            "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. The value false
indicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throt
tling is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.

            "concurrent":1,// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

- Configure SAP HANA Reader to read data from a database or table that is sharded

> **Note** When you configure a synchronization node to read data from a sharded database or table, you can select multiple SAP HANA tables with the same schema.

```
{
    "type": "job",
    "version": "1.0",
    "configuration": {
        "reader": {
            "plugin": "saphana",
            "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": {
        }
    }
}
```

2.1.4.5.4. Configure the writer

2.1.4.5.4.1. 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 user interface (UI) and code editor.

Prerequisites

Data Integration can synchronize data to AnalyticDB for MySQL 2.0 in real time. You must create a table to which you want to write data in the destination AnalyticDB for MySQL 2.0 database before synchronization. In real-time synchronization mode, data can be more efficiently synchronized, and the process is simple.

Before you configure AnalyticDB for MySQL 2.0 Writer, you must configure an AnalyticDB for MySQL 2.0 data source.

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
Boolean	BOOLEAN

Parameters

Parameter	Description	Required	Default value
url	The URL used to connect to the AnalyticDB for MySQL 2.0 database. Specify this parameter in the IP address:Port format.	Yes	No default value
database	The name of the AnalyticDB for MySQL 2.0 database.	Yes	No default value
Access Id	The AccessKey ID used to connect to the AnalyticDB for MySQL 2.0 database.	Yes	No default value
Access Key	The AccessKey secret used to connect to the AnalyticDB for MySQL 2.0 database.	Yes	No default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table to which you want to write data.	Yes	No default value
partition	The names of the partitions to which you want to write data in the destination table. This parameter is required only when the destination table is a partitioned table.	No	No default value
writeMode	The write mode. Set the value to insert. In this mode, if a primary key conflict occurs, the conflicting rows are overwritten.	Yes	No default value
column	The columns to which you want to write data in the destination table. Separate the names with commas (,), such as ["a", "b", "c"]. To write data to all the columns in the destination table, set the value to [*].	Yes	No default value

Parameter	Description	Required	Default value
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 specified, the URL changes to a Java Database Connectivity (JDBC) connection string used to connect to the AnalyticDB for MySQL 2.0 database. For example, set the suffix parameter to `autoReconnect=true&failOverReadOnly=false&maxReconnects=10`.	No	No default value
batchSize	The number of data records to write at a time.	Required only when the writeMode parameter is set to insert	No default value
bufferSize	The size of the Data Integration data buffer, which is designed to improve the performance of AnalyticDB for MySQL 2.0. Source data 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 value of the batchSize parameter. We recommend that you set the bufferSize parameter to a value that is a multiple of the value of the batchSize parameter. This parameter takes effect only when the writeMode parameter is set to insert.	Required only when the writeMode parameter is set to insert	No default value

Configure AnalyticDB for MySQL 2.0 Writer by using the codeless UI

1. Configure data sources.

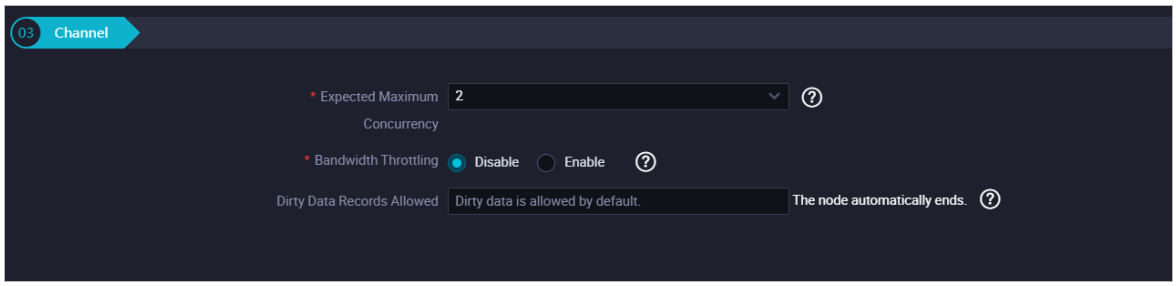
   Configure **Source** and **Target** for the synchronization node.

Parameter	Description
Connection	This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
Table	This parameter corresponds to the table parameter that is described in the preceding section.
Write Method	This parameter corresponds to the writeMode parameter that is described in the preceding section.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section. Fields in the source on the left have a one-to-one mapping with fields in the destination on the right.

Operation	Description
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specified rules.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
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 names of the partitions to which you want to write data.
                "datasource":"",// The name of the data source.
                "column":[// The names of the columns to which you want to write data.
                     "id"
                ],
                "writeMode":"insert",// The write mode.
                "batchSize":"256",// The number of data records to write at a time.
                "table":"",// The name of the table to which you want to write data.
                "overWrite":"true"// Specifies whether to overwrite the destination table when data
is written to AnalyticDB for MySQL 2.0. The value true indicates that the destination table is overw
ritten, and the value false indicates that the destination table is not overwritten,and new data is
appended to the existing data. This parameter 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. The value false in
dicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttlin
g is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1,// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.1.4.5.4.2. 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 and file. 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
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*

Parameter	Description	Required	Default value
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 t
opics to distinguish 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 be
fore sending 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 onl
y if you set this parameter to true.
        }
    },
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

2.1.4.5.4.3. Configure Db2 Writer

This topic describes the data types and parameters supported by Db2 Writer and how to configure it by using the code editor.

Db2 Writer allows you to write data to tables stored in Db2 databases. Specifically, Db2 Writer connects to a remote Db2 database through Java Database Connectivity (JDBC), and runs an `INSERT INTO` statement to write data to the Db2 database. Internally, data is submitted to Db2 database in batches.

Db2 Writer is designed for extract-transform-load (ETL) developers to import data from data warehouses to Db2 databases. Db2 Writer can also be used as a data migration tool by users such as database administrators (DBAs).

Db2 Writer obtains data from a Data Integration reader, and writes the data to the destination database by running 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, Db2 Writer makes `batch updates with the PreparedStatement method` and sets the `rewriteBatchedStatements` parameter to true. In this way, Db2 Writer buffers data, and submits a write request when the amount of data in the buffer reaches a specific threshold.

> **Note** A sync node that uses Db2 Writer must have at least the permission to run 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.

Db2 Writer supports most Db2 data types. Make sure that your data types are supported.

The following table lists the data types supported by Db2 Writer.

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
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 Db2 database.	Yes	None
password	The password for connecting to the Db2 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 (,). 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
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.	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 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 Db2 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 Db2 Writer by using the codeless UI

Currently, the codeless UI is not supported for Db2 Writer.

Configure Db2 Writer by using the code editor

In the following code, a node is configured to write data to a Db2 database.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"db2",// The writer type.
            "parameter":{
                "postSql":[],// The SQL statement to run after the sync node is run.
                "password":"",// The password for connecting to the Db2 database.
                "jdbcUrl":"jdbc:db2://ip:port/database",// The JDBC URL for connecting to the Db2 da
tabase.
                "column":[
                    "id"
                ],
                "batchSize":1024,// The number of data records to write at a time.
                "table":"", // The name of the destination table.
                "username":"", // The username for connecting to the Db2 database.
                "preSql":[]// The SQL statement to run 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 i
ndicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is thrott
led. 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"
            }
        ]
    }
}
```

2.1.4.5.4.4. Configure PolarDB-X Writer

This topic describes the data types and parameters supported by PolarDB-X Writer and how to configure it by using the codeless user interface (UI) and code editor.

PolarDB-X Writer can write data to tables stored in PolarDB-X databases. PolarDB-X Writer connects to the proxy of a remote PolarDB-X database by using Java Database Connectivity (JDBC), and executes a `REPLACE INTO` statement to write data to the PolarDB-X database.

> **Note**
> - To execute the `REPLACE INTO` statement, make sure that your table has the primary key or a unique index to avoid duplicate data.
> - You must add a PolarDB-X data source before you configure PolarDB-X Writer.

PolarDB-X Writer is designed for extract, transform, load (ETL) developers to import data from data warehouses to PolarDB-X databases. PolarDB-X Writer can also be used as a data migration tool by users such as database administrators.

PolarDB-X 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, the original rows are replaced by new rows. PolarDB-X Writer sends data to the PolarDB-X 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 the data is written to multiple tables.

> **Note** A synchronization node that uses PolarDB-X Writer must have the permissions 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, PolarDB-X Writer supports most PolarDB-X data types. Make sure that your data types are supported.

The following table lists the data types supported by PolarDB-X Writer.

Category	PolarDB-X 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

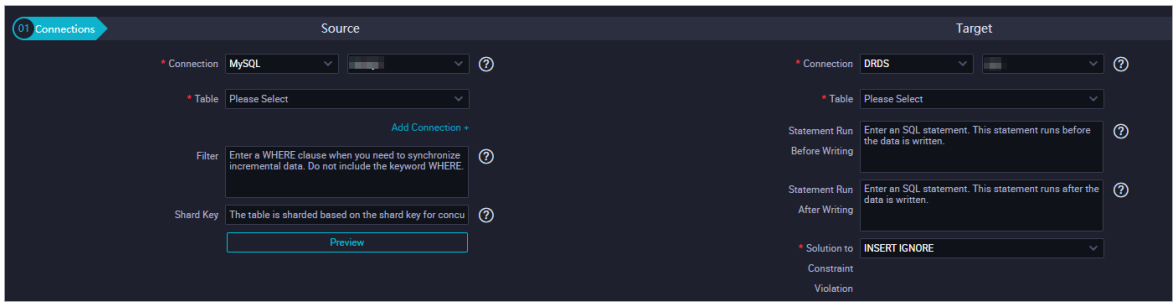
Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table to which you want to write data.	Yes	No default value

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 the original rows are replaced by new rows, and data is written to PolarDB-X. • *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, the original rows are deleted, and new rows are inserted. This indicates that all the fields of the original rows are replaced.	No	*insert*
column	The names of the columns to which you want to write data in the destination table. Separate the names with commas (,), such as "column": ["id","name","age"]. To write data to all the columns in the destination table, set the value to ["*"].	Yes	No default value
preSql	The SQL statement that you want to execute before the synchronization node is run. For example, you can set this parameter to the SQL statement that is used to clear outdated data. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	No default value
postSql	The SQL statement that you want to execute after the synchronization node is run. For example, you can set this parameter to the SQL statement that is used to add a timestamp. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	No default value
batchSize	The number of data records to write at a time. Set this parameter to an appropriate value based on your business requirements. This greatly reduces the interactions between Data Integration and PolarDB-X and increases throughput. If you set this parameter to an excessively large value, an out of memory (OOM) error may occur during data synchronization.	No	*1,024*

Configure PolarDB-X Writer by using the codeless UI

1. Configure data sources.

   Configure **Source** and **Target** for the synchronization node.



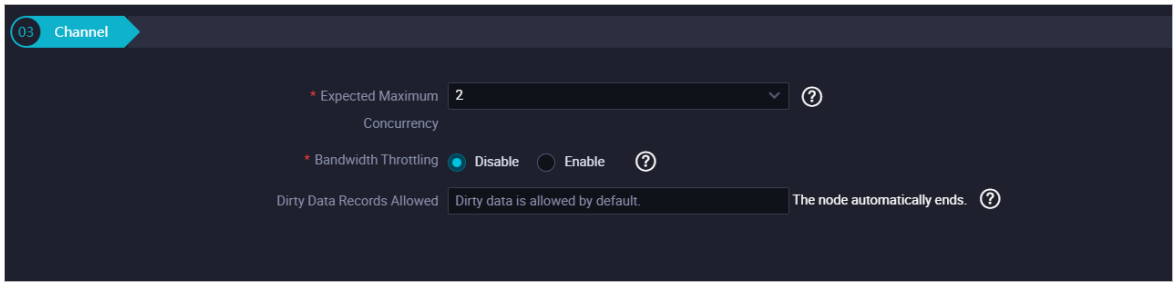
Parameter	Description
Connection	This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.

Parameter	Description
Table	This parameter corresponds to the table parameter that is described in the preceding section.
Statement Run Before Writing	This parameter corresponds to the preSql parameter that is described in the preceding section. Enter the SQL statement that you want to execute before the synchronization node is run.
Statement Run After Writing	This parameter corresponds to the postSql parameter that is described in the preceding section. Enter the SQL statement that you want to execute after the synchronization node is run.
Solution to Constraint Violation	This parameter corresponds to the writeMode parameter that is described in the preceding section. You can select the desired write mode.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section. Fields in the source on the left have a one-to-one mapping with fields in the destination on the right.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specified rules.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure PolarDB-X Writer by using the code editor

In the following code, a synchronization node is configured to write data to PolarDB-X:

```
{
    "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 that you want to execute after the synchronization node is run.
                "datasource":"",// The name of the data source.
                "column":[// The names of the columns to which you want to write data.
                "id"
                ],
                "writeMode":"insert ignore",
                "batchSize":"1024",// The number of data records to write at a time.
                "table":"test",// The name of the table to which you want to write data.
                "preSql":[]// The SQL statement that you want to execute before the synchronization 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. The value false indicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1,// The maximum number of parallel threads.
                }
            },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
                }
            ]
        }
    }
```

2.1.4.5.4.5. FTP Writer

This topic describes the parameters that are supported by FTP Writer and how to configure FTP Writer by using the codeless user interface (UI) and code editor.

FTP Writer can write one or more CSV files to a remote FTP server. FTP Writer converts the data that is obtained from a Data Integration reader to CSV files. Then, FTP Writer writes these files to a remote FTP server by using FTP-related network protocols.

> **Note** Before you configure FTP Writer, you must configure an FTP data source. For more information, see Configure an FTP connection.

FTP Writer can write files that store logical two-dimensional tables to an FTP server, such as CSV files that store text data.

FTP Writer converts the data that is obtained from a Data Integration reader to files and writes the files to an FTP server. The files on the FTP server store only unstructured data. FTP Writer provides the following features:

- Writes only files that store text data. The text data must be logical two-dimensional tables. FTP Writer cannot write files that store binary large object (BLOB) data, such as video data.
- Writes CSV-like and text files that contain custom delimiters.
- Uses parallel threads to write multiple files. Each thread is used to write one file.

FTP Writer does not support the following features:

- Uses parallel threads to write a single file.
- Distinguishes between data types. FTP does not distinguish between data types. Therefore, FTP Writer writes all data as strings to files on an FTP server.
- Writes compressed files to an FTP server.

Parameters

Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
timeout	The timeout of connections to the FTP server. Unit: milliseconds.	No	60000
path	The directory on the FTP server to which you want to write data. FTP Writer uses parallel threads to write multiple files to the directory based on the parallelism setting.	Yes	No default value
fileName	The name prefix of the files that you want to write to the FTP server. A random suffix is appended to the specified prefix to form the actual file name that is used by each thread.	Yes	No default value

Parameter	Description	Required	Default value
writeMode	The mode in which FTP Writer writes files. Valid values: • truncate: deletes all existing files with the specified file name prefix in the destination directory before files are written to the directory. • 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.	Yes	No default value
fieldDelimiter	The column delimiter that is used in the files that you want to write to the FTP server. The delimiter must be a single character.	Yes	No default value
compress	The compression format of the files that you want to write to the FTP server. The GZIP and BZIP2 compression formats are supported.	No	No compression
encoding	The encoding format of the files that you want to write.	No	utf-8
nullFormat	The string that represents a null pointer. No standard strings can represent a null pointer 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	No default value
dateFormat	The format in which the data of the DATE type is serialized in a file, such as "dateFormat":"yyyy-MM-dd".	No	No default value
fileFormat	The format in which the files are written to the FTP server. 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 delimiters, the column delimiters are escaped by double quotation marks ("). If a file is written as a text file, the data in the file is separated by the column delimiters. In this case, the column delimiters are not escaped.	No	text
header	The table header if the files are written as text files, such as ['id', 'name', 'age'].	No	No default value
markDoneFileName	The name of the file that is used to indicate that the synchronization node is successfully run, such as `xxx.ok` and `xxx.done`. Data Integration generates the file after data synchronization. You must set this parameter to the full path name of a file.	No	No default value

Parameter	Description	Required	Default value
singleFileOutput	Specifies whether to append a random suffix to the name prefix of the files that you want to write to the FTP server. The name prefix is specified by the fileName parameter. By default, a random suffix is appended to the specified name prefix to form the actual file name that is used by each thread. If you do not want to append a random suffix, set the singleFileOutput parameter to *true*.	No	No default value

Configure FTP Writer by using the codeless UI

1. Configure data sources.

   Log on to the DataWorks console. The **DataStudio** page appears. On the DataStudio page, move the pointer over the icon and choose **Data Integration > Batch Synchronization**. In the **Create Node** dialog box, configure the parameters to create a batch synchronization node.

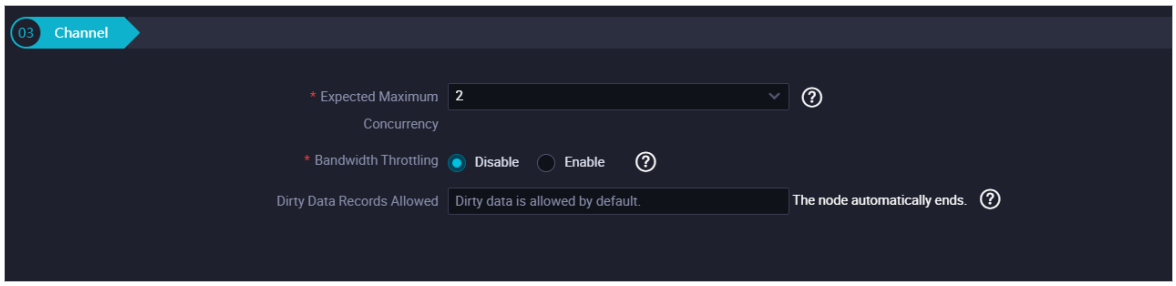
   Configure **Source** and **Target** for the synchronization node.

Parameter	Description
Connection	The name of the data source to which you want to write data. This parameter corresponds to the datasource parameter that is described in the preceding section.
File Path	The file path. This parameter corresponds to the path parameter that is described in the preceding section.
File Type	The format of the files that you want to write to the FTP server. The default format is CSV.
Field Delimiter	The column delimiter. This parameter corresponds to the fieldDelimiter parameter that is described in the preceding section. The default delimiter is commas (,).
Encoding	The encoding format. This parameter corresponds to the encoding parameter that is described in the preceding section. The default encoding format is *UTF-8*.
Null String	The string that represents a null pointer. This parameter corresponds to the nullFormat parameter that is described in the preceding section.
Time Format	The time format. This parameter corresponds to the dateFormat parameter that is described in the preceding section.
Solution to Duplicate Prefixes	This parameter corresponds to the writeMode parameter that is described in the preceding section.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section. Fields in the source on the left have a one-to-one mapping with fields in the destination on the right.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure FTP Writer by using the code editor

In the following code, a synchronization node is configured to write files to an FTP server:

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"ftp",// The writer type.
            "parameter":{
                "path":"",// The directory on the FTP server to which you want to write files.
                "fileName":"",// The name prefix of the files that you want to write to the FTP server.
                "nullFormat":"null",// The string that represents a null pointer.
                "dateFormat":"yyyy-MM-dd HH:mm:ss",// The time format.
                "datasource":"",// The name of the data source.
                "writeMode":"",// The write mode.
                "fieldDelimiter":",",// The column delimiter.
                "encoding":"",// The encoding format.
                "fileFormat":""// The format in which FTP Writer writes files.
            },
            "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. The value false indicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1,// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.1.4.5.4.6. 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

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
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

Parameter	Description	Required	Default value

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

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*
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
```

```
    "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":"_"
                    }
                ],
                "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 i
ndicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is thrott
led. 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"
            }
        ]
    }
}
```

2.1.4.5.4.7. 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*
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 ad
dresses 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.

2.1.4.5.4.8. 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

Category	Hive data type
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
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
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

Parameter	Description	Required	Default value
dataxParquetMode	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:	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@**.***.	Required if the haveKerberos parameter is set to true	None

```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"
    }
}
```

> **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@**.***"
> ```

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 throt
tled. The maximum transmission rate takes effect only if you set this parameter to true.
        }
    },
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

2.1.4.5.4.9. Configure MaxCompute Writer

This topic describes the data types and parameters supported by MaxCompute Writer and how to configure it by using the codeless user interface (UI) and code editor.

MaxCompute Writer is designed for developers to insert data to or update data in MaxCompute. MaxCompute Writer can be used to transfer data at the GB or TB level to MaxCompute.

> **Note** Before you configure MaxCompute Writer, you must configure a MaxCompute data source. For more information, see Configure a MaxCompute connection.

MaxCompute Writer writes data to MaxCompute by using Tunnel based on the specified information such as the source project, table, partition, and field.

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
Date and time	Datetime
Boolean	Boolean

Parameters

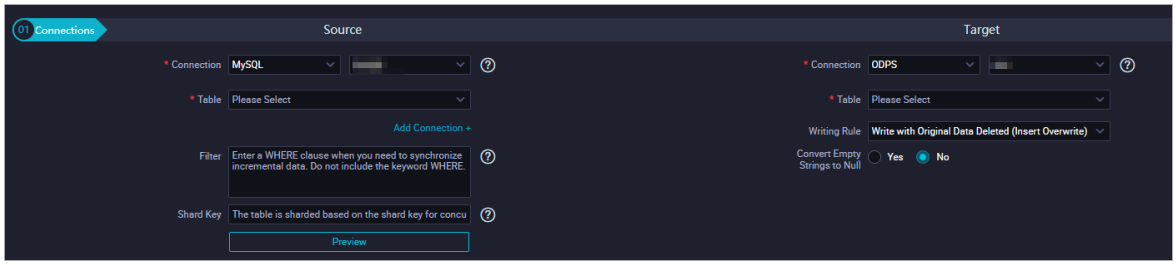
Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table to which data you want to write data. The name is not case-sensitive. You can specify only a name.	Yes	No default value
partition	The partitions to which you want to write data. 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, such as `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 data write operations 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 partitioned tables	No default value
column	The names of the columns to which you want to write data. To write data to all the columns in the destination table, set the value to `["*"]`. To write data to some columns in the destination table, set the value to the names of the specified columns. Separate the names with commas (,), such as `"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 enter `"column": ["c","b"]`. During data synchronization, column a is automatically set to null. • The column parameter must explicitly specify all the columns to which you want to write data. This parameter cannot be left empty.	Yes	No default value

Parameter	Description	Required	Default value
truncate	To ensure the idempotence of write operations, set the `truncate` parameter to true. When a failed synchronization node is rerun due to a write failure, MaxCompute Writer deletes the data that has been written before it writes the source data again. This ensures 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 parallel nodes delete data from the same table or partition. To avoid this issue, we recommend that you do not run parallel data definition language (DDL) nodes to write data to the same partition. You can create different partitions for nodes that need to be run in parallel.	Yes	No default value

Configure MaxCompute Writer by using the codeless UI

1. Configure data sources.

   Configure **Source** and **Target** for the synchronization node.



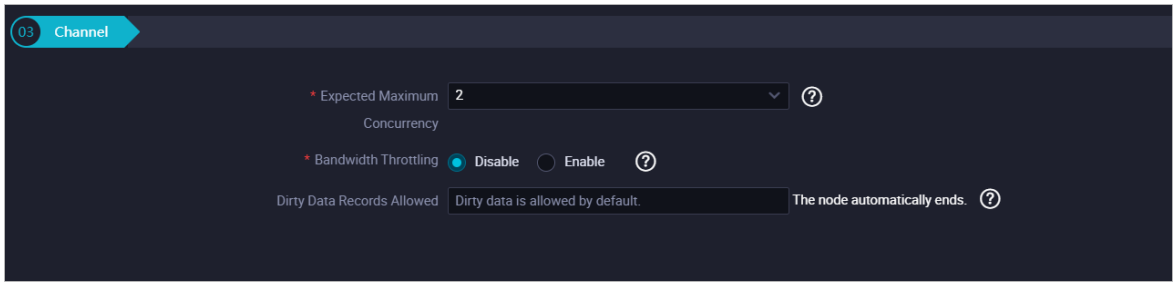
Parameter	Description
Connection	This parameter corresponds to the datasource parameter in the preceding parameter description. Select the name of a data source that you configured.
Table	This parameter corresponds to the table parameter that is described in the preceding section.
Partition Key Column	To write data to all the columns in the destination table, enter "column": ["*"]. The partition parameter allows you to use wildcards and specify one or more partitions. ○ `"partition":"pt=20140501/ds=*"` indicates that data is written to all ds partitions with pt=20140501. ○ `"partition":"pt=top?" ?` indicates that data is written to the partitions with pt=top and pt=to. You can specify the partition key columns to which you want to write data. For example, the partition key column of a MaxCompute table is pt=${bdp.system.bizdate}. In this case, you can add the pt column to the source table in the Mappings section. If the column is marked as unidentified, ignore the mark and proceed to the next step. ○ To write data to all partitions, enter pt=*. ○ To write data to specific partitions, specify the required dates.

Parameter	Description
Writing Rule	◦ **Write with Original Data Deleted (Insert Overwrite)**: All data in the table or partition is deleted before MaxCompute Writer writes data. This rule is equivalent to the `INSERT OVERWRITE` statement. ◦ **Write with Original Data Retained (Insert Into)**: No data is deleted before MaxCompute Writer writes data. 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. Synchronization 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 synchronization 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 mappings. This operation is equivalent to setting the column parameter that is described in the preceding section. Fields in the source on the left have a one-to-one mapping with fields in the destination on the right.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specified rules.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure MaxCompute Writer by using the code editor

In the following code, a synchronization node is configured to write data to MaxCompute. 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":"odps",// The writer type.
            "parameter":{
                "partition":"",// The partitions to which you want to write data.
                "truncate":true,// The write rule.
                "compress":false,// Specifies whether to enable compression.
                "datasource":"odps_first",// The name of the data source.
            "column": [// The names of the columns to which you want to write data.
                "id",
                "name",
                "age",
                "sex",
                "salary",
                "interest"
                ],
                "emptyAsNull":false,// Specifies whether to convert empty strings to null.
                "table":""// The name of the table to which you want to write data.
            },
            "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. The value false indicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1,// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

Additional information

- Column filter

MaxCompute Writer allows you to perform operations that MaxCompute does not support, such as filtering columns, reordering columns, and setting empty fields to null. To write data to all the columns in the destination table, set the column parameter to `["*"]`.

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 to `["c","b"]`. The first column and the second column of the source table are written to column c and column b in the MaxCompute table. During data synchronization, column a is automatically set to null.

- Handling for column configuration errors

  To avoid losing the data of redundant columns and ensure high data reliability, MaxCompute Writer returns an error message if the number of columns that are 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 will 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, such as `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 synchronization node is rerun due to a write failure, MaxCompute Writer deletes the data that has been written before it writes the source data again. This ensures that the same data is written for each rerun. If a synchronization node is interrupted due to other exceptions, the data cannot be rolled back and the node cannot be automatically rerun. 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 in the specified partition or table is deleted before a rerun. Exercise caution when you set this parameter to true.

2.1.4.5.4.10. 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

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

Parameter	Description	Required	Default value
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 i
ndicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is thrott
led. 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"
            }
        ]
    }
}
```

2.1.4.5.4.11. 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
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

Parameter	Description	Required	Default value
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

Parameter	Description	Required	Default value

preSql	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"}`.
	  ○ *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

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. I
f the replacekey parameter is set to _id, set the type parameter to ObjectId. If you set the type pa
rameter to String, the 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 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.
            "mbps": "1"// The maximum transmission rate.
        }
    },
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

2.1.4.5.4.12. Configure MySQL Writer

This topic describes the data types and parameters supported by MySQL Writer and how to configure it by using the codeless user interface (UI) and code editor.

MySQL Writer writes data to tables stored in MySQL databases. Specifically, MySQL Writer connects to a remote MySQL database by using Java Database Connectivity (JDBC) and executes an `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**
> - Before you configure MySQL Writer, you must configure a MySQL data source.
> - MySQL Writer does not support MySQL 8.0 or later.

MySQL Writer can be used as a data migration tool by users such as database administrators. 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 synchronization node that uses MySQL Writer must have the required permissions 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
Boolean	BOOLEAN
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

Parameters

Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of table to which you want to write data.	Yes	No 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 the original rows are replaced by new rows, and data is written to MySQL. • *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, the original rows are deleted, and new rows are inserted. This indicates that all fields of the original rows are replaced.	No	*insert*
column	The names of the columns to which you want to write data in the destination table. Separate the names with commas (,), such as `"column":["id","name","age"]`. To write data to all the columns in the destination table, set the value to `["*"]`.	Yes	No default value
preSql	The SQL statement that you want to execute before the synchronization node is run. For example, you can set this parameter to the SQL statement that is used to delete outdated data. 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 SQL statements cannot be executed in the same transaction.	No	No default value

Parameter	Description	Required	Default value
postSql	The SQL statement that you want to execute after the synchronization node is run. For example, you can set this parameter to the SQL statement that is used to add a timestamp. 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 SQL statements cannot be executed in the same transaction.	No	No default value
batchSize	The number of data records to write at a time. Set this parameter to an appropriate value based on your business requirements. This greatly reduces the interactions between Data Integration and MySQL on the network and increases throughput. If you set this parameter to an excessively large value, an out of memory (OOM) error may occur during data synchronization.	No	*1,024*

Configure MySQL Writer by using the codeless UI

1. Configure data sources.

   Configure **Source** and **Target** for the synchronization node.

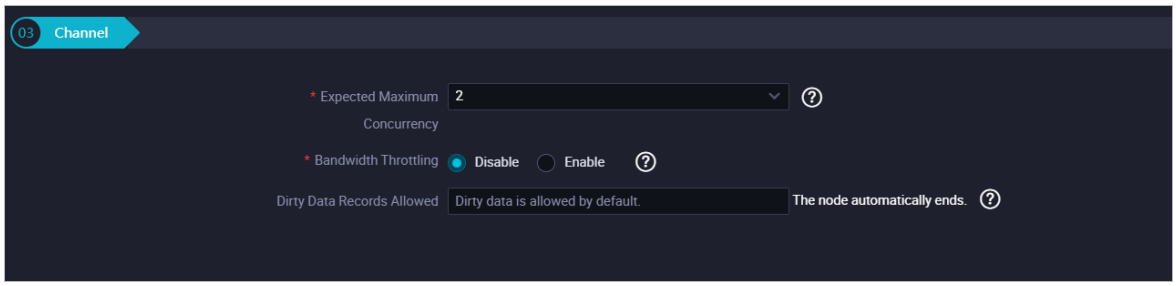
Parameter	Description
Connection	This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
Table	This parameter corresponds to the table parameter that is described in the preceding section.
Statement Run Before Writing	This parameter corresponds to the preSql parameter that is described in the preceding section. Enter the SQL statement that you want to execute before the synchronization node is run.
Statement Run After Writing	This parameter corresponds to the postSql parameter that is described in the preceding table. Enter the SQL statement that you want to execute after the synchronization node is run.
Solution to Primary Key Violation	This parameter corresponds to the writeMode parameter that is described in the preceding section. You can select the desired write mode.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section. Fields in the source on the left have a one-to-one mapping with fields in the destination on the right.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that are established.

Operation	Description
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specific rules.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure MySQL Writer by using the code editor

In the following code, a synchronization node is configured to write data to MySQL. 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.
            "parameter":{
                "postSql":[],// The SQL statement that you want to execute after the synchronization
node is run.
                "datasource":"",// The name of the data source.
                "column":[// The names of the columns to which you want to write data.
                    "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 which you want to write data.
                "preSql":[]// The SQL statement that you want to execute before the synchronization
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 number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.1.4.5.4.13. Oracle Writer

This topic describes the data types and parameters that are supported by Oracle Writer and how to configure Oracle Writer by using the codeless user interface (UI) and code editor.

Oracle Writer writes data to tables stored in primary Oracle databases. Oracle Writer connects to a remote Oracle database by using Java Database Connectivity (JDBC) and executes the `INSERT INTO` statement to write data to the Oracle database.

> **Note** Before you configure Oracle Writer, you must configure an Oracle data source. For more information, see Configure an Oracle connection.

Oracle Writer is designed for extract, transform, load (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 database administrators.

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 the data types of your database are supported.

The following table lists the data types that are 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 name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	If the table uses the default schema for the destination database, set this parameter to the name of the table to which you want to write data. If the table does not use the default schema for the destination database, specify this parameter in the Schema name.Name of the table to which you want to write data format.	Yes	No default value

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 the original rows are replaced by new rows, and data is written to Oracle. • *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, the original rows are deleted, and new rows are inserted. This indicates that all fields of the original rows are replaced.	No	*insert*
column	The names of the columns to which you want to write data in the destination table. Separate the names with commas (,), such as `"column": ["id","name","age"]`. If you want to write data to all the columns in the destination table, set this parameter to an asterisk (*), such as `"column":["*"]`.	Yes	No default value
preSql	The SQL statement that you want to execute before the synchronization node is run. For example, you can set this parameter to the SQL statement that is used to delete outdated data. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	No default value
postSql	The SQL statement that you want to execute after the synchronization node is run. For example, you can set this parameter to the SQL statement that is used to add a timestamp. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	No default value
batchSize	The number of data records to write at a time. Set this parameter to an appropriate value based on your business requirements. This greatly reduces the interactions between Data Integration and Oracle and increases throughput. If you set this parameter to an excessively large value, an out of memory (OOM) error may occur during data synchronization.	No	*1,024*

Parameter	Description	Required	Default value

driverVersion	The version of the driver that can be used to access the database. Valid values: ojdbc5, ojdbc6, ojdbc7, ojdbc8, and ojdbc14.
	If you want to use a driver of a specific version that is compatible with the database, add the following configurations by using the code editor:
	`"driverVersion": "ojdbc5" // Driver version of ojdbc5-11.2.0.3.jar`
	`"driverVersion": "ojdbc6" // Driver version of ojdbc6-12.1.1.jar`
	`"driverVersion": "ojdbc7" // Driver version of ojdbc7-12.1.0.2.jar`
	`"driverVersion": "ojdbc8" // Driver version of ojdbc8-12.2.0.1.jar`
	`"driverVersion": "ojdbc14" // Driver version of ojdbc14-10.2.0.3.0.jar`
		No	No default value

Configure Oracle Writer by using the codeless UI

1. Configure data sources.

   Log on to the DataWorks console. The **DataStudio** page appears. On the DataStudio page, move the pointer over the icon and choose **Data Integration > Batch Synchronization**. In the **Create Node** dialog box, configure the parameters to create a batch synchronization node.

   Configure **Source** and **Target** for the synchronization node.

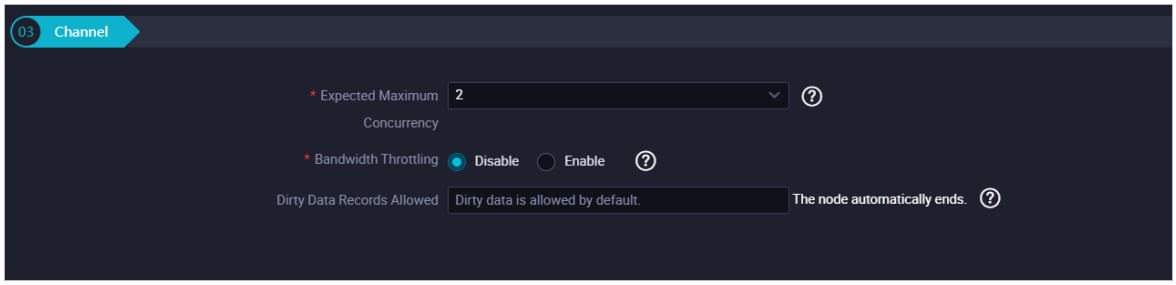
Parameter	Description
Connection	The name of the data source to which you want to write data. This parameter corresponds to the datasource parameter that is described in the preceding section.
Table	The name of the table to which you want to write data. This parameter corresponds to the table parameter that is described in the preceding section.
Statement Run Before Writing	The SQL statement that you want to execute before the synchronization node is run. This parameter corresponds to the preSql parameter that is described in the preceding section.
Statement Run After Writing	The SQL statement that you want to execute after the synchronization node is run. This parameter corresponds to the postSql parameter that is described in the preceding section.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in

the preceding section. Fields in the source on the left have a one-to-one mapping with fields in the destination on the right.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specific rules.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure Oracle Writer by using the code editor

In the following code, a synchronization node is configured to write data to Oracle:

```
{
    "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 that you want to execute after the synchronization node is run.
                "datasource":"",
                "session":[],// The settings of the session to the database.
                "column":[// The names of the columns to which you want to write data.
                    "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 table to which you want to write data.
                "preSql":[]// The SQL statement that you want to execute before the synchronization 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. The value false indicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1 // The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.1.4.5.4.14. Configure OSS Writer

This topic describes the data types and parameters supported by OSS Writer and how to configure it by using the codeless user interface (UI) and code editor.

OSS Writer writes one or more CSV-like files to Object Storage Service (OSS).

> **Note** Before you configure OSS Writer, you must configure an OSS data source.

OSS Writer can write files that store logical two-dimensional tables, such as CSV files that store text data, to OSS.

OSS Writer converts the data obtained from a Data Integration reader to files and write the files to OSS. OSS stores only unstructured data. OSS Writer supports the following features:

- Writes only files that store text data. The text data must be logical two-dimensional tables.
- Writes data stored in CSV-like files with custom delimiters.
- Uses parallel threads to write multiple files. Each thread is used to write one file.
- Supports object rotation. OSS Writer can write the excess data in a file to another object when the size of the file or the number of rows in the file exceeds a specific value.

OSS Writer does not support the following features:

- Uses parallel 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 OSS.

The following table lists the data types supported by OSS Writer.

Category	OSS data type
Integer	Long
Floating point	Double
String	String
Boolean	Bool
Date and time	Date

Parameters

Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
object	The prefix for the names of the files that you want to write to OSS. OSS simulates the directory effect by adding delimiters to object names. You can set the object parameter based on the following rules: • `"object": "datax"`: The names of the objects start with datax, which is followed by a random string as the suffix. • `"object": "cdo/datax"`: The names of the objects start with `/cdo/datax`, which is followed by a random string as the suffix. OSS uses forward slashes (/) in object names 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 `"writeSingleObject": "true"`. For more information, see writeSingleObject.	Yes	No default value

Parameter	Description	Required	Default value
writeMode	The mode in which OSS Writer writes the files. Valid values: • *truncate*: OSS Writer deletes all existing objects with the specified object name prefix before files are written to OSS. For example, you set the object parameter to `abc`. In this case, OSS Writer deletes all the objects whose names start with abc. • *append*: OSS Writer writes all files and ensures that the actual file names do not conflict with the names of existing objects by suffixing the file names with random UUIDs. For example, you set the object parameter to DI. In this case, the actual names of the files written to OSS are in the following format: DI_****_****_****. • *nonConflict*: OSS Writer returns an error message if an object whose name contains the specified prefix exists. For example, you set the `object` parameter to abc, and the object named abc123 exists. In this case, OSS Writer returns an error message.	Yes	No default value
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 column delimiters, the column delimiters are escaped by using double quotation marks ("). • If a file is written as a text file, the data in the file is separated by column delimiters. In this case, the column delimiters are not escaped.	No	*text*
fieldDelimiter	The column delimiter that is used in the files that you want to write to OSS.	No	,
encoding	The encoding format of the files that you want to write to OSS.	No	*utf-8*
nullFormat	The string that represents a null pointer. No standard strings can represent a null pointer in TXT files. Therefore, Data Integration provides the nullFormat parameter to define which string represents a null pointer. For example, you set `nullFormat="null"`. In this case, Data Integration considers null as a null pointer.	No	No default value
header (advanced parameter, available only in the code editor)	The table header in the files to write to OSS, such as ['id','name','age'].	No	No default value
maxFileSize (advanced parameter, available only in the code editor)	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 each part is 10 MB. This size is the minimum granularity for file rotation. This indicates that if you set the maxFileSize parameter to a value that is less than 10 MB, the minimum size of a file is 10 MB. You can call the InitiateMultipartUploadRequest operation to write a maximum of 10,000 parts at a time. 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

Parameter	Description	Required	Default value
suffix (advanced parameter, available only in the code editor)	The file name extension of the files that you want to write to OSS. For example, you set the suffix parameter to .csv. In this case, the final name of a file written to OSS is in the fileName****.csv format.	No	No default value

Configure OSS Writer by using the codeless UI

1. Configure data sources.

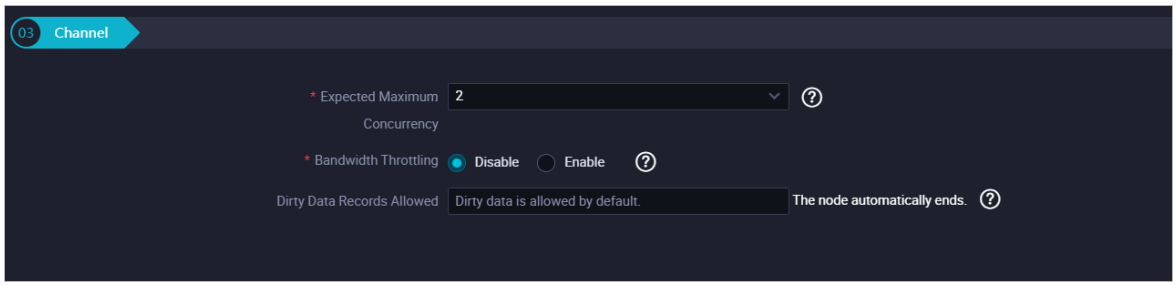
   Configure **Source** and **Target** for the synchronization node.

Parameter	Description
Connection	This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
Object Name Prefix	This parameter corresponds to the object parameter that is described in the preceding section. Enter the path of the directory that stores the files. Do not include the name of the OSS bucket in the path.
File Type	This parameter corresponds to the fileFormat parameter that is described in the preceding section. Valid values: *csv* and *text*.
Field Delimiter	This parameter corresponds to the fieldDelimiter parameter that is described in the preceding section. The default delimiter is commas (,).
Encoding	This parameter corresponds to the encoding parameter that is described in the preceding section. The default encoding format is *UTF-8*.
Null String	This parameter corresponds to the nullFormat parameter that is described in the preceding section. Enter a string that represents a null pointer. If the source contains the string, the string is replaced with a null pointer.
Time format	The format in which data of the DATE type is serialized in an object, such as `"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, the system can replace the object with the new object, insert the new object, or return an error message.

2. Configure field mappings. Fields in the source on the left have a one-to-one mapping with fields in the destination on the right.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that are established.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure OSS Writer by using the code editor

In the following code, a synchronization node is configured to write data 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 a null pointer.
                "dateFormat":"",// The format in which data of the DATE type is serialized in an obj
ect.
                "datasource":"",// The name of the data source.
                "writeMode":"",// The write mode.
                "encoding":"",// The encoding format.
                "fieldDelimiter":","// The column delimiter.
                "fileFormat":"",// The format in which files are written to OSS.
                "object":""// The prefix for the names of the files that you want to write to OSS.
            },
            "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. The value false in
dicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttlin
g is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.1.4.5.4.15. Configure PostgreSQL Writer

This topic describes the data types and parameters supported by PostgreSQL Writer and how to configure it by using the codeless user interface (UI) and code editor.

PostgreSQL Writer can write data to PostgreSQL. PostgreSQL Writer connects to a remote PostgreSQL database by using Java Database Connectivity (JDBC), and executes an SQL statement to write data to the PostgreSQL database.

> **Note** Before you configure PostgreSQL Writer, you must configure a PostgreSQL data source.

PostgreSQL Writer connects to a remote PostgreSQL database by using JDBC, generates a SELECT statement based on your configurations, and then executes the SQL statement to write data to the PostgreSQL database. Then, PostgreSQL Writer assembles the returned result into abstract datasets of CDP custom data types and sends the datasets to a 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
STRING	VARCHAR, CHAR, TEXT, BIT, and INET
DATE	DATE, TIME, and TIMESTAMP
BOOLEAN	BOOLEAN
BYTES	BYTEA

> **Note**
> - PostgreSQL Writer supports only the data types that are listed in the preceding table.
> - You can convert the MONEY, INET, and BIT data types by using syntax such as `a_inet::varchar`.

Parameters

Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table to which you want to write data.	Yes	No default value

Parameter	Description	Required	Default value
writeMode	The write mode. Valid values: • *insert*: executes the `INSERT INTO...VALUES...` statement to write data to PostgreSQL. If a primary key conflict or unique index conflict occurs, data cannot be written to PostgreSQL and is regarded as dirty data. We recommend that you us 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 use this mode when performance issues occur.	No	*insert*
column	The names of the columns to which you want to write data in the destination table. Separate the names with commas (,), such as `"column":["id","name","age"]`. To write data to all the columns in the destination table, set the value to `["*"]`.	Yes	No default value
preSql	The SQL statement that you want to execute before the synchronization node is run. For example, you can set this parameter to the SQL statement that is used to delete outdated data. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	No default value
postSql	The SQL statement that you want to execute after the synchronization node is run. For example, you can set this parameter to the SQL statement that is used to add a timestamp. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	No default value
batchSize	The number of data records to write at a time. Set this parameter to an appropriate value based on your business requirements. This greatly reduces the interactions between Data Integration and PostgreSQL on the network and increases throughput. If you set this parameter to an excessively large value, an out of memory (OOM) error may occur during data synchronization.	No	*1,024*

Parameter	Description	Required	Default value
pgType	The PostgreSQL configuration for converting data types. Valid values: bigint[], double[], text[], jsonb, and json. The following code provides a configuration example: `{ "job": { "content": [{ "reader": {...}, "writer": { "parameter": { "column": [// The names of the columns to which you want to write data in the destination table. "bigint_arr", "double_arr", "text_arr", "jsonb_obj", "json_obj"], "pgType": { // The configuration that is specific for PostgreSQL to convert 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	No default value

Configure PostgreSQL Writer by using the codeless UI

1. Configure data sources.

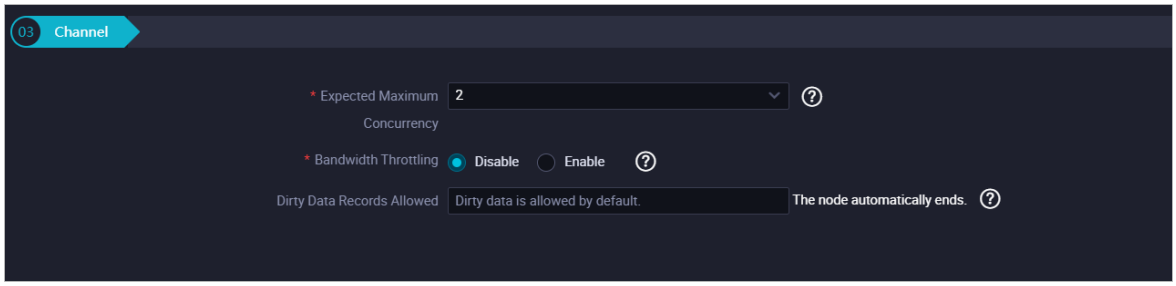
   Configure **Source** and **Target** for the synchronization node.

Parameter	Description
Connection	This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
Table	This parameter corresponds to the table parameter that is described in the preceding section.
Statement Run Before Writing	This parameter corresponds to the preSql parameter that is described in the preceding section. Enter the SQL statement that you want to execute before the synchronization node is run.
Statement Run After Writing	This parameter corresponds to the postSql parameter that is described in the preceding section. Enter the SQL statement that you want to execute after the synchronization node is run.
Write Method	This parameter corresponds to the writeMode parameter that is described in the preceding section. Valid values: *insert* and *copy*.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section. Fields in the source on the left have a one-to-one mapping with fields in the destination on the right.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specified rules.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure PostgreSQL Writer by using the code editor

In the following code, a synchronization node is configured to write data to PostgreSQL. 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 that you want to execute after the synchronization
node is run.
                "datasource":"// The name of the data source.
                    "col1",
                    "col2"
                ],
                "table":"",// The name of table to which you want to write data.
                "preSql":[]// The SQL statement that you want to execute before the synchronization
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. The value false ind
icates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling
is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.1.4.5.4.16. 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
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*

Parameter	Description	Required	Default value
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
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.
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*.

Type	Parameter	Description	Required
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*.
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 writ
ten 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 i
ndicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is thrott
led. 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"
            }
        ]
    }
}
```

2.1.4.5.4.17. 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 user interface (UI) and code editor.

SQL Server Writer writes data to tables stored in primary SQL Server databases. Specifically, SQL Server Writer connects to a remote SQL Server database by using Java Database Connectivity (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 extract, transform, load (ETL) developers to import data from data warehouses to SQL Server. SQL Server Writer can also be used as a data migration tool by users such as database administrators.

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 performs batch updates by using the `PreparedStatement` method and the setting of `rewriteBatchedStatements=true`. 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 to tables stored only in primary SQL Server databases.
> - A synchronization node that uses SQL Server Writer must have at least the permissions to execute the INSERT INTO statement. Whether other permissions are required depends on the SQL statements that you specify 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 name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table to which you want to write data.	Yes	No default value
column	The names of the columns to which you want to write data in the destination table. Separate the names with commas (,), such as `"column":["id","name","age"]`. To write data to all the columns in the destination table, set the value to `["*"]`.	Yes	No default value
preSql	The SQL statement that you want to execute before the synchronization node is run. For example, you can set this parameter to the SQL statement that is used to delete outdated data. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	No default value

Parameter	Description	Required	Default value
postSql	The SQL statement that you want to execute after the synchronization node is run. For example, you can set this parameter to the SQL statement that is used to add a timestamp. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	No default value
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*
batchSize	The number of data records to write at a time. Set this parameter to an appropriate value based on your business requirements. This greatly reduces the interactions between Data Integration and SQL Server on the network and increases throughput. If you set this parameter to an excessively large value, an out of memory (OOM) error may occur during data synchronization.	No	*1,024*

Configure SQL Server Writer by using the codeless UI

1. Configure data sources.

   Configure **Source** and **Target** for the synchronization node.

Parameter	Description
Connection	This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
Table	This parameter corresponds to the table parameter that is described in the preceding section.
Statement Run Before Writing	This parameter corresponds to the preSql parameter that is described in the preceding section. Enter the SQL statement that you want to execute before the synchronization node is run.
Statement Run After Writing	This parameter corresponds to the postSql parameter that is described in the preceding section. Enter the SQL statement that you want to execute after the synchronization node is run.
Solution to Primary Key Violation	This parameter corresponds to the writeMode parameter that is described in the preceding section. You can select the desired write mode.

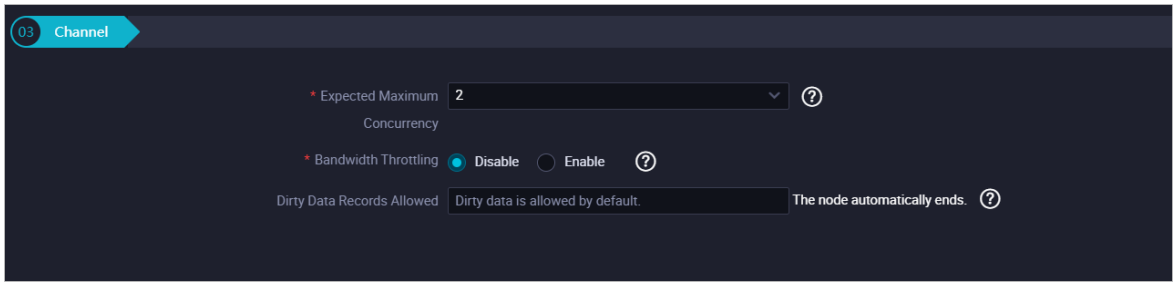
2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination 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.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that are established.

Operation	Description
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit the 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 as Unidentified.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure SQL Server Writer by using the code editor

In the following code, a synchronization node is configured to write data to SQL Server. 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":"sqlserver",// The writer type.
            "parameter":{
                "postSql":[],// The SQL statement that you want to execute after the synchronization node is run.
                "datasource":"",// The name of the data source.
                "column":[// The names of the columns to which you want to write data.
                    "id",
                    "name"
                ],
                "table":"",// The name of the table to which you want to write data.
                "preSql":[]// The SQL statement that you want to execute before the synchronization 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. The value false indicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttling is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1,// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.1.4.5.4.18. 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*
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*

Parameter	Description	Required	Default value
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
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:
	`{ "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:
		Yes	None

Parameter	Description	Required	Default value

	<pre>{
	"name": "col_geo_shape",
	"type": "geo_shape",
	"tree": "quadtree",
	"precision": "10m"
	}</pre>
	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:
	<pre>{
	"name": "col_integer_array",
	"type": "integer",
	"array": true
	}</pre>		

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*
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.

2.1.4.5.4.19. 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

Parameter	Description	Required	Default value
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 throt
tled. The maximum transmission rate takes effect only if you set this parameter to true.
        }
    },
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

2.1.4.5.4.20. 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 into or update data in Open Search. Open Search Writer is designed 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 in strict 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.

For example, 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 that is 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, 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 overwritten based on IDs. Therefore, the data written to Open Search must contain an 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.

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 of the account that is used to access Open Search.	Yes	None
accessKey	The AccessKey secret that corresponds to the AccessKey ID. The AccessKey secret is equivalent to a logon password.	Yes	None
host	The endpoint of Open Search. You can view the endpoint in the Apsara Uni-manager Management 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 allow you to import 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 this parameter to an asterisk (*), such as `"column":["*"]`. If you want to write data only to specific columns in the destination table, set this parameter to the columns. Separate the columns with commas (,), such as `"column":["id","name"]`. Open Search Writer can filter columns and change the order of columns. For example, 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 of `"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. Multiple data records are written to Open Search at a time. 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 access 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*
writeMode	The write mode. To ensure the idempotence of write operations, set this parameter to add/update. • 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 multiple data records to Open Search at a time 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

Parameter	Description	Required	Default value
ignoreWriteError	Specifies whether to ignore failed write operations. Example: `"ignoreWriteError":true`. If multiple data records are written to Open Search at a time, this parameter specifies whether to ignore failed write operations in the current batch. If you set this parameter to true, Open Search Writer continues to perform other write operations. If you set this parameter to false, the synchronization node ends, and an error message is returned. We recommend that you use the default value.	No	*false*
version	The version of Open Search, such as `"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 synchronization 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
            }
        }
    }
}
```

2.1.4.5.4.21. Tablestore Writer

This topic describes the data types and parameters that are supported by Tablestore Writer and how to configure Tablestore Writer by using the codeless user interface (UI) and code editor.

Tablestore is a NoSQL database service that is built on the Apsara distributed operating system. The service allows you to store and access large volumes of structured data in real time. Tablestore organizes data into instances and tables. It uses data sharding and load balancing technologies to seamlessly expand the data scale.

Tablestore Writer connects to the Tablestore server by using Tablestore SDK for Java and writes data to the server by using the SDK. Tablestore Writer greatly optimizes the write process, including retry after write timeouts, retry after exceptions, and batch submission.

Tablestore Writer writes data to Tablestore in one of the following modes:

- PutRow: the PutRow API operation for Tablestore, which is used to insert data to a specific row. If the specified row does not exist, a new row is added. Otherwise, the original row is overwritten.
- UpdateRow: the UpdateRow API operation for Tablestore, which is used to update the data of a specific row. If the specified row does not exist, a new row is added. Otherwise, the values of the specified columns are added, modified, or removed as requested.

Tablestore Writer supports all Tablestore data types. The following table lists the data types that are supported by Tablestore Writer.

Category	Tablestore data type
Integer	INTEGER
Floating point	DOUBLE
String	STRING
Boolean	BOOLEAN
Binary	BINARY

Note To write INTEGER-type data, set the data type to INT in the code editor. Then, DataWorks converts the INT type into the INTEGER type. If you directly set the data type to INTEGER, an error is reported in the log, and the node fails.

Parameters

Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
endPoint	The endpoint of the Tablestore server.	Yes	No default value
accessId	The AccessKey ID of the account that is used to access Tablestore.	Yes	No default value
accessKey	The AccessKey secret of the account that is used to access Tablestore.	Yes	No default value

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 Tablestore console before you can create and manage tables. Instances are the basic units that you can use to manage Tablestore resources. Access control and resource measurement for applications are implemented at the instance level.	Yes	No default value
table	The name of the table to which you want to write data. You can specify only one table. Multi-table synchronization is not required for Tablestore.	Yes	No default value
primaryKey	The primary key of the destination table. Specify the primary keys in a JSON array. Tablestore is a NoSQL database service. You must specify the primary key of the destination table for Tablestore Writer to write data. **Note** The primary keys in Tablestore must be of the STRING or INT type. Therefore, you must set the data type of a primary key to STRING or INT in the code editor. Data Integration supports data type conversion. Tablestore Writer can convert data that is not of the STRING or INT type to the STRING or INT type. The following code provides a configuration example: `"primaryKey" : [ {"name":"pk1", "type":"string"}, null ],`	Yes	No default value
column	The names of the columns to which you want to write data. Specify the columns in a JSON array. Specify this parameter in the following 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 Tablestore include STRING, INT, DOUBLE, BOOLEAN, and BINARY.	Yes	No default value

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: • PutRow: the PutRow API operation for Tablestore, which is used to insert data to a specific row. If the specified row does not exist, a new row is added. Otherwise, the original row is overwritten. • UpdateRow: the UpdateRow API operation for Tablestore, which is used to update the data of a specific row. If the specified row does not exist, a new row is added. Otherwise, the values of the specified columns are added, modified, or removed as requested. • DeleteRow: deletes a row.	Yes	No default value

Configure Tablestore Writer by using the codeless UI

This method is not supported.

Configure Tablestore Writer by using the code editor

In the following code, a synchronization node is configured to write data to Tablestore:

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"ots",// The writer type.
            "parameter":{
                "datasource":"",// The name of the data source.
                "column":[// The names of the columns to which you want to write data.
                    {
                        "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 table to which you want to write data.
                "primaryKey":[// The primary key of the destination table.
                    {
                        "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. The value false in
dicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttlin
g is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1,// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.1.4.5.4.22. 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 selec
t sql, retrieve data from the ResultSet. warn: The more you know about the database, the less pr
oblems 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
```

2.1.4.5.4.23. 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 i
ndicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is thrott
led. 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"
            }
        ]
    }
}
```

2.1.4.5.4.24. Hive Writer

Hive Writer writes data to the Hadoop distributed file system (HDFS) and loads the data to Hive. This topic describes how Hive Writer works, the parameters that are supported by Hive Writer, and how to configure Hive Writer by using the codeless user interface (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 connects to a Hive metastore and obtains the storage path, format, and column delimiter of the HDFS file to which you want to write data. Then, Hive Writer writes data to the HDFS file and loads data in the HDFS file to the destination Hive table by using Java Database Connectivity (JDBC).

The underlying logic of Hive Writer is the same as that of HDFS Writer. You can configure parameters for 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 name of the data source. It must be the same as the name of the added data source.	Yes	No default value
column	The names of the columns to which you want to write data, such as `"column": ["id", "name"]`. • You can select specific columns to write. • The column parameter must explicitly specify all the columns to which you want to write data. The parameter cannot be left empty. • The column order cannot be changed.	Yes	No default value
table	The name of the Hive table to which you want to write data. **Note** The name is case-sensitive.	Yes	No default value
partition	The partition in the Hive table. This parameter is required for partitioned Hive tables. After you specify this parameter, Hive Writer writes data to the partition that is specified by this parameter.	No	No default value

Parameter	Description	Required	Default value
writeMode	The mode in which Hive Writer loads data to the Hive table. After data is written to an 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 mode in which Hive Writer loads data from the HDFS file to the Hive table. Valid values: • *truncate*: Hive Writer deletes existing data before it loads data to the Hive table. • *append*: Hive Writer retains the existing data and appends data to the Hive table. • Other: Hive Writer writes data to the HDFS file but does not load the data 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 prevent data from being incorrectly deleted. This parameter must be used together with the hiveConfig parameter.	Yes	No default value

Parameter	Description	Required	Default value

parquet schema	The schema of the destination files. This parameter is available only when the FileType parameter is set to *Parquet*.
	
	The parquet schema parameter is specified in the following format:
	
	`message MessageTypeName {`
	`required, dataType, columnName;`
	`.....;`
	`}`
	
	• MessageTypeName: the name of the MessageType object. You can customize the name.
	• dataType: the data type. Valid values: BOOLEAN, INT32, INT64, INT96, FLOAT, DOUBLE, BINARY, and FIXED_LEN_BYTE_ARRAY.
	
	Note Each line, including the last line, 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;`
	`}`
		Specifies whether the desired columns are required in the parquet schema parameter. Valid values:
		• required
		• optional
		
		Note We recommend that you use optional for the desired columns.
			No default value

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, execute the `LOAD DATA INPATH` statement to load data based on the mode that is specified by the writeMode parameter. The client that is 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 that is specified by the writeMode parameter. `"hiveConfig": { "hiveCommand": "", "jdbcUrl": "", "username": "", "password": "" }` • Hive Writer uses an HDFS client to write data to HDFS files. You can use the hiveConfig parameter to specify advanced settings for the HDFS client.	Yes	No default value

Configure Hive Writer by using the codeless UI

Log on to the DataWorks console. The **DataStudio** page appears. On the **DataStudio** page, double-click the desired synchronization node and configure the node.

Perform the following steps on the configuration tab of the node:

1. Configure data sources.

   Configure **Source** and **Target** for the synchronization node.

Parameter	Description
Connection	The name of the data source to which you want to write data. This parameter corresponds to the datasource parameter that is described in the preceding section.
Table	The name of the table to which you want to write data. This parameter corresponds to the table parameter that is described in the preceding section.
Partition info	The partition to which you want to write data. You must specify the last-level partition. Hive Writer can write data only to a single partition.
Write mode	The write mode. This parameter corresponds to the writeMode parameter that is described in the preceding section.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section. Fields in the source on the left have a one-to-one mapping with fields in the destination on the right.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specific rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure Hive Writer by using the code editor

In the following code, a synchronization node is configured to write data to Hive in the 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 you want to write dat
a.
                "datasource": "hive_ha_shanghai", // The name of the data source.
                "table": "partitiontable2", // The name of the table to which you want to write data
.
                "column": [ // The names of the columns to which you want to write data.
                    "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
        }
    }
}
```

2.1.4.5.4.25. 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

Parameter	Description	Required	Default value
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
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 i
ndicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is thrott
led. 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"
            }
        ]
    }
}
```

2.1.4.5.4.26. 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

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. 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
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"
            }
        ]
    }
}
```

2.1.4.5.4.27. KingbaseES Writer

This topic describes the parameters that are supported by KingbaseES Writer and how to configure KingbaseES Writer by using the codeless user interface (UI) and code editor.

Context

KingbaseES Writer writes data to tables stored in KingbaseES databases. KingbaseES Writer connects to a remote KingbaseES database by using Java Database Connectivity (JDBC) and executes the `INSERT INTO` or `REPLACE INTO` statement to write data to the database. KingbaseES uses the InnoDB engine so that data is written to the database in batches.

KingbaseES Writer can also be used as a data migration tool by users such as database administrators. KingbaseES Writer obtains protocol data from a Data Integration reader, and writes the data to the destination database based on the value of the writeMode parameter.

> **Note** A synchronization node that uses kingbaseES Writer must have at least the permissions to execute the `INSERT INTO` or `REPLACE INTO` statement. Whether other permissions are required depends on the SQL statements that you specify in the preSql and postSql parameters when you configure the node.

Parameters

Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table to which you want to write data.	Yes	No default value
column	The names of the columns to which you want to write data. Separate the names with commas (,), such as `"column": ["id", "name", "age"]`. If you want to write data to all the columns in the destination table, set this parameter to an asterisk (*), such as `"column":["*"]`.	Yes	No default value
preSql	The SQL statement that you want to execute before the synchronization node is run. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor. For example, you can set this parameter to the following statement to delete outdated data before the synchronization node is run: `truncate table tablename` **Note** If you specify multiple SQL statements, they may not be executed in the same transaction.	No	No default value
postSql	The SQL statement that you want to execute after the synchronization node is run. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor. For example, you can set this parameter to `alter table tablenameadd colname timestamp DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP` to add a timestamp after the synchronization node is run.	No	No default value

Parameter	Description	Required	Default value
batchSize	The number of data records to write at a time. Set this parameter to an appropriate value based on your business requirements. This greatly reduces the interactions between Data Integration and the database and increases throughput. If you set this parameter to an excessively large value, an out of memory (OOM) error may occur during data synchronization.	No	*2048*

Configure KingbaseES Writer by using the codeless UI

1. Configure data sources.

   Log on to the DataWorks console. The **DataStudio** page appears. On the DataStudio page, move the pointer over the icon and choose **Data Integration > Batch Synchronization**. In the **Create Node** dialog box, configure the parameters to create a batch synchronization node.

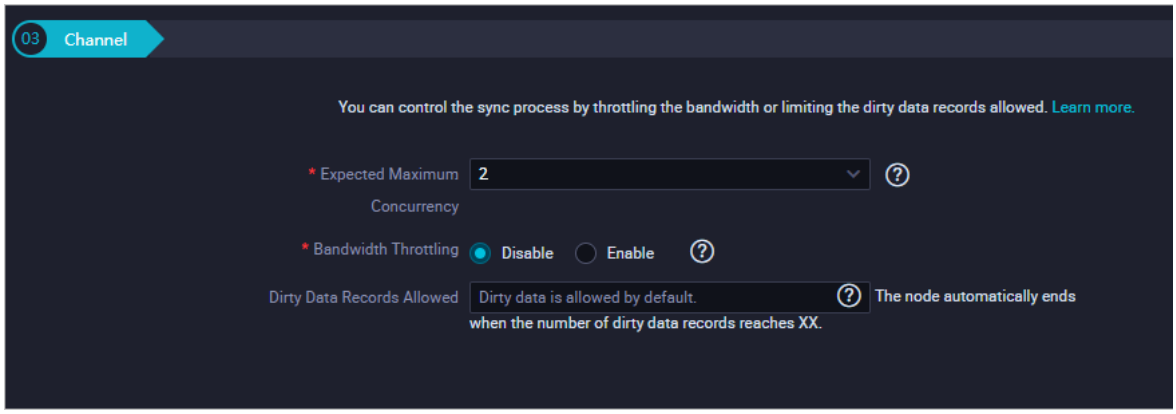
   On the configuration tab of the batch synchronization node, configure Source and **Target** for the node.

Parameter	Description
Connection	The name of the data source to which you want to write data. This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
Table	The name of the table to which you want to write data. This parameter corresponds to the table parameter that is described in the preceding section.
Pre sql	The SQL statement that you want to execute before the synchronization node is run. This parameter corresponds to the preSql parameter that is described in the preceding section. Enter the SQL statement that you want to execute before the synchronization node is run.
Post sql	The SQL statement that you want to execute after the synchronization node is run. This parameter corresponds to the postSql parameter that is described in the preceding table. Enter the SQL statement that you want to execute after the synchronization node is run.
Data Records Per Write	The number of data records to write at a time. This parameter corresponds to the batchSize parameter that is described in the preceding section. Valid values: *2048*. You can specify this parameter based on your business requirements.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section. Fields in the source on the left have a one-to-one mapping with fields in the destination on the right.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on the specified rules.
Add	Click **Add** to add a field. Take note of the following rules when you 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 that are supported by relational databases, such as now() and count(1). ○ If the value that you entered cannot be parsed, the value of Type for the field is Unidentified.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure KingbaseES Writer by using the code editor

In the following code, a synchronization node is configured to write data to KingbaseES:

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"kingbasees",// The writer type.
            "parameter":{
                "postSql":[],// The SQL statement that you want to execute after the synchronization
node is run.
                "datasource":"",// The name of the data source.
                "column":[// The names of the columns to which you want to write data.
                    "id",
                    "value"
                ],
                "batchSize":2048,// The number of data records to write at a time.
                "table":"",// The name of the table to which you want to write data.
                "preSql":[
                     "delete from XXX;"// The SQL statement that you want to execute before the sync
hronization 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 parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.1.4.5.4.28. SAP HANA Writer

This topic describes the parameters that are supported by SAP HANA Writer and how to configure SAP HANA Writer by using the codeless user interface (UI) and code editor.

Context

SAP HANA Writer writes data to tables stored in SAP HANA databases. SAP HANA Writer connects to a remote SAP HANA database by using Java Database Connectivity (JDBC) and executes the `INSERT INTO` or `REPLACE INTO` statement to write data to the SAP HANA database. SAP HANA uses the InnoDB engine so that data is written to the database in batches.

SAP HANA Writer can also be used as a data migration tool by users such as database administrators.

> **Note** A synchronization node that uses SAP HANA Writer must have at least the permissions to execute the `INSERT INTO` or `REPLACE INTO` statement. Whether other permissions are required depends on the SQL statements that you specify in the preSql and postSql parameters when you configure the node.

Parameters

Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table to which you want to write data.	Yes	No default value
column	The names of the columns to which you want to write data. Separate the names with commas (,), such as `"column": ["id", "name", "age"]`. If you want to write data to all the columns in the destination table, set this parameter to an asterisk (*), such as `"column":["*"]`.	Yes	No default value
preSql	The SQL statement that you want to execute before the synchronization node is run. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor. For example, you can set this parameter to the following SQL statement that is used to delete outdated data: `truncate table tablename` **Note** If you specify multiple SQL statements in the code editor, the SQL statements cannot be executed in the same transaction.	No	No default value
postSql	The SQL statement that you want to execute after the synchronization node is run. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor. For example, you can set this parameter to `alter table tablenameadd colname timestamp DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP` that is used to add a timestamp.	No	No default value

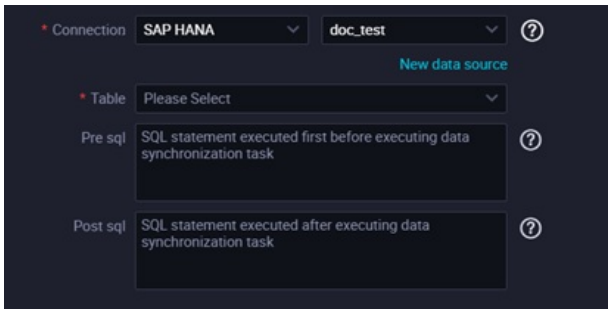
Parameter	Description	Required	Default value
batchSize	The number of data records to write at a time. Set this parameter to an appropriate value based on your business requirements. This greatly reduces the interactions between Data Integration and SAP HANA and increases throughput. If you set this parameter to an excessively large value, an out of memory (OOM) error may occur during data synchronization.	No	*2048*

Configure SAP HANA Writer by using the codeless UI

1. Configure data sources.

   Log on to the DataWorks console. The **DataStudio** page appears. On the DataStudio page, move the pointer over the icon and choose **Data Integration > Batch Synchronization**. In the **Create Node** dialog box, configure the parameters to create a batch synchronization node.

   Configure Source and **Target** for the synchronization node.

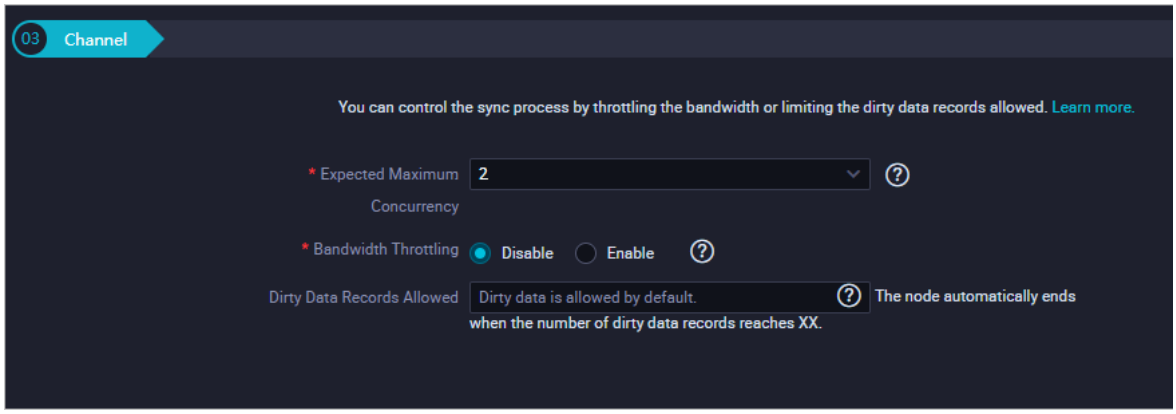
   

Parameter	Description
Connection	The name of the data source to which you want to write data. This parameter corresponds to the datasource parameter that is described in the preceding section.
Table	The name of the table to which you want to write data. This parameter corresponds to the table parameter that is described in the preceding section.
Pre sql	The SQL statement that you want to execute before the synchronization node is run. This parameter corresponds to the preSql parameter that is described in the preceding section.
Post sql	The SQL statement that you want to execute after the synchronization node is run. This parameter corresponds to the postSql parameter that is described in the preceding section.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section. Fields in the source on the left have a one-to-one mapping with fields in the destination on the right.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specific rules.
Add	Click **Add** to add a field. Take note of the following rules when you 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 that are supported by relational databases, such as now() and count(1). ○ If the field that you entered cannot be parsed, the value of Type for the field is Unidentified.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization 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 workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure SAP HANA Writer by using the code editor

In the following code, a synchronization node is configured to write data to SAP HANA:

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"saphana",// The writer type.
            "parameter":{
                "postSql":[],// The SQL statement that you want to execute after the synchronization
node is run.
                "datasource":"",// The name of the data source.
                "column":[// The names of the columns to which you want to write data.
                    "id",
                    "value"
                ],
                "batchSize":1024,// The number of data records to write at a time.
                "table":"",// The name of the table to which you want to write data.
                "preSql":[
                     "delete from XXX;" // The SQL statement that you want to execute before the syn
chronization 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. The value false in
dicates that bandwidth throttling is disabled, and the value true indicates that bandwidth throttlin
g is enabled. The mbps parameter takes effect only when the throttle parameter is set to true.
            "concurrent":1 // The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.1.4.5.5. 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 re
    ateTime<'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.

2.1.4.6. Synchronize data in real time

2.1.4.6.1. Overview

This topic describes the definition, highlights, supported data sources, and architecture of the real-time synchronization feature. This topic also describes how to use this feature.

Definition

The real-time synchronization feature allows you to synchronize the data changes in a source database to a destination database in real time. For example, after you add, modify, or delete the data in a source database, the real-time synchronization feature synchronizes these changes to a destination database in real time. If a real-time synchronization is performed during a batch synchronization for full historical data, all the data in your source database is synchronized to your destination database. The real-time synchronization feature helps ensure that the data in your destination database is updated in real time and consistent with that in your source database.

Highlights

- Diverse data sources
  - Star-shaped combination is supported. Each supported source can be combined with a supported destination to synchronize data.

  - The feature will soon allow you to synchronize data from a single source to multiple destinations at the same time.
- Comprehensive synchronization solutions
  - You can configure a synchronization solution to synchronize the full data or incremental data from a database to MaxCompute or Hologres.
  - The feature synchronizes full data first and then continuously synchronizes incremental data to the destination database based on the synchronization solution that you configure.
- Real-time data extraction

  The feature can extract data from the table shards, a single table, and multiple tables in a database, and extract the data definition language (DDL) statements of tables.
- Data processing

  The feature can filter data and replace strings during synchronization.
- Monitoring and alerting
  - The feature monitors the service latency, failovers, dirty data, heartbeats, and failures during synchronization.
  - Alert notifications can be sent by email, phone call, and DingTalk message.
- Small impact on the source

  The feature is optimized to have a small impact on the source.
- Graphical development
  - You can develop real-time synchronization nodes in a graphical user interface (GUI) without the need to write code.
  - The feature is easy to use even for beginners.

Supported data sources

- Source: MySQL Binlog, Oracle CDC, Kafka, DataHub, LogHub, and PolarDB.
- Destination: MaxCompute, Hologres, Kafka, and DataHub.
- The feature can filter data and replace strings during synchronization.

How to use the feature

You can use the real-time synchronization feature to create real-time synchronization solutions or synchronization nodes.

- For more information about how to create a real-time synchronization node, see Create a real-time synchronization node.
- For more information about how to create a synchronization solution, see Go to the Sync Solutions page.

2.1.4.6.2. Data sources supported by real-time synchronization

Real-time synchronization supports reader, writer, and conversion plug-ins. The following table lists the plug-ins.

Category	Plug-in	References
	MySQL Binlog Reader	Configure MySQL Binlog Reader
	Oracle Change Data Capture (CDC) Reader	Configure Oracle CDC Reader

Category	Plug-in	References
Reader	DataHub Reader	Configure DataHub Reader
	LogHub Reader	Configure LogHub Reader
	Kafka Reader	Configure Kafka Reader
Writer	Hologres Writer	Configure Hologres Writer
	DataHub Writer	Configure DataHub Writer
	Kafka Writer	Configure Kafka Writer
	MaxCompute Writer	Configure MaxCompute Writer
Conversion	Data filtering	Configure data filtering
	String replacement	Configure string replacement

Note A real-time synchronization node cannot be directly run on the node configuration tab. Instead, you must save and commit the node before the real-time synchronization node is run in the production environment.

Basic configuration

After you configure the reader, writer, and conversion plug-ins, you can click the **Basic configuration** panel in the right-side navigation pane to configure the synchronization node.

Parameter	Description
Description	The description of the real-time synchronization node.
JVM parameters	The Java virtual machine (JVM) memory allocated for the real-time synchronization node. If this parameter is not specified, Data Integration automatically allocates JVM memory based on your node configurations.

2.1.4.6.3. Create, configure, commit, and manage real-time synchronization nodes

DataWorks allows you to synchronize data in real time. This topic describes how to create, configure, commit, and manage real-time synchronization nodes.

Create a real-time synchronization node

1. Log on to the DataWorks console.
2. On the DataStudio page, move the pointer over the icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can find the required workflow, right-click the workflow name, and then choose **Create > Data Integration > Real-time synchronization**.
3. In the **Create Node** dialog box, configure the parameters.

Parameter	Description
Node Type	The type of the node. Default value: **Real-time synchronization**.
Sync Method	The method used to synchronize data. Valid values: ○ **End-to-end ETL**: synchronizes data in one table to one or more tables. Data type conversion is supported during the synchronization. ○ **Migration to Hologres**: synchronizes all or some tables in a database to Hologres. Destination tables can be automatically created in Hologres. ○ **Migration to MaxCompute**: synchronizes all or some tables in a database to MaxCompute. ○ **Migration to DataHub**: synchronizes all or some tables in a database to DataHub.
Node Name	The name of the node. The name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
Location	The directory in which the real-time synchronization node is stored.

4. Click **Commit**.

Configure the real-time synchronization node

The operations that you can perform on the configuration tab of the real-time synchronization node vary based on the synchronization method that you selected.

- To configure the real-time synchronization node for which **Sync Method** is set to **End-to-end ETL**, perform the following steps:
  i. Click **Basic configuration** in the right-side navigation pane. In the Basic configuration panel, select the required resource group from the **Resource Group** drop-down list.

No.	Description
1	The left-side navigation tree, which consists of the **Input**, **Output**, and **Conversion** sections.
2	The configuration canvas of the real-time synchronization node. You can drag a component from the navigation tree to the canvas and configure the component.
3	The property configuration panel of the real-time synchronization node. This panel appears after you click **Basic configuration** in the right-side navigation pane.

  ii. Drag components from the navigation tree to the canvas, and draw lines to connect the components. This way, the components synchronize data based on the connections.
  iii. Click each component on the canvas. In the panel that appears, configure the parameters.
  iv. Click the icon in the top toolbar.

- To configure the real-time synchronization node for which **Sync Method** is set to **Migration to Hologres**, perform the following steps:
  i. Click **Basic configuration** in the right-side navigation pane. In the Basic configuration panel, select the required resource group from the **Resource Group** drop-down list.

> **Notice** You must select a **resource group** before you commit the node. Otherwise, the system returns an error when you commit the node.

ii. In the **Data source** section, specify **Type** and **Data source**.

iii. In the **Select the source table for synchronization** section, select the tables that you want to synchronize in the **SOURCE Table** list and click the > icon to move the tables to the **Selected Source table** list.

The SOURCE Table list displays all the tables in the source. You can select all or some tables to synchronize them at a time.

> **Notice** If a selected table does not have a primary key, the table cannot be synchronized in real time.

iv. (Optional)In the **Set synchronization rules** section, click **Add rule** and select an option to configure naming rules for destination tables.

Supported options include **Table name conversion rules** and **Target table name rule**.

- **Table name conversion rules**: the rule for converting the names of source tables to those of destination tables.
- **Target table name rule**: the rule for adding a prefix and suffix to the converted names of destination tables.

v. Click **Next Step**.

vi. In the **Set target table** step, specify **Target Hologres data source** and **Schema**.

vii. Click **Reload source table and Hologres Table mapping** to configure the mappings between the source tables and destination Hologres tables.

viii. View the mapping progress, source tables, and mapped destination tables, and click **Next Step**.

You can view the mapping progress between the source tables and destination tables. The mapping may take a long period of time if you want to synchronize a large number of tables.

An error message appears if the selected source table does not have a primary key. The synchronization can be performed if one of the selected source tables has a primary key. Source tables without primary keys are ignored during the synchronization.

You can set Table creation method to **Create tables automatically** or **Use existing Table**. The name of the destination table that appears in the **Hologres Table name** column varies based on the setting of Table creation method.

- If you set **Table creation method** to **Create tables automatically**, the name of the destination table that is automatically created appears. You can click the table name to view and modify the table creation statements.
- If you set Table creation method to **Use existing Table**, you must select a table from the drop-down list in the **Hologres Table name** column.

ix. In the **Run resource settings** step, specify **Maximum number of connections supported by source read** and **Number of concurrent writes on the target side**. Then, click the icon in the top toolbar.

- To configure the real-time synchronization node for which **Sync Method** is set to **Migration to MaxCompute**, perform the following steps:

i. Click **Basic configuration** in the right-side navigation pane. In the Basic configuration panel, select the required resource group from the **Resource Group** drop-down list.

ii. In the **Data source** section, specify **Type** and **Data source**.

iii. In the **Select the source table for synchronization** section, select the tables that you want to synchronize in the **SOURCE Table** list and click the > icon to move the tables to the **Selected Source table** list.

The SOURCE Table list displays all the tables in the source. You can select all or some tables to synchronize them at a time.

> **Notice** If a selected table does not have a primary key, the table cannot be synchronized in real time.

iv. In the **Set synchronization rules** section, click **Add rule** and select an option to configure naming rules for destination tables.

Supported options include **Table name conversion rules** and **Target table name rule**.

- **Table name conversion rules**: the rule for converting the names of source tables to those of destination tables.
- **Target table name rule**: the rule for adding a prefix and suffix to the converted names of destination tables.

v. Click **Next Step**.

vi. In the **Set target table** step, select a data source from the **Target MaxCompute data source** drop-down list and click the icon next to **MaxCompute time automatic partition settings**. In the **Edit** dialog box, set the partition interval of tables in MaxCompute to day or hour.

vii. Click **Reload source table and MaxCompute Table mapping** to configure the mappings between the source tables and destination MaxCompute tables.

viii. View the mapping progress, source tables, and destination tables. Then, click **Next Step**.

You can view the mapping progress between the source tables and destination tables. The mapping may take a long period of time if you want to synchronize a large number of tables.

An error message appears if the selected source table does not have a primary key. The synchronization can be performed if one of the selected source tables has a primary key. Source tables without primary keys are ignored during the synchronization.

You can set Table creation method to **Create tables automatically** or **Use existing Table**. The name of the destination table that appears in the **MaxCompute Table name** column varies based on the setting of Table creation method.

- If you set **Table creation method** to **Create tables automatically**, the name of the destination table that is automatically created appears. You can click the table name to view and modify the table creation statements.
- If you set **Table creation method** to **Use existing Table**, you must select a table name from the drop-down list in the MaxCompute Table name column.

ix. In the **Run resource settings** step, specify **Maximum number of connections supported by source read** and **Number of concurrent writes on the target side**. Then, click the icon in the top toolbar.

- To configure the real-time synchronization node for which **Sync Method** is set to **Migration to DataHub**, perform the following steps:

i. On the node configuration tab that appears, click **Basic configuration** in the right-side navigation pane. In the Basic configuration panel, select the required resource group from the **Resource Group** drop-down list.

ii. In the **Data source** section, specify **Type** and **Data source**.

iii. In the **Select the source table for synchronization** section, select the tables that you want to synchronize in the **SOURCE Table** list and click the icon to move the tables to the **Selected Source table** list.

The SOURCE Table list displays all the tables in the source. You can select all or some tables to synchronize them at a time.

> **Notice** If a selected table does not have a primary key, the table cannot be synchronized in real time.

iv. In the **Set synchronization rules** section, click **Add rule** and then select an option to configure naming rules for destination DataHub topics.

Supported options include **SOURCE table name and Topic conversion rules** and **Target Topic rules**.

v. Click **Next Step**.

vi. In the **Set target table** step, select a data source from the **Target DataHub data source** drop-down list and then click **Reload source table and DataHub Topic mapping** to configure the mappings between the source tables and destination DataHub topics.

vii. View the mapping progress, source tables, and destination topics. Then, click **Next Step**.

You can view the mapping progress between the source tables and destination topics. The mapping may take a long period of time if you want to synchronize a large number of tables.

You can set Topic creation method to **Create tables automatically** or **Use existing Topic** . The message that appears in the **DataHub Topic** column varies based on the setting of Topic creation method.

- If you set Topic creation method to **Create tables automatically**, the **Create tables automatically** dialog box appears after you click **Next Step**. Click **Start table building** in the dialog box, and then click **Close** after the topic is created.
- If you set Topic creation method to **Use existing Topic**, you must select a topic from the drop-down list in the **DataHub Topic** column.

viii. In the **Run resource settings** step, specify **Maximum number of connections supported by source read** and **Number of concurrent writes on the target side**. Then, click the icon in the top toolbar.

Commit the real-time synchronization node

1. On the configuration tab of the real-time synchronization node, click the icon in the top toolbar.
2. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
3. Click **OK**.

   If the workspace that you use is in standard mode, you must click **Deploy** in the upper-right corner after you commit the real-time synchronization node.

Manage the real-time synchronization node

1. After you commit or deploy the real-time synchronization node, click **Operation Center** in the upper-right corner of the DataStudio page to manage the node on the **Real Time DI** page.
2. On the **Real Time DI** page, find the real-time synchronization node, click the node name, and then view the O&M details about the node.

   On the Real Time DI page, you can start, stop, undeploy, or configure alert settings for the real-time synchronization node.

   - To start a node that is not running, perform the following steps:
     a. Find the node and click **Start** in the Operation column.

b. In the **Start** dialog box, configure the parameters.

Parameter	Description
Whether to reset the site	Specifies whether to set the time point for the next startup. If you select Reset site, the **Start time point** and **Time zone** parameters are required.
Start time point	The date and time for starting the real-time synchronization node.
Time zone	The time zone where the source resides. Select a time zone from the **Time zone** drop-down list.
Failover	■ The condition for automatically terminating the real-time synchronization node. You can specify the maximum number of dirty data records allowed. If you set this parameter to 0, no dirty data records are allowed. If this parameter is not specified, the node continues to run no matter whether dirty data records exist. ■ You can also specify the maximum number of failover times. If you do not specify the times, the node is automatically terminated if the node fails 100 times within 5 minutes. This prevents resource occupation caused by frequent startups.

c. Click **OK**.

- To stop a running node, perform the following steps:
  a. Find the node and click **Stop** in the Operation column.
  b. In the message that appears, click **Stop**.
- To undeploy a node that is not running, perform the following steps:
  a. Find the node and click **Offline** in the Operation column.
  b. In the message that appears, click **Offline**.
- Find the node and click **Alarm settings** in the Operation column. Then, you can view alert event information and alert rules on the **Alert event** and **Alarm rules** tabs.
- To configure alert settings for a node, perform the following steps:
  a. Select the node and click **New Alarm** in the lower part of the page.

b. In the **New rule** dialog box, configure the parameters.

Parameter	Description
Name	Required. The name of the rule that you want to create.
Description	The description of the rule.
Indicators	The metrics in the rule that you want to create. Valid values: **Task Status**, **Business latency**, **Failover**, **Dirty Data**, and **DDL error**.
Threshold	The threshold for reporting an alert. The default value is 5 minutes for both **WARNING** and **CRITICAL** alerts.
Alarm interval	The interval at which an alert is reported. The default value is 5 minutes.
WARNING	The method used to send alert notifications. The value of this parameter can be only **Mail**.
CRITICAL	
Recipient	The alert recipient. Select a recipient from the **Receiver** drop-down list.

c. Click **OK**.

- To modify alert settings for a node, perform the following steps:
  a. Select the node whose alarm settings you want to modify and click **Operation alarm** in the lower part of the page.
  b. In the **Operation alarm** dialog box, specify **Operation type** and **Alarm indicators**.

     DataWorks automatically modifies all the rules for the selected alert types at a time.
  c. Click **OK**.

2.1.4.6.4. Reader

2.1.4.6.4.1. Configure MySQL Binlog Reader

MySQL Binlog Reader reads data from tables in your MySQL database in real time after you subscribe to real-time binary logs.

Context

MySQL Binlog Reader supports the following versions of ApsaraDB RDS for MySQL: V5.1, V5.5, V5.6, V5.7, and V8.0.

If the binary log write feature is enabled for a read-only database and the binlog-format parameter is set to ROW, MySQL Binlog Reader can read data from this read-only database.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create > Real-time synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

> **Notice** The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **MySQL Binlog** in the **Input** section to the canvas on the right.
6. Click the **MySQL Binlog** node. In the panel that appears, configure the parameters.

Parameter	Description
Data source	The MySQL Binlog data source that you have added. You can select only a MySQL Binlog data source. If no data source is available, click **New data source** to add one on the **Data Source** page.
Table	The name of the table from which you want to read data. You can click **Data preview** on the right to preview the selected table. In the case of sharding, MySQL Binlog Reader can read data from multiple tables and databases in real time. **Notice** To prevent errors, the tables must use the same schema.
Output field	The fields that you want to synchronize. Valid values: ○ **Manage fields**: additional fields that are automatically added when the fields in the source tables are synchronized to the destination. These fields facilitate data management, sorting, and deduplication. ○ **Data Field**: the fields in the original tables that you want to synchronize.

MySQL Binlog Reader supports sharding. You can click **Add a database and table data source** and select the required **data source** and **table** from the Data source drop-down list. This way, multiple data sources are added, and data in these data sources is synchronized at the same time.

> **Notice** To prevent errors, the tables must use the same schema.

7. Click the icon in the toolbar.

2.1.4.6.4.2. Configure Oracle CDC Reader

Oracle Change Data Capture (CDC) Reader synchronizes data by using the triggers created in Oracle databases. You must activate the Oracle CDC service before you use Oracle CDC Reader.

Context

Oracle CDC Reader V11g is supported.

Configure Oracle CDC Reader

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create >**

Real-time synchronization.

3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Notice** The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **Oracle CDC** in the **Input** section to the canvas on the right.
6. Click the **Oracle CDC** node. In the panel that appears, configure the parameters.

Parameter	Description
Data source	The Oracle CDC data source that you have added. You can select only an Oracle CDC data source. If no data source is available, click **New data source** to add one on the **Data Source** page.
Table	The name of the table from which you want to read data. You can click **Data preview** on the right to preview the selected table.
Output field	The fields from which you want to read data.

7. Click the icon in the toolbar.

Create and authorize an Oracle subscriber

Common users have no permissions to read data from the production environment. If you want the system to run a real-time synchronization node to read data from the production environment, you must create an Oracle subscriber and grant the user the minimum permissions.

1. Execute the following statement to create an Oracle subscriber:

```
create user cdc_pubuser identified by cdc_pubuserpwd default tablespace ts_cdcpub QUOTA UNLIMITE
D ON ts_cdcpub;
```

2. Execute the following statements to grant permissions to the Oracle subscriber:

```
grant create session to cdc_pubuser;
grant create table to cdc_pubuser;
grant select_catalog_role to cdc_pubuser;
grant execute_catalog_role to cdc_pubuser;
grant connect,resource to cdc_pubuser;
# You can grant the Oracle subscriber the permissions to synchronize specific tables or all tabl
es in real time based on your business requirements.
grant select on schema.tableName to cdc_pubuser;
```

2.1.4.6.4.3. Configure DataHub Reader

DataHub Reader reads data from DataHub in real time by using the DataHub SDK.

Context

DataHub Reader keeps running after it is started and reads data from DataHub when new data is stored to DataHub. DataHub Reader provides the following features:

- Reads data in real time.

- Reads data in parallel based on the number of shards in DataHub.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create > Real-time synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Notice** The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **DataHub** in the **Input** section to the canvas on the right.
6. Click the **DataHub** node. In the configuration panel that appears, set the parameters.

Parameter	Description
Data source	The configured DataHub data source. In this example, you can select only a DataHub data source. If no data source is available, click **New data source** on the right to add one on the **Data Source** page.
Topic	The name of the DataHub topic from which you want to read data. You can click **Data preview** on the right to preview the selected topic.
Output field	The fields from which you want to read data.

7. Click the icon in the toolbar.

2.1.4.6.4.4. Configure LogHub Reader

LogHub Reader reads data from LogHub topics in real time by using the LogHub SDK and supports shard merge and split. After shards are merged or split, duplicate data records may exist but no data is lost.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create > Real-time synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Notice** The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **Loghub** in the **Input** section to the canvas on the right.

6. Click the **LogHub** node. In the panel that appears, configure the parameters.

Parameter	Description
Data source	The LogHub data source that you have configured. You can select only a LogHub data source. If no data source is available, click **New data source** on the right to add one on the **Data Source** page.
Logstore	The name of the Logstore from which you want to read data. You can click **Data preview** on the right to preview the selected Logstore.
Advanced Configuration	Specifies whether to split data in the Logstore. If you select **Split** for Split tasks, you must specify **Split rules**.
Output field	The fields that you want to synchronize.

7. Click the icon in the toolbar.

2.1.4.6.4.5. Configure Kafka Reader

Kafka Reader reads data from Kafka in real time by using a Kafka SDK.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create > Real-time synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   Notice The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **Kafka** in the **Input** section to the canvas on the right.
6. Click the **Kafka** node. In the panel that appears, configure the parameters.

Parameter	Description
server	The broker server address of Kafka, in the format of `ip:port`.
topic	The name of the Kafka topic from which you want to read data. Kafka maintains the feeds of messages in categories called topics. Each message published to a Kafka cluster is assigned to a topic. Each topic contains a group of messages. **Note** Kafka Reader can read data from only one topic for each synchronization node.

Parameter	Description
keyType	The type of the key in Kafka.
valueType	The type of the value in Kafka.
Initiation Site	The start time of data synchronization.
Configuration parameters	The extended parameters specified when KafkaConsumer is created, such as bootstrap.servers, auto.commit.interval.ms, and session.timeout.ms. You can control the data consumption behaviors of KafkaConsumer by setting parameters in KafkaConfig.

Parameter	Description
Output field	The output fields, which can be customized. ○ Click **Add more fields**. In the line that appears, enter a field name and select a data type to customize the field. **Note** The output fields of Kafka are in the JSON format by default. You do not need to specify the **Value Method** parameter. To use the entire value of a Kafka field, you can customize the output field __value__. You can use the .Sub-fieldor [Element in the data array] syntax to obtain the data in the complex JSON format. The following code shows data in a Kafka message: `{ "a": { "a1": "hello" }, "b": "world", "c":[ "xxxxxxx", "yyyyyyy" ], "d":[ { "AA":"this", "BB":"is_data" }, { "AA":"that", "BB":"is_also_data" } ] }` You can set the Output field parameter to the following values based on the preceding code: ■ To synchronize all the data in the Kafka message, set this parameter to __value__. ■ To synchronize "hello" in the a1 field, set this parameter to a.a1. ■ To synchronize "world in the b field, set this parameter to b. ■ To synchronize "yyyyyyy" in the c field, set this parameter to c[1]. ■ To synchronize "this" in the AA field, set this parameter to d[0].AA. ○ To delete a field, move the pointer over the field and click the icon.

7. Click the icon in the toolbar.

2.1.4.6.4.6. Configure PolarDB Reader

PolarDB Reader can read data only from PolarDB for MySQL databases. It cannot read data from PolarDB for PostgreSQL databases.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create > Real-time synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Notice** The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **PolarDB** in the **Input** section to the canvas on the right.
6. Click the **PolarDB** node. In the panel that appears, configure the parameters.

Parameter	Description
Data source	The PolarDB for MySQL data source that you have added. You can select only a PolarDB for MySQL data source. If no data source is available, click **New data source** to add one on the **Data Source** page.
Table	The name of the table from which you want to read data. You can click **Data preview** on the right to preview the selected table.
Output field	The fields from which you want to read data.

7. Click the icon in the toolbar.

2.1.4.6.5. Writer

2.1.4.6.5.1. Configure MaxCompute Writer

MaxCompute offers a comprehensive data import solution to support fast computing for large amounts of data.

Prerequisites

A reader or conversion node is configured.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create > Real-time synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Notice** The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **MaxCompute** in the **Output** section to the canvas on the right. Connect the MaxCompute node to the configured reader or conversion node.
6. Click the **MaxCompute** node. In the panel that appears, configure the parameters.

Parameter	Description
Data source	The MaxCompute data source that you have configured. You can select only a MaxCompute data source. If no data source is available, click **New data source** on the right to add one on the **Data Source** page.
Table	The name of the MaxCompute table to which you want to write data. You can click **One-Click table creation** on the right to create a table, or click **Data preview** to preview the selected table. **Notice** Before you create a table, connect the MaxCompute node to a reader node, and make sure that the output field parameters are specified for the reader node.
Partition message	The information about the partitioned MaxCompute table.
Field Mapping	The field mappings between the source and destination. Click **Field Mapping** and configure field mappings. The synchronization node synchronizes data based on the field mappings.

If you want to create a table, click **One-Click table creation** next to Table. In the **New data table** dialog box, configure the parameters.

Parameter	Description
Table name	The name of the MaxCompute table.
Life cycle	The lifecycle of the MaxCompute table.
Data field structure	The field structure of the MaxCompute table. To add a field, click **Add**.
Partition settings	The partitions of the MaxCompute table. **Notice** You must configure at least two levels of partitions, which are yearly and monthly partitions. You can configure a maximum of five levels of partitions, which are yearly, monthly, daily, hourly, and minutely partitions.

7. Click the icon in the toolbar.

2.1.4.6.5.2. Configure Hologres Writer

You can build a real-time data warehouse by using the real-time write capability of Hologres.

Prerequisites

A reader or conversion node is configured.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create > Real-time synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Notice** The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **Hologres** in the **Output** section to the canvas on the right. Then, draw a line to connect it to the configured reader or conversion node.
6. Click the **Hologres** node. In the panel that appears, configure the parameters.

Parameter	Description
Data source	The Hologres data source that you configured. You can select only a Hologres data source. If no data source is available, click **New data source** to add one on the **Data Source** page.
Table	The name of the Hologres table to which you want to write data. You can click **One-Click table creation** on the right to create a table, or click **Data preview** to preview the selected table.
Dynamic Time Partition	If the Hologres table is a partitioned table, you must specify a dynamic time-based partition. The dynamic time-based partition parses the value of a source field in the yyyymmdddhhmmss format. After the value is parsed, you can use the dynamic partition whose name is a string of variables in the destination table. The destination partition varies based on the value of the source field. For example, the value of the source field is 20200816, and the name of the destination partition is in the {yyyy}-{mm}-{dd} format. In this case, the value is written to the 2020-08-16 partition.
Job type	The type of the data write operation. Valid values: **Replay (replay operation log to restore data)** and **Insert (direct archive save)**. ○ **Replay (replay operation log to restore data)**: indicates that Hologres Writer performs the same operation on the Hologres destination as that performed on the source. For example, if the `INSERT` statement is executed to add a record to the source, Hologres Writer executes the `INSERT` statement to add the same record to the Hologres destination. If the `UPDATE` or `DELETE` statement is executed in the source, Hologres Writer executes the `UPDATE` or `DELETE` statement in the Hologres destination. ○ **Insert (direct archive save)**: indicates that Hologres Writer uses the Hologres destination as streaming data storage. Data is synchronized from the source to the Hologres destination by using the `INSERT` statement.

Parameter	Description
Writer conflict policy	The solution to data write conflicts. Valid values: ○ **Cover (Overwrite)**: indicates that Hologres Writer uses the new data synchronized from the source to overwrite the existing data in the Hologres destination. ○ **Ignore (Ignore)**: indicates that Hologres Writer ignores the new data synchronized from the source and retains the existing data in the Hologres destination.
Field Mapping	The field mappings between the source and destination. Click **Field Mapping** and configure field mappings between the source and destination. The synchronization node synchronizes data based on the field mappings.

7. Click the icon in the toolbar.

2.1.4.6.5.3. Configure DataHub Writer

DataHub is a platform that is 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.

Prerequisites

A reader or conversion node is configured.

Context

DataHub Writer writes data to DataHub by using DataHub SDK for Java. The following code provides an example of DataHub SDK for Java:

```
<dependency>
    <groupId>com.aliyun.datahub</groupId>
    <artifactId>aliyun-sdk-datahub</artifactId>
    <version>2.5.1</version>
</dependency>
```

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create > Real-time synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   Notice The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **DataHub** in the **Output** section to the canvas on the right. Connect the new node to the configured reader or conversion node.
6. Click the new **DataHub** node. In the configuration panel that appears, set the parameters.

Parameter	Description
Data source	The configured DataHub data source. In this example, you can select only a DataHub data source. If no data source is available, click **New data source** on the right to add one on the **Data Source** page.
Topic	The name of the DataHub topic to which you want to write data. You can click **Data preview** on the right to preview the selected topic.
Batch number	The number of records to write at a time.
Field Mapping	The mappings between fields in the source and destination. DataWorks synchronizes data based on the field mappings.

7. Click the icon in the toolbar.

2.1.4.6.5.4. Configure Kafka Writer

To configure Kafka Writer, select a table and configure field mappings.

Prerequisites

A reader or conversion node is configured.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create > Real-time synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Notice** The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **Kafka** in the **Output** section to the canvas on the right. Connect the new node to the configured reader or conversion node.
6. Click the new **Kafka** node. In the configuration panel that appears, set the parameters.

Parameter	Description
server	The broker server address of Kafka, in the format of `ip:port`.
topic	The name of the Kafka topic to which you want to write data. Kafka maintains feeds of messages in categories called topics. Each message that is published to a Kafka cluster is assigned to a topic. Each topic contains a group of messages.
keyColumn	The column that is specified as the key.

Parameter	Description
valueColumn	The column that is specified as the value. If this parameter is not specified, all columns are concatenated by the delimiter that is specified by the fieldDelimiter parameter to form the value.
keyType	The type of the key in Kafka.
valueType	The type of the value in Kafka.
batchSize	The number of data records to write at a time. Default value: *1024*.
Configuration parameters	The extended parameters specified when KafkaConsumer is created, such as bootstrap.servers, auto.commit.interval.ms, and session.timeout.ms. You can set parameters in kafkaConfig to control the data consumption behavior of KafkaConsumer.

7. Click the icon in the toolbar.

2.1.4.6.6. Transform

2.1.4.6.6.1. Configure data filtering

The Data Filtering plug-in is used to filter data based on specific rules, such as the field size. Only data that meets the rules is retained.

Prerequisites

A reader node is configured.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create > Real-time synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Notice** The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **Data Filtering** in the **Conversion** section to the canvas on the right. Connect the Data Filtering node to the configured reader node.
6. Click the new **Data Filtering** node. In the configuration panel that appears, set the parameters.
   - **Node configuration**

     Rules: the rules for filtering data in data sources. Only data that meets the rules is retained.
   - **Output field**

     The names and types of output fields after filtering.
7. Click the icon in the toolbar.

2.1.4.6.6.2. Configure string replacement

The String Replace plug-in is used to replace the field values of the STRING type.

Prerequisites

A reader node is configured.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create > Real-time synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Notice** The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **String Replace** in the **Conversion** section to the canvas on the right. Connect the String Replace node to the configured reader node.
6. Click the **String Replace** node. In the configuration panel that appears, set the parameters.

Parameter	Description
Rules	The rules that are used to replace the field values of the STRING type. Each rule is defined by the following parameters: ○ **Field**: the field of the parent node that you want to use as the input field. ○ **Regular matching**: specifies whether a regular expression is used to search for the original string. ○ **Original string**: the original string to search. ○ **New string**: the new string that is used to replace the original string. ○ **Case Sensitive**: specifies whether the value is case-sensitive during the search.
Add condition	Click this button to add more string replacement rules.
Output field	The output fields after string replacement.

7. Click the icon in the toolbar.

2.1.4.7. Data synchronization solutions

2.1.4.7.1. Go to the Sync Solutions page

The Data Integration service of DataWorks allows you to create and configure a synchronization solution to synchronize data from a source to a destination in real time. You can use a synchronization solution to synchronize multiple tables at a time or synchronize both full and incremental data. If you want to synchronize both full and incremental data, you can synchronize the incremental data after the full data is synchronized.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
3. In the left-side navigation pane, click **Sync Solutions**. The **Solution Task list** page appears.

   You can create synchronization solutions and view the details and statuses of the created synchronization solutions on this page. A synchronization solution has the following states:

   - **Not running**: The synchronization solution is not run. You can click **Start execution** in the Operation column that corresponds to the synchronization solution to run the synchronization solution.

     > **Note** You can click **Task configuration** in the Operation column that corresponds to a synchronization solution in the **Not running** state to edit the synchronization solution. If you click Task configuration in the Operation column that corresponds to a synchronization solution in another state, you can only view the information about the synchronization solution.

   - **Running**: The synchronization solution is running and cannot be terminated. You must wait until the synchronization solution is completed.
   - **Exception**: An error occurred during the running of the synchronization solution. You can click **Execution details** in the Operation column that corresponds to the synchronization solution to troubleshoot the error.
   - **Success**: The synchronization solution is completed. You can click **Execution details** in the Operation column that corresponds to the synchronization solution to view the running results of the synchronization solution.

2.1.4.7.2. Synchronize data to Hologres in real time

You can create and configure a data synchronization solution to synchronize data in a specified data source to Hologres in real time. This topic describes how to synchronize data to Hologres in real time.

Procedure

1. Go to the **Solution Task list** page.
   i. Log on to the DataWorks console.
   ii. Click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. In the left-side navigation pane, click **Sync Solutions**.
2. On the **Solution Task list** page, click **New Task** in the upper-right corner.
3. In the **New resolution task** dialog box, click **One-click real-time synchronization to Hologres**.
4. In the **Set synchronization sources and rules** step, configure the parameters.

i. In the **Basic configuration** section, configure the relevant parameters.

Parameter	Description
Scheme name	The name of the synchronization solution. The name can be a maximum of 50 characters in length.
Description	The description of the synchronization solution. The description can be a maximum of 50 characters in length.
Destination task storage location	If you select Automatically establish workflow, DataWorks automatically creates a workflow named in the format of clone_database_Source name+to+Destination name in the **Data Integration** directory. All synchronization nodes generated by the synchronization solution are placed in the directory of this workflow. If you do not select **Automatically establish workflow**, you must select a directory from the **Select Location** drop-down list. All synchronization nodes generated by the synchronization solution are placed in the specified directory.

ii. In the **Data source** section, specify **Type** and **Data source**.

iii. In the **Select the source table for synchronization** section, select the tables that you want to synchronize in the **SOURCE Table** list and click **>** to move the tables to the **Selected Source table** list.

The SOURCE Table list displays all the tables in the source. You can select all or some tables to synchronize them at a time.

> **Notice** If a selected table does not have a primary key, the table cannot be synchronized in real time.

iv. In the **Set synchronization rules** section, click **Add rule** and select an option to configure naming rules for destination tables.

Supported options include **Table name conversion rules** and **Target table name rule**.

- **Table name conversion rules**: the rule for converting the names of source tables to those of destination tables.
- **Target table name rule**: the rule for adding a prefix and suffix to the converted names of destination tables.

v. Click **Next Step**.

5. In the **Set target table** step, configure the parameters.

i. Specify **Target Hologres data source** and **Schema**. **Write Hologres policy** is set to **Replay (replay operation log to restore data)** by default and cannot be changed.

ii. Click **Reload source table and Hologres Table mapping** to configure the mappings between the source tables and destination Hologres tables.

iii. View the mapping progress, source tables, and mapped destination tables.

You can view the mapping progress between the source tables and destination tables. The mapping may take a long period of time if you want to synchronize a large number of tables.

An error message appears if the selected source table does not have a primary key. The synchronization can be performed if one of the selected source tables has a primary key. Source tables without primary keys are ignored during the synchronization.

You can set Table creation method to **Create tables automatically** or **Use existing Table**. The name of the destination table that appears in the **Hologres Table name** column varies based on the setting of Table creation method.

- If you set **Table creation method** to **Create tables automatically**, the name of the destination table that is automatically created appears. You can click the table name to view and modify the table creation statements.
- If you set **Table creation method** to **Use existing Table**, you must select a table from the drop-down list in the Hologres Table name column.

iv. Click **Next Step**.

6. In the **Run resource settings** step, configure the parameters.

Parameter	Description
Realtime task resource group	The resource group used for running the batch synchronization node and real-time synchronization nodes generated by the synchronization solution.
Offline task resource group	
Select scheduling Resource Group	The resource group for scheduling used for running the nodes generated by the synchronization solution.
Maximum number of connections supported by source read	The maximum number of Java Database Connectivity (JDBC) connections that are allowed for the source. Specify an appropriate number based on the resources of the source.
Offline task name rules	The name of the batch synchronization node that is used to synchronize the full data of the source. After a synchronization solution is created, a batch synchronization node is generated first to synchronize full data, and then real-time synchronization nodes are generated to synchronize incremental data.

7. Click **Complete configuration**.
8. On the **Solution Task list** page, find the newly created synchronization solution and click **Start execution** in the Operation column.

After the running of the synchronization solution is successful, you can perform the following operations on the synchronization solution:

- Click **Task configuration** in the Operation column to view the information about or edit the synchronization solution.

> **Note** You can click **Task configuration** in the Operation column that corresponds to the data synchronization solution in the **Not running** state to edit the data synchronization solution. If you click Task configuration in the Operation column of a synchronization solution in another state, you can only view information about the synchronization solution.

- Click **Execution details** in the Operation column to view the points in time at which the synchronization solution was started and ended and the status of each node.
- Click **Delete** in the Operation column to delete the synchronization solution. In the **Delete** message, click

Confirm.

> **Note** After you click Confirm, only the configuration record of the data synchronization solution is deleted. The generated synchronization nodes and tables are not affected.

2.1.4.7.3. Synchronize data to MaxCompute in real time

You can create a synchronization solution to synchronize data from a specified data source to MaxCompute in real time.

Procedure

1. Go to the **Solution Task list** page.
   i. Log on to the DataWorks console.
   ii. Click the ☰ icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. In the left-side navigation pane, click **Sync Solutions**.
2. On the **Solution Task list** page, click **New Task** in the upper-right corner.
3. In the **New resolution task** dialog box, click **One-click real-time synchronization to MaxCompute**.
4. In the **Set synchronization sources and rules** step, configure the parameters.
   i. In the **Basic configuration** section, configure the relevant parameters.

Parameter	Description
Scheme name	The name of the synchronization solution. The name can be a maximum of 50 characters in length.
Description	The description of the synchronization solution. The description can be a maximum of 50 characters in length.
Destination task storage location	If you select Automatically establish workflow, DataWorks automatically creates a workflow named in the format of clone_database_Source name+to+Destination name in the **Data Integration** directory. All synchronization nodes generated by the synchronization solution are placed in the directory of this workflow. If you do not select **Automatically establish workflow**, you must select a directory from the **Select Location** drop-down list. All synchronization nodes generated by the synchronization solution are placed in the specified directory.

   ii. In the **Data source** section, specify **Type** and **Data source**.
   iii. In the **Select the source table for synchronization** section, select the tables that you want to synchronize in the **SOURCE Table** list and click **>** to move the tables to the **Selected Source table** list.

   The SOURCE Table list displays all the tables in the source. You can select all or some tables to synchronize them at a time.

> **Notice** If a selected table does not have a primary key, the table cannot be synchronized in real time.

iv. In the **Set synchronization rules** section, click **Add rule** and select an option to configure naming rules for destination tables.

Supported options include **Table name conversion rules** and **Target table name rule**.

- **Table name conversion rules**: the rule for converting the names of source tables to those of destination tables.
- **Target table name rule**: the rule for adding a prefix and suffix to the converted names of destination tables.

v. Click **Next Step**.

5. In the **Set target table** step, configure the parameters.

i. Select a data source from the **Target MaxCompute data source** drop-down list and specify **Write mode**.

ii. Click next to **MaxCompute time automatic partition settings**. In the **Edit** dialog box, modify the partition settings for the destination tables. You can configure daily and hourly partitions.

iii. Click **Reload source table and MaxCompute Table mapping** to configure the mappings between the source tables and destination MaxCompute tables.

iv. View the mapping progress, source tables, and mapped destination tables.

You can view the mapping progress between the source tables and destination tables. The mapping may take a long period of time if you want to synchronize a large number of tables.

An error message appears if the selected source table does not have a primary key. The synchronization can be performed if one of the selected source tables has a primary key. Source tables without primary keys are ignored during the synchronization.

You can set Table creation method to **Create tables automatically** or **Use existing Table**. The name of the destination table that appears in the MaxCompute Table name column varies based on the setting of Table creation method.

- If you set **Table creation method** to **Create tables automatically**, the name of the destination table that is automatically created appears. You can click the table name to view and modify the table creation statements.
- If you set **Table creation method** to **Use existing Table**, you must select a table name from the drop-down list in the MaxCompute Table name column.

v. Click **Next Step**.

6. In the **Run resource settings** step, configure the parameters.

Parameter	Description
Synchronization engine	Default value: **Default embedded engine**.
Realtime task resource group	The resource group that is used to run the nodes generated by the real-time synchronization solution.
Real-time synchronization task name	The name of the real-time synchronization solution.
Select scheduling Resource Group	The resource group that is used to run the real-time synchronization nodes and batch synchronization node generated by the synchronization solution. Synchronization solutions can run on shared resource groups and custom resource groups for Data Integration.
Offline task resource group	

Parameter	Description
Maximum number of connections supported by source read	The maximum number of Java Database Connectivity (JDBC) connections that are allowed for the source database. Specify an appropriate number based on the resources of the source database.
Offline task name rules	The name of the batch synchronization node that is used to synchronize the full data of the source database. After a data synchronization solution is created, DataWorks first generates a batch synchronization node to synchronize full data, and then generates real-time synchronization nodes to synchronize incremental data.

7. Click **Complete configuration**.
8. On the **Solution Task list** page, find the newly created synchronization solution and click **Start execution** in the Operation column.

   After the synchronization solution is run, you can perform the following operations on the synchronization solution:

   - Click **Task configuration** in the Operation column to view information about or configure the synchronization solution.

     > **Note** You can configure a synchronization solution only when it is in the **Not running** state. If you click Task configuration in the Operation column of a synchronization solution that is in another state, you can only view information about the synchronization solution.

   - Click **Execution details** in the Operation column to view the time at which the synchronization solution was started and ended and the status of each node.
   - Click **Delete** in the Operation column to delete the synchronization solution. In the **Delete** message, click **Confirm**.

     > **Note** After you click Confirm, only the configuration record of the data synchronization solution is deleted. The generated synchronization nodes and tables are not affected.

2.1.4.8. Resource groups

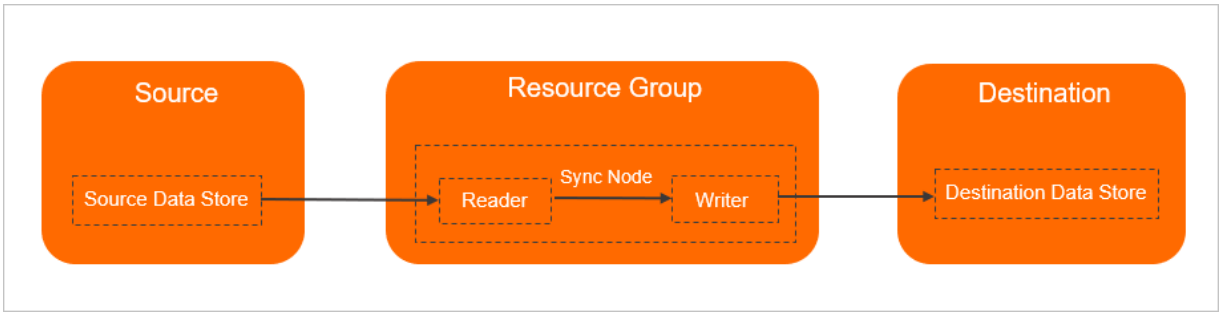
2.1.4.8.1. Overview

This topic introduces the definition of resource groups. It also describes how the connectivity and performance of resource groups affect data synchronization.

Definition

A resource group is a collection of computing resources on which synchronization nodes of Data Integration are run. In most cases, a resource group is one or more servers that consist of CPU, memory, and network resources.

In the process of running a synchronization node, the resource group pulls data from the source and pushes the data to the destination.



Connectivity and performance

When you use resource groups, you must pay attention to their connectivity and performance.

- Connectivity

  To ensure that data can be synchronized, a resource group must be connected to the source and destination. Connectivity is the most important factor that affects data synchronization.

  Data Integration cannot build networks. Before you use Data Integration to synchronize data, you must make sure that the resource group is connected to the data sources. If the resource group is disconnected from the data sources, synchronization nodes cannot be run.

- Performance

  Synchronization nodes consume the CPU, memory, and network resources on the servers where the nodes are run. Insufficient resources may lead to various issues. For example, the nodes fail to be started, wait for resources for a long period of time after startup, transmit data at a low rate, or fail to generate data. To ensure the smooth running of synchronization nodes, you must allocate adequate resources for them.

2.1.4.8.2. Shared resource groups

Data Integration of DataWorks provides shared resource groups for you to create and run synchronization nodes.

Shared resource groups for Data Integration are created and maintained by Data Integration. Shared resource groups compose a public resource pool. Nodes that use resources in the public resource pool may not be run as scheduled due to insufficient resources. We recommend that you prepare sufficient resources to ensure the efficient running of synchronization nodes.

> **Note**
>
> - You can run a maximum of 25 parallel nodes on a shared resource group for Data Integration when the shared resource group is not in use.
> - You cannot change the memory size of a shared resource group. Instead, you can change the number of parallel nodes that can be run on the shared resource group.
>
>   The following formula is used to calculate the memory size: Memory size = `Number of parallel nodes × 512 MB`.

2.1.4.8.3. Create a custom resource group for Data Integration

This topic describes how to create a custom resource group for Data Integration and select a resource group for Data Integration to run a batch synchronization node.

Prerequisites

An Elastic Compute Service (ECS) instance is available.

Context

If the shared resource groups of DataWorks do not support your data sources or you want to speed up data transmission, you can create custom resource groups to run your synchronization nodes.

A workspace administrator can create or modify custom resource groups on the **Custom Resource Groups** page of Data Integration.

> **Note**
> - The admin permission is required to access some files on the ECS instance that hosts a custom resource group. For example, the admin permission is required to call shell or Structured Query Language (SQL) files on the ECS instance when you write a shell script for a node.
> - Resource groups for scheduling are used to run nodes. These resource groups have limited resources and are not suitable for computing nodes. Therefore, we recommend that you do not create custom resource groups on the ECS instances of a resource group for scheduling. MaxCompute can process large amounts of data. We recommend that you use MaxCompute for big data computing.

Custom resource groups for Data Integration are subject to the following limits:

- The difference between the time of the ECS instance where a custom resource group for Data Integration resides and the current Internet time must be within 2 minutes. Otherwise, service requests may time out, and nodes may fail to be run on the custom resource group for Data Integration.
- You can add only one custom resource group for Data Integration on an ECS instance. You can select only one network type for each custom resource group for Data Integration.
- Custom resource groups added on the **Custom Resource Groups** page of Data Integration can run synchronization nodes created only in the current workspace.

  Custom resource groups for Data Integration that you added on the Custom Resource Groups page cannot run synchronization nodes in a manually triggered workflow.

If the timeout error message `response code is not 200` exists in the log file of alisatasknode, the custom resource group for Data Integration was not accessible within the specific period in time. The ECS instance that hosts the custom resource group for Data Integration can continue to work if the exception persists for no more than 10 minutes. To find the exception details, view the heartbeat.log file in the `/home/admin/alisatasknode/logs` directory.

Create a custom resource group for Data Integration

1. Go to the **Data Integration** page.
   i. Log on to the DataWorks console.
   ii. Click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
2. In the left-side navigation pane, click **Custom Resource Group**.
3. On the **Custom Resource Groups** page, click **Add Resource Group** in the upper-right corner.

   > **Notice** By default, the Custom Resource Groups page displays only your custom resource groups and does not display your shared resource groups.

4. In the **Add Resource Group** wizard, perform the following steps:

i. In the **Create Resource Group** step, set the **Resource Group Name** parameter.

> **Note** The name can contain letters, digits, and underscores (_) and must start with a letter.

ii. Click **Next**.

iii. In the **Add Server** step, set the parameters.

Parameter	Description
Network Type	The network type. Valid values: **Classic Network** and **VPC**.
Server Name or **ECS UUID**	The hostname or the universally unique identifier (UUID) of the ECS instance that hosts the custom resource group. ■ If you set Network Type to **Classic Network**, you must set the **Server Name** parameter. To obtain the hostname, log on to the ECS instance and run the `hostname` command. ■ If you set Network Type to **VPC**, you must set the **ECS UUID** parameter. To obtain the UUID, log on to the ECS instance and run the `dmidecode | grep UUID` command.
Server IP Address	The private IP address of the ECS instance.
Server CPU (Cores)	The number of CPU cores on the ECS instance. We recommend that you configure at least four CPU cores for an ECS instance that hosts a custom resource group.
Server RAM (GB)	The memory of the ECS instance. We recommend that you configure at least 8 GB RAM and 80 GB disk space for an ECS instance that hosts a custom resource group.

iv. Click **Next**.

v. Perform the steps that are listed in the **Install Agent** step.

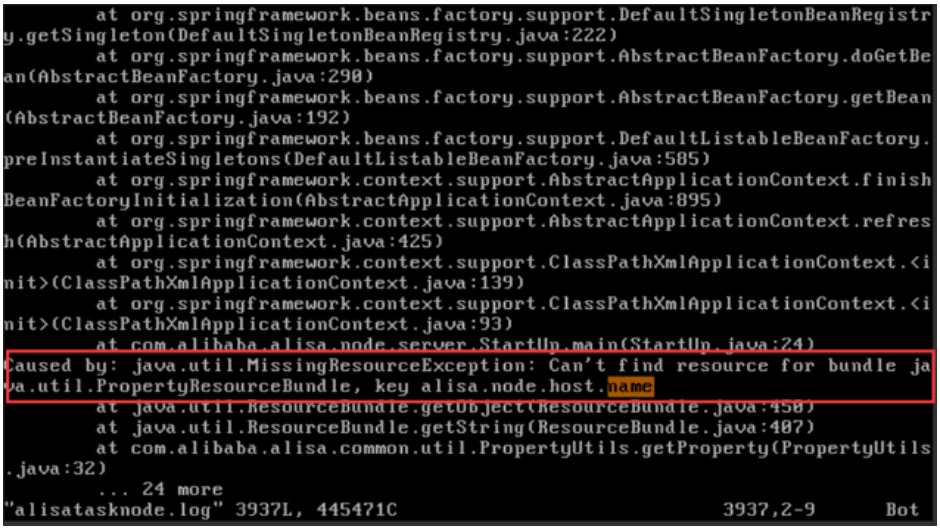
> **Note** If an error occurs when you run the `install.sh` script or you need to run it again, run the `rm -rf install.sh` command in the same directory as the `install.sh` script to delete the generated file. Then, run the `install.sh` script again.
>
> The commands to run during the installation and initialization process differ for each user. Run relevant commands based on the instructions on the initialization interface.

vi. Click **Next**.

vii. In the **Test Connection** step, click **Refresh** and check the status of the instance.

viii. Click **Complete**.

If the instance status remains **Stopped** after the preceding steps, the hostname may not be bound to an IP address, as shown in the following figure.



1. Log on to the ECS instance by using the admin user.
2. Run the `hostname -i` command to view the hostname binding information.
3. Run the `vim/etc/hosts` command to add the binding of the IP address and hostname.
4. Refresh the instance status and check whether the ECS instance is registered.

   If the ECS instance is still in the Stopped state after you refresh the page, perform the following steps to restart alisatasknode:

   i. Log on to the ECS instance by using the admin user.

   ii. Run the following command:

   ```
   /home/admin/alisatasknode/target/alisatasknode/bin/serverctl restart
   ```

   > **Note** You must enter your AccessKey pair when you run this command. Keep your AccessKey secret strictly confidential.

Configure the resource group for Data Integration

1. Click the icon in the upper-left corner of the Data Integration page and choose **All Products > Data Development > DataStudio**.
2. In the upper-left corner of the page that appears, select the workspace where your resource group for Data Integration resides.
3. On the **Data Analytics** tab, expand the workflow where the batch synchronization node you want to configure resides, find the batch synchronization node in the Data Integration folder, and then double-click it.
4. On the configuration tab of the node, click the **Resource Group configuration** tab in the right-side navigation pane.
5. On the **Resource Group configuration** tab, set **Programme** and select a resource group based on your business requirements.
6. On the configuration tab of the node, click the icon in the top toolbar.

2.1.4.9. Full-database migration

2.1.4.9.1. 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.

2.1.4.9.2. 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.
Advanced Settings	You can configure the rules for converting the table name, column names, and data types.

Functional module	Description
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('${bdp.system.bizdate}', '%Y%m%d') <= gmt_modified AND gmt_modified < DATE_ADD(STR_T
   O_DATE('${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.

2.1.4.9.3. 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.

2.1.5. Data Analytics

2.1.5.1. 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:

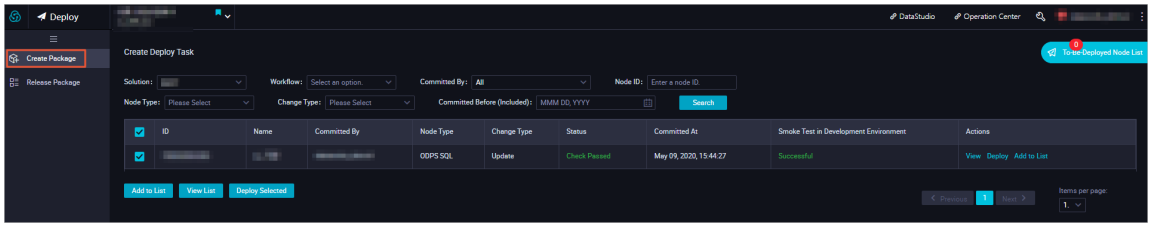
- Workflows facilitate business-oriented code development. Nodes in a workflow are organized by type. A hierarchical directory structure is supported. We recommend that you create a maximum of four levels of sub-directories. To create a sub-directory, right-click the target node type and select **Create 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.

2.1.5.2. Guidelines and specifications of SQL coding

This topic describes the basic guidelines and detailed specifications of SQL coding.

SQL coding guidelines

When you write SQL code, take note of the following guidelines:

- The code is comprehensive.
- Code lines are clear, neat, well-organized, and structured.
- The optimal execution speed is considered during SQL coding.

- Comments need to be added if 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.
- All SQL keywords and reserved words must be all uppercase or lowercase, such as select/SELECT, from/FROM, where/WHERE, and/AND, or/OR, union/UNION, insert/INSERT, delete/DELETE, group/GROUP, having/HAVING, and count/COUNT. Do not use mixed-case letters, such as Select or seLECT.
- A unit of indentation contains four spaces. All indentations must be the integral multiple of an indentation unit. The code is aligned based on its hierarchy.
- You cannot use an `asterisk (*)` to specify a column name. The column name must be specified for all statements.
- Matching opening and closing parentheses must be placed in the same column.

SQL coding specifications

When you write SQL code, take note of the following specifications:

- 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 users can add change records in the future. Each line can contain a maximum of 80 characters. The following code provides an example template:

```
-- 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.
  - Reserve one unit of indentation between the word SELECT and the first selected field.
  - Start each of the other field names in a new line with two units of indentation and a comma (,).
  - Place the comma (,) between two fields right before the second field.
  - Place the AS statement in the same line as its matching field and keep 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 in compliance with the following requirements:

- Use a line for each clause.
- The clauses are left aligned with the SELECT statement.
- Reserve two units of indentation between the first word of a clause and its content.
- Keep the logical operators such as AND and OR in a WHERE clause left aligned with the word WHERE.
- If the length of a clause name exceeds two units of indentation 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

  Reserve one space before and after each arithmetic operator and logical operator. Keep all the operators in the same line unless the length of the code exceeds 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)
```

- Arrangement for a SELECT CASE statement

  The SELECT CASE statement is used to evaluate the value of a variable. Take note of the following specifications on SELECT CASE statements:

  - Write the WHEN clause one unit of indentation after the CASE statement in the same line.
  - Use a line for each WHEN clause. Wrap a line if the clause is excessively long.

```
,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
```

  - The CASE statement must contain the ELSE clause. The ELSE clause must be aligned with the WHEN clause.

- Nested query

Nested queries are often used to implement the extract, transform, and load (ETL) process of data warehouse systems. The following figure shows an example of arranging nested queries.

```
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
  - If an alias is defined for a table in a SELECT statement, you must use the alias whenever you reference the table in the statement. Specify an alias for each table in the SELECT statement.
  - We recommend that you define the table aliases by using letters in alphabetical order.
  - 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 4after the letter that represents the level.

```
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 to clauses that are difficult to understand.
- Add a comment to 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.

2.1.5.3. GUI elements

2.1.5.3.1. 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.

No.	GUI element	Description
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.
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.
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.

2.1.5.3.2. Workflow Parameters

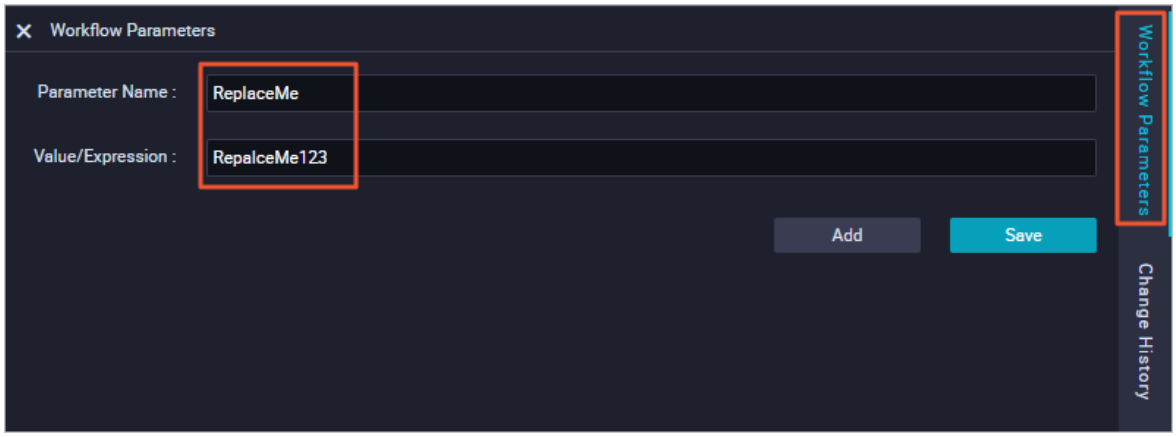
On the Workflow Parameters tab, you can assign a value to a variable or replace the value of a parameter for all nodes in the current workflow. This topic describes how to configure a workflow parameter by assuming that you want to replace the value of the ReplaceMe parameter with ReplaceMe123 in a manually 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.

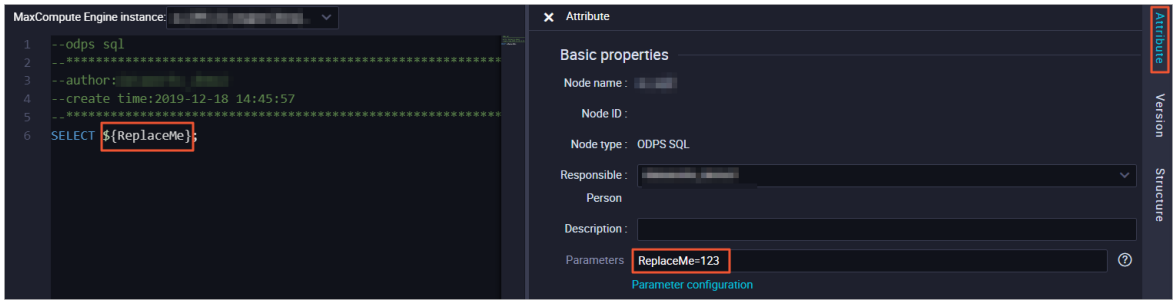


5. Click in the toolbar.

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.

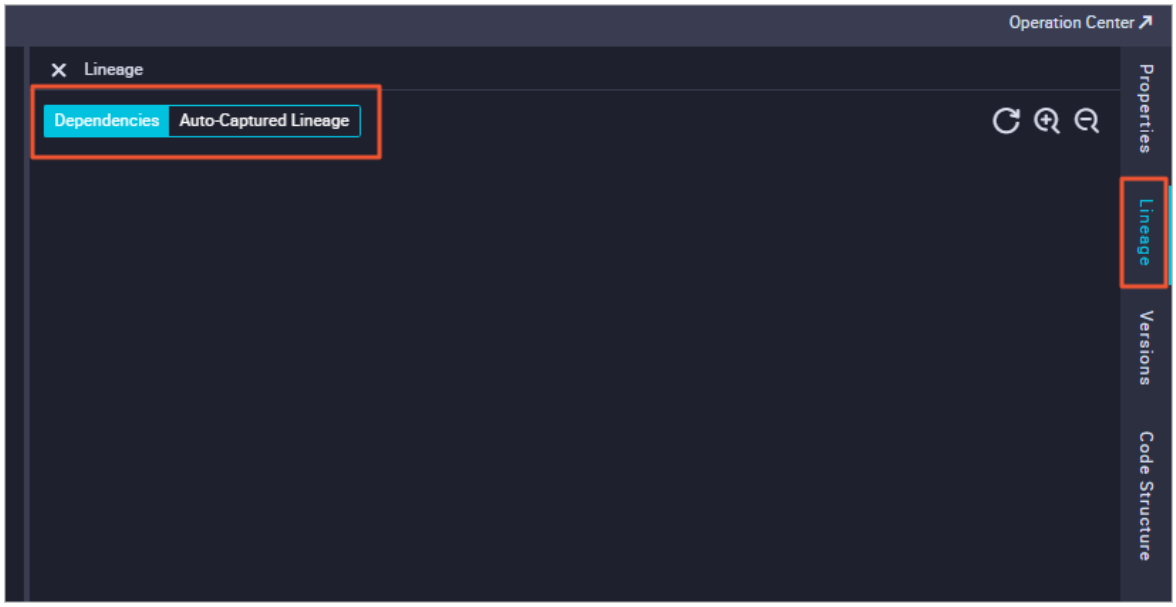
If you have not assigned a value to a workflow parameter on the **Workflow Parameters** tab for a manually triggered workflow, you must assign a value to the workflow parameter every time you run the workflow in the production environment.

2.1.5.3.3. 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.

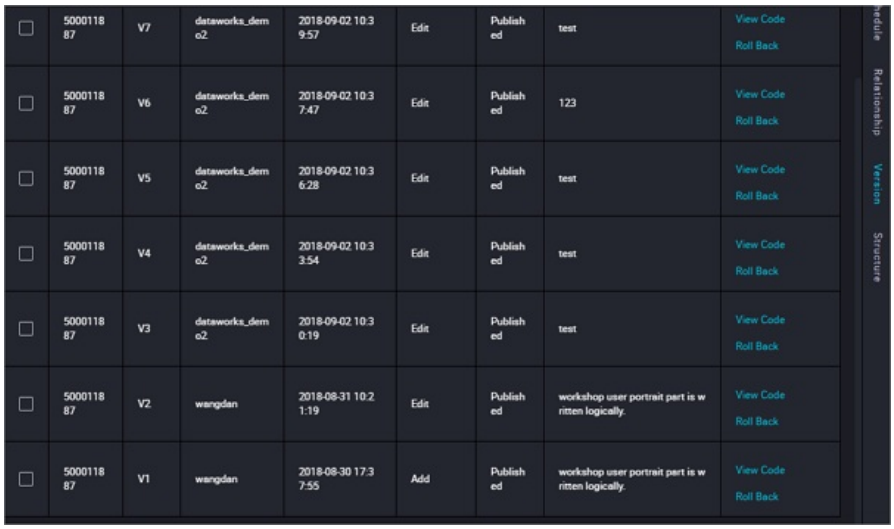
2.1.5.3.4. Versions

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.

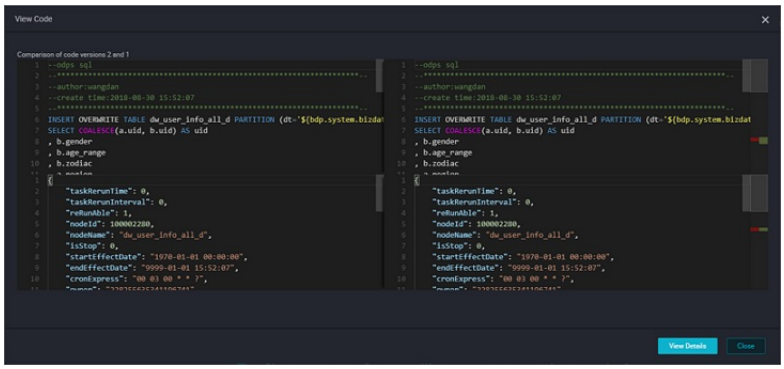
> **Note** Only a committed node has the version information. Every time a node is committed, a version is generated and added to the **Versions** tab.

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.



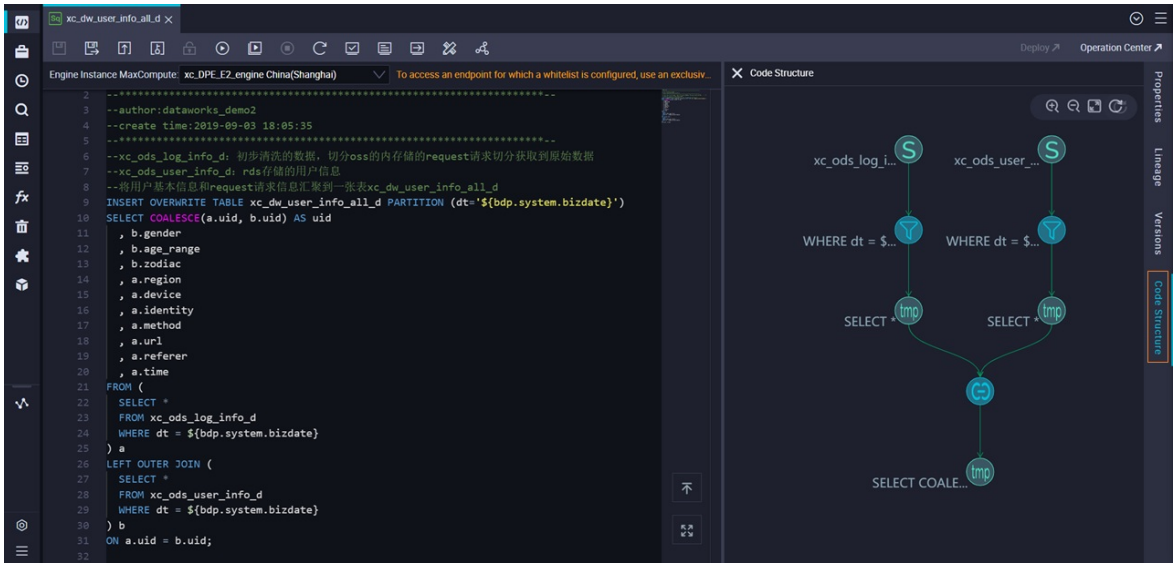
GUI element	Description
File ID	The ID of the node.
Versions	The version of the node. A version is generated each time the node is committed and deployed. V1 indicates version 1 and V2 indicates version 2. The version number is incremented by 1 each time.
Committed By	The user who committed the version.
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.
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.

GUI element	Description
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.

2.1.5.3.5. 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.

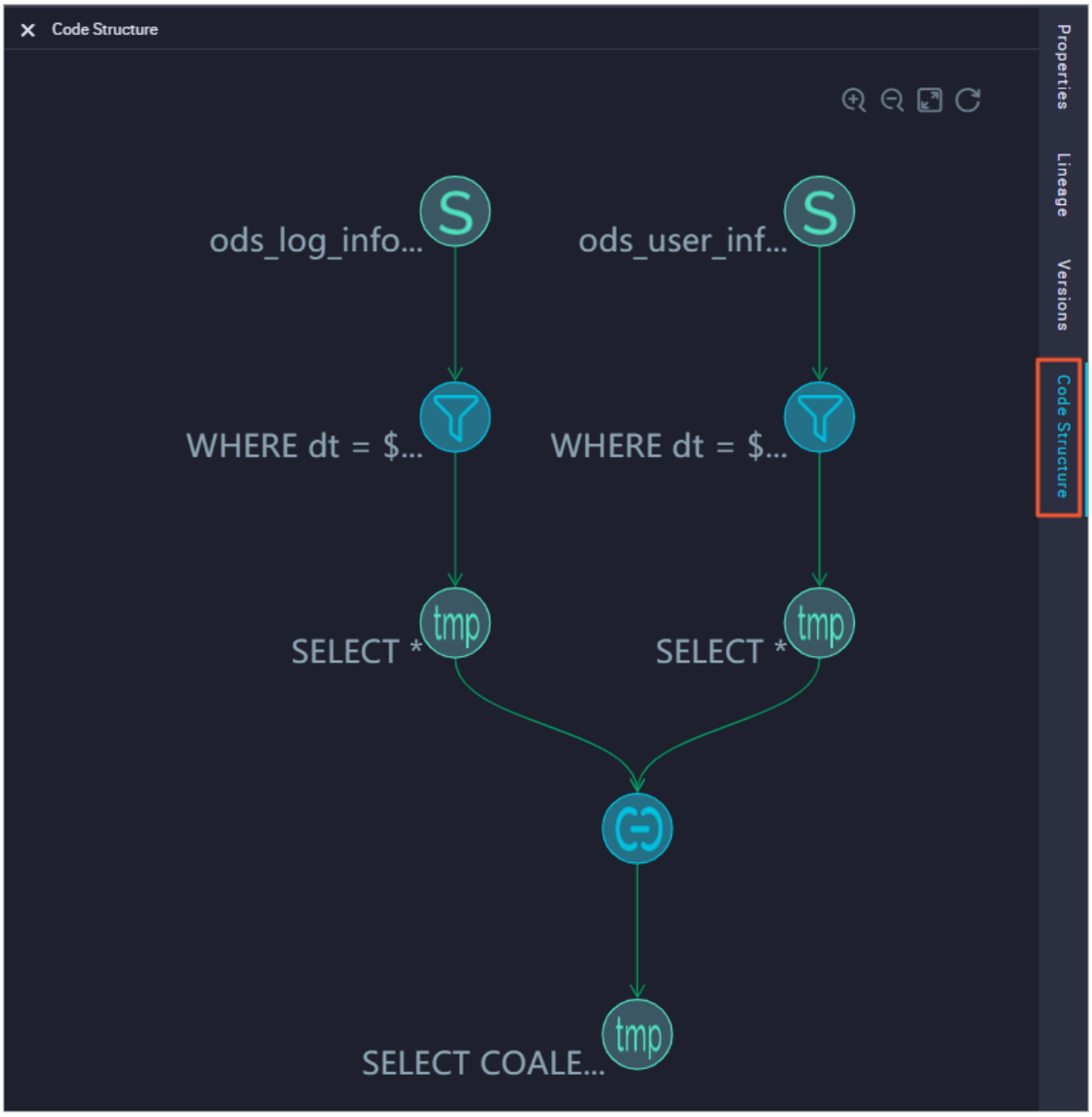
1. Log on to the DataWorks console.
2. Double-click the ODPS SQL node whose code structure you want to view. For more information about how to create a node, see Create an ODPS SQL node.
3. On the node configuration tab that appears, click the **Code Structure** tab in the right-side navigation pane.



In this example, the ODPS SQL node contains the following SQL statement:

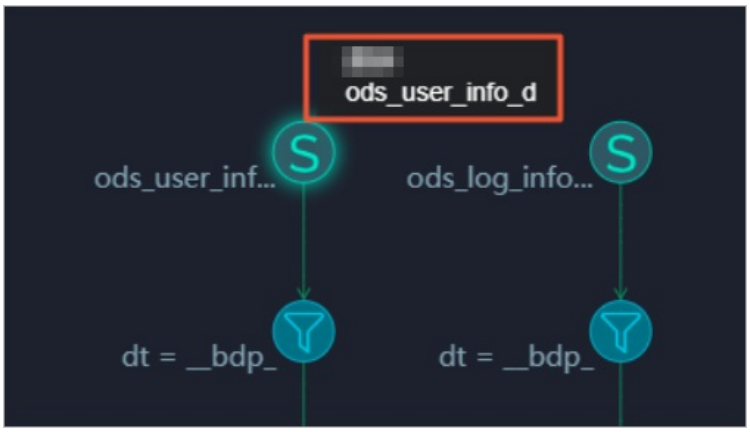
```
INSERT OVERWRITE TABLE dw_user_info_all_d PARTITION (dt='${bdp.system.bizdate}')
SELECT COALESCE(a.uid, b.uid) AS uid
  , b.gender
  , b.age_range
  , b.zodiac
  , a.region
  , a.device
  , a.identity
  , a.method
  , a.url
  , a.referer
  , a.time
FROM (
  SELECT *
  FROM ods_log_info_d
  WHERE 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.

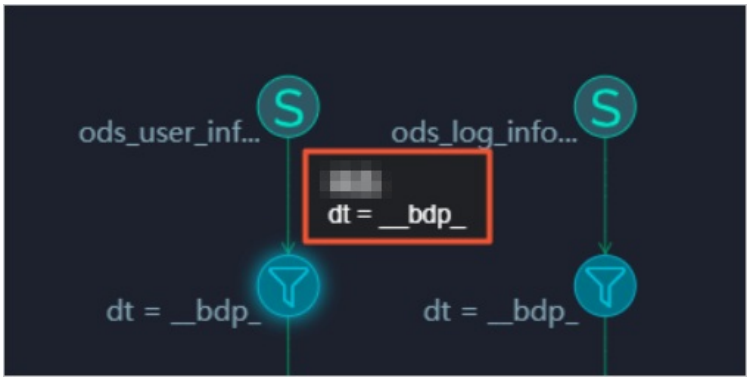


Move the pointer over a circle to view the description.

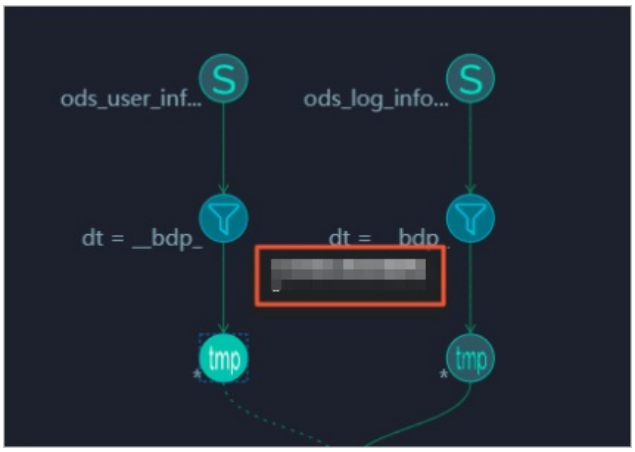
- **Source table**: the table to be queried.



- **Filter**: the condition for filtering the partitions in the table to be queried.

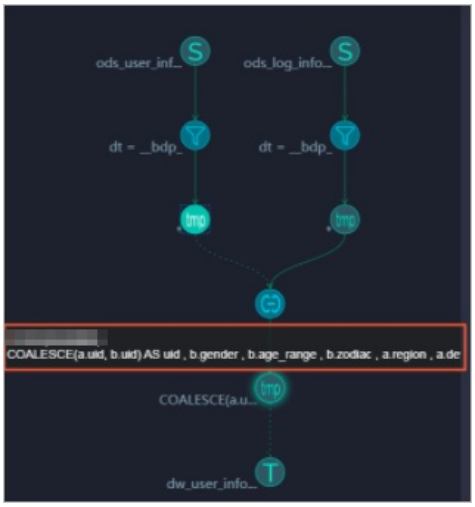


- **First intermediate table (view)**: the temporary table that stores the query results.

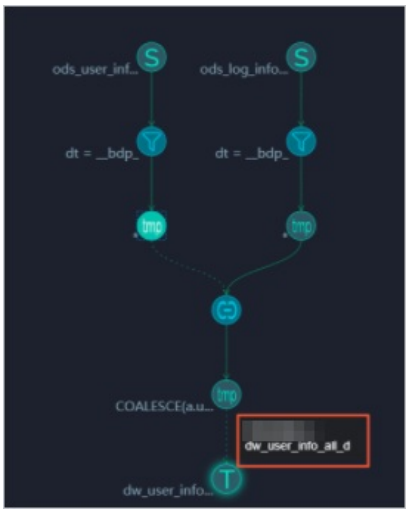


- **Join**: the operation for joining the query results.

- **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.



2.1.5.4. Business flows

2.1.5.4.1. Overview

DataWorks organizes different types of nodes in a workflow by business category. This allows you to develop code by business.

DataWorks provides dashboards for different types of nodes in each workflow and provides tools for you to optimize and manage nodes in each workflow. This implements easy, 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 directed acyclic graph (DAG) for the workflow 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 excessively complex. We recommend that you split the workflow into two or more workflows and add the 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 can be a maximum of 128 characters in length.

5. Click **Create**.

Nodes in a workflow

A workflow consists of the following types of nodes:

- **Data integration**

  Click **Data Integration** in a workflow to view all the data integration nodes in the workflow.

- **MaxCompute**

  The MaxCompute compute engine supports various data analytics nodes, such as MaxCompute SQL, SQL Snippet, MaxCompute Spark, PyODPS, MaxCompute Script, and MaxCompute MR nodes. You can also view and create tables, resources, and functions.

  - **Data analytics**

    Right-click **MaxCompute** in a workflow to create a data analytics node.

  - **Table**

    Right-click **MaxCompute** in a workflow and choose **Create > Table** to create a table. You can also view all the tables that are created in the current MaxCompute project.

  - **Resource**

    Right-click **MaxCompute** in a workflow, choose **Create > Resource**, and then select a specific resource type to create a resource. You can also view all the resources that are created in the current MaxCompute project.

  - **Function**

    Right-click **MaxCompute** in a workflow and choose **Create > Function** to create a function. You can also view all the functions that are created in the current MaxCompute project.

- **AnalyticDB for PostgreSQL**

  You can create AnalyticDB for PostgreSQL nodes and tables.

  > **Note** The AnalyticDB for PostgreSQL folder is displayed on the DataStudio page only after you associate an AnalyticDB for PostgreSQL instance with the current workspace on the **Project Management** page.

- **EMR**

  The E-MapReduce (EMR) compute engine supports various data analytics nodes, such as EMR Hive, EMR MR, EMR Spark, and EMR Spark SQL nodes. You can also view and create EMR resources.

  > **Note** The **EMR** folder is displayed only after you associate an EMR compute engine instance with the current workspace on the **Project Management** page.

- **Data analytics**

  Right-click **Data Analytics** in the **EMR** folder in a workflow and select Create to create a data analytics node of a specific type.

- **Resource**

  Right-click **Resource** in the **EMR** folder in a workflow and select Create to create a resource of a specific type. You can also view all the resources that are created in the current EMR compute engine instance.

- **Algorithm**

  Right-click **Algorithm** in a workflow and 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.

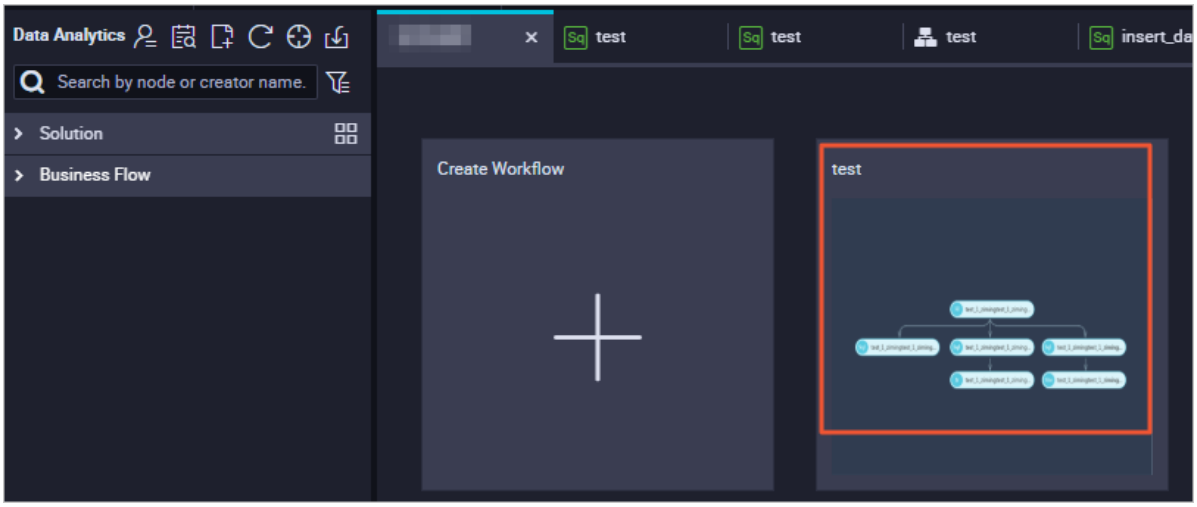
- **General**

  Right-click **General** in a workflow and 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 the workflows that are created in the current workspace.

Click a workflow. The dashboard of the workflow appears.



View the dashboard for each type of node

DataWorks provides a dashboard for each type of node in a workflow. On the dashboard, each node is presented by a card that offers operation and optimization suggestions. This way, 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 the icon in the top toolbar.
2. In the **Commit** dialog box, select the nodes that you want to commit, set **Change description**, and then select **Ignore I/O Inconsistency Alerts**.
3. Click **Commit**.

> **Note** A committed node whose code is not modified 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.

2.1.5.4.2. Create and reference a node group

This topic describes how to create and reference a node group.

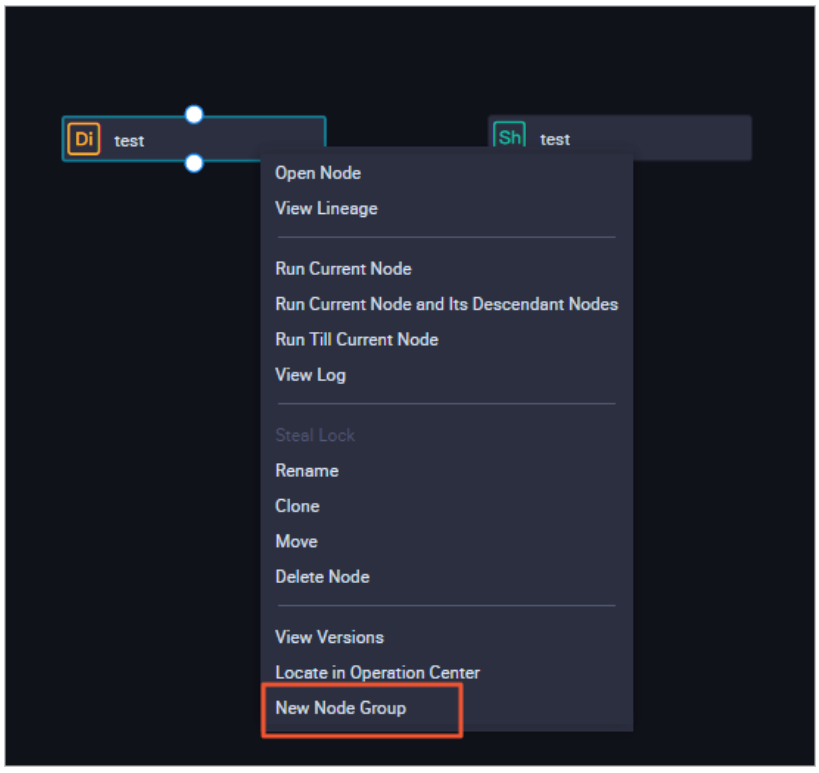
Context

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.

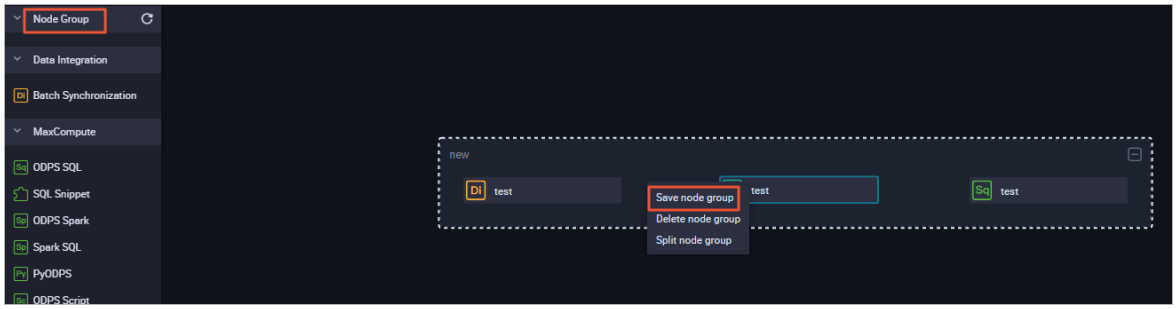
Create a node group

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.

2.1.5.5. Node types

2.1.5.5.1. Data Integration

2.1.5.5.1.1. Create a batch synchronization node

Batch synchronization nodes support various types of data sources, such as MaxCompute, MySQL, PolarDB-X, SQL Server, PostgreSQL, Oracle, MongoDB, Db2, Tablestore, Object Storage Service (OSS), FTP, HBase, LogHub, HDFS, and Stream.

Context

When you enter a table name, a drop-down list appears, displaying all matching tables. Only exact matches are supported. Therefore, you must enter a complete table name. Tables are labeled as unsupported if they are not supported by batch synchronization nodes.

If you move the pointer over a table in the list, the details of the table are displayed, such as the database, IP address, and owner of the table. The details help you select the correct table. After you select and click a table, the column information is automatically entered. You can add, move, or remove columns.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **Data Integration > Batch Synchronization**.

   Alternatively, you can click the desired workflow in the Business Flow section, right-click **Data Integration**, and then choose **Create > Batch Synchronization**.
3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Configure the batch synchronization node. For more information, see Overview.
6. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.

   > **Notice** You can commit the node only after you specify the **Rerun** and **Parent Nodes** parameters.

   i. Click the icon in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Publish** in the upper-right corner after you commit the node. For more information, see Publish nodes.
8. Test the node. For more information, see Manage auto triggered nodes.

2.1.5.5.2. MaxCompute

2.1.5.5.2.1. 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, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

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.

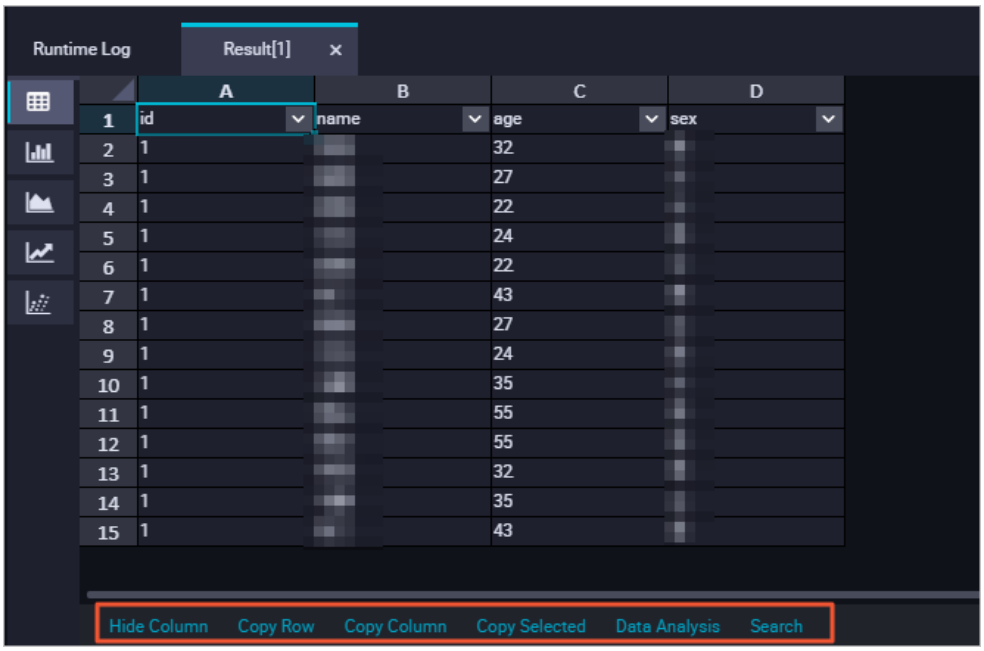
The `INSERT INTO` statement may result in unexpected data duplication. Although DataWorks does not re-execute the `INSERT INTO` statement, it may rerun corresponding nodes. We recommend that you avoid using the `INSERT INTO` statement. When DataWorks executes the INSERT INTO statement, the following information appears in logs:

```
The INSERT INTO statement in SQL may cause repeated data insertion. Although SQL-level retri
es 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 synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure 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 node. For more information, see Manage auto triggered nodes.

2.1.5.5.2.2. 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, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

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 synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
8. Commit the node.

   > **Notice** You can commit the node only after you specify the **Rerun** and **Parent Nodes** parameters.

   i. Click the icon in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Publish** in the upper-right corner after you commit the node. For more information, see Publish nodes.
9. Test the node. For more information, see Manage auto triggered nodes.

2.1.5.5.2.3. 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**.

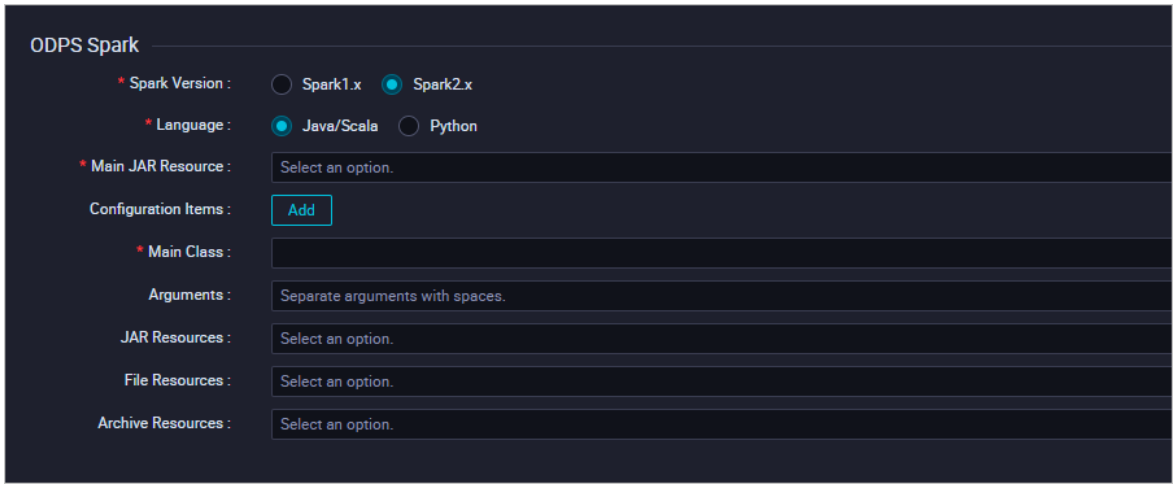
   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.
3. In the **Create Resource** dialog box, set **Resource Name** and **Location**.
4. Click **Upload** and select the target 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, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
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.
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 synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
6. Commit the node.

> **Notice** You can commit the node only after you specify the **Rerun** and **Parent Nodes** parameters.

i. Click the icon in the toolbar.

ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iii. Click **OK**.

In a workspace in standard mode, you must click **Publish** in the upper-right corner after you commit the node. For more information, see Publish nodes.

7. Test the node. For more information, see Manage auto triggered nodes.

2.1.5.5.2.4. Create a PyODPS 2 node

DataWorks supports PyODPS 2 nodes, which are integrated with Alibaba Cloud MaxCompute SDK for Python. You can edit Python code in PyODPS 2 nodes to process data in MaxCompute.

Context

You can also use MaxCompute SDK for Python to process data in MaxCompute.

> **Note**
> - The Python version of PyODPS 2 nodes is V2.7.
> - Each PyODPS 2 node can process a maximum of 50 MB of data and can occupy a maximum of 1 GB of memory. Otherwise, DataWorks terminates the PyODPS 2 node. Avoid writing excessive data processing code for a PyODPS node.

PyODPS 2 nodes are designed to use MaxCompute SDK for Python. 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 2 node

1. Log on to the DataWorks console.
2. On the DataStudio page, move the pointer over the icon and choose **MaxCompute > PyODPS 2**.

   Alternatively, you can find the required workflow, right-click the workflow name, and then choose **Create > MaxCompute > PyODPS 2**.

   > **Notice** The **MaxCompute** folder is displayed on the DataStudio page only after you associate a **MaxCompute** compute engine instance with the current workspace on the **Workspace Management** page.

3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. Configure the PyODPS 2 node on the node configuration tab.

i. Use the MaxCompute entry point.

In DataWorks, each PyODPS 2 node includes the global variable odps or o, which is the MaxCompute entry. Therefore, you do not need to manually specify the MaxCompute entry point.

```
print(odps.exist_table('PyODPS_iris'))
```

ii. Execute SQL statements.

PyODPS 2 nodes allow you to execute MaxCompute SQL statements to query data and obtain query results. If you use the execute_sql or run_sql method, the running instances are returned.

Not all statements that can be executed on the MaxCompute client are supported by PyODPS 2 nodes. To execute statements other than data definition language (DDL) or data manipulation language (DML) statements, you must use alternative methods.

For example, to execute a GRANT or REVOKE statement, use the run_security_query method. To run a Machine Learning Platform for AI (PAI) command, use the run_xflow or execute_xflow method.

```
o.execute_sql('select * from dual') # Execute the statement in synchronous mode. Other nodes
are blocked until the execution of the SQL statement is complete.
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() # Block other nodes until the execution of the SQL statement is
complete.
```

iii. Configure runtime parameters.

You can use the hints parameter to configure 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 specify the sql.settings parameter for the global configuration, you must configure 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 configured
based on 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 2 nodes do not support relative time variables.

6. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.

7. Commit the node.

   > **Notice** You can commit the node only after you specify the **Rerun** and **Parent Nodes** parameters.

   i. Click the icon in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Publish** in the upper-right corner after you commit the node. For more information, see Publish nodes.

8. Test the node. For more information, see Manage auto triggered nodes.

Built-in services of PyODPS 2 nodes

A PyODPS 2 node contains the following built-in services:

- setuptools
- cython
- psutil
- pytz
- dateutil
- requests
- pyDes
- numpy
- pandas
- scipy
- scikit_learn
- greenlet
- six
- Other built-in services in Python 2.7, such as smtplib

2.1.5.5.2.5. 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, specify **Node Name** and **Location**.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. Edit the SQL script of the ODPS Script node as required.
6. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.

> **Notice** You can commit the node only after you specify the **Rerun** and **Parent Nodes** parameters.

   i. Click the icon in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iii. Click **OK**.

   In a workspace in standard mode, you must click **Publish** in the upper-right corner after you commit the node. For more information, see Publish nodes.

8. Test the node. For more information, see Manage auto triggered nodes.

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.

2.1.5.5.2.6. 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, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

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 curren
t workspace 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-
click 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.mapred
.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 synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
6. Commit the node.

   > **Notice** You can commit the node only after you specify the **Rerun** and **Parent Nodes** parameters.

   i. Click the icon in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Publish** in the upper-right corner after you commit the node. For more information, see Publish nodes.
7. Test the node. For more information, see Manage auto triggered nodes.

2.1.5.5.2.7. 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.
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.

Parameter or button	Description
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.
	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**.
	Field Name	The name of the field. The name can contain letters, digits, and underscores (_).
	Display Name	The display name of the field.

Section	Parameter or button	Description
Schema	**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.
	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	**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 *yyyymmmdd* or *yyyy-mm-dd*.

Section	Parameter or button	Description
Note This section is available only when Partitioning under Physical Model is set to Partitioned Table.	**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
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.

2.1.5.5.2.8. 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**.

2.1.5.5.2.9. 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 the save icon in the toolbar.
6. Commit the function.
   i. Click the commit icon in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.
   iii. Click **OK**.

2.1.5.5.3. EMR

2.1.5.5.3.1. Modes for associating an EMR cluster with a DataWorks workspace

After you associate an EMR cluster with a DataWorks workspace, you can create nodes such as EMR Hive, EMR MR, EMR Presto, and EMR Spark SQL nodes based on an EMR compute engine and configure EMR workflows. You can also schedule the nodes and manage metadata. This improves your data output.

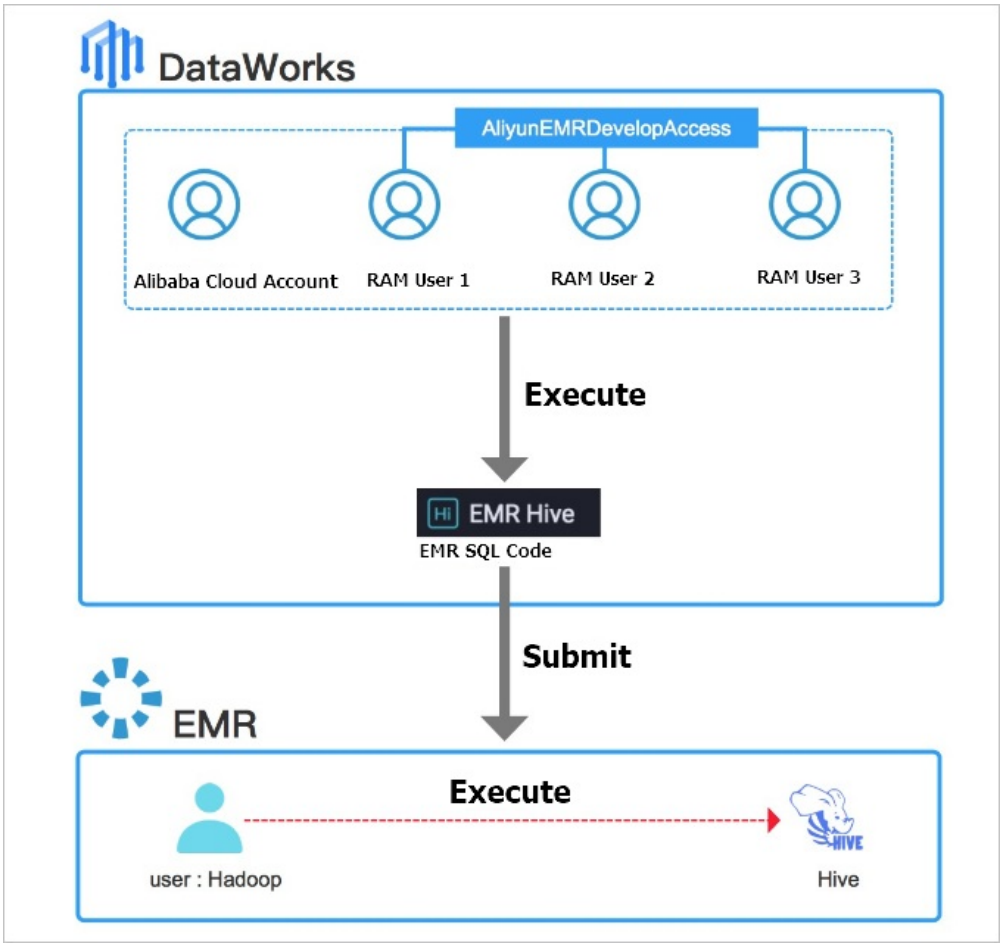
DataWorks provides two modes for you to associate an EMR cluster with a workspace: **Shortcut mode** and **Security mode**. The two modes can meet the security requirements of various enterprises. If you associate an EMR cluster with a workspace by using the **Shortcut mode**, you can create and run EMR nodes to generate data. If you associate an EMR cluster with a workspace by using the **Security mode**, you can create and run EMR nodes to generate data and manage permissions on the data to ensure higher security.

Shortcut mode

In **Shortcut mode**, if you run or schedule EMR nodes in DataWorks by using your Apsara Stack tenant account or as a RAM user, the code is committed to the EMR cluster and run by the Hadoop user of the EMR cluster.

> **Notice**
>
> - The Hadoop user has all the permissions on the EMR cluster. Proceed with caution when you use the Shortcut mode to associate an EMR cluster with a workspace.
> - Before you use the **Shortcut mode** to associate an EMR cluster with a workspace, you must attach the AliyunEMRDevelopAccess policy to workspace roles such as developers and administrators. This way, the roles can be used to create and run EMR nodes in DataStudio.
>   - The AliyunEMRDevelopAccess policy is attached to Apsara Stack tenant accounts by default.
>   - To run EMR nodes as a RAM user, you must attach the AliyunEMRDevelopAccess policy to the RAM user.

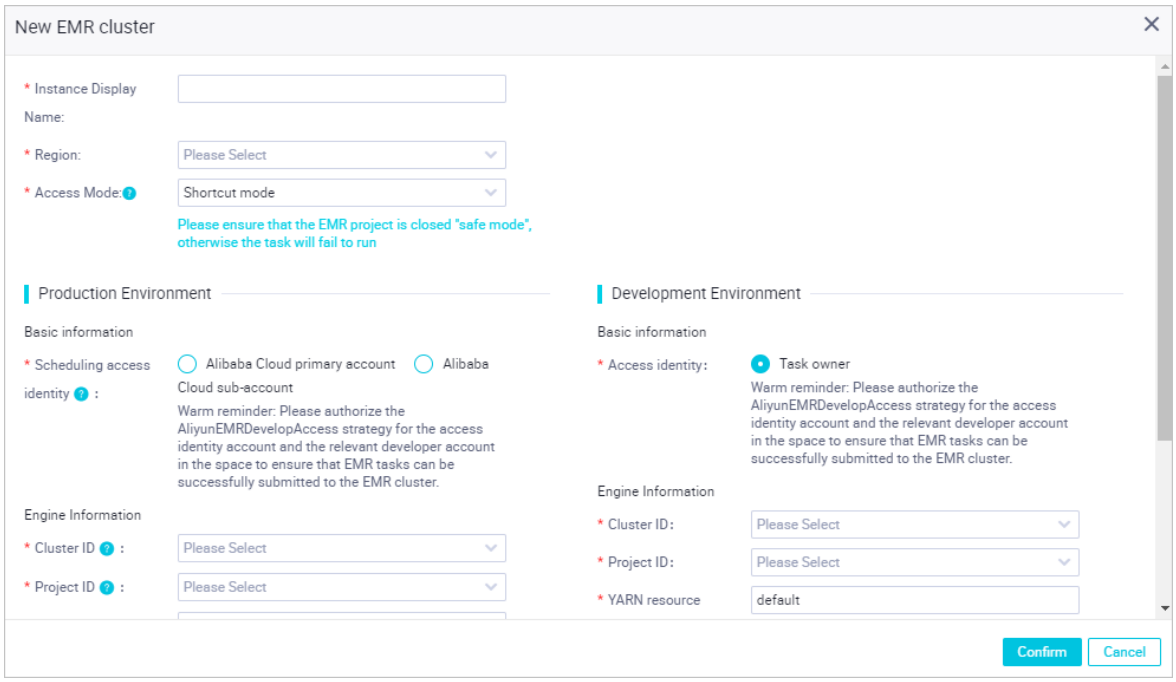
The **Shortcut mode** is suitable for workspaces that do not require strict permission management for users who run nodes.



To associate an EMR cluster with a workspace in **Shortcut mode**, perform the following steps:

1. Log on to the DataWorks console.
2. In the upper-right corner of the page, click the icon to go to the Workspace Management page.
3. In the **Compute Engine Information** section, click the **E-MapReduce** tab.
4. On the **E-MapReduce** tab, click **Add Instance**.
5. In the **New EMR cluster** dialog box, set the parameters.

Parameters in the New EMR cluster dialog box vary based on the mode in which your DataWorks workspace runs. The following table describes the parameters for a DataWorks workspace in standard mode. You must set the parameters for both the production environment and the development environment.



Parameter	Description
Instance Display Name	The display name of the EMR compute engine instance.
Region	The region of the current workspace.
Access Mode	The access mode of the EMR cluster. Select **Shortcut mode** from the drop-down list.

Scheduling access identity	The identity that is used to commit the code of an EMR node to the EMR cluster. The code is committed when the node is committed to the scheduling system of DataWorks in the production environment. Valid values: **Alibaba Cloud primary account** and **Alibaba Cloud sub-account**.
	Note
	- This parameter is available only for the production environment.
	- Before you use the **Shortcut mode** to associate an EMR cluster with a workspace, you must attach the AliyunEMRDevelopAccess policy to workspace roles such as developers and administrators. This way, the roles can be used to create and run EMR nodes in DataStudio.
	  - The AliyunEMRDevelopAccess policy is attached to Apsara Stack tenant accounts by default.
	  - To run EMR nodes as a RAM user, you must attach the AliyunEMRDevelopAccess policy to the RAM user.

Parameter	Description
Access identity	The identity that is used to commit the code of an EMR node in the development environment to the EMR cluster. Default value: **Task owner**. **Note** ◦ This parameter is available only for the development environment of a workspace in standard mode. ◦ **Task owner** can be an Apsara Stack tenant account or a RAM user. Before you use the **Shortcut mode** to associate an EMR cluster with a workspace, you must attach the AliyunEMRDevelopAccess policy to workspace roles such as developers and administrators. This way, the roles can be used to create and run EMR nodes in DataStudio. ■ The AliyunEMRDevelopAccess policy is attached to Apsara Stack tenant accounts by default. ■ To run EMR nodes as a RAM user, you must attach the AliyunEMRDevelopAccess policy to the RAM user.
Cluster ID	The ID of the EMR cluster that you want to associate with the workspace. Select an ID from the drop-down list. The EMR cluster with the selected ID is used as the runtime environment of EMR nodes.
Project ID	The ID of the EMR project that you want to associate with the workspace. Select an ID from the drop-down list. The selected EMR project with the selected ID is used as the runtime environment of EMR nodes. **Note** The IDs of the EMR projects in **Security mode** are not displayed and cannot be selected.
YARN resource queue	The name of the resource queue in the EMR cluster. Unless otherwise specified, set this parameter to *default*.
Endpoint	The endpoint of the EMR cluster. Unless otherwise specified, set this parameter to *default*.

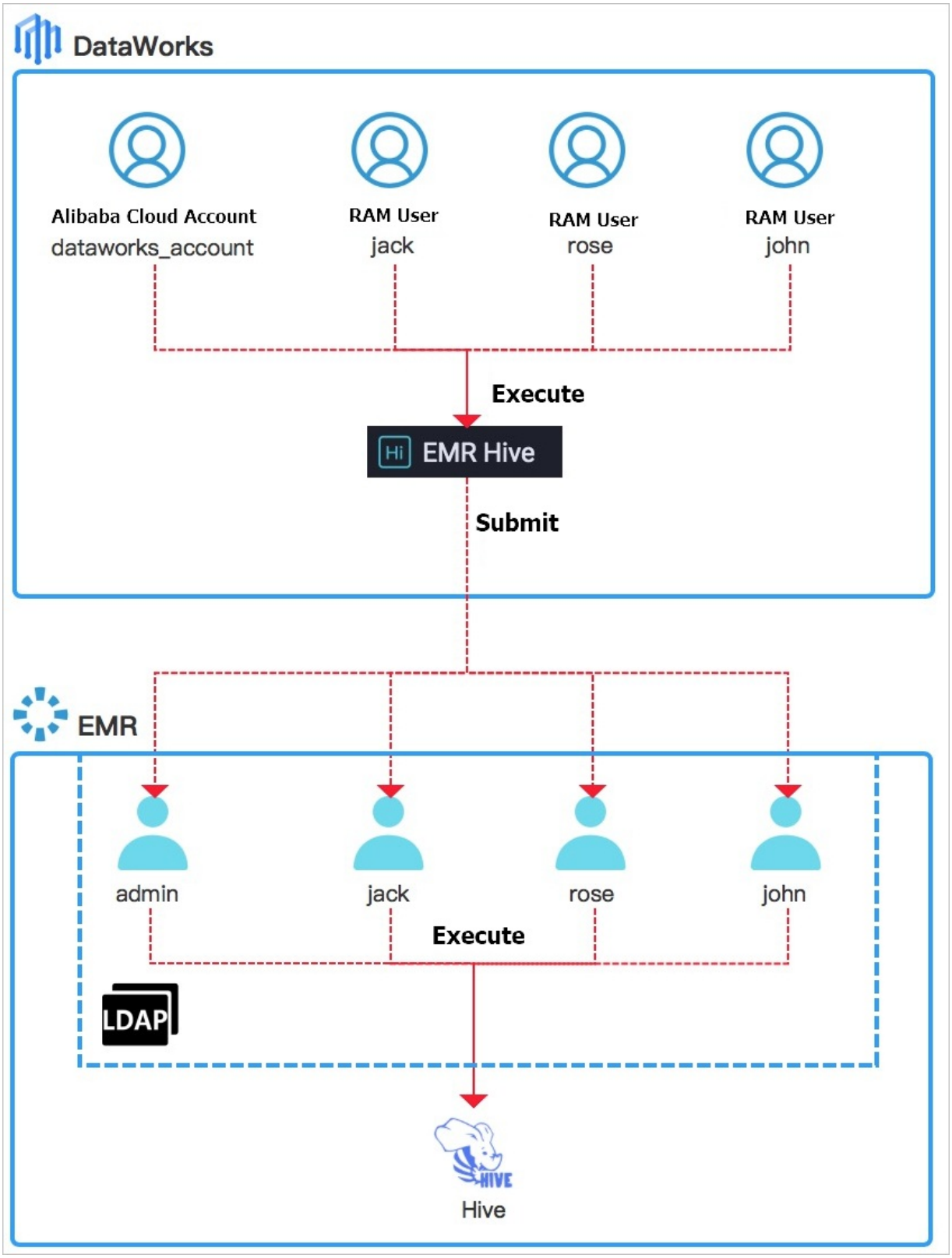
6. Click **Confirm**.

Security mode

In **Security mode**, if you commit the code of EMR nodes by using an Apsara Stack tenant account or as a RAM user to an EMR cluster, the code is run by a user that has the same name as the Apsara Stack tenant account or RAM user. EMR Ranger can be used to manage the permissions of each Hadoop user in the EMR cluster. This ensures that different Apsara Stack tenant accounts, node owners, or RAM users have different data permissions when they run EMR nodes in DataWorks. This provides higher data security.

> **Note**
>
> Before you use the **Security mode** to associate an EMR cluster with a workspace, you must add the credentials of workspace roles such as developers and administrators to the Lightweight Directory Access Protocol (LDAP) directory of the EMR cluster. In addition, you must attach the AliyunEMRDevelopAccess or AliyunEMRFullAccess policy and grant relevant data permissions to the workspace roles. This way, the roles can be used to create and run EMR nodes in DataStudio.
>
> - The credentials of Apsara Stack tenant accounts are in the LDAP directory of the EMR cluster by default. The AliyunEMRDevelopAccess and AliyunEMRFullAccess policies are also attached to Apsara Stack tenant accounts by default.
> - To run EMR nodes as a RAM user, you must add the credential of the RAM user to the LDAP directory of the EMR cluster. For more information, see the *Add the credentials of specific RAM users to the LDAP directory of the EMR cluster* step. In addition, you must attach the AliyunEMRDevelopAccess or AliyunEMRFullAccess policy to the RAM user.

The **Security mode** is suitable for workspaces that require strict management and isolation of data permissions for users who run nodes.



To use the **Security mode** to associate an EMR cluster with a workspace, perform the following steps:

1. Turn on Security Mode for the EMR project.

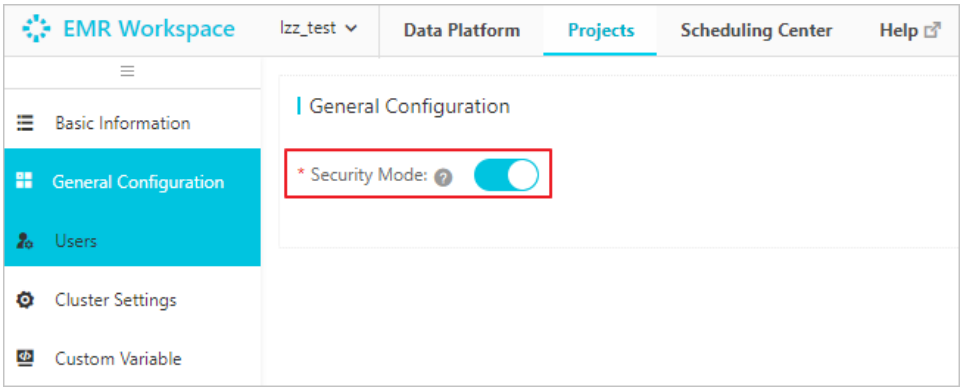
i. Log on to the EMR console.

ii. In the top navigation bar, click **Data Platform**.

iii. In the **Projects** section, find the project for which you want to enable the Security mode and click **Edit Job** in the Actions column.

iv. On the page that appears, click the **Projects** tab in the top navigation bar.

v. In the left-side navigation pane, click **General Configuration**. On the General Configuration page, turn on **Security Mode**.



2. Add the credentials of specific RAM users to the LDAP directory of the EMR cluster.

i. Go back to the homepage of the EMR console. In the top navigation bar, click **Cluster Management**.

ii. Find the cluster that you want to manage and click **Details** in the Actions column.

iii. In the left-side navigation pane, click **Users**.

iv. On the **Users** page, click **Add User**.

v. In the **Add User** dialog box, set the parameters.

We recommend that you add the credentials of the following RAM users to the LDAP directory of the EMR cluster:

- RAM users that create, test, and run EMR nodes in DataStudio
- RAM users that create, commit, and deploy EMR nodes in DataStudio

vi. Click **OK**.

3. Configure EMR Ranger and manage the permissions of the Hadoop users that correspond to your Apsara Stack tenant account and RAM users.

4. Associate the EMR cluster with the current DataWorks workspace.

i. Log on to the DataWorks console.

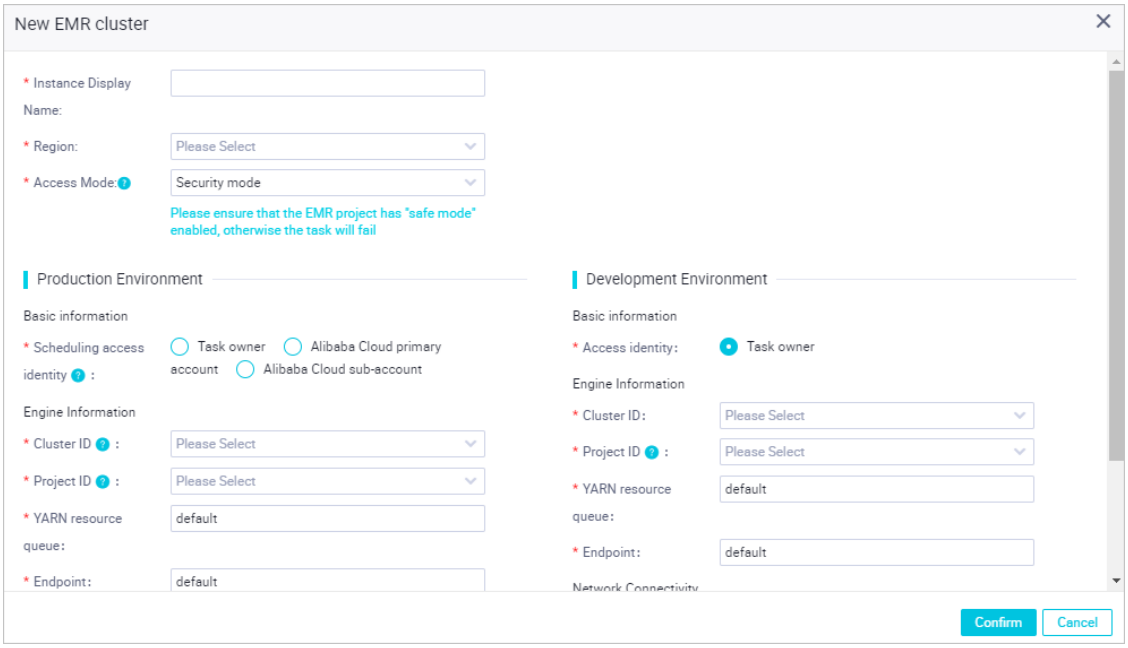
ii. In the upper-right corner of the page, click the icon to go to the Workspace Management page.

iii. In the **Compute Engine Information** section, click the **E-MapReduce** tab.

iv. On the **E-MapReduce** tab, click **Add Instance**.

v. In the **New EMR cluster** dialog box, set the parameters.

Parameters in the New EMR cluster dialog box vary based on the mode in which your DataWorks workspace runs. The following table describes the parameters for a DataWorks workspace in standard mode. You must set the parameters for both the production environment and the development environment.



Parameter	Description
Instance Display Name	The display name of the EMR compute engine instance.
Region	The region of the current workspace.
Access Mode	The access mode of the EMR cluster. Select **Security mode** from the drop-down list and click **Confirm** in the **Please note** message. **Note** You cannot use multiple modes to associate an EMR cluster with a DataWorks workspace at the same time. Proceed with caution when you change the access mode of the EMR cluster because a mode change leads to permission changes.

Parameter	Description
Scheduling access identity	The identity that is used to commit the code of an EMR node to the EMR cluster. The code is committed when the node is committed and deployed to the DataWorks scheduling system in the production environment. The Hadoop user that corresponds to this identity runs the code. Valid values: **Task owner**, **Alibaba Cloud primary account**, and **Alibaba Cloud sub-account**. ■ **Task owner**: commits and runs the code of an EMR node as the node owner. If you select this value, the data permissions of Hadoop users are isolated. **Task owner** can be an Apsara Stack tenant account or a RAM user. ■ **Alibaba Cloud primary account**: commits the code of an EMR node to the EMR cluster by using an Apsara Stack tenant account. ■ **Alibaba Cloud sub-account**: commits the code of an EMR node to the EMR cluster as a RAM user. **Note** ■ This parameter is available only for the production environment. ■ The credentials of Apsara Stack tenant accounts are in the LDAP directory of the EMR cluster by default. The AliyunEMRDevelopAccess and AliyunEMRFullAccess policies are also attached to Apsara Stack tenant accounts by default. ■ To run EMR nodes as a RAM user, you must add the credential of the RAM user to the LDAP directory of the EMR cluster. For more information, see the *Add the credentials of specific RAM users to the LDAP directory of the EMR cluster* step. In addition, you must attach the AliyunEMRDevelopAccess or AliyunEMRFullAccess policy to the RAM user.

Parameter	Description
Access identity	The identity that is used to commit the code of an EMR node in the development environment to the EMR cluster. Default value: **Task owner**. The Hadoop user that corresponds to the user who runs the node runs the code. **Note** ■ This parameter is available only for the development environment of a workspace in standard mode. ■ Make sure that the credential of the user who runs the node is added to the LDAP directory of the EMR cluster. In addition, make sure that the AliyunEMRDevelopAccess or AliyunEMRFulAccess policy is attached to the user and relevant data permissions are granted to the user. This way, the user can run EMR nodes in DataStudio. **Task owner** can be an Apsara Stack tenant account or a RAM user. ■ The credentials of Apsara Stack tenant accounts are in the LDAP directory of the EMR cluster by default. The AliyunEMRDevelopAccess and AliyunEMRFullAccess policies are also attached to Apsara Stack tenant accounts by default. ■ To run EMR nodes as a RAM user, you must add the credential of the RAM user to the LDAP directory of the EMR cluster. For more information, see the *Add the credentials of specific RAM users to the LDAP directory of the EMR cluster* step. In addition, you must attach the AliyunEMRDevelopAccess or AliyunEMRFullAccess policy to the RAM user.
Cluster ID	The ID of the EMR cluster that you want to associate with the workspace. Select an ID from the drop-down list. The EMR cluster with the selected ID is used as the runtime environment of EMR nodes.
Project ID	The ID of the EMR project that you want to associate with the workspace. Select the ID of an EMR project in Security mode from the drop-down list. **Note** The IDs of the EMR projects that are not in **Security mode** are not displayed and cannot be selected.
YARN resource queue	The name of the resource queue in the EMR cluster. Unless otherwise specified, set this parameter to *default*.
Endpoint	The endpoint of the EMR cluster. Unless otherwise specified, set this parameter to *default*.

vi. Click **Confirm**.

2.1.5.5.3.2. 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, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
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 synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure 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 node. For more information, see Manage auto triggered nodes.

2.1.5.5.3.3. 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, specify **Node Name** and **Location**.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

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 synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure 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 node. For more information, see Manage auto triggered nodes.

2.1.5.5.3.4. 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, specify **Node Name** and **Location**.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

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 synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure 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 node. For more information, see Manage auto triggered nodes.

2.1.5.5.3.5. 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, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
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 synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure 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 node. For more information, see Manage auto triggered nodes.

2.1.5.5.3.6. Create and use an EMR Shell node

You can create an EMR Shell node and run the node by using the code editor.

Prerequisites

- An EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the EMR cluster belongs:
  - Action: Allow
  - Protocol type: Custom TCP

- Port range: 8898/8898
- Authorization object: 100.104.0.0/16

- An EMR compute engine instance is associated with the current workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Associate an EMR cluster with a workspace.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned when you run EMR nodes.
  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

  ```
  hive.security.authorization.sqlstd.confwhitelist.append=tez.*|spark.*|mapred.*|mapreduce.*|ALISA.*|SKYNET.*
  ```

  > **Note** In the code, `ALISA.*` and `SKYNET.*` are configurations in DataWorks.

  ii. After the whitelist configurations are modified, restart the Hive service to make the configurations take effect.

Create an EMR Shell node and use the node to develop data

1. Log on to the DataWorks console.
2. Create an **EMR Shell** node.
   i. On the DataStudio page, move the pointer over the icon and choose **EMR > EMR Shell**.

   You can also find the workflow in which you want to create the EMR Shell node, right-click the workflow name, and then choose **Create > EMR > EMR Shell**.

   ii. In the **Create Node** dialog box, set the **Node Name**, **Node Type**, and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

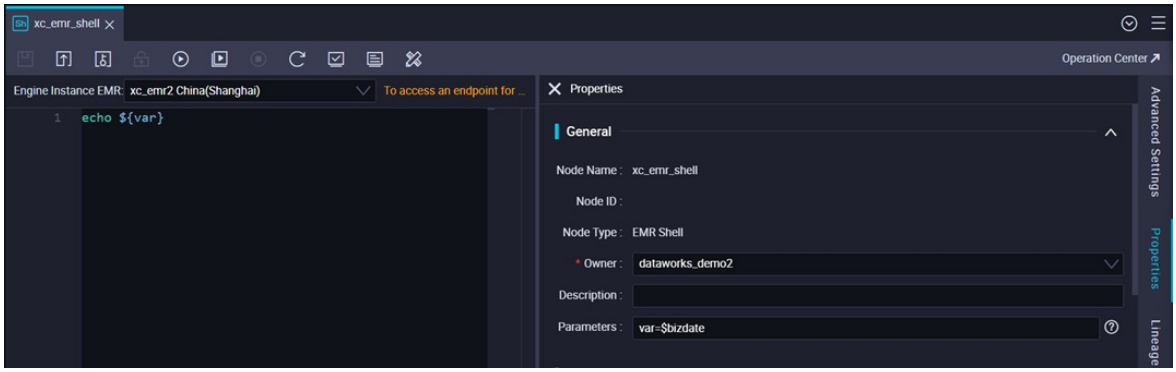
   iii. Click **Commit**. Then, the configuration tab of the **EMR Shell** node appears.
3. Use the **EMR Shell** node to develop data.

   The following code provides an example:

   ```
   DD=`date`;
   echo "hello world, $DD"
   ## Scheduling parameters are supported.
   echo ${var};
   ```

   For more information about the scheduling parameters, see Scheduling parameters.

   If you want to change the values that are assigned to the parameters in the code, click the Run with Parameters icon in the top toolbar.



4. Click the **Advanced Settings** tab in the right-side navigation pane. In the Advanced Settings panel, change the values of the parameters.
   - "USE_GATEWAY":true: If you set this parameter to true, the EMR Shell node is automatically committed to the master node of an EMR gateway cluster.
   - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters that are required to run Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
   - "queue": the scheduling queue to which jobs are committed. Default value: default.
   - "vcores": the number of CPU cores. Default value:1.
   - "memory": the memory that is allocated to the launcher. Unit: MB. Default value: 2048.
   - "priority": the priority. Default value: 1.
   - "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. A value of false indicates that only one SQL statement is executed at a time. A value of true indicates that multiple SQL statements are executed at a time.
5. Configure scheduling properties for the EMR Shell node.

   If you want the system to periodically run the EMR Shell node, you can click the **Properties** tab in the right-side navigation pane to configure scheduling properties for the node based on your business requirements.
6. Commit and deploy the EMR Shell node.
   i. Click the icon in the top toolbar to save the node.
   ii. Click the icon in the top toolbar to commit the node.
   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iv. Click **OK**.
7. View the EMR Shell node.
   i. On the configuration tab of the EMR Shell node, click **Operation Center** in the upper-right corner to go to Operation Center.
   ii. View the scheduled EMR Shell node. For more information, see Manage auto triggered nodes.

2.1.5.5.3.7. Create and use an EMR Spark Shell node

You can create an EMR Spark Shell node and run the node by using the code editor.

Prerequisites

- An EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the EMR cluster belongs:
  - Action: Allow
  - Protocol type: Custom TCP

- Port range: 8898/8898
- Authorization object: 100.104.0.0/16

- An EMR compute engine instance is associated with the current workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Associate an EMR cluster with a workspace.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned if you run EMR nodes.
  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

  ```
  hive.security.authorization.sqlstd.confwhitelist.append=tez.*|spark.*|mapred.*|mapreduce.*|ALISA.*|SKYNET.*
  ```

  > **Note** In the code, `ALISA.*` and `SKYNET.*` are configurations in DataWorks.

  ii. After the whitelist configurations are modified, restart the Hive service to make the configurations take effect.

Create an EMR Spark Shell node and use the node to develop data

1. Log on to the DataWorks console.
2. Create an **EMR Spark Shell** node.
   i. On the DataStudio page, move the pointer over the icon and choose **EMR > EMR Spark Shell**.

   You can also find the workflow in which you want to create the EMR Spark Shell node, right-click the workflow name, and then choose **Create > EMR > EMR Spark Shell**.

   ii. In the **Create Node** dialog box, set the **Node Name**, **Node Type**, and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

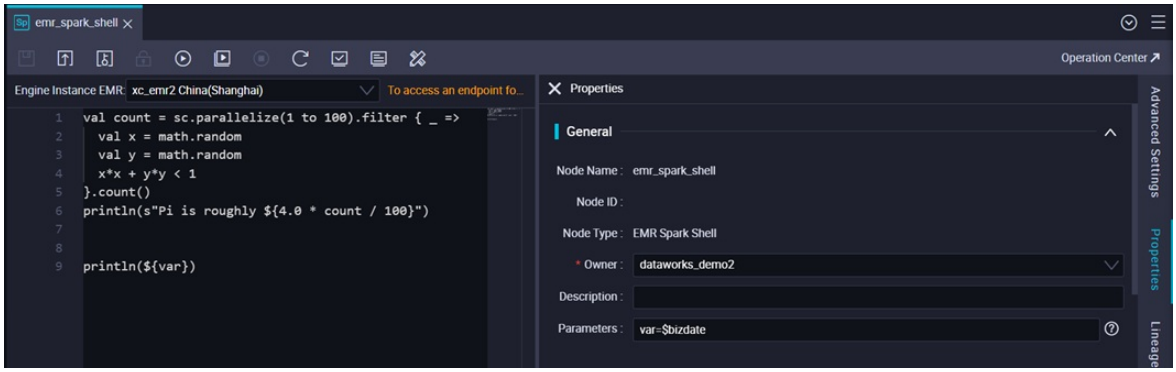
   iii. Click **Commit**. Then, the configuration tab of the **EMR Spark Shell** node appears.
3. Use the **EMR Spark Shell** node to develop data.

   The following code provides an example:

   ```
   val count = sc.parallelize(1 to 100).filter { _ =>
     val x = math.random
     val y = math.random
     x*x + y*y < 1
   }.count()
   println(s"Pi is roughly ${4.0 * count / 100}")
   println(${var})
   ```

   Scheduling parameters are supported. For more information, see Scheduling parameters

   If you want to change the values that are assigned to the parameters in the code, click the Run with Parameters icon in the top toolbar.



4. Click the **Advanced Settings** tab in the right-side navigation pane. In the Advanced Settings panel, change the values of the parameters.
   - "USE_GATEWAY":true: If you set this parameter to true, the EMR Spark Shell node is automatically committed to the master node of an EMR gateway cluster.
   - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters that are required to run Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
   - "queue": the scheduling queue to which jobs are committed. Default value: default.
   - "vcores": the number of CPU cores. Default value:1.
   - "memory": the memory that is allocated to the launcher. Unit: MB. Default value: 2048.
   - "priority": the priority. Default value: 1.
   - "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. A value of false indicates that only one SQL statement is executed at a time. A value of true indicates that multiple SQL statements are executed at a time.
5. Configure scheduling properties for the EMR Spark Shell node.

   If you want the system to periodically run the EMR Spark Shell node, you can click the **Properties** tab in the right-side navigation pane to configure scheduling properties for the node based on your business requirements.
6. Commit and deploy the EMR Spark Shell node.
   i. Click the icon in the top toolbar to save the node.
   ii. Click the icon in the top toolbar to commit the node.
   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iv. Click **OK**.
7. View the EMR Spark Shell node.
   i. On the configuration tab of the EMR Spark Shell node, click **Operation Center** in the upper-right corner to go to Operation Center.
   ii. View the scheduled EMR Spark Shell node. For more information, see Manage auto triggered nodes.

2.1.5.5.3.8. Create an EMR Impala node

This topic describes how to create an EMR Impala node. EMR Impala nodes allow you to perform interactive analysis and queries by executing SQL statements on petabytes of data.

Prerequisites

- An EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the EMR cluster belongs:
  - Action: Allow

- Protocol type: Custom TCP
- Port range: 8898/8898
- Authorization object: 100.104.0.0/16

- An EMR compute engine instance is associated with the current workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Associate an EMR cluster with a workspace.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned if you run EMR nodes.
  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

```
hive.security.authorization.sqlstd.confwhitelist.append=tez.*|spark.*|mapred.*|mapreduce.*|ALISA.*|SKYNET.*
```

  > **Note** In the code, `ALISA.*` and `SKYNET.*` are configurations in DataWorks.

  ii. After the whitelist configurations are modified, restart the Hive service to make the configurations take effect.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, move the pointer over the icon and choose **EMR > EMR Impala**.

   You can also find the required workflow, right-click the workflow name, and then choose **Create > EMR > EMR Impala**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**. Then, the configuration tab of the **EMR Impala** node appears.
5. On the node configuration tab, enter the code for the node.

   Sample code:

```
-- SQL statement example
-- The size of SQL statements cannot exceed 130 KB.
show tables;
-- Scheduling parameters are supported.
CREATE TABLE IF NOT EXISTS userinfo (
ip STRING COMMENT'IP address',
uid STRING COMMENT'User ID'
)PARTITIONED BY(
dt STRING
);
ALTER TABLE userinfo ADD IF NOT EXISTS PARTITION(dt=${bizdate});
-- The system automatically adds limit 10000 to the SELECT statement.
select * from userinfo ;
```

   For more information about the scheduling parameters, see Scheduling parameters.

If you want to change the values that are assigned to the parameters in the code, click the Run with Parameters icon in the top toolbar.

> **Note** If multiple EMR compute engine instances are associated with the current workspace, you must select one EMR compute engine instance. If only one EMR compute engine instance is associated with the current workspace, you do not need to make a choice.

6. Click the **Advanced Settings** tab in the right-side navigation pane. In the Advanced Settings panel, change the values of the parameters.
   - "USE_GATEWAY":true: If you set this parameter to true, the EMR Impala node is automatically committed to the master node of an EMR gateway cluster.
   - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters that are required to run Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
   - "queue": the scheduling queue to which jobs are committed. Default value: default.
   - "vcores": the number of CPU cores. Default value:1.
   - "memory": the memory that is allocated to the launcher. Unit: MB. Default value: 2048.
   - "priority": the priority. Default value: 1.
   - "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. A value of false indicates that only one SQL statement is executed at a time. A value of true indicates that multiple SQL statements are executed at a time.
7. Configure scheduling properties for the EMR Impala node.

   If you want the system to periodically run the EMR Impala node, you can click the **Properties** tab in the right-side navigation pane to configure scheduling properties for the node based on your business requirements.
8. Commit and deploy the EMR Impala node.
   i. Click the icon in the top toolbar to save the node.
   ii. Click the icon in the top toolbar to commit the node.
   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iv. Click **OK**.
9. View the EMR Impala node.
   i. On the configuration tab of the EMR Impala node, click **Operation Center** in the upper-right corner to go to Operation Center.
   ii. View the scheduled EMR Impala node. For more information, see Manage auto triggered nodes.

2.1.5.5.3.9. Create and use an EMR Presto node

This topic describes how to create an EMR Presto node in DataWorks. EMR Presto nodes allow you to perform interactive analysis and queries on large amounts of structured and unstructured data.

Prerequisites

- An EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the EMR cluster belongs:
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with the current workspace. The EMR folder is displayed only

after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Associate an EMR cluster with a workspace.

- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned if you run EMR nodes.
  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

  ```
  hive.security.authorization.sqlstd.confwhitelist.append=tez.*|spark.*|mapred.*|mapreduce.*|ALISA.*|SKYNET.*
  ```

  > **Note** In the code, `ALISA.*` and `SKYNET.*` are configurations in DataWorks.

  ii. After the whitelist configurations are modified, restart the Hive service to make the configurations take effect.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, move the pointer over the icon and choose **EMR > EMR Presto**.

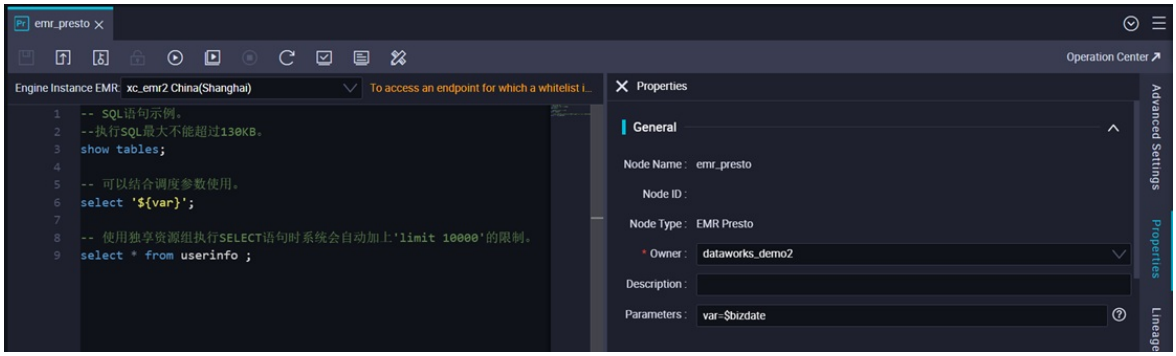
   You can also find the workflow in which you want to create the EMR Presto node, right-click the workflow name, and then choose **Create > EMR > EMR Presto**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**. Then, the configuration tab of the **EMR Presto** node appears.
5. On the node configuration tab, write code for the node.

   ```
   -- SQL statement example
   -- The size of SQL statements cannot exceed 130 KB.
   show tables;
   -- Scheduling parameters are supported.
   select '${var}';
   -- The system automatically adds limit 10000 to the SELECT statement.
   select * from userinfo ;
   ```

   For more information about the scheduling parameters, see Scheduling parameters.

   If you want to change the values that are assigned to the parameters in the code, click the Run with Parameters icon in the top toolbar.



> **Note** If multiple EMR compute engine instances are associated with the current workspace, you must select one EMR compute engine instance. If only one EMR compute engine instance is associated with the current workspace, you do not need to make a choice.

6. Click the **Advanced Settings** tab in the right-side navigation pane. In the Advanced Settings panel, change the values of the parameters.
   - "USE_GATEWAY":true: If you set this parameter to true, the EMR Presto node is automatically committed to the master node of an EMR gateway cluster.
   - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters that are required to run Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
   - "queue": the scheduling queue to which jobs are committed. Default value: default.
   - "vcores": the number of CPU cores. Default value:1.
   - "memory": the memory that is allocated to the launcher. Unit: MB. Default value: 2048.
   - "priority": the priority. Default value: 1.
   - "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. A value of false indicates that only one SQL statement is executed at a time. A value of true indicates that multiple SQL statements are executed at a time.
7. Configure scheduling properties for the EMR Presto node.

   If you want the system to periodically run the EMR Presto node, you can click the **Properties** tab in the right-side navigation pane to configure scheduling properties for the node based on your business requirements.
8. Commit and deploy the EMR Presto node.
   i. Click the icon in the top toolbar to save the node.
   ii. Click the icon in the top toolbar to commit the node.
   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iv. Click **OK**.
9. View the EMR Presto node.
   i. On the configuration tab of the EMR Spark Shell node, click **Operation Center** in the upper-right corner to go to Operation Center.
   ii. View the scheduled EMR Presto node. For more information, see Manage auto triggered nodes.

2.1.5.5.3.10. Create and use an EMR JAR resource

DataWorks allows you to create EMR JAR resources in the DataWorks console. You can upload a Java Archive (JAR) file that contains user-defined functions (UDFs) or open source MapReduce code as an EMR JAR resource. Then, you can reference the resource in compute nodes such as an EMR MR node. This topic describes how to create an EMR JAR resource by uploading a file, commit the resource, and reference the resource in compute nodes such as an EMR MR node.

Prerequisites

- An EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the EMR cluster belongs:
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16

- An EMR compute engine instance is associated with the current workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned if you run EMR nodes.
  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

  ```
  hive.security.authorization.sqlstd.confwhitelist.append=tez.*|spark.*|mapred.*|mapreduce.*|ALISA.*|SKYNET.*
  ```

  > **Note** In the code, `ALISA.*` and `SKYNET.*` are configurations in DataWorks.

  ii. After the whitelist configurations are modified, restart the Hive service to make the configurations take effect.

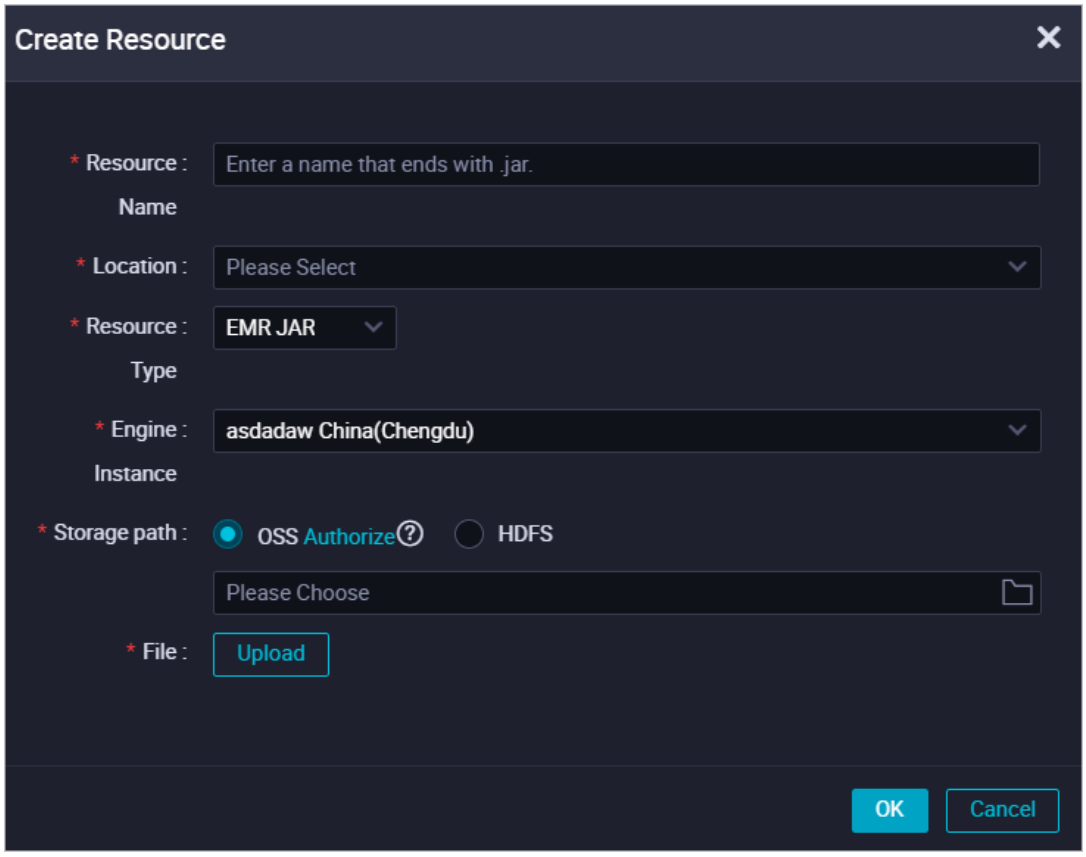
Limits

If Kerberos authentication is enabled for an EMR cluster., you cannot create tables, resources, and functions in a visualized manner for this cluster.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, move the pointer over the icon and choose **EMR > Resource > EMR JAR**.

   You can also find the required workflow, right-click the workflow name, and then choose **Create > EMR > Resource > EMR JAR**.
3. In the **Create Resource** dialog box, set the following parameters.

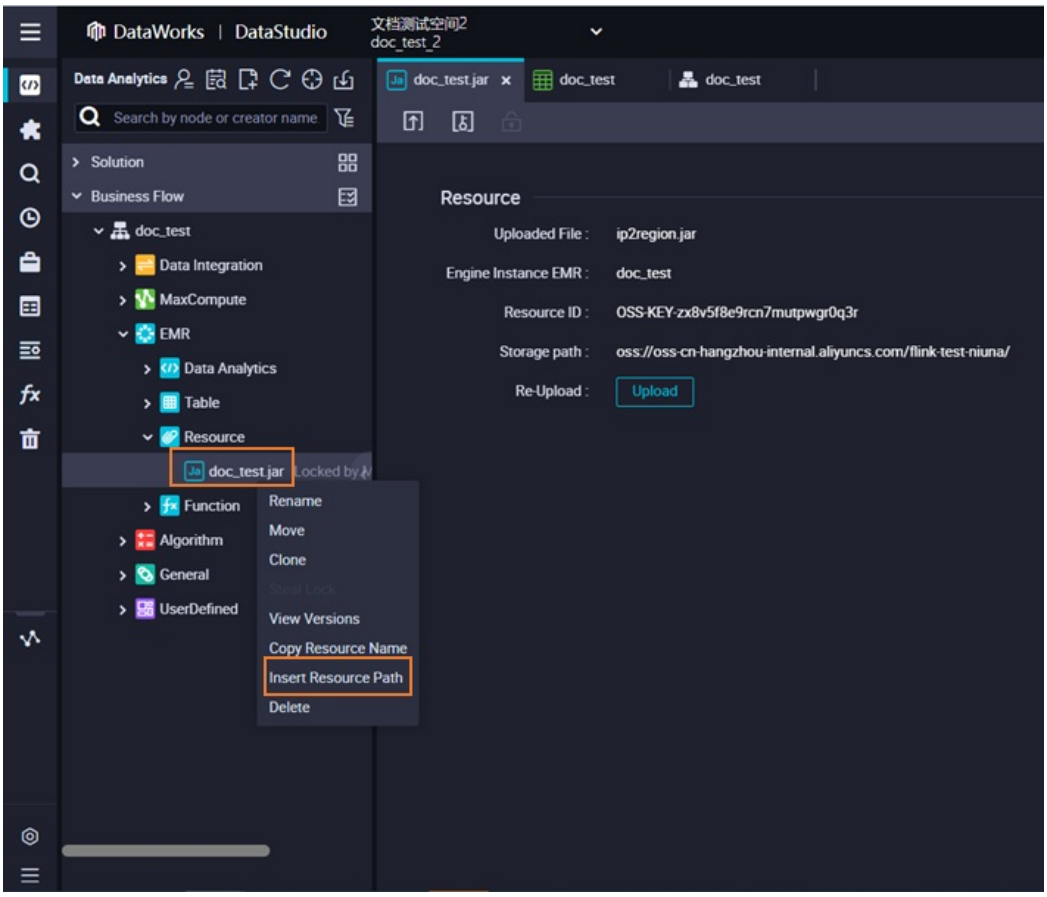


Parameter	Description
Resource Name	The name of the resource that you want to create. The resource name must have the suffix .jar.
Location	The folder for storing the resource. The default value is the path of the current folder. You can modify the path based on your business requirements.
File Type	The type of the resource. Set the parameter to EMR JAR.
Engine Instance	The EMR compute engine instance to which the resource belongs. Select an instance from the drop-down list.
Storage path	The storage path of the resource. Valid values: **OSS** and **HDFS**. ◦ If you select **OSS**, you must click Authorize next to OSS to authorize DataWorks and EMR to access Object Storage Service (OSS). Then, select a folder. ◦ If you select **HDFS**, enter a storage path.
File	The file that you want to upload. You can click **Upload**, select a file from your on-premises machine, and then click **Open**.

4. Click **Create**.
5. Click the and icons in the top toolbar to save and commit the resource to the development environment.

What's next

After you create an EMR JAR resource, you can reference the resource in the code of compute nodes such as an EMR MR node. The following figure shows how to reference the resource. For more information, see Create an EMR MR node.



2.1.5.5.3.11. Create an EMR table

This topic describes how to create an EMR table.

Prerequisites

- An EMR compute engine instance is associated with the current workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page.
- The metadata of an EMR data source is collected in Data Map so that you can select an EMR database when you create a table.

Limits

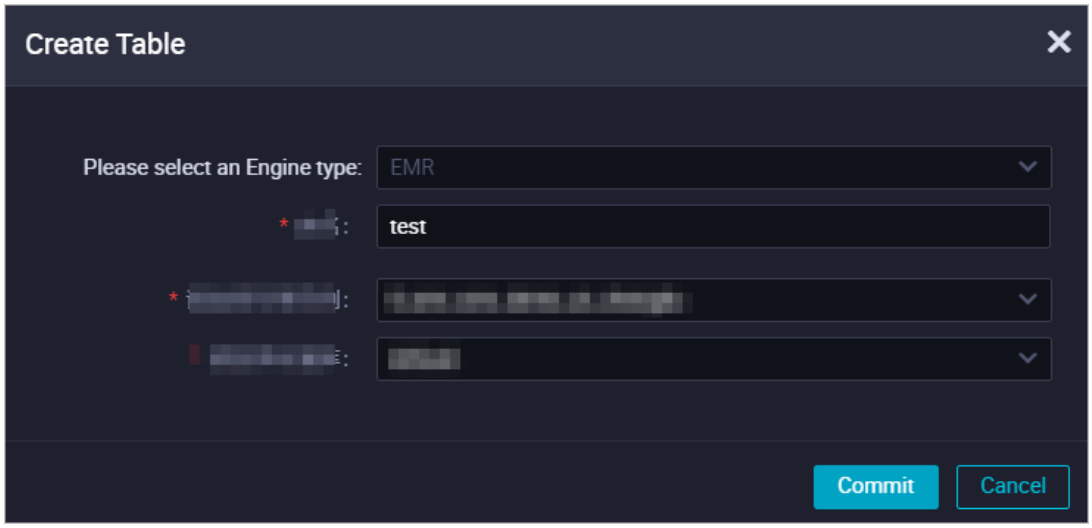
If Kerberos authentication is enabled for an EMR cluster., you cannot create tables, resources, and functions in a visualized manner for this cluster.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **EMR > table**.

   You can also find the workflow in which you want to create an EMR table, right-click **EMR**, and then choose **Create > Table**.

3. In the **Create Table** dialog box, set the parameters as required.

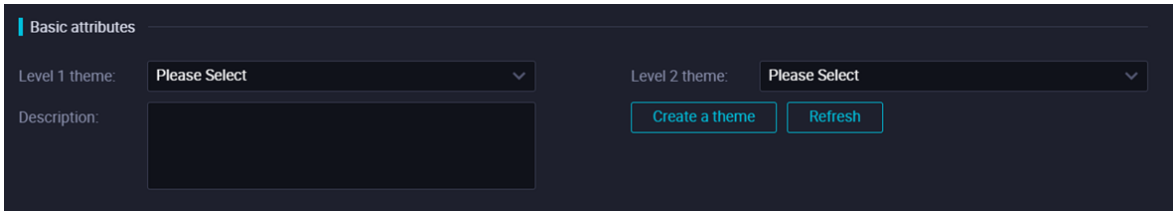


Parameter	Description
Engine type	The default value is EMR, which cannot be changed.
Table Name	The name of the EMR table.
Engine Instance	Select a required compute engine instance from the drop-down list.
Database	Select the database in which the compute engine instance resides from the drop-down list. **Note** You must collect metadata before you can select a database.

4. Click **Create**. The table configuration tab appears.

   The upper part of the tab shows the configurations that you specified in the **Create Table** dialog box. You can change the database where the EMR compute engine instance resides. To create a database, click **Create a database**. In the **Create a database** dialog box, set the parameters as required and click **OK**.

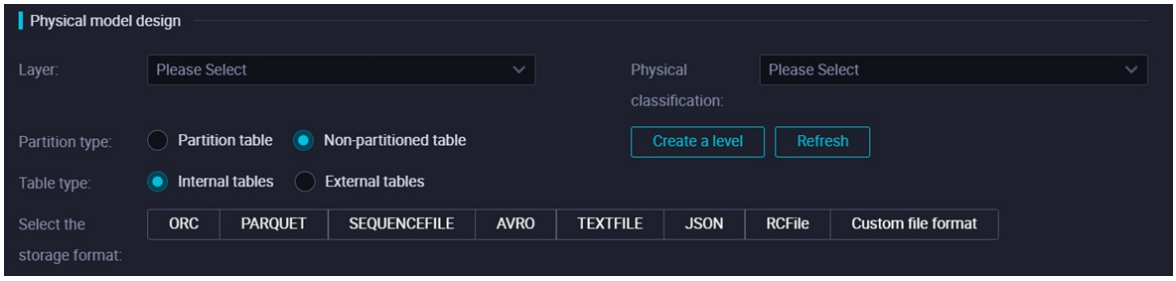
5. In the **Basic attributes** section, set the parameters as required.



Parameter	Description
Level 1 theme	The name of the level-1 folder where the table resides. **Note** The level-1 and level-2 folders show the table locations in DataWorks for you to manage tables with ease.

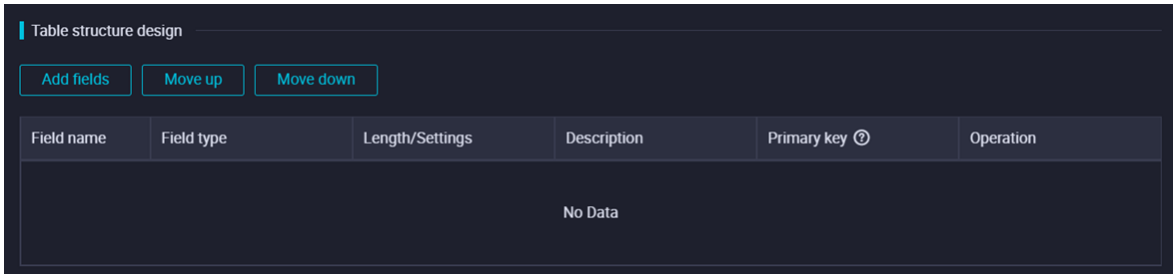
Parameter	Description
Level 2 theme	The name of the level-2 folder where the table resides.
Create a theme	Click **Create a theme** to go to the **Folder Management** tab. On this tab, you can create level-1 and level-2 folders.
Refresh	After you create a folder, click **Refresh**.
Description	The description of the table.

6. In the **Physical model design** section, set the parameters as required.



Parameter	Description
Layer	The levels and categories of the table. Select the appropriate level and category from the drop-down lists. To add levels and categories, click **Create a level** to go to the **Level Management** tab. After you create levels and categories, click **Refresh**.
Physical classification	
Partition type	Valid values: **Partition table** and **Non-partitioned table**.
Table type	Valid values: **Internal tables** and **External tables**.

7. In the **Table structure design** section, set the parameters as required.



Parameter	Description
Add fields	To add a field, click **Add fields**, configure the field information, and then click **Save** in the Operation column.
Move up	Adjusts the field sequence of a table that has not been created. If you want to adjust the sequence of fields in an existing table, you must delete the table and create another table with the same name. These operations are forbidden in the production environment.
Move down	
Field name	The name of the field, which can contain letters, digits, and underscores (_).

Parameter	Description
Data type	The EMR table supports the following data types: TINYINT, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, DECIMAL, VARCHAR, CHAR, STRING, BINARY, DATETIME, DATE, TIMESTAMP, BOOLEAN, ARRAY, MAP, and STRUCT.
Length/Settings	You must set this parameter if the data type that you specify for the field has a length limit.
Description	The description of the field.
Primary key	Specifies whether the field serves as the primary key. The primary key ensures that each record is unique for your business. DataWorks does not impose a limit on the field that can be specified as the primary key.
Edit	After you save the field, you can click **Edit** to edit the field and then click **Save**.
Delete	Deletes a created field. **Note** If you want to delete a field from an existing table and then commit the table, you must delete the table and create another table with the same name. These operations are forbidden in the production environment.
Add partitions	If you set the **Partition type** parameter to **Partition table** in the **Physical model design** section, you must configure a partition for the table. You can add a partition to the current table. If you want to add a partition to an existing table and then commit the table, you must delete the table and create another table with the same name. These operations are forbidden in the production environment.

8. Click the icon in the top toolbar to commit the EMR table to the production environment.

   If you are using a workspace in standard mode, commit the table to the development environment and the production environment in sequence.

   Notice You cannot create an EMR table in DDL mode.

2.1.5.5.3.12. Create an EMR function

This topic describes how to create an EMR function.

Prerequisites

- An EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the EMR cluster belongs:
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with the current workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page.

- The required resources are uploaded.

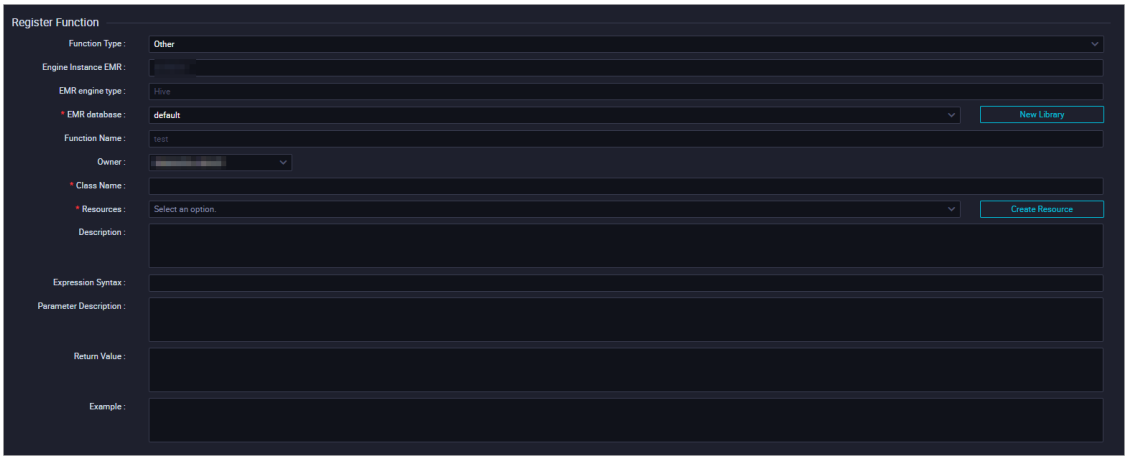
Limits

If Kerberos authentication is enabled for an EMR cluster., you cannot create tables, resources, and functions in a visualized manner for this cluster.

Procedure

1. Log on to the DataWorks console.
2. Create a workflow. For more information, see Overview.
3. Write code in a local Java environment and compress the code to a JAR package. Then, create a JAR resource and commit the resource. For more information, see Create and use an EMR JAR resource.
4. Create a function.
   i. Click the workflow in the Scheduled Workflow pane, right-click **EMR**, and then choose **Create > Function**.
   ii. In the **Create Function** dialog box, set the **Function Name**, **Engine Instance**, and **Location** parameters.
   iii. Click **Create**.

iv. In the **Function information** section of the configuration tab that appears, set the parameters.



Parameter	Description
Function Type	The type of the function. Valid values: **Mathematical Operation Functions**, **Aggregate Functions**, **String Processing Functions**, **Date Functions**, **Window Functions**, and **Other Functions**.
Engine Instance	The EMR compute engine instance. By default, the system automatically selects the EMR compute engine instance. You cannot change the value.
Engine Type	The type of the compute engine instance. By default, the system automatically selects EMR. You cannot change the value.
EMR database	The database where the EMR cluster resides. Select a database from the drop-down list. To create a database, click **New Library**. In the **New Library** dialog box, set the parameters and click **OK**.
Function Name	The name of the function. You can use this name to reference the function in SQL statements. The function name must be globally unique and cannot be changed after the function is created.
Owner	The value of this parameter is automatically displayed.
Class Name	Required. The name of the class that implements the function.
Resource	Required. The resource to be used in the function. Select a resource from the ones that are created in the current workspace from the drop-down list. To create a resource, click **Create Resource**. In the **Create Resource** dialog box, set the parameters and click **Create**.
Description	The description of the function.
Expression Syntax	The syntax of the function. Example: `test`.
Parameter Description	The description of the input and output parameters that are supported.
Return Value	Optional. The return value. Example: 1.
Example	Optional. The example of the function.

5. Click the icon in the top toolbar.
6. Commit the function.

i. Click the icon in the top toolbar.

ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iii. Click **OK**.

2.1.5.5.4. Hologres

2.1.5.5.4.1. Create a Hologres SQL node

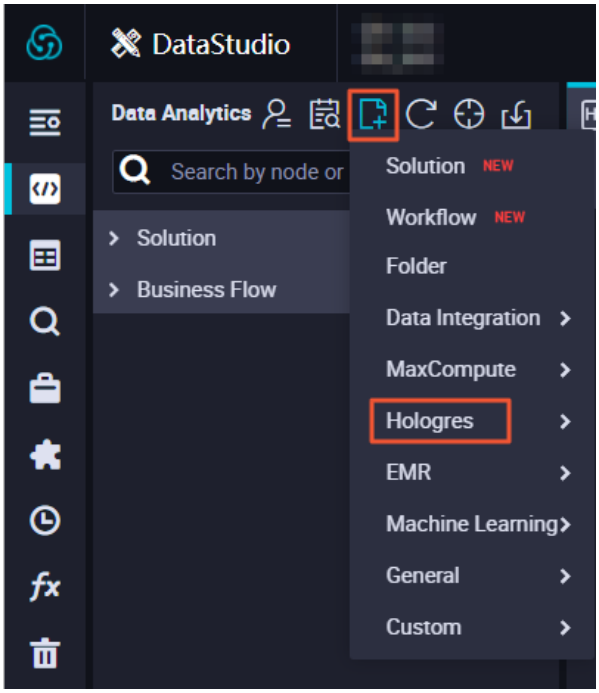
This topic describes how to create a Hologres SQL node. Hologres seamlessly integrates with MaxCompute at the underlying layer. This integration allows you to use standard PostgreSQL statements to query and analyze large volumes of data stored in MaxCompute. In this case, you do not need to transfer data. This allows you to quickly obtain query results.

Prerequisites

A **Hologres** compute engine instance is added on the **Project Management** page. This ensures that the **Hologres** folder is displayed on the page on which you want to create a Hologres SQL node.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the icon and choose **Hologres > Hologres SQL**.



3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

> **Note** The name of the node must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the tab that appears, edit and run the code in the code editor.

   After the Hologres SQL node is created, edit the code in compliance with the required SQL syntax.
6. On the configuration tab of the node, click the **Properties** tab in the right-side navigation pane. On the

Properties tab, configure properties for the node. For more information, see Basic properties.

7. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you commit the node.

   i. Click the icon in the top toolbar to save the node.

   ii. Click the icon in the top toolbar.

   iii. (Optional)In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   If the workspace that you use is in standard mode, you must click **Deploy** in the top navigation bar after you commit the node. For more information, see Deploy nodes.

8. Test the node. For more information, see Manage auto triggered nodes.

2.1.5.5.5. AnalyticDB for PostgreSQL

2.1.5.5.5.1. Create an AnalyticDB for PostgreSQL node

You can create AnalyticDB for PostgreSQL nodes in the DataWorks console to build an online extract, transform, load (ETL) process.

Prerequisites

The AnalyticDB for PostgreSQL compute engine is associated with the workspace in which you want to create an AnalyticDB for PostgreSQL node. The AnalyticDB for PostgreSQL folder is displayed in a workspace only after you associate an AnalyticDB for PostgreSQL compute engine instance with the workspace on the **Workspace Management** page. For more information, see Create an AnalyticDB for PostgreSQL node.

Procedure

1. Log on to the DataWorks console.
2. On the **DataStudio** page, move the pointer over the icon and choose **AnalyticDB > ADB for PostgreSQL**.

   Alternatively, you can find the required workflow, right-click the workflow name, and then choose **Create > AnalyticDB for PostgreSQL > ADB for PostgreSQL**.

3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The name of the node must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. Configure the AnalyticDB for PostgreSQL node.

   i. Select a data source from the **Select a connection** drop-down list.

   > **Notice**
   > - When you associate the AnalyticDB for PostgreSQL compute engine with the workspace, DataWorks automatically creates an AnalyticDB for PostgreSQL data source.
   > - You can select a data source that is added only by using the connection string mode.

ii. Compile SQL statements.

After you select a data source, compile SQL statements based on the syntax that is supported by AnalyticDB for PostgreSQL.

iii. Click the icon in the top toolbar to save the SQL statements.

iv. Click the icon in the top toolbar to execute the SQL statements.

6. Click **Properties** in the right-side navigation pane. In the Properties panel, configure scheduling properties for the node. For more information, see Basic properties.
7. Save and commit the node.

> **Notice** You must specify the **Rerun** and **Parent Nodes** parameters in the Properties panel before you commit the node.

i. Click in the top toolbar to save the node.

ii. Click the icon in the top toolbar.

iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iv. Click **OK**.

If the workspace that you use is in standard mode, you must click **Deploy** in the upper-right corner after you commit the AnalyticDB for PostgreSQL node. For more information, see Deploy nodes.

8. Test the node. For more information, see Manage auto triggered nodes.

2.1.5.5.5.2. Create an AnalyticDB for PostgreSQL table

This topic describes how to create an AnalyticDB for PostgreSQL table.

Prerequisites

- An AnalyticDB for PostgreSQL instance is associated with the workspace in which you want to create an AnalyticDB for PostgreSQL table. The AnalyticDB for PostgreSQL folder is displayed on the DataStudio page of a workspace only after you associate an AnalyticDB for PostgreSQL instance with the workspace on the **Project Management** page. For more information, see Create an AnalyticDB for PostgreSQL table.
- The metadata of the associated AnalyticDB for PostgreSQL instance is collected on the **Data Map** page. For more information, see Collect metadata from an AnalyticDB for PostgreSQL data source.

Procedure

1. Log on to the DataWorks console.
2. On the **DataStudio** page, move the pointer over the icon and choose **AnalyticDB > table**.

   Alternatively, you can find your workflow in the Business Flow section, right-click **AnalyticDB for PostgreSQL**, and then choose **Create > ADB visual table creation**.
3. In the **Create Table** dialog box, set **Table Name**.

> **Notice**
>
> - The table name must be in the format of schema_name.table_name.
> - The values of schema_name and table_name must be 1 to 63 characters in length and can contain letters, digits, and underscores (_). The values must start with a letter or underscore (_).
> - If you associate multiple AnalyticDB for PostgreSQL instances with the current workspace, you must select one based on your business requirements.

4. Click **Commit**. The table configuration tab appears.

   The upper part of the table configuration tab displays the table name and AnalyticDB for PostgreSQL instance name.

5. In the **General** section, set the parameters.

Parameter	Description
Level 1 theme	The name of the level-1 folder where the table resides. **Note** Level-1 and level-2 folders show the table locations in DataWorks for you to manage tables more conveniently.
Level 2 theme	The name of the level-2 folder where the table resides.
Create a theme	Click **Create a theme** to go to the **Folder Management** tab. On this tab, you can create level-1 and level-2 folders for tables. After you create a folder, click the icon next to Create Folder to synchronize the folder.
Description	The description of the table.

6. In the **Physical model design** section, set the parameters.

Parameter	Description
Level selection	The layer where the table data is stored or processed. A data warehouse consists of the operational data store (ODS), common data model (CDM), and application data store (ADS) layers. You can customize a name for each layer.
Physical classification	The category of the table. Tables are categorized into basic services, advanced services, and other services. You can customize a name for each category. **Note** Categories are designed only for your management convenience and do not involve underlying implementation.
New Level	The levels and categories that you want to create. To add levels and categories, click **New Level** to go to the **Hierarchical management** tab. After levels and categories are created, click the icon.

7. In the **AnalyticDB for PostgreSQL table design** section, set the parameters.

   You can configure the schema of an AnalyticDB for PostgreSQL table on the following tabs: **Column information settings**, **Index settings**, **Sub-table design**, and **Partition settings (optional)**.

Tab	Parameter	Description
Column information settings	**New columns**	Allows you to click the button and set the relevant parameters to create a field.
	Name	The name of the field.
	Field type	The data type of the field.
	Field length	The length of the field. You can specify the length for fields only of some specific data types.
	Default value	The default value of the field.
	Allow to be empty	Specifies whether the field can be empty.
	Is it the primary key?	Specifies whether the field serves as the primary key.
	Foreign key	Specifies whether the field serves as a foreign key.
	Operation	◦ You can perform the following operations on a new field: save, cancel, delete, move up, and move down. ◦ You can perform the following operations on an existing field: modify, delete, move up, and move down.
Index settings	**New columns**	Allows you to click the button and set the relevant parameters to create an index.
	Index name	The name of the index. Make sure that you specify a unique name.
	Include columns	The field on which the index will be created. To select a field, click **Edit**. In the **Select at least one index** dialog box, click the **+** icon. All the created fields appear in the Column information drop-down list. Select the field from the **Column information** drop-down list and click **Save**.
	Index type	The type of the index. Valid values: **Normal**, **Primary Key**, and **Unique**.
	Index mode	The mode for indexing data in the fields. Valid values: **B-tree**, **Bitmap**, and **GiST**.
	Operation	◦ You can perform the following operations on a new index: save, cancel, delete, move up, and move down. ◦ You can perform the following operations on an existing index: modify, delete, move up, and move down.
Sub-table design	**Hash (Recommended)**, **Copy Schema**, and **Random (Not Recommended)**	The way in which the partition key is generated. Take **Hash (Recommended)** as an example. Click **New columns** and select the target field from the **Name** drop-down list. The information about the selected field appears. Click **Save**. For more information, see the **Column information settings** section of this table.

Tab	Parameter	Description
Partition settings (optional)	**Partition settings (optional)**	The partitions of the table. You can configure the partitions based on your business requirements.

8. Click **Submit to development environment** and **Submit to production environment** in sequence.

   If you are using a workspace in basic mode, you need only to click **Submit to production environment**.

9. In the **Submit changes** dialog box, confirm that the table creation statements are correct, select a resource group from the **Select a resource group** drop-down list, and then click **Confirm execution**.

What's next

After the AnalyticDB for PostgreSQL table is created, you can query the table data, modify the table, or delete the table. For more information, see Manage tables.

2.1.5.5.6. AnalyticDB for MySQL

2.1.5.5.6.1. Create and use an AnalyticDB for MySQL node

You can create an AnalyticDB for MySQL node and use SQL statements to develop data for an AnalyticDB for MySQL data source. This topic describes how to create and use an AnalyticDB for MySQL node.

Prerequisites

- An AnalyticDB for MySQL instance is purchased and associated with the current DataWorks workspace in the **Workspace Management** page of DataWorks. For more information, see Associate an AnalyticDB for MySQL instance with a workspace.
- A workflow is created. For more information, see Create a workflow.

Create an AnalyticDB for MySQL node and use the node to develop data

1. Log on to the DataWorks console.
2. On the **DataStudio** page, move the pointer over the icon and choose **AnalyticDB for MySQL > ADB for MySQL**.

   You can also find the required workflow, right-click the workflow name, and then choose **Create > AnalyticDB for MySQL > ADB for MySQL**.

3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The name of the node must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Configure the AnalyticDB for MySQL node.
   i. Select a data source from the **Select Data Source** drop-down list.

   > **Notice**
   > - When you associate the AnalyticDB for MySQL compute engine instance with the workspace, DataWorks automatically adds an AnalyticDB for MySQL data source.
   > - You can select a data source that is added only by using the connection string mode.

ii. Write the SQL statements of the node.

After you select a data source, write the SQL statements of the node based on your business requirements.

iii. Click the icon in the top toolbar to save the SQL statements.

iv. Click the icon in the top toolbar to execute the SQL statements.

5. Click the **Properties** tab in the right-side navigation pane. In the Properties panel, configure scheduling properties for the node. For more information, see Basic properties.
6. Save and commit the node.

> **Notice** You must set the **Rerun** and **Parent Nodes** parameters in the Properties panel before you commit the node.

i. Click the icon in the top toolbar to save the node.

ii. Click the icon in the top toolbar.

iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iv. Click **OK**.

If the workspace that you use is in standard mode, you must click **Deploy** in the upper-right corner after you commit the AnalyticDB for MySQL node. For more information, see Deploy nodes.

7. Test the node. For more information, see Manage auto triggered nodes.

2.1.5.5.7. Algorithm

2.1.5.5.7.1. 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, specify **Node Name** and **Location**.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

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 synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure 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 node. For more information, see Manage auto triggered nodes.

2.1.5.5.8. General

2.1.5.5.8.1. 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**.

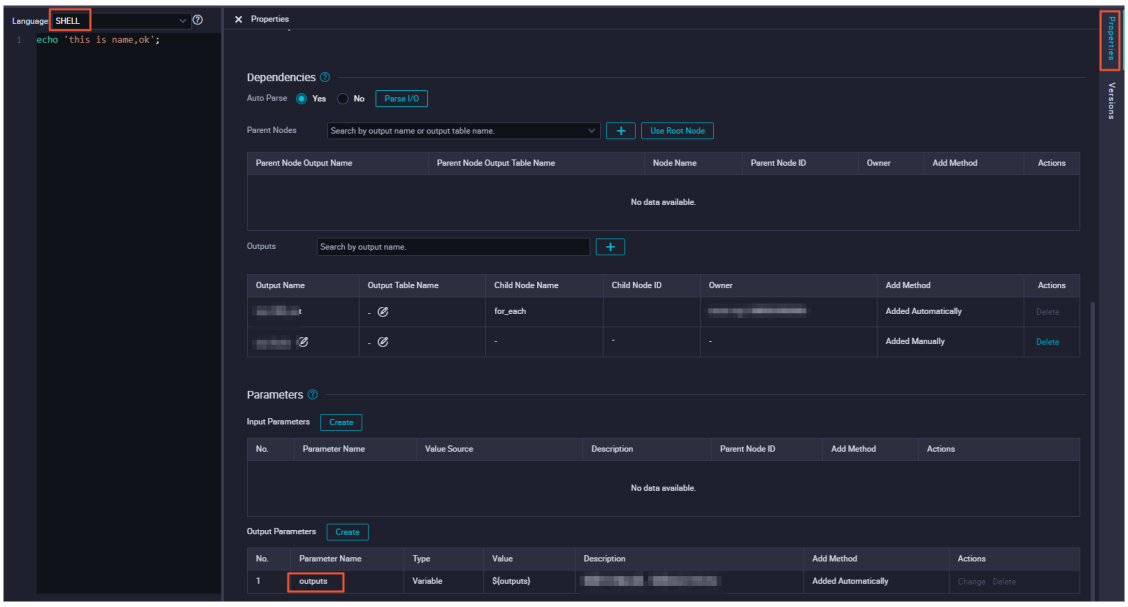
   Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > for-each**.
3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
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 synchronization node, click the **Properties** tab in the right-side

navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.

7. Commit the node.

   > **Notice** You can commit the node only after you specify the **Rerun** and **Parent Nodes** parameters.

   i. Click the icon in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Publish** in the upper-right corner after you commit the node. For more information, see Publish nodes.

8. Test the node. For more information, see Manage auto triggered nodes.

2.1.5.5.8.2. 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**.

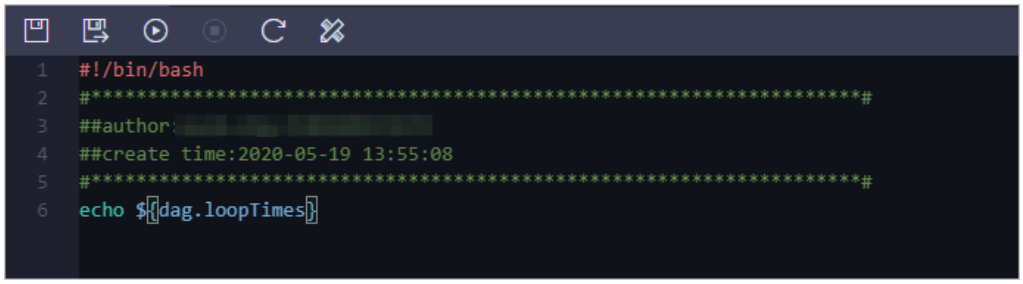
3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

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

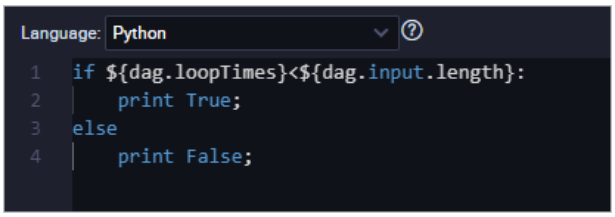
In addition to simple scenarios, do-while nodes can also be used in complex scenarios where each row of data is processed in sequence by using a loop. Before you process data in such scenarios, make sure that:

- 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.

2.1.5.5.8.3. 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.

2.1.5.5.8.4. 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 ! =.
>	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.

2.1.5.5.8.5. 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.

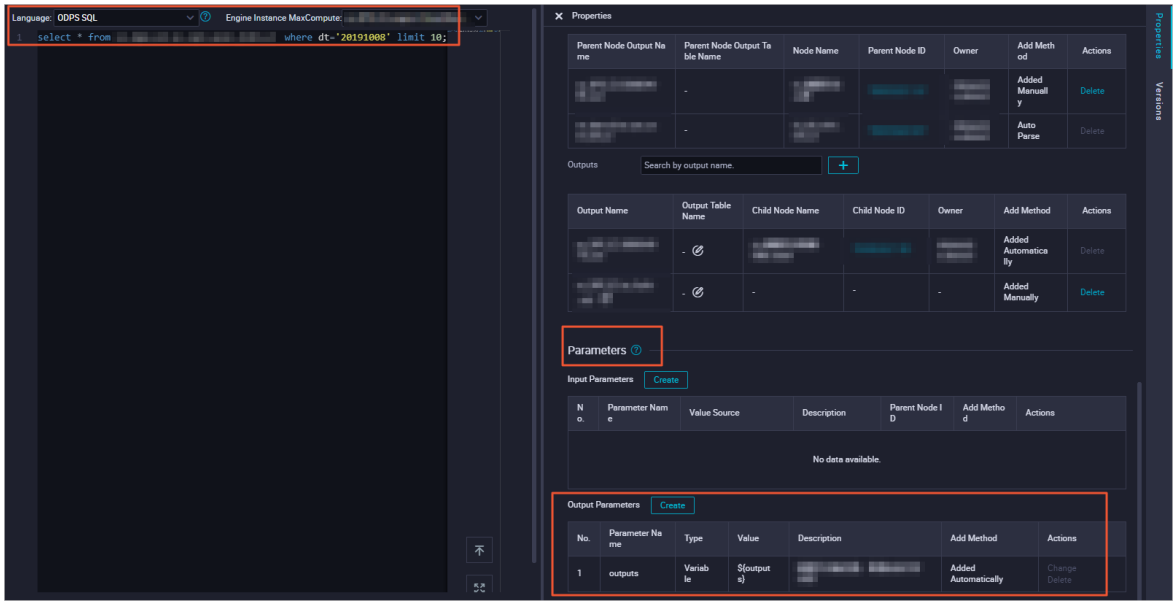
   This following sections describe how to use assignment nodes that use the Python, MaxCompute SQL, and SHELL languages respectively to pass data between a parent node and a child node named Assignment node value comparison_shell by using context-based parameters.

   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.

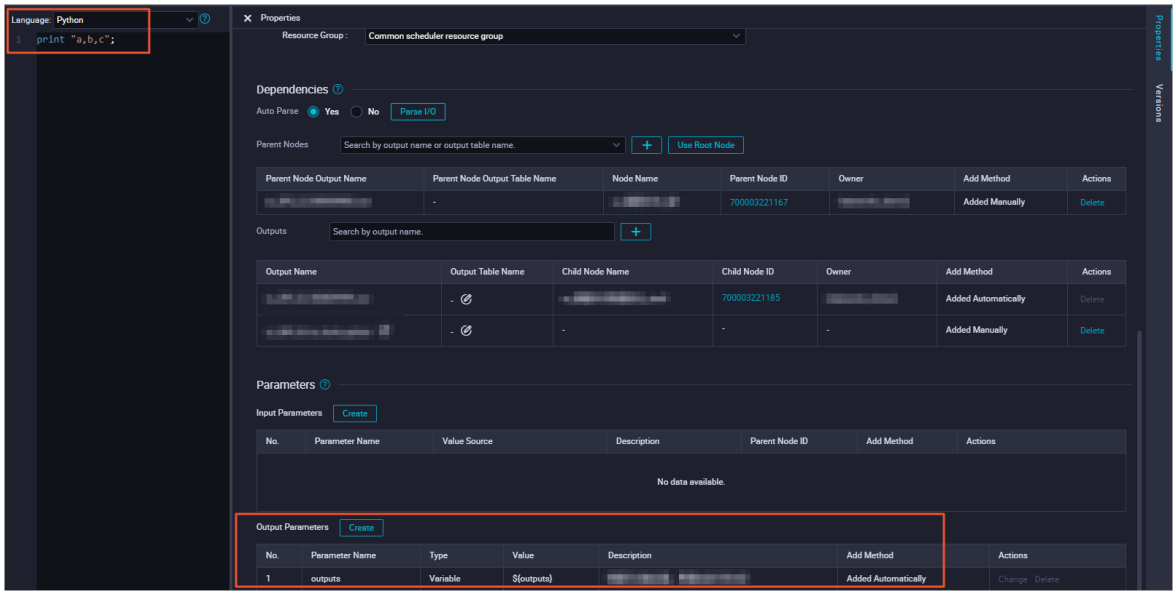
```
echo '${sql_inputs}';
echo 'Use the value in the first line in the output of the fuzhi_sql node as the input'${sql_inp
uts[0]};
echo 'Use the value in the second line in the output of the fuzhi_sql node as the input'${sql_in
puts[1]};
echo 'Use the value of the second field in the first line in the output of the fuzhi_sql node as
the input'${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.

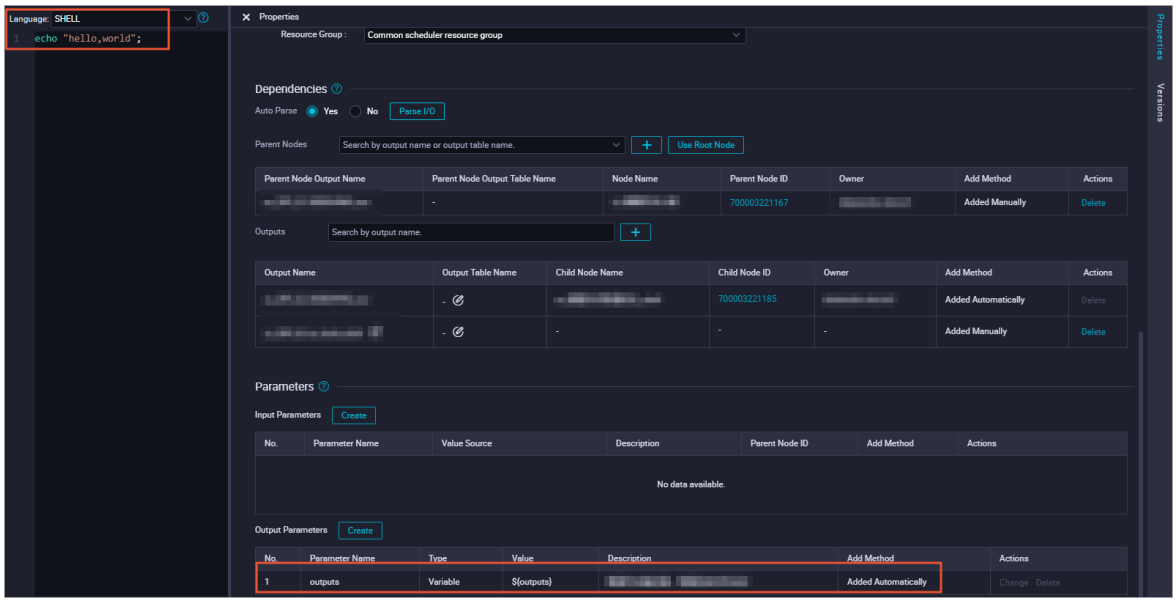
```
echo 'The output of the fuzhi_python node'${python_inputs};
echo 'Use the first value in the output of the fuzhi_python node as the input'${python_inputs[0]
};
echo 'Use the second value in the output of the fuzhi_python node as the input'${python_inputs[1
]};[1]}
```

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.

2.1.5.5.8.6. 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.

2.1.5.5.8.7. 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.

2.1.5.5.8.8. 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.

2.1.5.5.8.9. 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.

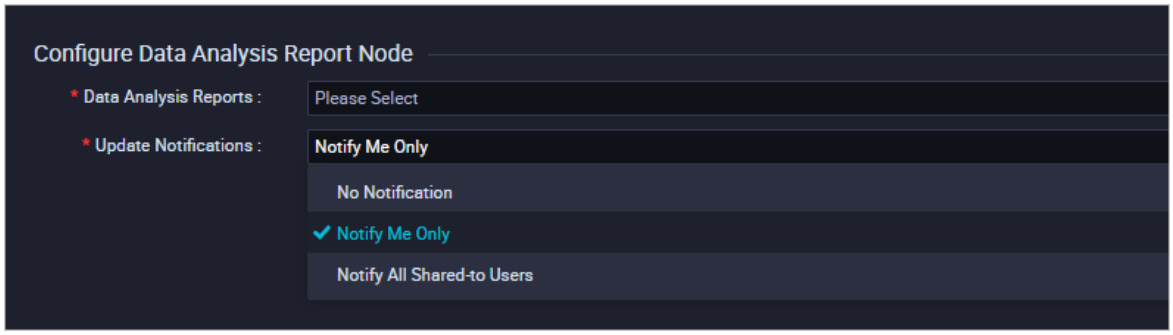
Procedure

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, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

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 synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure 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 Manage auto triggered nodes.

   After the node is run in the production environment, you can view updates of the report on the Report page of the **DataAnalysis** module.

2.1.5.5.9. Custom

2.1.5.5.9.1. 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, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
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 synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure 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 node. For more information, see Manage auto triggered nodes.

2.1.5.6. Schedule

2.1.5.6.1. 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.

2.1.5.6.2. 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

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 Manage auto triggered nodes. 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 Manage auto triggered nodes. 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.

2.1.5.6.3. 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.

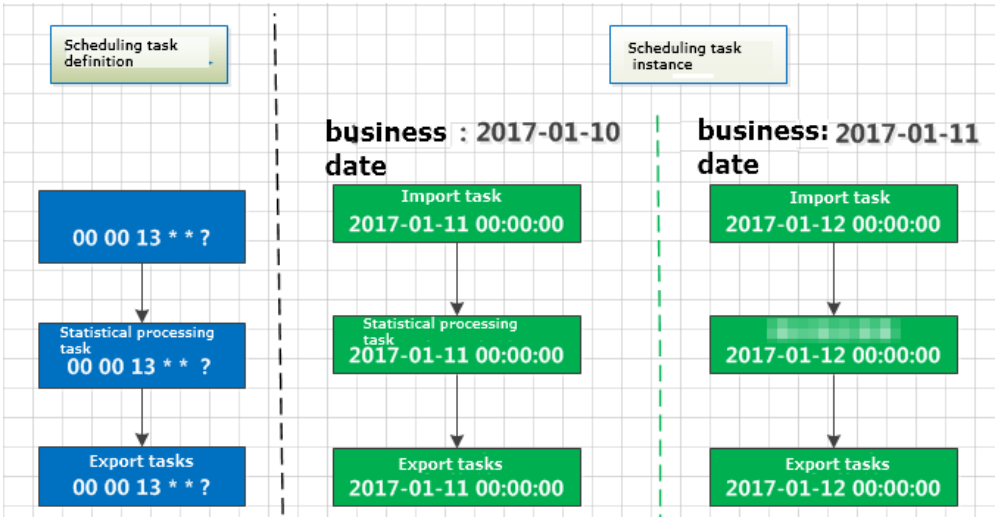
  > **Note** An auto triggered node can be run only when all the ancestor instances are run and the scheduled time arrives. Both prerequisites are indispensable and have no specific chronological order.

- If you clear Customize Runtime, the scheduled time of the node is randomly set in the range of 00:00 to 00:30.

Scenarios:

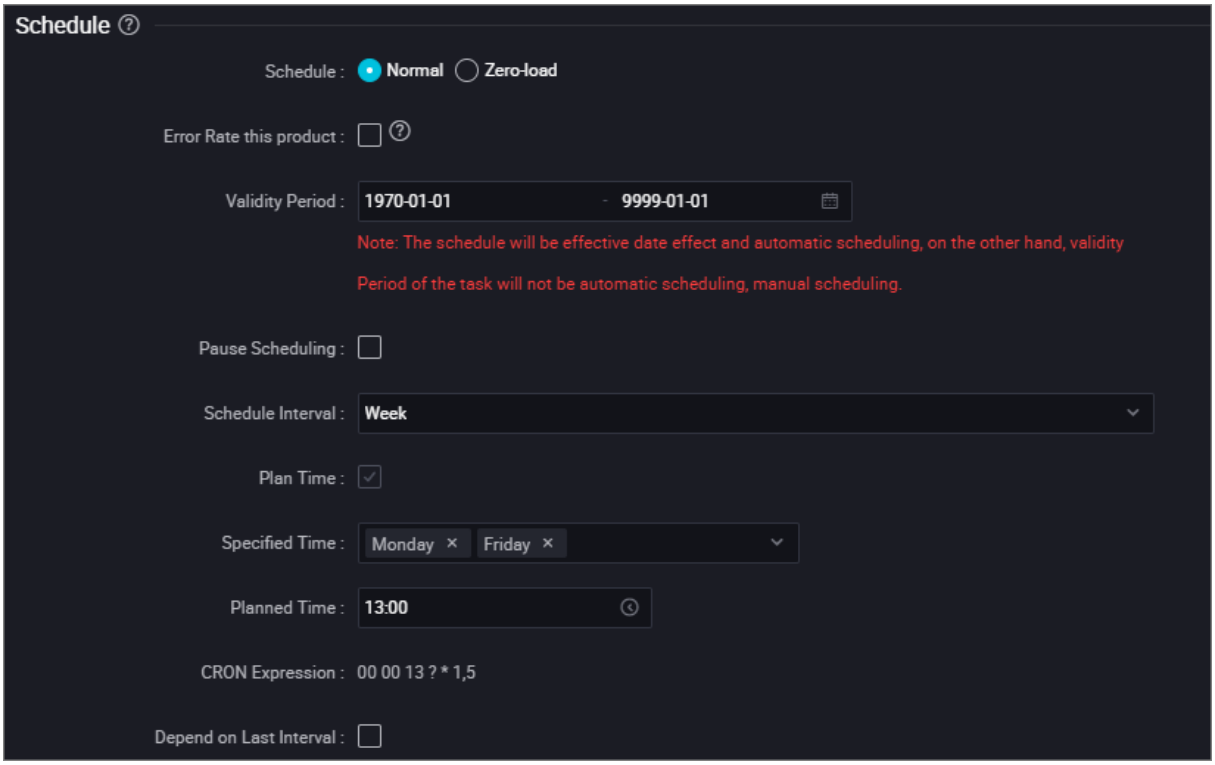
For example, you have created an import node, an analytics node, and an export node. They are all scheduled to run at 13:00 every day. The analytics node depends on the import node, and the export node depends on the analytics node. The following figure shows that the analytics node is configured to 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.



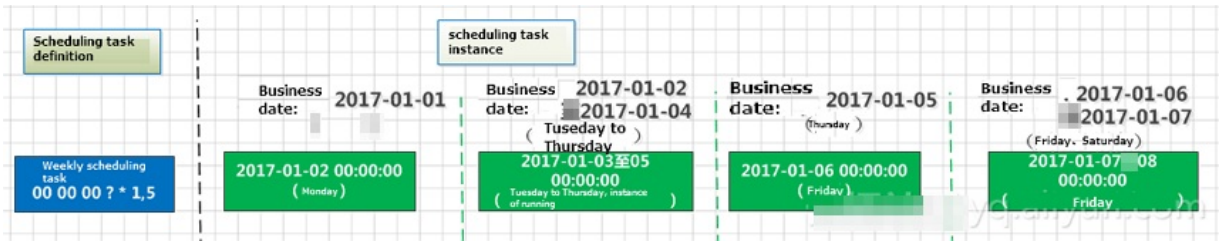
Scheduled by week

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.



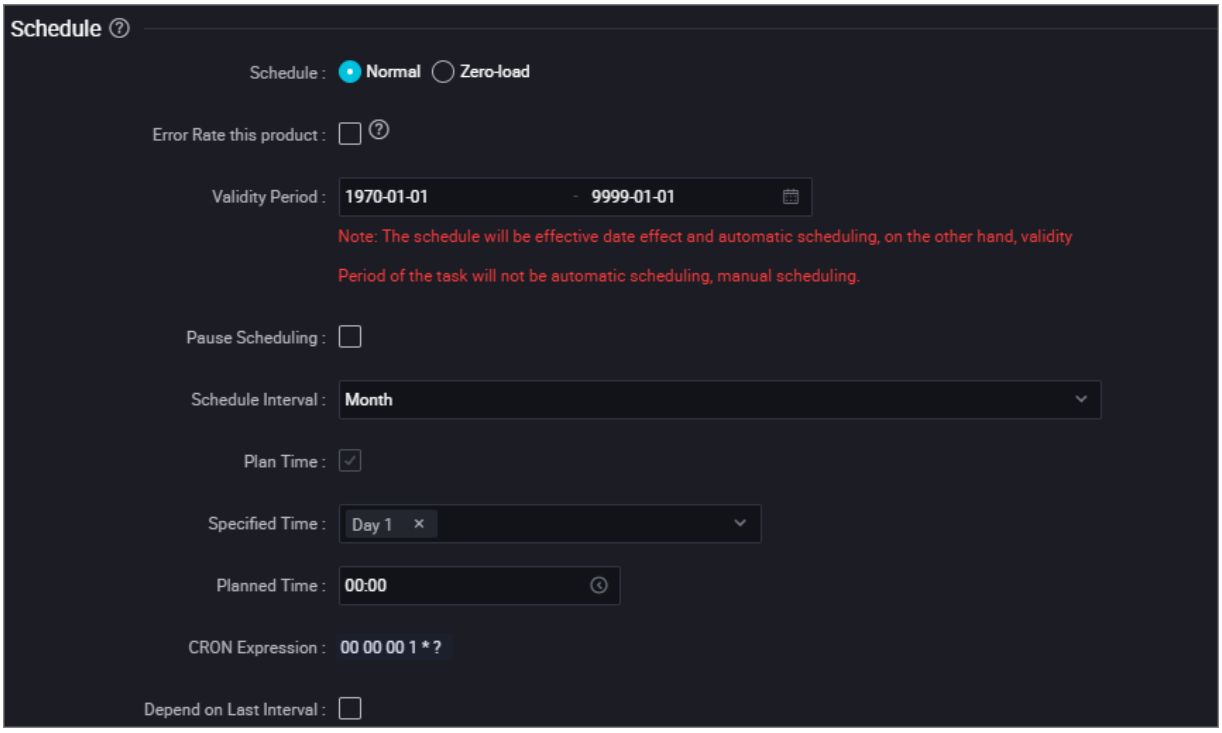
For example, you have created a node. As shown in the preceding figure, the scheduling system runs instances generated on Mondays and Fridays, but returns success responses without running the code for instances generated on Tuesdays, Wednesdays, Thursdays, Saturdays, and Sundays.

Based on the preceding node scheduling properties, the scheduling system automatically generates and runs instances for the node, as shown in the following figure.



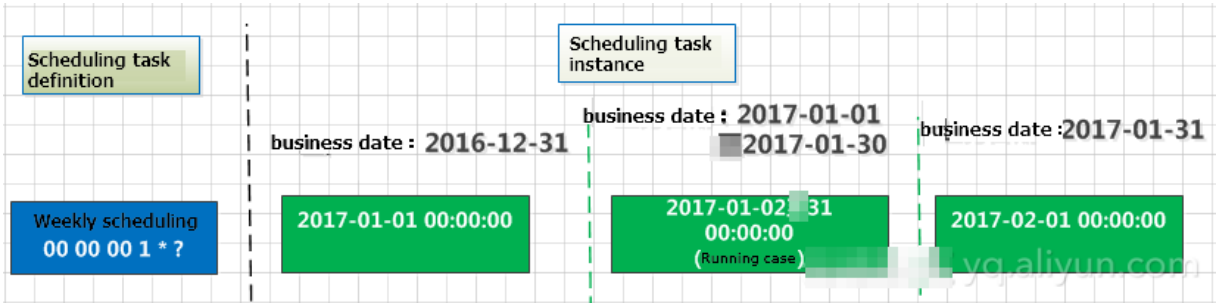
Scheduled by month

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.

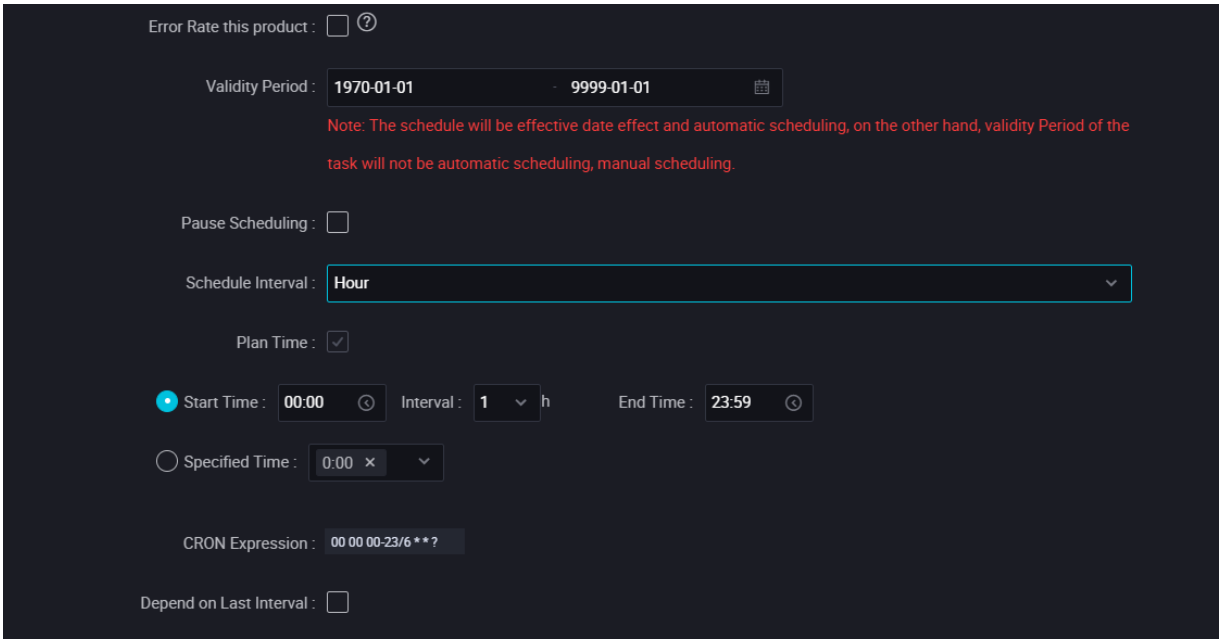
Based on the preceding node scheduling properties, the scheduling system automatically generates and runs instances for the node, as shown in the following figure.



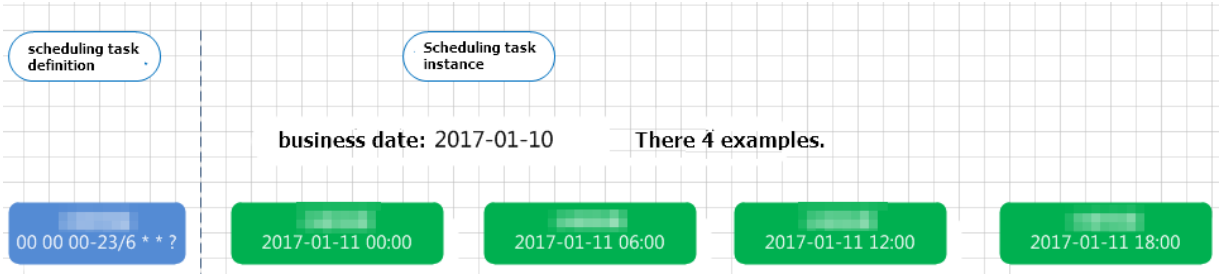
Scheduled by hour

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.



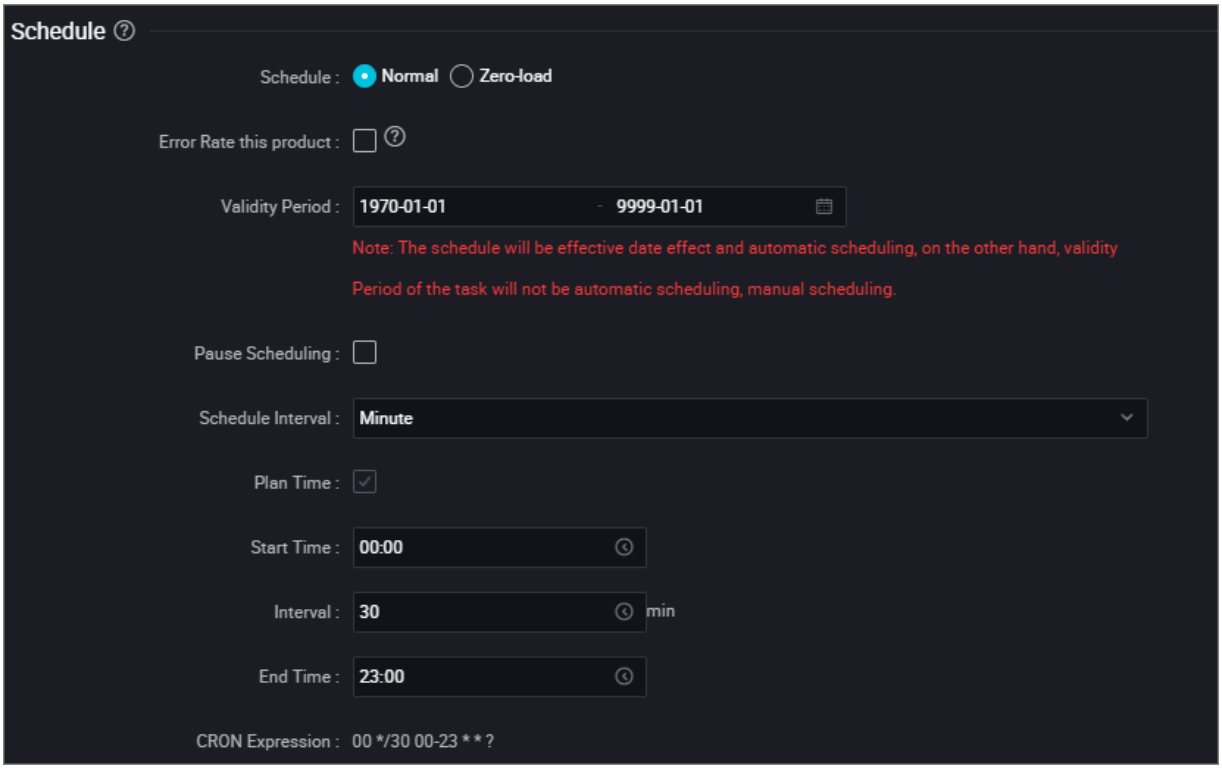
For example, you have created a node. As shown in the preceding figure, the node is automatically run every 6 hours in the period from 00:00 to 23:59 every day. In this case, the scheduling system automatically generates and runs instances for the node, as shown in the following figure.



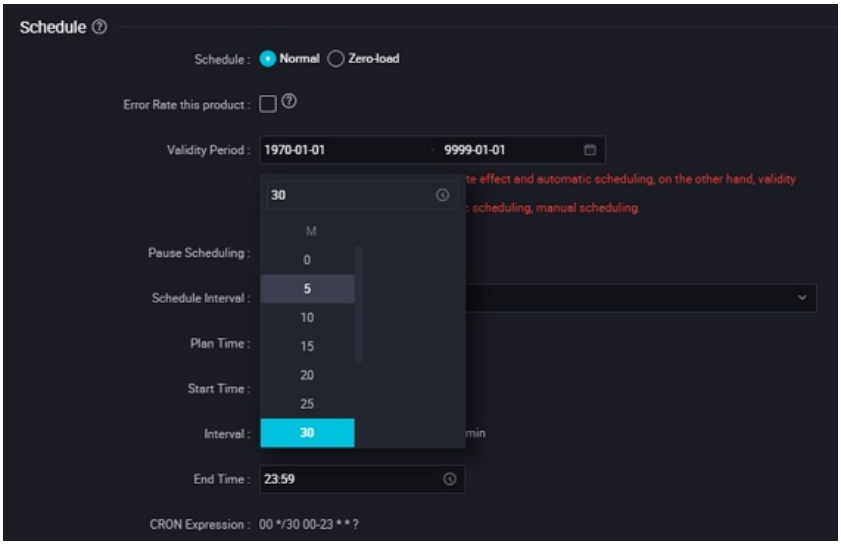
Scheduled by minute

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?

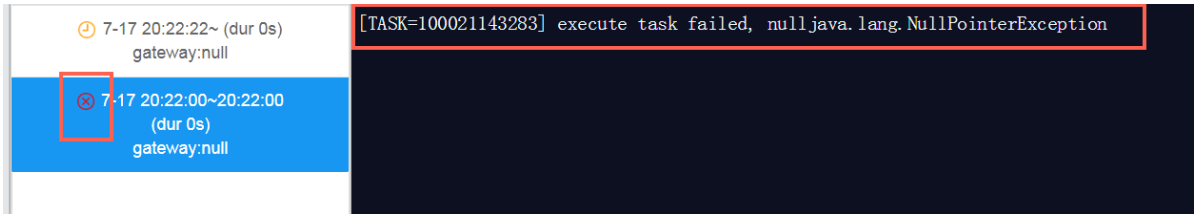
  A: When configuring the scheduling properties of Node A, select **Cross-Cycle Dependencies** and select **Instances of Current Node** from the Depend On drop-down list. When configuring the scheduling properties of Node B, configure Node B to depend on Node A and set the scheduled time of Node B to 00:00 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.

2.1.5.6.4. 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.

2.1.5.7. Components

2.1.5.7.1. 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.

Field	Data type	Description
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.

2.1.5.7.2. 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.

Icon or tab	Description
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.

2.1.5.8. Custom plug-ins

2.1.5.8.1. 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.

2.1.5.8.2. 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 (_).

Parameter	Description
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.

2.1.5.8.3. 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
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.

Parameter	Description
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.

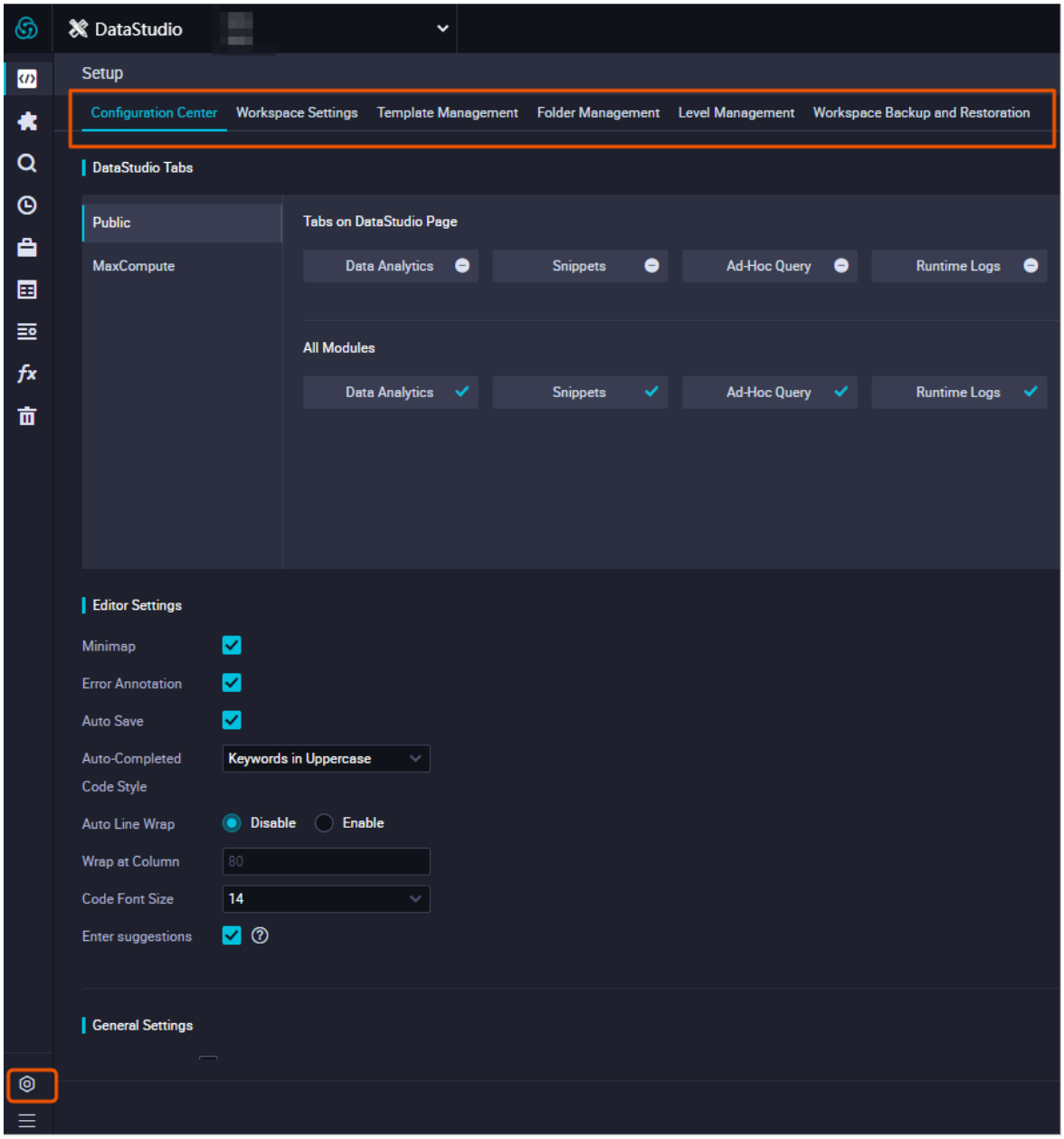
2.1.5.9. Setup

2.1.5.9.1. 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
- Template management
- Folder management
- Level management

2.1.5.9.2. 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**

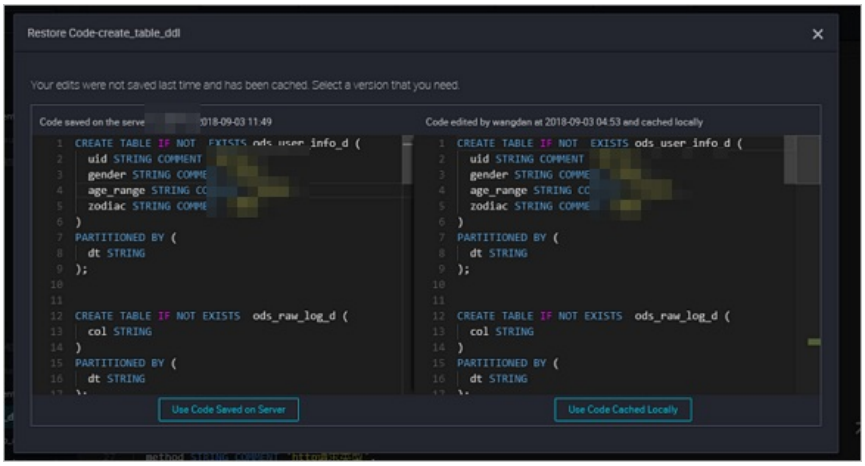
  The masked code in the current interface is displayed in the minimap in the upper-right corner of the page. When the code is long, you can move the pointer to specify the code block to be displayed in the minimap.

- **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.

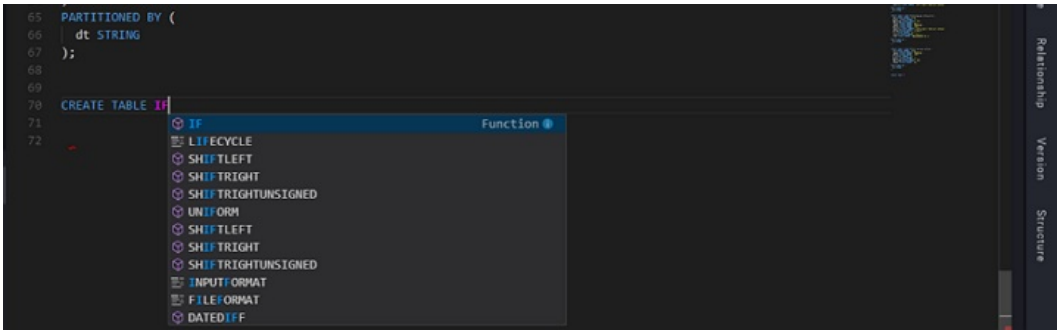
- **Auto Save**

  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**

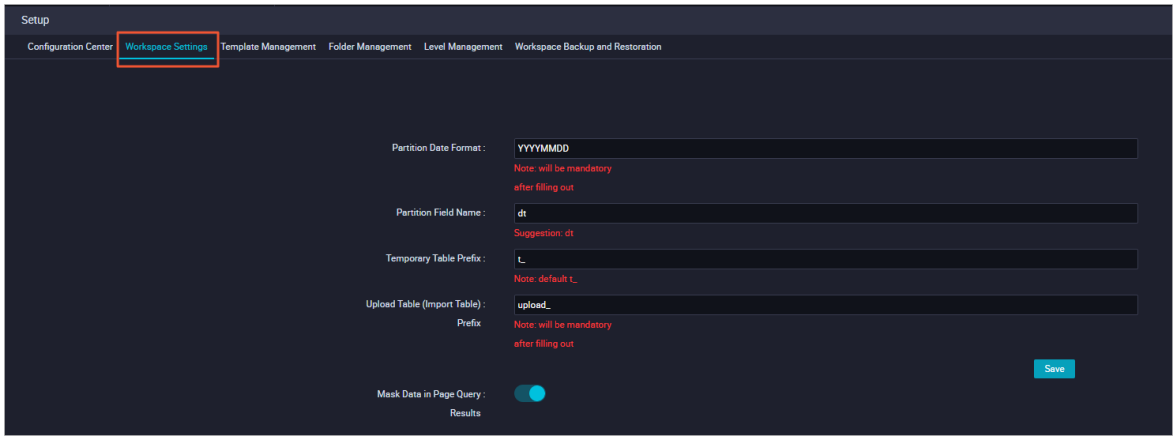
  You can set the DataStudio theme to black or white.

2.1.5.9.3. Configure a workspace

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 **DataStudio** page, 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 based on your business requirements.

Parameter	Description
Partition Field Name	The default name of the 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 mask the data in query results. If the switch is turned on, the result returned for an ad hoc query node in the current workspace is masked.

Enable data masking for DataWorks workspaces

Data masking needs to be enabled for DataWorks workspaces one by one. After data masking is enabled, the results returned for ad hoc query nodes in the current workspace are masked. The data stored at underlying layers is not affected because only dynamic data masking is performed. For example, data masking is enabled for Workspace A but is disabled for 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**. Then, click **Save** in the lower-right corner of the tab. The results returned for ad hoc query nodes in the current workspace will be masked.

> **Note** By default, the Mask Data in Page Query Results switch is turned off, and you are not allowed to download data.

After data masking is enabled for a DataWorks workspace, data of the types listed in the following table is masked by default.

Type	Data masking rule	Raw data	Data after masking
ID card number	Only the first and last digits in a 15- or 18-digit ID card number are displayed in plaintext. All the other digits are displayed as asterisks (*).	512345678943215678	5**********8
Mobile phone 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 or deposit card number are displayed in plaintext. All the other digits are displayed as asterisks (*).	• 1234576834509782 • 643257829145430986	• **********9782 • **********0986

Type	Data masking rule	Raw data	Data after masking
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:**:**:**:**:**
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

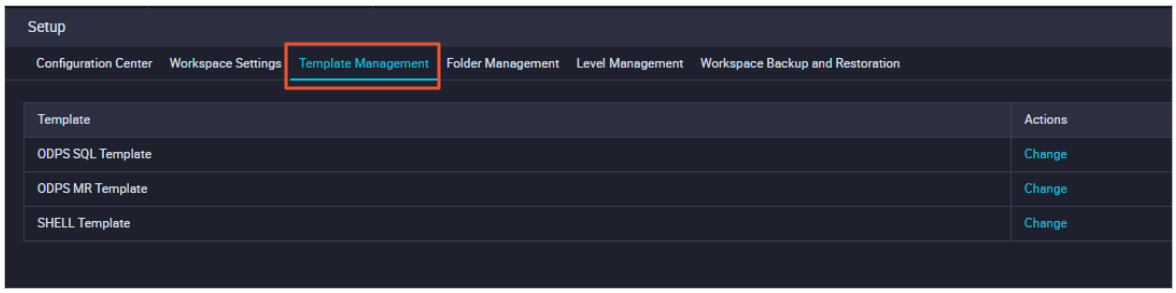
Note If you want to mask more types of data or have special requirements on the formats in which data is masked, complete your data masking settings in Data Protection. The data masking feature for DataWorks workspaces must work with Data Protection. For more information, see Overview of Data Protection.

2.1.5.9.4. 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**.

2.1.5.9.5. Folder management

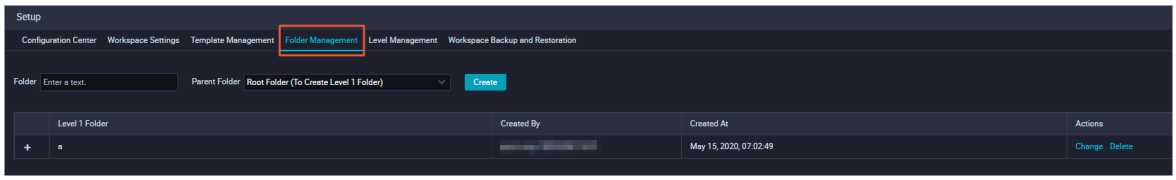
Each workspace can hold a great number of tables. For easy management, you can organize tables in two levels of folders.

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

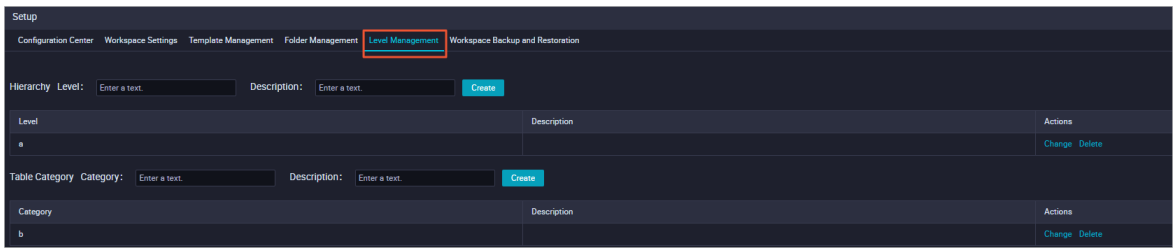
  Find the target folder and click **Delete** in the Actions column. In the **Delete Folder** message, click **OK**.

2.1.5.9.6. 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**.

2.1.5.9.7. 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.

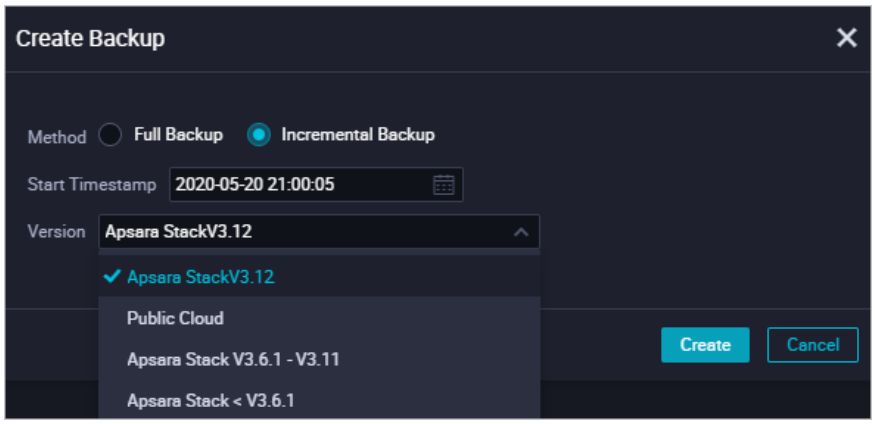
   On the page that appears, you can back up and restore workspaces.
   - On the Backup tab, you can compress the node code, node dependencies, resources, and functions in a workspace into one package.
   - On the Restore tab, you can restore a workspace to its original scheduling settings. After the workspace 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**.
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.

2.1.5.10. Deploy

2.1.5.10.1. 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.

2.1.5.10.2. 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.

2.1.5.10.3. 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.

2.1.5.11. 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.

2.1.5.12. 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.

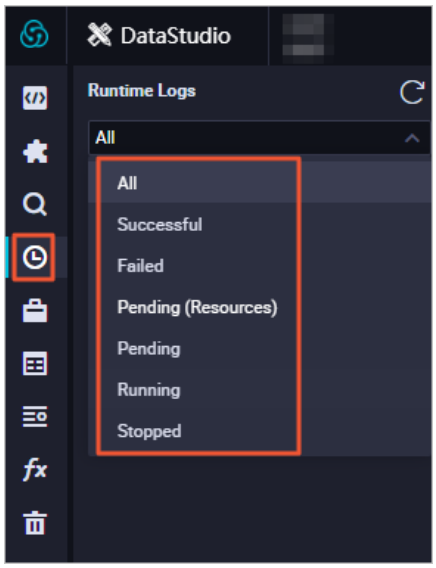
Context

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.

2.1.5.13. View tenant tables

On the Tenant Tables tab, you can view tables of all workspaces of the current tenant account.

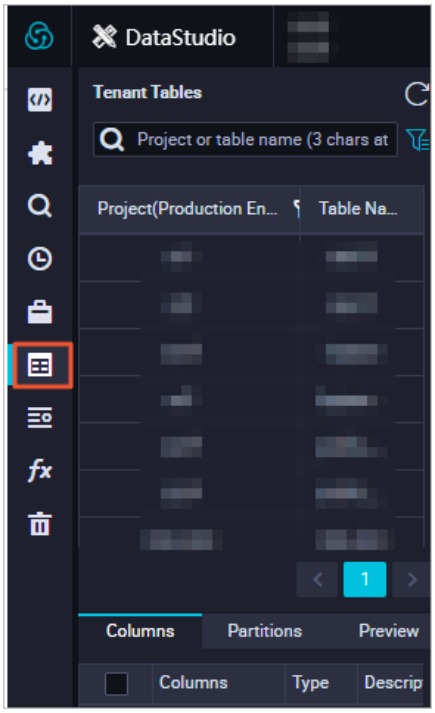
Prerequisites

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.
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.

2.1.5.14. 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
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.

2.1.5.15. 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.

2.1.5.16. 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.

2.1.5.17. 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.
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.

No.	Icon or tab	Description
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.

2.1.5.18. 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.

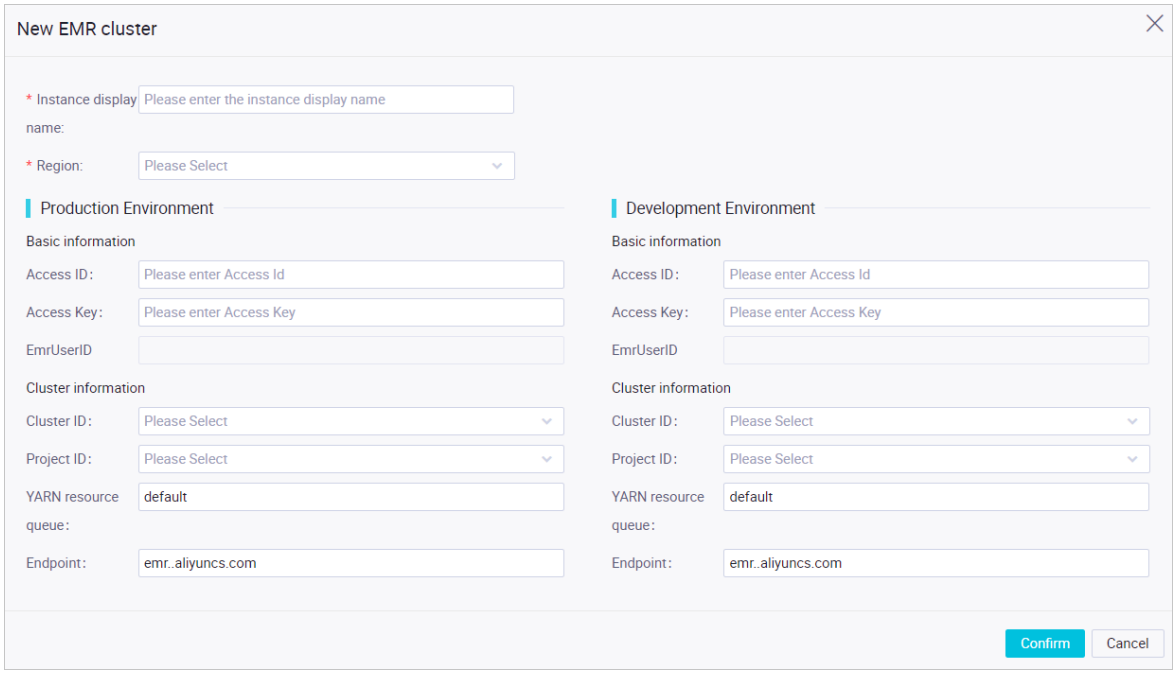
2.1.5.19. Use E-MapReduce in DataWorks

This topic describes how to use E-MapReduce (EMR) in DataWorks.

Bind an EMR project to a DataWorks workspace

Note Before you bind an EMR project to a DataWorks workspace, you must obtain the information about the EMR project.

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-right corner. The **Project Management** page appears.
3. In the **Computing Engine information** section of the **Project Management** page, click the **E-MapReduce** tab. On this tab, you can view the information about all available EMR compute engine instances in the current workspace.
4. Click **Add instances**.
5. In the **New EMR cluster** dialog box, configure the parameters.



Parameter	Description
Instance display name	The display name of the compute engine instance.
Region	The region where the current workspace resides.
Access ID	The AccessKey ID of the account that is authorized to access the EMR cluster.

Parameter	Description
Access Key	The AccessKey secret of the account that is authorized to access the EMR cluster.
Cluster ID	The ID of the EMR cluster.
EmrUserID	The ID of the user who created the EMR cluster.
Project ID	The ID of the project that is associated with the EMR cluster.
YARN resource queue	The name of the resource queue in the EMR cluster. Unless otherwise specified, set the parameter to *default*.
Endpoint	The endpoint of the EMR cluster. You can obtain the endpoint from the EMR console.

6. Click **Confirm**. After the EMR project is bound to your workspace, you can create EMR nodes on the **DataStudio** page.

> **Note** If the binding fails, check whether the failure is caused by one of the following reasons:
> - The EMR user ID is bound to another tenant account.
> - The specified cluster name already exists.

Create an EMR node

EMR nodes are categorized into four types: EMR Hive, EMR Spark SQL, EMR Spark, and EMR MR. For more information, see Create an EMR node.

Reference resource files

EMR resource files are categorized into two resource types: EMR JAR and EMR File.

Reference EMR resource files by using the following methods:

- For EMR Hive and EMR MR nodes, add `--@resource_reference{"Resource name"}` to the first line of the code.
- For EMR Spark nodes, add `##@resource_reference{"Resource name"}` to the first line of the code.

Manage data

DataWorks allows you to query EMR metadata and synchronize it for data development.

2.1.5.20. Migrate nodes in DataStudio

This topic describes how to migrate nodes in DataStudio that are created in a workflow of an earlier version to a workflow of a later version.

Migrate nodes in DataStudio

1. Log on to the DataWorks console. The DataStudio page appears.
2. Click **Business Flow** in the DataStudio pane to show all the created workflows. In the preceding figure, the workflow indicated by 1 is a workflow of an earlier version, and the workflow indicated by 2 is a workflow of a later version.
3. Click the desired workflow to open it and click Data Integration. Then, right-click a node that you want to migrate in the Data Integration folder, and select **Move**.
4. In the **Move Node** dialog box, set **Location** to the path of the destination workflow of a later version.

5. Click **OK**.

2.1.6. HoloStudio

2.1.6.1. Overview

Holo Studio provides Hologres users with standardized, easy-to-use development management services and end-to-end real-time data warehouse construction services by using a visualized user interface and a wizard. In addition to standard management available in PostgreSQL, Holo Studio provides more interactive analytics features.

Description

- **Hologres management**

  Holo Studio allows you to create a PostgreSQL table on a visual user interface or by using SQL statements.

- **Data development**

  Holo Studio allows you to create multiple external tables at a time and synchronize data based on the underlying capabilities of DataWorks. Holo Studio also provides the end-to-end, stable, and efficient extract, transform, load (ETL) service.

- **SQL Console**

  The SQL Console of Holo Studio allows you to use an SQL editor to perform data development and queries.

- **Visualized control**

  Holo Studio seamlessly interconnects with the Hologres console and allows you to easily and rapidly manage Hologres instances and instance objects, such as users and databases.

2.1.6.2. Bind a Hologres database to Holo Studio

Before you use Holo Studio on Apsara Stack, you must bind a Hologres instance to Holo Studio. If you are not authorized to access the instance, use your Alibaba Cloud account to apply for the access permissions on the Hologres instance. This topic describes how to bind a Hologres instance on which you have obtained the access permissions to Holo Studio as a superuser.

> **Note**
>
> - The user who has obtained the access permissions on the Hologres instance is the superuser of the instance. Other users can access the Hologres instance only after the users are granted the related permissions by the superuser.
> - It is optional for common users to apply for the access permissions on a Hologres instance. A common user can use Hologres if the user can use a database in the instance. To obtain more permissions, a common user must submit a request to the superuser.

1. **Log on to Holo Studio**

   Log on to the Apsara Uni-manager Management Console. In the top navigation bar, choose **Products > DataWorks** to go to the DataStudio page. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Holo Studio** to go to Holo Studio.

2. **Bind a database to Holo Studio**

On the Holo Studio page, click the icon in the left-side navigation pane to go to the PG management tab.

Move the pointer over the icon and select **Database**. In the Create Database dialog box, configure the parameters. After you configure all the parameters, click **Test connectivity**. If the system displays a message indicating that the connectivity test is passed, the database is connected. Then, click **Completed**.

Parameter	Description	Remarks
Connect To	The type of the data source. Default value: Interactive Analytics.	The value cannot be changed.
Server	The endpoint of the Hologres instance.	The value of this parameter is automatically generated after the Hologres instance is created.
Port	The port number of the Hologres instance.	The value of this parameter is automatically generated after the Hologres instance is created.
Database Name	The name of the Hologres database.	By default, a database named **postgres** is generated after the access permissions on the Hologres instance are requested. You can directly bind the database to the workspace. For actual business scenarios, you need to bind a new database.
User name	The AccessKey ID of your Apsara Stack tenant account.	You can click User Info in the upper-right corner to view the AccessKey ID.
Password	The AccessKey secret of your Apsara Stack tenant account.	You can click User Info 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	You can click this button to test whether the database is connected.	None.

Note If the Hologres instance is deleted, the instance and all databases within the instance are unavailable. You must create another instance and bind it to Holo Studio again.

3. **Create another database to develop data**

After the Hologres instance is bound to the workspace, click the icon on the **PG management** tab, and the new database appears. Then, you can use the new database to develop data.

By default, a database named postgres is created after a Hologres instance is created. However, the resources that are allocated to this database are insufficient. Therefore, to meet your business requirements, we recommend that you create a database and use the database to develop data. For more information about how to create a database, see Database management.

2.1.6.3. SQL Console

The SQL Console in Holo Studio is an editor for executing SQL statements. In the SQL Console, you can execute SQL statements to analyze data in Hologres and quickly obtain the query results. This topic describes the basic features and usage of the SQL Console in Holo Studio.

Folder

The Folder module stores new ad hoc queries, which allows you to easily manage ad hoc queries.

Go to the DataStudio page. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Holo Studio**.

In the left-side navigation pane of the **Holo Studio** page, click the icon. The **SQL Console** tab appears. Move the pointer over the icon and select **Folder**. In the Create Folder dialog box, specify Folder Name to create a folder. The name can contain letters and underscores. You can create an ad hoc query in the folder and execute standard SQL statements to perform operations on tables. You can also right-click a table in the folder and select the relevant options to move, rename, or delete the table.

SQL Console

The SQL Console module can generate an SQL editor in which you can execute standard SQL statements.

1. In the left-side navigation pane of the **Holo Studio** page, click the icon. Then, the **SQL Console** tab appears.
2. Move the pointer over the icon and select **SQL Console**. In the Create Node dialog box, configure basic information for the node.

   You can also click **Create SQL Console** on the right side of the **SQL Console** tab to configure the node.

Parameter	Description
Node Name	The name of the ad hoc query node. The name can contain letters, digits, underscores (_), and periods (.).
Location	The folder in which the ad hoc query node is stored.
Database	The database to which the ad hoc query applies.

3. Write SQL statements used in the ad hoc query and click the icon in the top toolbar to run the ad hoc query node. Then, you can view the query result. The following example shows how to create a table, import data to the table, and query the data in 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,    'e
ach slyly above the careful'),
(6,    'Supplier#000000006',    'tQxuVm7s7CnK',    14,    '24-696-997-4969',    136579,    'fina
l 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', 88
990, '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;
```

Note The SQL Console does not support transaction-related statements, such as `BEGIN, COMMIT, and CALL_SET_TABLE`. If you want to execute such SQL statements, go to the DataStudio page.

Parameter	Description
SQL editor	You can write SQL statements in the SQL editor.
Save	You can click this icon to save all statements in the SQL editor.
Run	You can click this icon to run all statements in the SQL editor. The result appears at the lower part of the tab. You can also select an SQL statement to be run. In this case, the system only runs this statement.
Reload	You can click this icon to refresh the content in the SQL editor. The system retains only the saved content after the refresh.
Stop	You can click this icon to stop running SQL statements.
Operational Log	You can check the running results and error messages.
Result	You can check the table content after the SQL statements are run.

Holo Studio 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.

2.1.6.4. PostgreSQL management

2.1.6.4.1. Manage databases

The PG management module of Holo Studio allows you to manage databases and tables with one click in a visualized manner. In addition, Holo Studio supports interactive queries with a response time within seconds. This topic describes how to manage databases on the PG management tab.

Bind a Hologres database to a workspace

Before you bind a Hologres database to a workspace, you must create a Hologres database. You can create a database in the Hologres console in a visualized manner. You can also create a database in the SQL Console by using SQL statements. To create a Hologres database, perform the following steps:

1. Create a database.

   Log on to the Hologres console. Then, click **Instances** in the left-side navigation pane. On the Instances page, click **DataWorks**. On the DataStudio page, click the icon in the upper-left corner, move the pointer over **All Products**, and then choose **All Products** > **Data Development** > **Holo Studio**. On the **Holo Studio** page, click the icon in the left-side navigation pane. Then, the SQL Console tab appears.

   Enter the following SQL statement in the SQL Console to create a database:

   ```
   create database dbname;
   create database testdb; // Create a database named testdb.
   ```

2. Bind the database.

   After the database is created, click the PG management icon in the left-side navigation pane of the Holo Studio page to bind the database. On the PG management tab, move the pointer over the first icon on the right side of PG management and select **Database** to bind the database.

3. Configure the database.

   In the **Create Database** dialog box, configure the parameters and click **Test connectivity**. If the system displays a message indicating that the connectivity test is passed, the specified database is connected. Then, click **Completed**.

Parameter	Description	Remarks
Connect To	The type of the data source. Default value: Interactive Analytics.	The value 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.

Parameter	Description	Remarks
Database Name	The name of the created database.	The value of this parameter must be the same as the name of the database created by using the SQL statement, such as **testdb**.
User name	The AccessKey ID of your Apsara Stack tenant account.	You can click User Info in the upper-right corner to view the AccessKey ID.
Password	The AccessKey secret of your Apsara Stack tenant account.	You can click User Info 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	You can click this button to check whether the database is connected.	If the system displays a message indicating that the connectivity test is passed, the specified database is connected.

Delete a database

The PG management module of Holo Studio allows you to delete a database. On the left-side navigation pane, click the **PG management** icon. On the PG management tab, find the database that you want to delete, right-click the database name, and then select **Delete Database**.

In the Delete Database message, click **Ok**.

> **Note** Only a superuser of a database or the database owner configured by a superuser can delete the database.

2.1.6.4.2. Manage tables

Similar to PostgreSQL, Hologres manages data by using tables. The PG management module of Holo Studio allows you to manage tables in a visualized manner. You can quickly create, check, or delete a table. This topic describes how to use the PG management module of Holo Studio to manage tables.

Create a table

1. **Create a table.**

   On the Holo Studio page, click the  in the left-side navigation pane to go to the PG management tab. On the PG management tab, move the pointer over the  icon and select **Table** to create a table.

   You can also click **Create Table** on the right side of the **PG management** tab to create a table.

2. **Configure the table.**

   On the tab that appears, configure the parameters for the table and click **Commit**. The following figure shows an example of how to create a common column-oriented table that has a primary key.

Item	Parameter	Description
General	Interactive Analytics Database	The database to which the table belongs.
	Table Name	The name of the table.
	Description	The description of the table.
Field	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.
	Optional	Specifies whether the field can be empty.
	Array	Specifies whether the field is an array.
	Description	The description of the field.
	Actions	The operations that you can perform on the field. For example, you can delete the field from the table, or move up 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 data in the table. Default value: Permanent.
	Clustering Index	The index used for sorting.
	Dictionary Code Columns	The column whose values are used to build a dictionary mapping.
	Bitmap Column	The column on which bit code is built.
Partitioned Table	PARTITION BY LIST	The partition field.

View a table

1. **Generate DDL statements.**

   In the left-side navigation pane, click **PG management**. On the PG management tab, double-click the table that you want to view and click **Generate DDL Statement**. Then, you can view the SQL statement used to create the table.

2. **Preview data.**

   In the left-side navigation pane, click **PG management**. Double-click the table you want to view and click **Data Preview** to view the table data. If the table contains no data, you can only view the fields of the table.

Delete a table

In the left-side navigation pane, click **PG management**. Right-click the table that you want to delete and select **Delete Table** to delete the table.

In the Delete message, click **OK**.

2.1.6.4.3. 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.

2.1.6.5. Data analytics

2.1.6.5.1. 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**.

2.1.6.5.2. 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.

2.1.6.5.3. 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
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

Parameter	Description	Remarks
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.

2.1.6.5.4. 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.

Section	Parameter	Description	Remarks
	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.
	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.

2.1.6.5.5. 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
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.

2.1.6.6. Hologres console

2.1.6.6.1. 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 trillions of data records. Hologres allows you to use mainstream Business Intelligence (BI) tools to gain an analytical insight into data from multiple dimensions and explore business data in an efficient and cost-effective manner.

For the convenience of business, Apsara Stack provides the Hologres console independent of the DataWorks console for you to manage Hologres instances, users, and databases.

Log on to the Apsara Uni-manager Management Console. In the top navigation bar, choose **Products > Hologres**. On the page that appears, specify Organization and Region, and click Access as Administrator to go to the Hologres console.

2.1.6.6.2. View the instance list

The Instances page lists all the Hologres instances that belong to your Apsara Stack tenant account. On this page, you can perform operations such as viewing instance statuses, changing instance configurations, and creating instances. You can also click Manage in the Actions column that corresponds to an instance to go to the instance details page. On this page, you can manage objects in the instance, including databases and users.

Instance list

Log on to the Apsara Uni-manager Management Console. In the top navigation bar, choose **Products > Hologres**. On the page that appears, specify Organization and Region, and click Interactive Analytics to go to the Hologres console. In the left-side navigation pane of the Hologres console, click **Instances**.

1. **Create Instance button**

   On the Instances page, click **Create Instance**. In the Create Instance dialog box, enter an instance name and select 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 desired instance.

3. **Status column**

The Status column displays the running status of each Hologres instance. An instance can be in one of the following states:

- Running: The instance is running as expected.
- Creating: The instance is being created. You must wait for 3 minutes to 5 minutes.
- Stop: The instance is suspended, and you cannot connect to it.

Actions column

The Actions column allows you to perform the following operations on Hologres instances:

1. **Manage**

   Find the desired instance and click **Manage** in the Actions column. On the page that appears, you can view and manage objects in the instance, including databases and users.

2. **Configure**

   If your instance cannot meet your business requirements or has a large number of surplus resources, you can click **Configure** in the Actions column. In the **Configure** dialog box, change the specifications of the instance based on your business requirements to upgrade or downgrade the instance.

3. **Stop**

   Find the desired instance and click **Stop** in the Actions column to suspend the instance. After the state of the instance changes to **Stop**, the instance is suspended, and you cannot connect to it.

2.1.6.6.3. 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
   ```

2.1.6.6.4. 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.

2.1.6.6.5. 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.

2.1.7. DataAnalysis

2.1.7.1. Overview

DataAnalysis provides the features of building and sharing workbooks, dimension tables, and visual reports. These features help developers and business personnel analyze data.

Benefits

Compared with offline data analysis, DataAnalysis has the following benefits:

- High efficiency: DataAnalysis analyzes data in a database by using online data analysis tools, such as pivot tables. For example, in a user profile table, you can create a pivot table for a partition that belongs to September. When data for October is obtained, you can update the source data and reuse the configuration of the pivot table for September to obtain a pivot table for October. This avoids repeated operations.
- High capacity: DataAnalysis efficiently analyzes large amounts of data with the help of compute engines.
- Data sharing: DataAnalysis can analyze data obtained from the databases of different business systems. DataAnalysis allows you to export data to MaxCompute tables. It also allows you to share data with specified members and grant them access permissions. This way, data can be shared between different systems and different users.
- High security: DataAnalysis allows you to analyze data online without the need to download data. It also allows you to manage the permissions that allow users to analyze and share data.

Description

- **Workbook**

  Workbooks are the core feature of DataAnalysis. A workbook is a workspace in which you can obtain, explore, and analyze data. A workbook is in the form of an online table and offers common table features. After you import data from data sources or import local data to workbooks, you can perform data pivoting and profiling.

- **Dimension table**

  The dimension table feature allows you to create MaxCompute tables without the need to write SQL code. It also allows you to collaboratively edit MaxCompute tables with other users online. The dimension table feature also allows you to import data to MaxCompute tables in a visualized manner. For more information, see Create and manage dimension tables.

- **Report**

  You can create and design visual reports by dragging and configuring controls without the need to execute SQL statements.

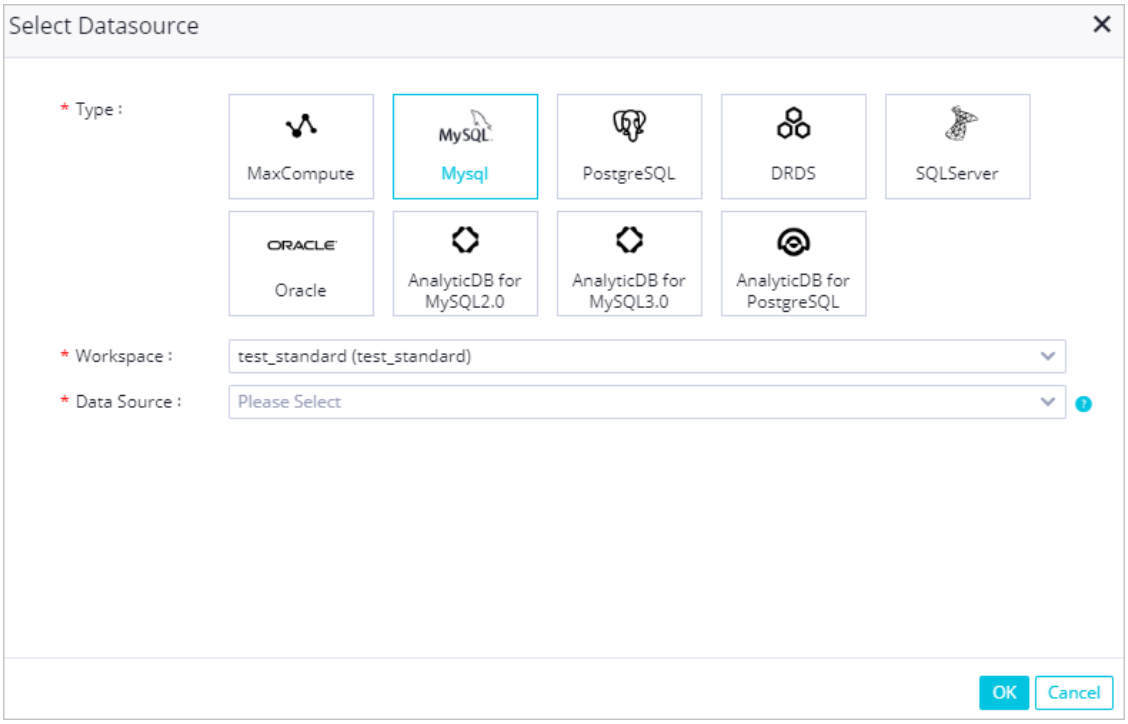
2.1.7.2. SQL queries

DataWorks allows you to import data from a data source and perform queries and analysis of data by using the SQL query feature. This topic describes how to use the SQL query feature.

Create an SQL query task

1. Go to the DataAnalysis page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click the icon and choose **All Products > Data Development > DataAnalysis**. The **DataAnalysis** page appears.
2. On the **DataAnalysis** page, click **SQL Query** in the top navigation bar to go to the **SQL Query** page.
3. Specify the data source to be queried.
   i. On the left side of the **SQL Query** page, click the + icon in the **Data source** section.

   The first time you go to the **SQL Query** page, you must click **Add Now** first.
   ii. In the **Select Datasource** dialog box, set the parameters as required.

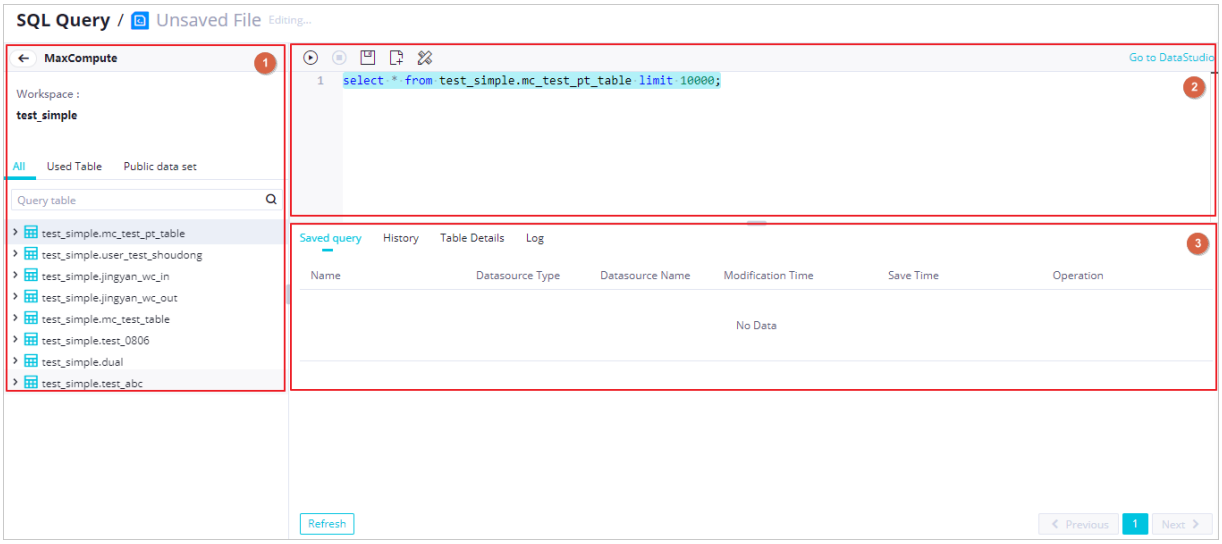


Parameter	Description
Type	Specify the type of the data source to be queried.
Workspace	The workspace in which the specified data source resides.
Data Source	Select the data source to be queried from the drop-down list. **Note** If you set the **Type** parameter to MaxCompute, this parameter is not displayed.

   iii. Click **OK** to create an SQL query task. On the SQL Query page, you can use SQL statements to query and analyze data for the created SQL query task.

Perform and manage SQL queries

On the SQL Query page, you can use SQL statements to query and analyze data from the specified data source. You can also view saved queries, execution history, and logs of the SQL queries.



Area No.	Description
1	This section displays the tables that are contained in the specified data source for which you perform the SQL queries in the current workspace. You can also search for a table by entering keywords in the search box.
2	This section allows you to run SQL statements to query data. After you enter the SQL statement to be executed, click the icon in the toolbar to run the statement. You can also click **Go to DataStudio** to copy the SQL statement and go to the DataStudio page. On the DataStudio page, you can develop data and schedule ETL tasks.
3	In this section, you can view the details of the saved queries, execution history, and logs of SQL queries. You can also preview, load, rename, and delete historical SQL queries on the Saved query tab.

2.1.7.3. Workbook

2.1.7.3.1. Create and manage workbooks

Before data analysis, you must create a workbook to store the data to be analyzed. This topic describes how to create, view, and manage workbooks.

Create a workbook

1. Go to the DataAnalysis page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click the icon and choose **All Products > Data Development > DataAnalysis**. The **DataAnalysis** page appears.
2. In the top navigation bar of the DataAnalysis page, click **Spreadsheet**.

3. On the **Spreadsheet** page, click the + icon in the **New Spreadsheet** section to go to the workbook editing page. On this page, you can import data for analysis.

View and manage workbooks

1. On the workbook editing page, click **Spreadsheet** in the upper-left corner or **Spreadsheet** in the top navigation bar to go back to the Spreadsheet page.
2. In the **All Spreadsheets** section of the **Spreadsheet** page, select **I created** or **Share it with me** from the drop-down list in the upper-right corner to view the workbooks in the corresponding category.
3. Click the file name of a workbook to go to the workbook editing page.

   In the All Spreadsheets section, you can also perform the following operations on a workbook:

   - To rename a workbook, find the workbook and click the icon in the Operation column. In the **Rename** dialog box, enter a new name in the **File Name** field and click **OK**.
   - To change the owner of a workbook, find the workbook and click the icon in the Operation column. In the **Change Owner** dialog box, select an owner from the drop-down list and click **OK**.
   - To clone a workbook, find the workbook and click the icon in the Operation column. The generated workbook appears in the workbook list. The name of the generated workbook contains the _copy suffix.
   - To delete a workbook, find the workbook and click the icon in the Operation column. In the **Delete** message, click **OK**.

2.1.7.3.2. 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. Go to the DataAnalysis page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click the icon and choose **All Products > Data Development > DataAnalysis**. The **DataAnalysis** page appears.
2. On the DataAnalysis page, click **Spreadsheet** in the top navigation bar to go to the **Spreadsheet** page.
3. On the Spreadsheet page, find the workbook that you want to edit in the **All Spreadsheets** section and click the name of the workbook that you want to edit in the **File Name** column to go to the workbook editing page.

Apply a template to a workbook or save a workbook as a template

You can perform the following steps to apply an existing template to the current workbook:

1. In the upper-right corner of the workbook editing page, choose **Template > Import Template**.
2. In the **Import Template** dialog box, click the template you want to use.

   > **Note** The data of the template will overwrite that of the current workbook.

3. Click **OK**.

You can also perform the following steps to save the current workbook as a template:

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 cannot exceed 256 characters in length, and the template description cannot exceed 1,024 characters in length.

3. Click **OK**.

Import the data on your on-premises machine

On the workbook editing page, move the pointer over **Import** in the upper-right corner and select **Local File** or **File Data** to import the data on your on-premises machine.

- If you select **Local File**, you can import only Excel files from your on-premises machine. Data in all the sheets of a selected Excel file is imported.

  Choose **Import > Local File**. Select the Excel file that you want to import and click **Open** to import data in all the sheets of the Excel file to the workbook.

- If you select **File Data**, you can import data from workbooks or import CSV files or Excel files from your on-premises machine. If you import data from a workbook or an Excel file, you can specify the sheet from which you want to import data.

  Choose **Import > File Data**. In the **Import** dialog box, select one of the following types of source files based on your business requirements:

  - **Spreadsheet**

    In the **Import** dialog box, click **Spreadsheet**, set the parameters, and then click **OK**.

Parameter	Description
Spreadsheet	The workbook from which you want to import data. Select a workbook from the **Spreadsheet** drop-down list.
Sheet	The sheet from which you want to import data. Select a sheet from the **Sheet** drop-down list.
Data Preview	Displays the data in the selected workbook.
Import Start Row	The row from which the data starts to be imported. Default value: 1.
Placement Location	The location where the data you want to import is placed. Valid values: **Current Worksheet** and **New Worksheet**.
Placement Method	The way in which the data you want to import is placed. Valid values: **Append**, **Overwrite**, and **Active Cell**.

- **Local CSV File**

  In the **Import** dialog box, click **Local CSV File**, set the parameters, and then click **OK**.

Parameter	Description
File	The CSV file from which you want to import data. Click **Select File(.csv)**, select the required CSV file from your on-premises machine, and then click **Open**.
Original Character Set	The character set that is used by the selected CSV file. Valid values: **UTF-8** and **GBK**. If garbled characters appear, you can change the character set.
Separator	The row delimiter and column delimiter. ■ The following row delimiters are supported: \r\n, \n, and \r. ■ The following column delimiters are supported: commas (,), semicolons (;), and \t. If the cell data cannot be correctly separated, you can change the delimiters.
Data Preview	Displays the data in the selected CSV file.
Import Start Row	The row from which the data starts to be imported. Default value: 1.
Placement Location	The location where the data you want to import is placed. Valid values: **Current Worksheet** and **New Worksheet**.
Placement Method	The way in which the data you want to import is placed. Valid values: **Append**, **Overwrite**, and **Active Cell**.

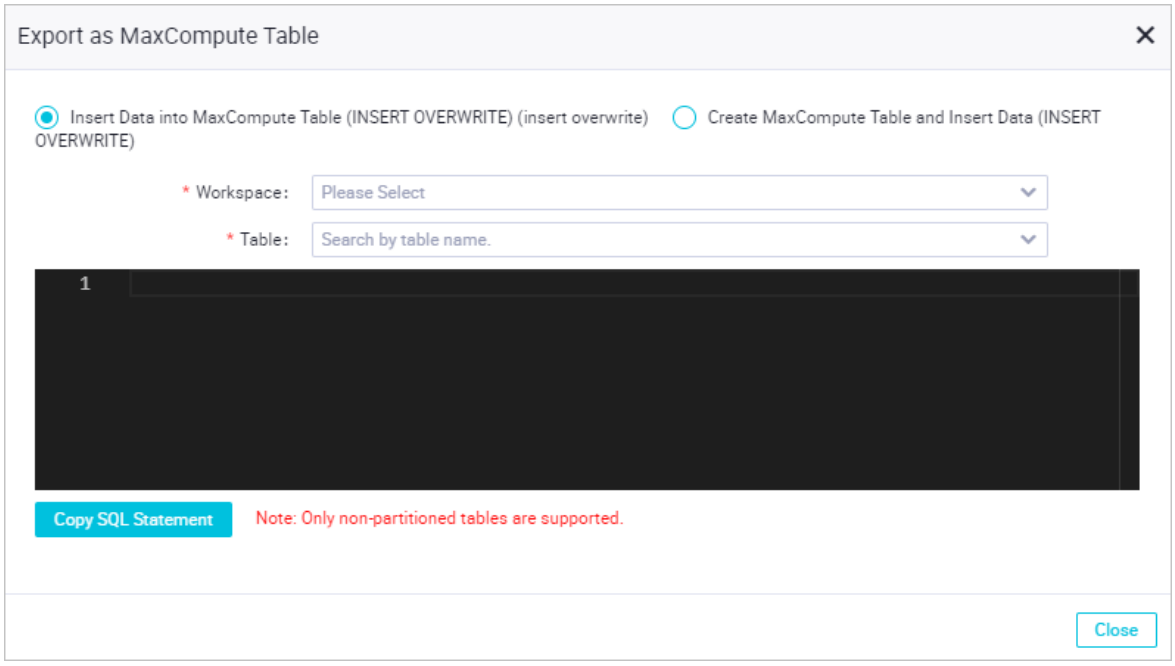
- **Local Excel file**

  In the **Import** dialog box, click **Local Excel File**, set the parameters, and then click **OK**.

Parameter	Description
File	The Excel file from which you want to import data. Click **Select File(.xlsx)**, select the required Excel file from your on-premises machine, and then click **Open**.
Sheet	The sheet from which you want to import data. Select a sheet from the **Sheet** drop-down list.
Data Preview	Displays the data in the selected Excel file.
Import Start Row	The row from which the data starts to be imported. Default value: 1.
Placement Location	The location where the data you want to import is placed. Valid values: **Current Worksheet** and **New Worksheet**.
Placement Method	The way in which the data you want to import is placed. Valid values: **Append**, **Overwrite**, and **Active Cell**.

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 the parameters.



Insert mode	Parameter	Description
Insert Data into MaxCompute Table (INSERT OVERWRITE) (insert overwrite)	**Workspace**	The workspace corresponding to the MaxCompute project 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 corresponding to the MaxCompute project to which the MaxCompute table belongs.
	Table Name	The name of the MaxCompute table that you want to create and to which you want to insert data. Make sure that the table name is unique. You can click **Check Duplicate Names** to check whether the table name already 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, specify the data that you want to analyze.

   You can set the Choose Data parameter to **Select Range** or **Use External Data Source**.

   - **Select Range**

The value of the **Range** parameter changes when you select the cells for which you want to create a pivot table.

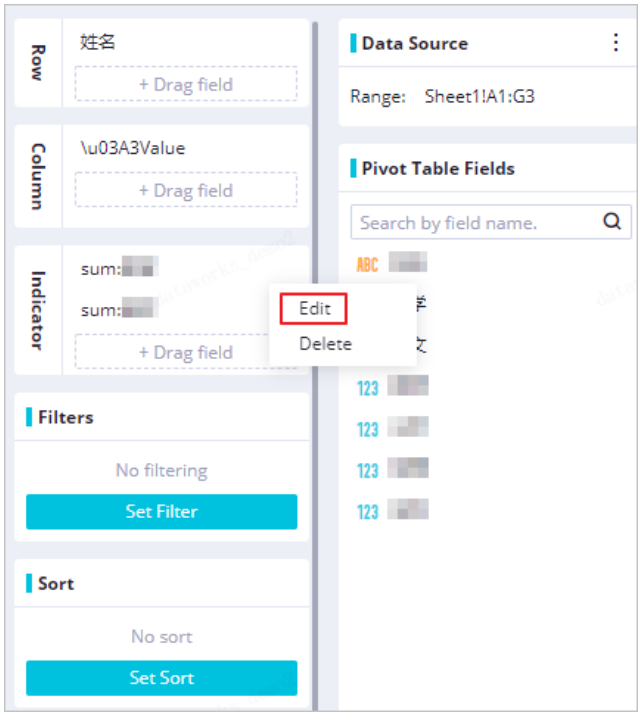
- **Use External Data Source**

  You can select a data source of the **MaxCompute**, **MySQL**, **Data Services**, or **PostgreSQL** type. If you use an external data source, prepare the required data source or API in advance and select the data source or API based on the business requirements. For more information about how to configure a data source, see Data sources.

3. Click **OK**. The pivot table editing panel appears.

   In this example, **Select Range** is selected.

   - **Data Source**: the range of the selected data in the workbook.
   - **Pivot Table Fields**: the names of the fields in the selected data.
   - **Row**: Drag a field to the **Row** section. Each value of the field added to the **Row** section occupies a row in the pivot table.
   - **Column**: Drag a field to the **Column** section. Each value of the field added to the **Column** section occupies a column in the pivot table.
   - **Indicator**: To modify the settings of a metric, move the pointer over the metric, click the ⋮ icon, and then select **Edit**.



   In the **Property settings** dialog box, set the **Summary method** parameter and click **OK**. By default, the display name of the metric is in the format of **Aggregation method:Source field name** and cannot be modified.

   - **Filters**: To filter data, click **Set Filter**. In the **Set Filter** dialog box, click **Add Condition**, specify the filter conditions, and then click **OK**.
   - **Sort**: To sort data based on a field specified in the **Row** section, click **Set Sort**. In the **Set Sort** dialog box, specify the sorting rule and click **OK**.

Share and download a workbook

> **Note** To share and download a workbook, you must choose More > Management in the top navigation bar, and turn on **Allow Sharing** and **Allow Download** in the Spreadsheet section of the **Configuration Management** tab. Then, you can share and download a workbook.

In the upper-right corner of the workbook editing page, click **Share**. In the dialog box that appears, configure the sharing mode.

You must configure the following information before you can share a workbook:

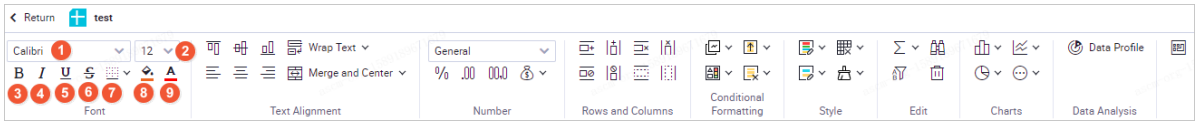
- **Link**: After you specify Users with Edit Access and Users with Read Access, click **Copy Link** to copy the URL and send the copied URL to specified users.
- **Users with Edit Access**: In the **Users with Edit Access** section, click **Add**. In the dialog box that appears, select members with whom you want to share the workbook and click **OK**.
- **Visible to All**: To share the workbook with all members, turn on **Visible to All**.
- **Users with Read Access**: To share the workbook with some members, turn off **Visible to All** and click **Add** in the **Users with Read Access** section. In the Share File with These Users dialog box, select members with whom you want to share the workbook and click **OK**.

> **Note** If you are prompted that the number of members with whom you want to share the workbook reaches the upper limit, you can upgrade the DataWorks edition to increase the upper limit.

In the upper-right corner of the workbook editing page, click **Download** to download the workbook to your on-premises machine.

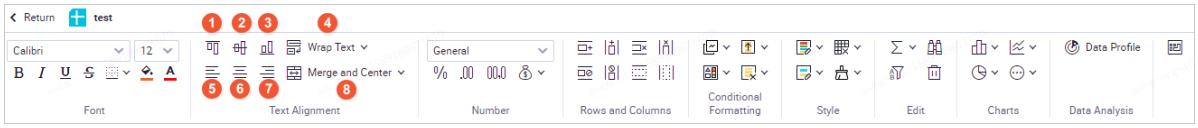
Menu bar

- **Font**



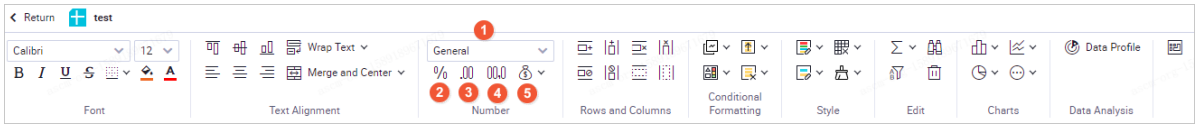
No.	GUI element	Description
①	**Font**	Select a font based on your business requirements.
②	**Font size**	Select a font size based on your business requirements.
③	**Bold**	Set text in bold.
④	**Italic**	Set text in italic.
⑤	**Underline**	Underline text.
⑥	**Strikethrough**	Add a strikethrough to text.
⑦	**Borders**	Add borders to text.
⑧	**Background color**	Specify the background color of text.
⑨	**Text color**	Specify the text color.

- **Text Alignment**



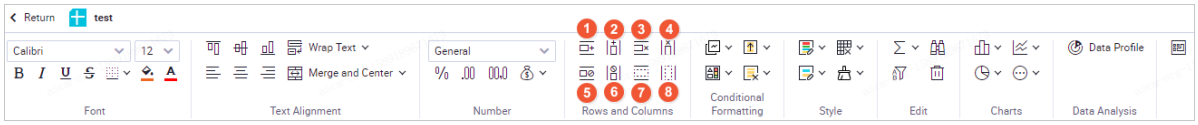
No.	GUI element	Description
①	**Top Align**	Align text to the top.
②	**Middle Align**	Vertically align text to the center.
③	**Bottom Align**	Align text to the bottom.
④	**Wrap Text**	Display long text in multiple lines to make it easy to view all the text.
⑤	**Align Left**	Align text to the left.
⑥	**Center**	Align text to the center.
⑦	**Align Right**	Align text to the right.
⑧	**Merge and Center**	Merge multiple cells to one cell and center the text in the cell.

- Number



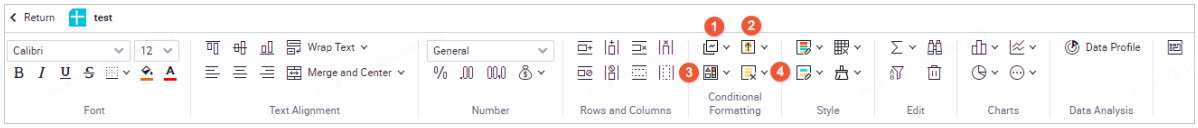
No.	GUI element	Description
①	**Number format**	Specify the number format for selected cells. You can select General, Number, Currency, Short Date, Long Date, Time, Percentage, Fraction, Scientific, or Text from the drop-down list.
②	**Percentage**	Apply the percentage format to numbers.
③	**Two Decimal Places**	Round numbers to two decimal places.
④	**1000 Separator**	Display numbers with thousands of separators, such as **1,005**.
⑤	**Currency**	Add a currency sign to numbers. Valid values: ¥ Chinese (PRC), $ English (United States), £ English (United Kingdom), € Euro, and Fr Franc.

- Rows and Columns



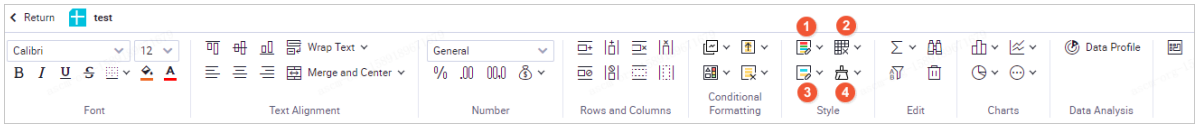
No.	GUI element	Description
①	**Insert Row**	Insert one or more rows to the workbook.
②	**Insert Column**	Insert one or more columns to the workbook.
③	**Delete Row**	Delete one or more selected rows from the workbook.
④	**Delete Column**	Delete one or more selected columns from the workbook.
⑤	**Lock Row**	Lock one or more selected rows in the workbook.
⑥	**Lock Column**	Lock one or more selected columns in the workbook.
⑦	**Hide Row**	Hide one or more selected rows in the workbook.
⑧	**Hide Column**	Hide one or more selected columns in the workbook.

- **Conditional Formatting**



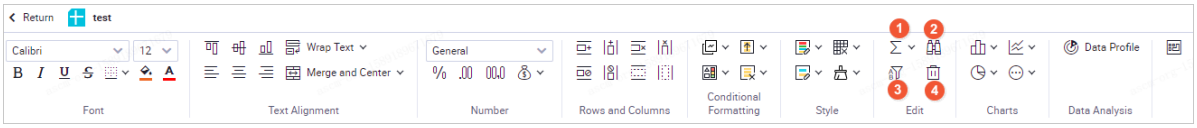
No.	GUI element	Description
①	**Highlight cell rules**	Specify the rules for highlighting cells. Highlighting rules are divided into two categories: **Highlight Cells Rules** and **Top/Bottom Rules**.
②	**Data Bar/Color Scale**	Format selected cells by using gradient or solid data bars and color scales.
③	**Icon Set**	Format selected 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**



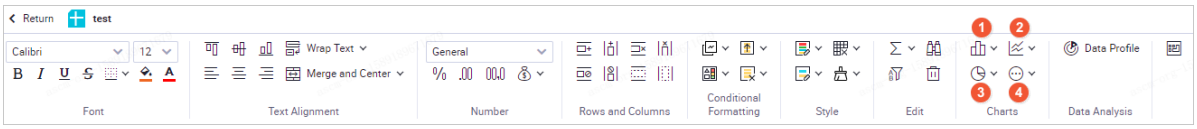
No.	GUI element	Description
①	**Apply table style**	Apply a table style.
②	**Delete**	Remove the applied table style.
③	**Cell Style**	Set the cell style.
④	**Clear**	Clear the content or style for selected cells. You can select **Clear All**, **Clear Content**, or **Clear Style** from the drop-down list.

- **Edit**



No.	GUI element	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 selected content.

- **Charts**

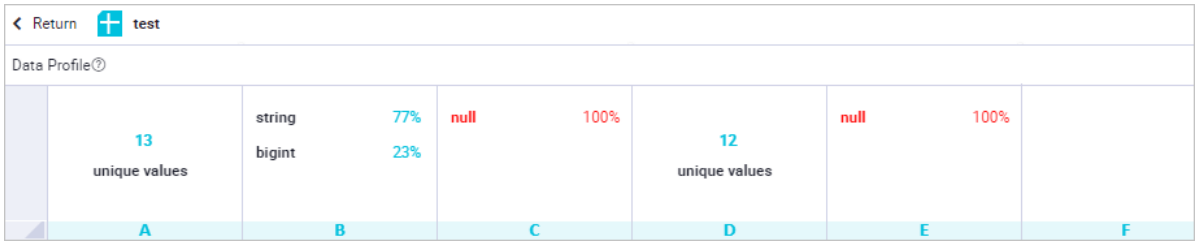


  - **Column Chart**: After you click the **Column Chart** icon, you can select **Column Chart**, **Stacked Column Chart**, and **100% Stacked Column Chart**.
  - **Line Chart**: After you click the **Line Chart** icon, you can select **Line Chart**, **Stacked Line Chart**, **100% Stacked Line Chart**, **Line Chart with Markers**, **Stacked Line Chart with Markers**, and **100% Stacked Line Chart with Markers**.
  - **Pie Chart**: After you click the **Pie Chart** icon, you can select includes **Pie Chart** and **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, detect, process, analyze, and visualize data. The data profiling feature analyzes data by column and allows you to view the data types and value distribution of each column.

Select the data that you want to analyze and click **Data Profile** in the menu bar. Then, you can view the data types and value distribution of each column above the workbook in the form of charts and rich text.



The simple mode of the data profiling feature has the following capabilities:

- For a column whose values are of the STRING or DATE data type: displays the values that rank top 2 and their percentages, and the percentage of other values in the form of rich text. If the number of unique values exceeds 50% of the total number of values, the number of unique values is displayed.
- For a column whose values are of the INTEGER or FLOAT type: displays the value distribution in the form of a histogram.
- For a column whose values are of the BOOLEAN type: displays the proportions of different values in the form of a pie chart.
- For a column whose values are of two or more data types: displays the proportions of different data types in the form of a pie chart. The system reminds you that the current column contains dirty data. After the dirty data is cleared, the simple mode displays value distribution in one of the preceding forms based on the data type.
- For a column whose values are null values: displays the percentage of the 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. The profiling result information includes the column name, value distribution information, data type, and data source.

The detailed mode of the data profiling has the following capabilities:

- For a column whose values are of the STRING or DATE type: displays basic information and the values that rank top 5 based on frequency. The basic information includes the number of fields, unique values, valid values, and the percentage of null values.
- For a column whose values are of the INTEGER or FLOAT type: displays basic information, the values that rank top 5 based on frequency, statistics, and a histogram. The basic information includes the number of fields, unique values, zeros, and the percentage of null values.
- For a column whose values are of the BOOLEAN type: displays basic information, the values that rank top 5 based on frequency, and a pie chart. The basic information includes the number of fields, unique values, zeros, and the percentage of null values.

> **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 the icon to view the shortcut keys for different features.

2.1.7.4. Dimension tables

2.1.7.4.1. Create and manage dimension tables

The dimension table feature allows you to create MaxCompute tables, import data on your on-premises machine to MaxCompute tables, and edit MaxCompute tables in a visualized manner.

Prerequisites

1. A MaxCompute compute engine instance is associated with a DataWorks workspace. For information about how to associate a MaxCompute compute engine instance, see Configure a workspace.
2. A MaxCompute table is created. For more information, see Create a MaxCompute table.

Limits

- To create a dimension table in DataAnalysis, you must be an administrator, an owner, or a developer of a DataWorks workspace.
- For a MaxCompute table that is created by using the dimension table feature, all fields in the MaxCompute table are of the STRING type. If you want to use fields of other data types, execute data definition language (DDL) statements to create a MaxCompute table on the **DataStudio** page.

Create a dimension table

1. Go to the DataAnalysis page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click the icon and choose **All Products > Data Development > DataAnalysis**. The **DataAnalysis** page appears.
2. In the top navigation bar of the DataAnalysis page, choose **More > Dimension Table** to go to the **Dimension Table** page.
3. On the **Dimension Table** page, click the icon in the **New Dimension Table** section.
4. In the **New Dimension Table** dialog box, set the parameters as required.

Parameter	Description
Target Workspace	The workspace corresponding to the MaxCompute project to which the MaxCompute table belongs.
Table Name	The name of the dimension table. A MaxCompute table will be used in the production environment. **Note** The table name can contain only letters, digits, and underscores (_), and must start with a letter.
Table Description	The description of the table, such as the purpose or features.
Field	The fields in the table. Only fields of the STRING type can be added.
Lifecycle	The lifecycle of the table. The table occupies storage resources in MaxCompute. To ensure that the resources can be recycled, select an appropriate lifecycle for the table from the drop-down list. If the specified lifecycle is exceeded, the table is deleted.

5. Select **I have known this risk and confirmed that as owner of this table, I am responsible for the subsequent changes to this table.** and click **OK** to go to the dimension table editing page to view and modify information about the table. For information about how to edit a dimension table, see Edit a dimension table.

   The MaxCompute table created in DataAnalysis is maintained in the production environment. The creator of the table is responsible for the creation and maintenance of the table.

View and manage dimension tables

1. On the dimension table editing page, click **Dimension Table** in the upper-left corner or **Dimension Table** in the top navigation bar to go back to the Dimension Table page.
2. In the **All Dimension Tables** section, select **I created** or **Share it with me** from the drop-down list in the upper-right corner to view the tables in the corresponding category.

   You can also share dimension tables with specific members. For information about how to share a dimension table, see Share a dimension table.
3. Click the file name of a required table or click **Edit** in the Operation column of the table to go to the dimension table editing page.

   In the All Dimension Tables section, you can perform the following operations on a dimension table:

   - To change the owner of a dimension table, find the table and click the icon in the Operation column. In the **Change Owner** dialog box, select an owner from the drop-down list and click **OK**.
   - To delete a dimension table, find the dimension table and click the icon in the Operation column. In the **Delete** message, click **OK**.

What's next

After you create a dimension table, go to the dimension table editing page and import data to this table. For information about how to edit a dimension table, see Import data to a dimension table.

2.1.7.4.2. Import data to a dimension table

After you create a dimension table, you can write data to the table for data analysis. You can also import data from a workbook, local CSV file, or local Excel file to the table for data analysis. This topic describes how to import data to a dimension table.

Prerequisites

A dimension table is created. For more information about how to create a dimension table, see Create a dimension table.

Procedure

1. Go to the DataAnalysis page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click the icon and choose **All Products > Data Development > DataAnalysis**. The **DataAnalysis** page appears.
2. In the top navigation bar of the DataAnalysis page, choose **More > Dimension Table** to go to the **Dimension Table** page.
3. In the **All Dimension Tables** section, click the name of the dimension table in the **File Name** column to go to the dimension table editing page.

   If no dimension table exists and you create one, the table editing page appears after the table is created. For more information about how to edit a dimension table, see Edit a dimension table.
4. On the dimension table editing page, click **Import** in the upper-right corner.
5. In the **Import** dialog box, select the type of the file that contains the data to be imported and set parameters.

   > **Note** Only data of the STRING type can be imported to a dimension table. Data that is not of the STRING type will be automatically converted to the STRING type when it is imported.

   - **Spreadsheet**

Parameter	Description
Spreadsheet	The workbook from which you want to import data. Select a workbook from the **Spreadsheet** drop-down list.
Sheet	The sheet from which you want to import data. Select a sheet from the **Sheet** drop-down list.
Data Preview	Displays the data in the selected sheet. When you preview the data in the selected sheet, you can determine whether to use the values in the first row as the column names by selecting or clearing **First Row as Field Names**.
Field Mapping	The mappings between the columns in the selected sheet and the fields in the dimension table.
Import Data Mode	The mode used to import data. Valid values: **Append** and **Overlay**.

- Local CSV File

Parameter	Description
File	The CSV file from which you want to import data. Click **Select File(.csv)**, select the required CSV file from your on-premises machine, and then click **Open**.
Original Character Set	The character set that is used by the selected CSV file. Valid values: **UTF-8** and **GBK**. If garbled characters appear, you can change the character set.
Separator	The row delimiter and column delimiter. ■ Valid values of row delimiters: \r\n, \n, and \r. ■ Valid values of column delimiters: commas (,), semicolons (;), and \t. If the cell data cannot be correctly separated, you can change the delimiters.
Data Preview	Displays the data in the selected sheet. When you preview the data in the selected sheet, you can determine whether to use the values in the first row as the column names by selecting or clearing **First Row as Field Names**.
Field Mapping	The mappings between the columns in the selected sheet and the fields in the dimension table.
Import Data Mode	The mode used to import data. Valid values: **Append** and **Overlay**.

- Local Excel file

Parameter	Description
File	The Excel file from which you want to import data. Click **Select File(.xlsx)**, select the desired Excel file from your on-premises machine, and then click **Open**.
Sheet	The sheet from which the data is to be imported. Select a sheet from the **Sheet** drop-down list.
Data Preview	Displays the data in the selected sheet. When you preview the data in the selected sheet, you can determine whether to use the values in the first row as the column names by selecting or clearing **First Row as Field Names**.

Parameter	Description
Field Mapping	The mappings between the columns in the selected sheet and the fields in the dimension table.
Import Data Mode	The mode used to import data. Valid values: **Append** and **Overlay**.

6. Click **OK**.
7. Click **Save** in the upper-right corner of the page.

   After you save the dimension table, you can click **Diff** in the upper-right corner of the page to check whether the changes meet expectations to prevent accidental operations.

2.1.7.4.3. Edit a dimension table

This topic describes how to edit a dimension table in a visualized manner to modify the information of a MaxCompute table that you created. You do not need to write SQL code.

Prerequisites

A dimension table is created. For more information about how to create a dimension table, see Create a dimension table.

Procedure

1. Go to the DataAnalysis page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click the icon and choose **All Products > Data Development > DataAnalysis**. The **DataAnalysis** page appears.
2. In the top navigation bar of the DataAnalysis page, choose **More > Dimension Table** to go to the **Dimension Table** page.
3. In the **All Dimension Tables** section of the **Dimension Table** page, click the name of the table that you want to edit in the **File Name** column to go to the dimension table editing page.
4. On the dimension table editing page, view and modify the information about the dimension table.

   In the left-side pane of the dimension table editing page, you can view the table information, such as the workspace, table name, table description, lifecycle, and field description. To view the details of the dimension table, click the link below **Table Details** to go to the **Data Map** page. For more information, see View the details of a table.

   To modify the settings of the dimension table, perform the following steps: Click **Modify field settings**. In the **Modify the field settings dimension table** dialog box, change the values of the **Table Description** and **Lifecycle** parameters and click OK. You can also add fields to the table in this dialog box.

   The right side of the dimension table editing page displays all the data in the MaxCompute table as a workbook. The values in the first row are field names. You can double-click a cell to modify the content of a field in the corresponding row.
5. Click **Save** in the upper-right corner of the page to save the changes.

   After you save the dimension table, you can view all the data in the table. You can also click **Diff** in the upper-right corner to view the differences from the previous version in the **Diff From the Previous Version** dialog box.

2.1.7.4.4. Share a dimension table

If you want to collaboratively edit a dimension table with multiple members, you can share the dimension table and grant the members the permissions to edit the table. This topic describes how to share a dimension table and grant edit or read permissions to specified members.

Prerequisites

Allow Sharing is turned on on the **Management** page.

Procedure

1. Go to the DataAnalysis page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click the icon and choose **All Products > Data Development > DataAnalysis**. The **DataAnalysis** page appears.
2. In the top navigation bar of the DataAnalysis page, choose **More > Dimension Table** to go to the **Dimension Table** page.
3. In the **All Dimension Tables** section of the **Dimension Table** page, click the name of the table that you want to edit in the **File Name** column to go to the dimension table editing page.

   If no dimension table exists and you created one, the dimension table editing page appears after the table is created. For information about how to edit a dimension table, see Edit a dimension table.
4. In the upper-right corner of the dimension table editing page, click **Share**. In the dialog box that appears, configure the sharing mode as required.

   You must configure the following information before you can share a dimension table with other members:

   - **Link**: After you specify Users with Edit Access and Users with Read Access, click **Copy Link** and send the copied URL to specified members.
   - **Users with Edit Access**: To specify members with permissions to edit the dimension table, click **Add** in the **Users with Edit Access** section. In the dialog box that appears, enter and select the names of the members to be granted the edit permissions, and click **OK**.

     > **Note** You can grant the edit permissions to a maximum of 10 members.

   - **Users with Read Access**: To specify members with permissions to read the dimension table, click **Add** in the **Users with Read Access** section. In the dialog box that appears, enter and select the names of the members to be granted the read permissions, and click **OK**.

     > **Note** You can grant the read permissions to a maximum of 30 members.

   After the sharing mode is configured, you can send the URL to specified members. The members can use the URL to access the dimension table. You can go back to the **Dimension Table** page to view the dimension tables that are shared with you.

2.1.7.5. Report

2.1.7.5.1. Create and manage reports

DataAnalysis allows you to explore data in a visualized manner and create reports. You can create reports by dragging and configuring controls without the need to write SQL code.

Create a report

1. Go to the DataAnalysis page.
   i. Log on to the DataWorks console.

ii. In the left-side navigation pane, click the icon and choose **All Products > Data Development > DataAnalysis**. The **DataAnalysis** page appears.

2. In the top navigation bar of the DataAnalysis page, choose **More > Reports** to go to the **Reports** page.
3. On the **Reports** page, click the + icon in the **New Report** section to go to the report editing page. You can edit the report on this page. For more information, see Edit a report.

   If you have templates within your account, you can click a template to create a report based on the template.

View and manage reports

1. On the report editing page, click **Reports** in the upper-left corner to go back to the Reports page.
2. In the **All Reports** section of the **Reports** page, you can view all reports.
3. Click the name of a report in the **File Name** column to go to the report editing page.

   On the Reports page, you can also perform the following operations on a report:

   - To rename a report, find the report and click the icon in the Operation column. In the **Rename** dialog box, enter a new name in the **File Name** field and click **OK**.
   - To delete a report, find the report and click the icon in the Operation column. In the **Delete** message, click **OK**.

What's next

After you create a report, go to the report editing page and edit the report in a visualized manner. For information about how to edit a report, see Edit a report.

2.1.7.5.2. Edit a report

This topic describes how to edit, preview, save, share, and publish a report, and save a report as a template.

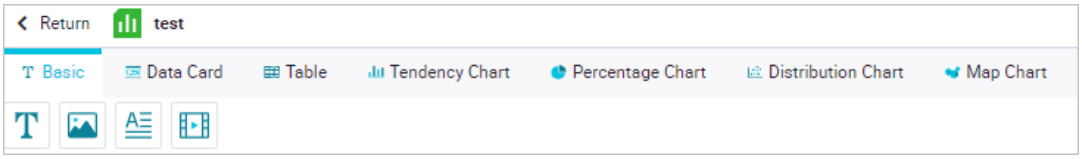
Procedure

1. Go to the DataAnalysis page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click the icon and choose **All Products > Data Development > DataAnalysis**. The **DataAnalysis** page appears.
2. In the top navigation bar of the DataAnalysis page, choose **More > Reports** to go to the **Reports** page.
3. Go to the editing page of a report.

   Use one of the following methods to go to the editing page of a report:

   - After you create a report, the editing page of the report appears. For more information, see Create and manage reports.
   - In the **All Reports** section of the Reports page, click the name of the report that you want to edit in the **File Name** column.
4. 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 canvas of 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.



5. Click the **Bar Chart** component. Then, add a data source on the **Data Config** tab in the right-side configuration section.

   If a data source is added, click the name of the data source on the **Data Config** tab.

   If you want to add a data source, click **Add** on the **Data Config** tab. In the **Add** dialog box, configure the parameters and click **OK**.

   You can set the Choose Data parameter to Select a spreadsheet or Use External Data Source.

   - **Select a spreadsheet**: You can select an editable worksheet in a workbook of the current user as the data source.

Parameter	Description
Choose Data	The range of the data that you want to analyze. Select **Select a spreadsheet**.
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.

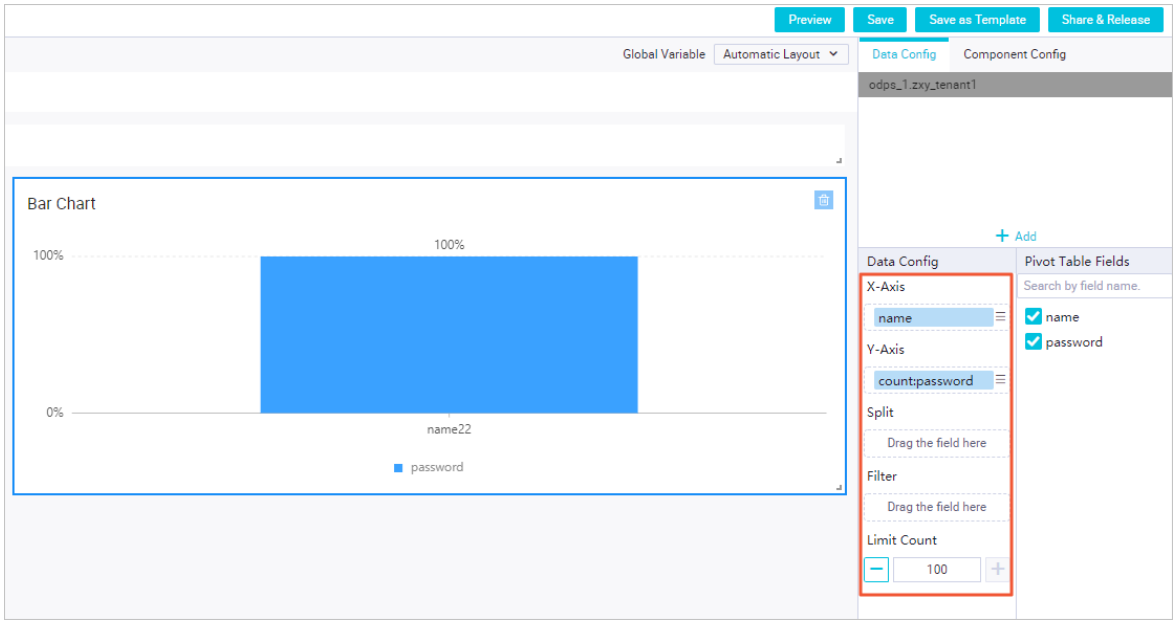
   - **Use External Data Source**: You can add an external data source. The following types of external data sources are supported: **Mysql**, **PostgreSQL**, and **Data Services**.

Parameter	Description
Choose Data	The range of the data that you want to analyze. Select **Use External Data Source**.
Type	The type of the data source. Valid values: **Mysql**, **PostgreSQL**, and **Data Services**.
Workspace	The workspace in which the data source resides.
Data Source	The name of the data source. To add a data source, perform the following steps: Go to the DataStudio page. 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, add a data source. **Note** This parameter is available only if the Type parameter is set to **Mysql**.
Table	The table of which the data is analyzed. **Note** This parameter is available only if the Type parameter is set to **Mysql**.

Parameter	Description
API Group	The API group to which the API belongs. **Note** This parameter is available only if the Type parameter is set to **Data Services**.
API	The API that you want to use as the data source. **Note** This parameter is available only if the Type parameter is set to **Data Services**.

6. Select fields as statistical items.

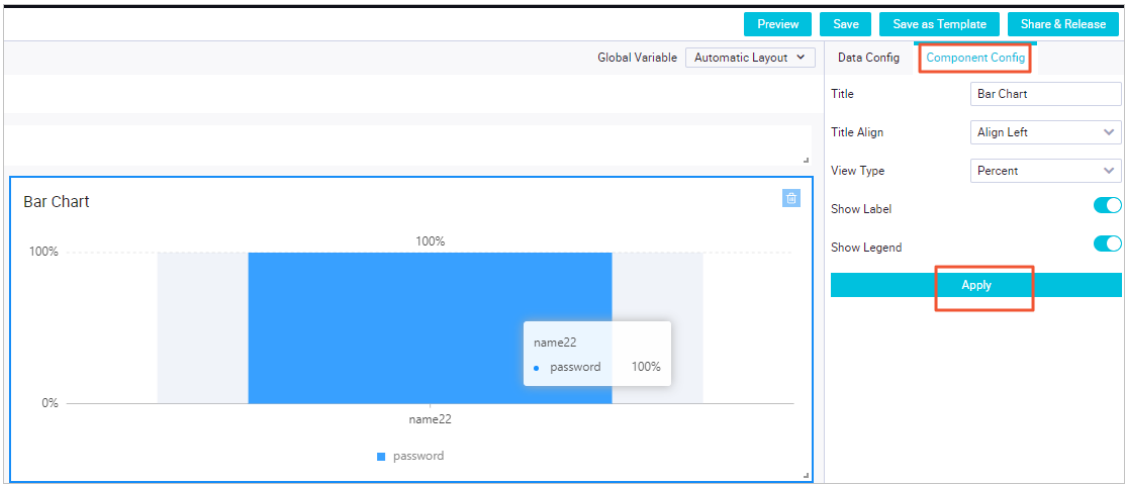
   The fields that are required vary based on the chart type. For example, you must set the **X-Axis**, **Y-Axis**, **Split**, and **Filter** parameters in the Data Config section for a column chart by dragging fields from the **Pivot Table Fields** section. You can also set the **Limit Count** parameter.



   Multiple charts can use the same data source. They can use the data source in different ways without affecting each other. A chart can use only one data source. If 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 source.

7. Configure the column chart.
   i. On the canvas, click the **Bar Chart** component.
   ii. Click the **Component Config** tab in the right-side configuration section.

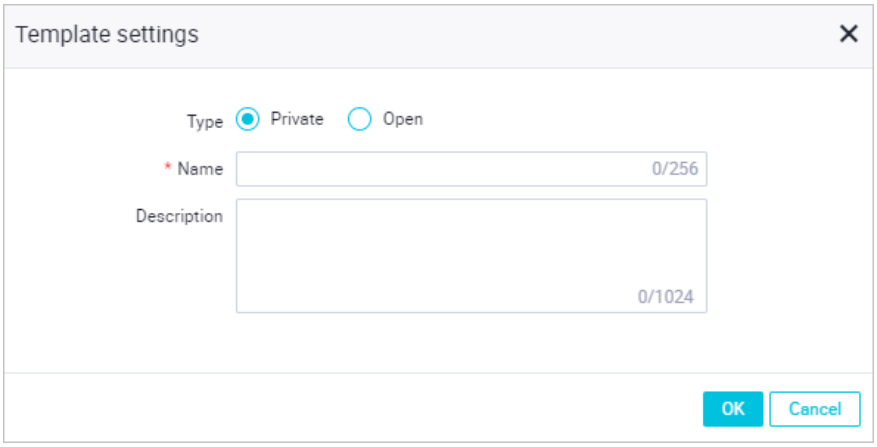
iii. On the **Component Config** tab, set the parameters.



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.

8. Click **Apply**.
9. Go back to the **Reports** page or preview, save, share, or publish the report based on your business requirements.
   - **Reports**: Click **Reports** in the upper-left corner to go back to the **Reports** page. On the Reports page, you can view other reports and go to the editing page of each of these reports.
   - **Preview**: Click **Preview** in the upper-right corner to preview the report.
   - **Save**: 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**: Save the report as a template. This way, you can use this template to create another report. Perform the following steps to save a report as a template:
     a. On the editing page of a report, click **Save as Template** in the upper-right corner.
     b. On the **Preview** page, click **Next Step (Template setup)**.

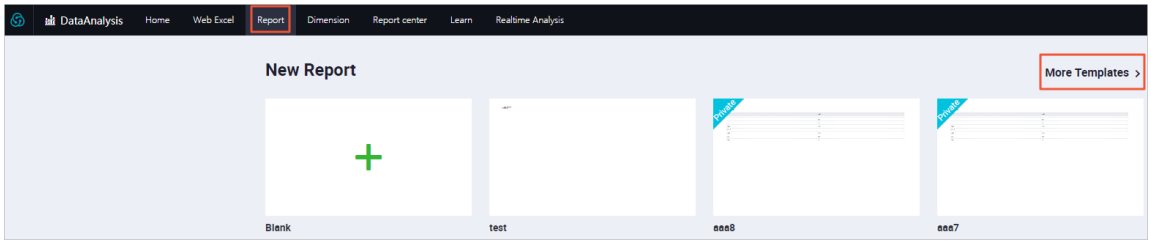
c. In the **Template settings** dialog box, set the parameters.



Parameter	Description
Type	Specifies whether to show the template to other users. Valid values: **Private** and **Open**.
Name	The name of the template. The name can be a maximum of 256 characters in length.
Description	The description of the template. The description can be a maximum of 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 **Reports** page to create a report.



- **Share & Release**: Click **Share & Release** in the upper-right corner to share and publish the report. You can share the report with specific or all users. If you want to share the report with specific users, click **Add** to specify the users.

2.1.8. Administration

2.1.8.1. Overview

Operation Center consists of the following modules: Overview, RealTime Task Maintenance, Cycle Task Maintenance, Alarm, and Engine Maintenance.

Module	Description
Overview	This module displays the running statuses of nodes in charts.

Module	Description
RealTime Task Maintenance	This module displays all the real-time data synchronization nodes that are committed to and run by the scheduling system. For more information, see Manage real-time synchronization nodes.
Cycle Task Maintenance	This module displays all the **auto triggered nodes** that are committed to and run by the scheduling system, the **instances** that are generated for the nodes, **retroactive instances**, and **test instances**. For more information, see Manage auto triggered nodes.
Trigger Task Maintenance	This module displays all the **manually triggered nodes** that are committed to the scheduling system and the **instances** generated for the nodes. For more information, see Manage manually triggered nodes.
Alarm	This module displays the running statuses of nodes that are monitored. You will receive alerts if monitored nodes are abnormal. For more information, see Overview.
Engine Maintenance	This module allows you to view the jobs, quota groups, and projects of MaxCompute. For more information, see MaxCompute engine O&M.

Scenarios

- You can view and manage your nodes and instances in Operation Center. You can also test nodes and generate retroactive data for the nodes.
- If you are using a workspace in standard mode, you can switch between the production and development environments by changing the URL of the Operation Center page. If the URL contains env=dev, Operation Center in the development environment is used. If the URL contains env=prod, Operation Center in the production environment is used. You can change the value of the env parameter to go to Operation Center of your desired environment.
- When you perform O&M for your nodes, you can view the instances of all the nodes and perform related operations on the instances. For example, you can stop and rerun instances.

> **Note** Instances are generated after nodes are run by the scheduling system. An instance is a snapshot of a node at a specific point in time. The instance contains information such as the time when the node is run, the running status of the node, and operational logs.

2.1.8.2. View the statistics on the Overview page

The Overview page displays the overall O&M information, including the metrics that require your special attention, overall running information about nodes, and trends on scheduling resource changes. It also displays information about Data Integration, including the distribution of batch synchronization nodes and real-time synchronization nodes in different states and the data synchronization progress of batch synchronization nodes and real-time synchronization nodes. This helps improve O&M efficiency.

Go to the Overview page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Task Operation > Operation Center**. The **Workbench Overview** tab on the **Overview** page appears.

View the statistics on the Workbench Overview tab

The Workbench Overview tab displays the statistics only on auto triggered nodes and auto triggered node instances. You can view the following information on the **Workbench Overview** tab:

- The **Focus on** section displays the following information:
  - The numbers of auto triggered node instances that require your special attention, including **failed instances**, **slow running instances**, and **instances waiting for resources**. Statistics are collected on auto triggered node instances whose data timestamp is the day before the current date. A node instance is considered a **slow running instance** if it meets the following conditions:
    - The node instance is running.
    - The running time of the node instance exceeds 30 minutes.
    - The running time of the node instance is at least 15 minutes longer than the average running time of the node instance during the last 10 days.
  - The numbers of **isolated nodes**, **paused nodes**, and **expired nodes**.
    - An isolated node is a node that does not have an ancestor node. In this case, the node cannot be run. For example, if you change the output name for the ancestor node of a node, the dependency becomes invalid.
    - After a node is paused, the node no longer generates node instances and cannot be scheduled.
    - If a node is not run at the specified time, the node becomes expired.

  The statistics in the Focus on section are updated when you load the page. You can click a type of node or node instance to go to the details page and view the specific nodes or node instances. We recommend that you manage these nodes and node instances at the earliest opportunity to avoid impacts on your business.
- The **Running Status Distribution** section displays the distribution of auto triggered node instances in different states. The data timestamp of these node instances is the day before the current date. The statistics in this section are updated when you load the page. You can click a sector in the pie chart to view the auto triggered node instances in a specific state.
- The **Node Completion** section displays the completion status of auto triggered node instances between 00:00 and 23:00 of the current date. You can view the number of auto triggered node instances that are successfully run or not run today and yesterday. You can also view the historical average number of auto triggered node instances that are successfully run or not run. In addition, you can select a node type to view the status of specific auto triggered node instances.

  The line chart displays the number of auto triggered node instances that are run today, the number of auto triggered node instances that are run yesterday, and the historical average number of auto triggered node instances that are run. If the deviations among the three numbers are large, an exception occurred during a specified time period. You must perform a further check and analysis.

  > **Note** The statistical aggregation method used by Operation Center has been changed. Only auto triggered node instances in the production environment are counted. Therefore, the line that represents the number of auto triggered node instances that are run today shows obvious fluctuations.
- The **Scheduling Resource Allocation** section displays the resource usage of a specific resource group and the number of auto triggered node instances that were running at different points in time within the last 24 hours. You can select a resource group from the **Resource Group** drop-down list in the upper-right corner of this section.

  Number of Instances represents the number of node instances in the current workspace, and **Resource Usage** represents the resource usage of the resource group in the current region.

  > **Note** The resources that are used by Data Integration nodes of the resource group are not recorded.
- The **Runtime Ranking** section ranks nodes based on their running time, time spent in waiting for resources, and slow running time. The statistics in this section are updated every day. Nodes that are run successfully on the day before the current date are ranked in this section.
- The **Error Ranking in Recent Month** section ranks nodes on which errors occurred within the last month and

displays the top 10 nodes. The statistics in this section are updated every day. You can view the settings of **Node ID**, **Node Name**, and **Errors** for each node.

- The **Number of Auto Triggered Nodes** section displays the trends on the numbers of auto triggered nodes and auto triggered node instances in a specified time range in the production environment. The statistics in this section are updated every day. You can view the trends on the numbers of auto triggered nodes and auto triggered node instances in a time range as wide as one year.
- The **Node Types** section displays the distribution of nodes in a pie chart. The statistics in this section are updated when you load the page. The pie chart displays a maximum of eight node types. If you have created more than eight types of nodes, specific node types are merged for display.

View the statistics on batch synchronization nodes on the Data integration tab

Click **Data Integration** on the **Overview** page. The Data integration tab appears. By default, the Data integration tab displays the statistics on **batch synchronization nodes** within a specified time range. You can specify **Time range** in the upper-right corner.

You can view the statistics on batch synchronization nodes in the following sections:

- The **RUNNING state distribution** section displays the distribution of batch synchronization node instances in different states. The data timestamp of these node instances is in the specified time range. The statistics in this section are updated when you load the page. You can click a sector in the pie chart to view the node instances in a specific state.
- The **Data Synchronization progress** section displays information about the data that is involved in batch synchronization within the specified time range. The information includes the **total amount of data**, **total public traffic**, and **total number of records**.
- The **Synchronize data volume statistics** section displays the curves of the data that is read from or written to different data sources within the specified time range.
- The **Latest list Top10** section displays the latest 10 node instances that **failed** to be run and the latest 10 node instances that are run **successfully**. The statistics provide you with an overview of the latest node instance status.
- The **Synchronization task execution details** section allows you to specify filter conditions to search for node instances. The filter conditions include **Submission time**, **Status**, and **Task name**. You can click the ID of a node instance to view the running details of the node instance.

View the statistics on real-time synchronization nodes on the Data integration tab

Click **Data Integration** on the **Overview** page. On the **Data integration** tab, click **Real-time synchronization**. You can view the statistics on real-time synchronization nodes in the following sections:

- The **RUNNING state distribution** section displays the distribution of real-time synchronization node instances in different states. The statistics in this section are updated when you load the page. You can click a sector in the pie chart to view the node instances in a specific state.
- The **Overview** section displays the total data transmission speed and total recording speed of all real-time synchronization node instances in the current workspace.
- The **Task delay Top10** section displays the top 10 node instances with the highest latency. This helps you find node instances with high latency.
- The **Task alarm information** section displays information about the latest alerts. This helps you efficiently view exceptions.
- The **Failover information** section displays information about failovers within a specified time period. This provides you with an overview of failovers.

2.1.8.3. Manage real-time synchronization nodes

This topic describes how to manage real-time synchronization nodes on the Operation Center page.

Procedure

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click the icon in the upper-left corner and choose **All Products > Operation Center**.
3. In the left-side navigation pane of the Operation Center page, choose **RealTime Task Maintenance > Real Time DI**.
4. On the **Real Time DI** page, find your desired real-time synchronization node, click the node name, and then view the operations and maintenance (O&M) details about the node.

   On this page, you can **start**, **stop**, **undeploy**, or **configure alerting** for the real-time synchronization node.

   - To start a node that is not running, perform the following steps:
     a. Find the node and click **Start** in the Operation column.
     b. In the **Start** dialog box, configure the parameters.

Parameter	Description
Whether to reset the site	Specifies whether to set the time point for the next startup. If you select Reset site, the **Start time point** and **Time zone** parameters are required.
Start time point	The date and time for starting the real-time synchronization node.
Time zone	The time zone where the source resides. Select a time zone from the **Time zone** drop-down list.
Failover	■ The condition for automatically terminating the real-time synchronization node. You can specify the maximum number of dirty data records allowed. If you set the value to 0, no dirty data records are allowed. If the value is empty, the running of the node continues no matter whether dirty data records exist. ■ You can also specify the maximum number of failover times. If you do not specify Failover, the node is automatically terminated if the node fails 100 times within 5 minutes. This avoids resource occupation caused by frequent startup.

     c. Click **OK**.
   - To stop a running node, perform the following steps:
     a. Find the node and click **Stop** in the Operation column.
     b. In the message that appears, click **Stop**.
   - To undeploy a node that is not running, perform the following steps:
     a. Find the node and click **Offline** in the Operation column.
     b. In the message that appears, click **Offline**.
   - Find the node and click **Alarm settings** in the Operation column. On the page that appears, you can view alert event information and alert rules on the **Alert event** and **Alarm rules** tabs.
   - To configure alert settings for a node, perform the following steps:
     a. Select your desired node and click **New Alarm** in the lower part of the page.

b. In the **New rule** dialog box, configure the parameters.

Parameter	Description
Name	Required. The name of the rule that you want to create.
Description	The description of the rule.
Indicators	The metrics in the rule that you want to create. Valid values: **Task Status**, **Business latency**, **Failover**, **Dirty Data**, and **DDL error**.
Threshold	The threshold for reporting an alert. The default value is 5 minutes for both **WARNING** and **CRITICAL** alerts.
Alarm interval	The interval at which an alert is reported. The default value is 5 minutes.
WARNING	The methods used to send alert notifications. Set the value to **Mail**.
CRITICAL	
Receiver	The alert recipient. Select a recipient from the **Receiver** drop-down list.

c. Click **OK**.

- To modify alert settings for a node, perform the following steps:
  a. Select your desired node and click **Operation alarm** in the lower part of the page.
  b. In the **Operation alarm** dialog box, specify the **Operation type** and **Alarm indicators** parameters.

     DataWorks automatically modifies all the rules for the selected alert types at a time.
  c. Click **OK**.

2.1.8.4. Auto triggered node O&M

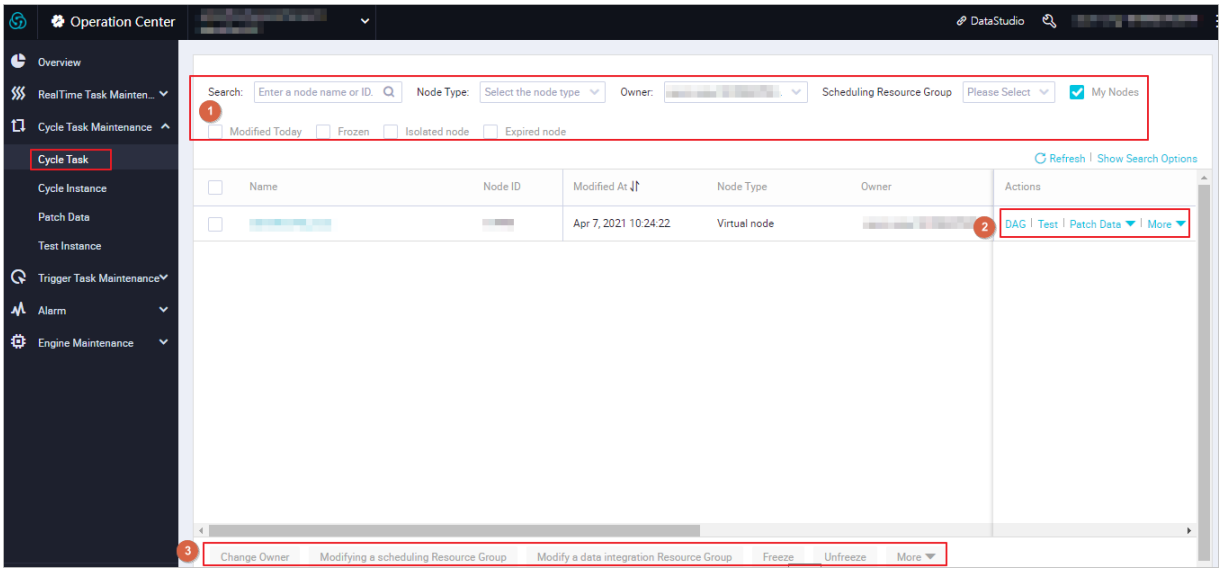
2.1.8.4.1. Manage auto triggered nodes

Auto triggered nodes are automatically run as scheduled after they are committed to the scheduling system. You can view the details of an auto triggered node from the directed acyclic graph (DAG) of the node or the auto triggered node list.

> **Note**
> - By default, the auto triggered node list displays nodes in all the workflows created by using 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 operations on the **projectname_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, the instances of auto triggered nodes cannot be run.

Manage auto triggered nodes in the node list

The **Cycle Task** page displays the auto triggered nodes that are committed to the scheduling system in a list.



Operation	Description
Filter	Allows you to specify filter conditions to search for the desired auto triggered nodes in the section marked with 1 in the preceding figure. You can search for the desired nodes by node name or node ID. You can also specify the filter conditions such as **Node Type**, **Owner**, **Scheduling Resource Group**, **My Nodes**, **Modified Today**, **Frozen Nodes**, **Isolated Nodes**, and **Expired Nodes** to perform the operation. **Note** When you search for the desired auto triggered nodes by node name, the search result is affected by other filter conditions that you specified. Only nodes that meet all filter conditions are displayed.
DAG	Allows you to view information such as the properties, operation logs, and code of the node. You can click **DAG** in the Actions column of the node to open the DAG in the section marked with 2 in the preceding figure.
Test	Allows you to test the node. You can click **Test** in the Actions column of the node in the section marked with 2 in the preceding figure. For more information, see Manage test instances.
Patch Data	Allows you to backfill data for the node. You can click **Patch Data** in the Actions column of the node in the section marked with 2 in the preceding figure. For more information, see Manage retroactive instances.

Operation	Description
More	Allows you to perform more operations on the node, such as freezing and unfreezing the node, and view the instances of the node. You can click **More** in the Actions column and perform the desired operation on the node in the section marked with 2 in the preceding figure. • Select **Freeze** to freeze the node. After the node is frozen, the system generates instances for the node but does not run the instances of the node and its dependent descendant instances. • Select **Unfreeze** to unfreeze the node. After the node is unfrozen, the system normally runs the instances of the node and its descendant instances. • Select **View Instances** to view the instances of the node. • Select **Configure Alert Rule** to configure alert rules 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 the workspace has multiple resource groups. • Select **Configure Data Quality Rules** to configure the rules that are used to monitor the data quality of the node. • Select **View Lineage** to view the lineage of the node. • Select **View Node Details** to go to the **Node Information** page. On this page, you can view the node information on the **Ancestor Nodes** and **Descendent Nodes** tabs.
Change Owner, **Modify Scheduling Resource Group**, **Modify Data Integration Resource Group**, **Freeze**, **Unfreeze**, **Configure Alert Rule**, **Add to Baseline**, and **Undeploy**	Allows you to select multiple nodes and perform operations on the nodes at a time. In the section marked with 3 in the preceding figure, you can click **Change Owner**, **Modify Scheduling Resource Group**, **Modify Data Integration Resource Group**, **Freeze**, or **Unfreeze**, or select **Configure Alert Rule**, **Add to Baseline**, or **Undeploy** after you click **More** to perform the related operation on the selected nodes.

Manage auto triggered nodes in a DAG

Find the desired node in the node list and click its name 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**	Allows you to view the ancestor or descendant instances of the node at one or more levels. If a workflow contains three or more nodes, the system hides some nodes. Select the largest number to view the most node dependencies.
View Node Details	Allows you to go to the **Node Information** page to view the node information, including the input table, output table, ancestor nodes, and descendant nodes.
View Code	Allows you to view the code of the node.

Operation	Description
Edit Node	Allows you to go to the DataStudio page and modify the node.
View Instances	Allows you to view the instances of the node.
View Lineage	Allows you to view the lineage of the node.
Test	Allows you to perform smoke testing for the node. You must specify the **Test Name** and **Data Timestamp** parameters in the **Test** dialog box and click **OK**. Then, the Test Instance page appears.
Run	Allows you to backfill data for the node. The following modes for backfilling data are supported: **Current Node Retroactively**, **Current and Descendant Nodes Retroactively**, and **Mass Nodes Retroactively**.
Freeze	Allows you to freeze the node and pause the scheduling of the node.
Unfreeze	Allows you to resume the scheduling of the frozen node.
Configure Data Quality Rules	Allows you to configure the rules that are used to monitor the data quality of the node.

2.1.8.4.2. Manage auto triggered node instances

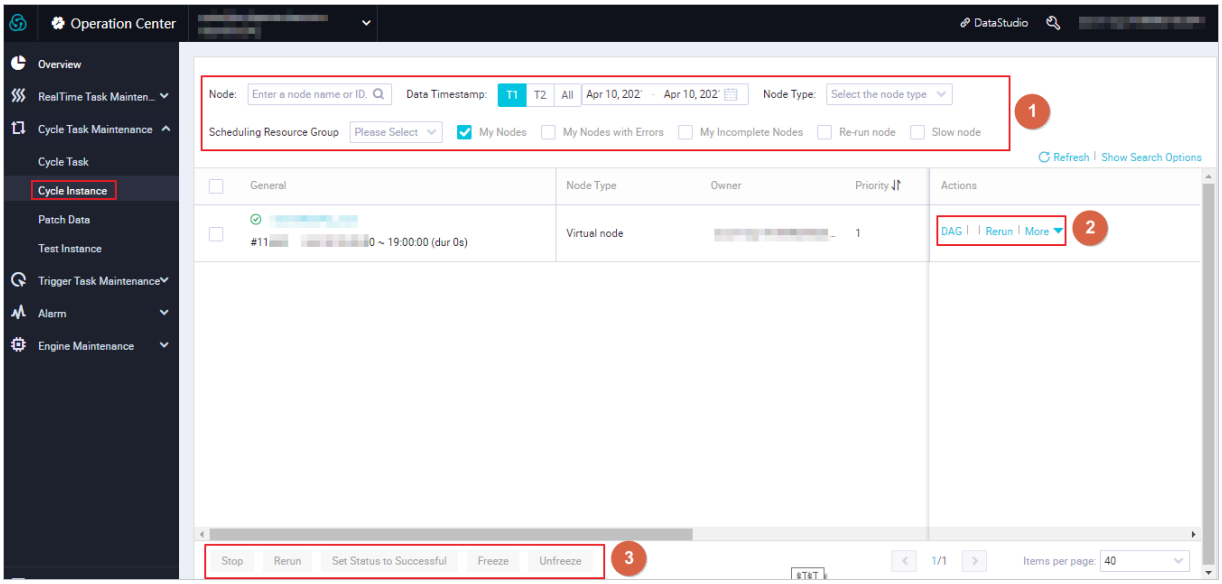
An auto triggered node instance is a snapshot that is taken for an auto triggered node at the time the node is scheduled to run.

An instance is generated each time an auto triggered node is run as scheduled. You can manage auto triggered node instances that have been scheduled. For example, you can view the status of instances, and stop, rerun, and unfreeze instances.

> **Note** Instances are generated as scheduled for auto triggered nodes. Each generated instance runs the latest code. If you modify and recommit the node code after instances are generated for the node, the latest code is run on the instances that have not been run.

Manage auto triggered node instances on the Cycle Instance page

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	You can specify filter conditions to search for your desired instance in the section marked with 1 in the preceding figure. You can search for instances by node name or node ID. You can also specify the following conditions to search for your desired instance: **Data Timestamp**, **Node Type**, **Scheduling Resource Group**, **My Nodes**, **My Nodes with Errors**, **My Incomplete Nodes**, **Rerun Nodes**, and **Slow Nodes**. **Note** By default, the date indicated by the value of the Data Timestamp parameter is one day before the current date.
Stop	Allows you to stop the instance. You can stop an instance only in the **Waiting time** or **Running** state. After you perform this operation, the instance enters the **Run failed** state.
Rerun	Allows you to rerun the instance. After the instance is rerun, its descendant instances that have not been run are run as scheduled. If an instance fails to be run or an instance is not run as scheduled, perform this operation. **Note** This operation applies only to the instances in the **Not Running**, **Succeeded**, or **Run failed** state.
Rerun Descendent Nodes	Allows you to rerun the instance and its descendant instances. You must specify the instances that you want to rerun. After they are run, the descendant instances that are not run are run as scheduled. Perform this operation to recover data. **Note** Only instances in the **Not Running**, **Succeeded**, or **Run failed** state can be selected. The value No appears in the **Meet Rerun Condition** column for instances in other states, and you cannot select these instances.

Operation	Description
Set Status to Successful	Allows you to set the state of the instance to **Succeeded** and run its descendant instances that are not run. If an instance fails to be run, perform this operation. **Note** This operation applies only to instances in the **Run failed** state.
Freeze	Allows you to freeze the instance in the running state.
Unfreeze	Allows you to unfreeze the instance that is frozen. • If the instance is not run, it is automatically run after the running of its ancestor instances is complete. • If all the ancestor instances are run, the state of the instance is directly set to Run failed. You must manually rerun the instance.
View Lineage	Allows you to view the lineage of the instance.
View Node Details	Allows you to view the details about the instance and its ancestor and descendant instances on the **Instance Information** page.
View Runtime Log	Allows you to view the operational logs of the instance.
Stop, Rerun, Set Status to Successful, Freeze, and Unfreeze	Allows you to perform operations on multiple instances at a time. You can click the following buttons in the section marked with 3 in the preceding figure to perform operations on multiple instances at a time: **Stop**, **Rerun**, **Set Status to Successful**, **Freeze**, and **Unfreeze**.

Manage an auto triggered node instance in a DAG

Click the name of an instance or **DAG** in the Actions column that corresponds to an instance 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**	Allows you to view the ancestor or descendant instances of the instance 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	Allows you to view the operational logs of the instance in the Running, Succeeded, or Run failed state.
View Code	Allows you to view the code of the instance.
Edit Node	Allows you to go to the DataStudio page and modify the node to which the instance belongs.
View Lineage	Allows you to view the lineage of the instance.
More	Allows you to view more information about the instance on the **General**, **Context**, **Runtime Log**, **Operation Log**, and **Code** tabs.
Stop	Allows you to stop the instance. Only instances in the Waiting time or Running state can be stopped. After you perform this operation, the instance enters the Run failed state.

Operation	Description
Rerun	Allows you to rerun the instance. After the instance is rerun, its descendant instances that have not been run are run as scheduled. If an instance fails to be run or an instance is not run as scheduled, perform this operation. **Note** This operation applies only to instances in the **Not Running**, **Succeeded**, or **Run failed** state.
Rerun Descendent Nodes	Allows you to rerun the instance and its descendant instances. You must specify the instances that you want to rerun. After they are run, the descendant instances that are not run are run as scheduled. Perform this operation to recover data. **Note** Only instances in the **Not Running**, **Succeeded**, or **Run failed** state can be selected. The value No appears in the **Meet Rerun Condition** column for instances in other states, and you cannot select these instances.
Set Status to Successful	Allows you to set the state of the instance to Succeeded and run its descendant instances that have not been run. If an instance fails to be run, perform this operation. **Note** This operation applies only to instances in the **Run failed** state.
Resume	Allows you to continue to run the instance after it fails.
Emergency Operations	Allows you to perform emergency operations in emergencies only. The emergency operations include **Delete Dependencies**, **Change Priority**, and **Force Rerun**. The emergency operations take effect only once on the current instance. Select **Delete Dependencies** to delete the dependencies of the current instance. If the ancestor instances fail and the current instance does not depend on the data of the ancestor instances, you can perform this operation to start the current instance.
Freeze	Allows you to freeze the instance in the Running state.
Unfreeze	Allows you to unfreeze the instance that is frozen. • If the instance is not run, it is automatically run after the running of its ancestor instances is complete. • If all the ancestor instances of the instance are successfully run, the state of the instance is directly set to Run failed. You must manually rerun the instance.

Instance states

No.	State	Icon
1	Succeeded	
2	Not Running	
3	Run failed	

No.	State	Icon
4	Running	
5	Waiting time	
6	Freeze	

2.1.8.4.3. Manage data backfill node instances

Data backfill node instances are generated when DataWorks backfills data for auto triggered nodes. You can perform O&M for and manage data backfill node instances. For example, you can view the statuses of data backfill node instances and stop, rerun, or unfreeze data backfill node instances.

Limits

- When DataWorks backfills data for a node for a specific time range, if one instance of the node fails on a day within the time range, the data backfill node instance for that day is also set to Failed. DataWorks will not run the instances of this node for the next day. To sum up, DataWorks runs the instances of a node on a day only when all its instances on the previous day are successfully run.
- For a self-dependent auto triggered node, if the first instance for which data needs to be backfilled has a last-cycle instance that is not run on the previous day, the data backfill node instance cannot be run. If the first instance for which data needs to be backfilled does not have a last-cycle instance on the previous day, the data backfill node instance is directly run.
- DataWorks generates alerts only for auto triggered node instances that fail.
- If an auto triggered node instance is running for a node, the data backfill node and test instances of the node can be run only after the running of the auto triggered node instance is complete.
- If both an auto triggered node instance and a data backfill node instance are running for a node, you must stop the data backfill node instance to ensure that the auto triggered node instance can be run as expected.

Backfill data

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Task Operation > Operation Center**.
3. On the Operation Center page, choose **Cycle Task Maintenance > Cycle Task** in the left-side navigation pane.
4. On the Cycle Task page, click the rightward arrow in the middle to show the node list. Find the required node, click **Patch Data** in the Actions column, and then select a mode to backfill data.

   You can also click DAG in the Actions column of the required node. In the panel that appears, right-click the node name in the directed acyclic graph (DAG), move the pointer over **Run**, and then select a mode to backfill data.

Backfill data for the current node

1. Find the required node and choose **Patch Data > Current Node Retroactively** in the Actions column.
2. In the **Patch Data** dialog box, configure the parameters.

Parameter	Description
Retroactive Instance Name	DataWorks automatically generates a data backfill node instance name for your node. You can modify the name.
Data Timestamp	The data timestamp of the data backfill node instance.
Node	The name of the node for which you want to backfill data. You cannot change the name.
Parallelism	Specifies whether to generate multiple data backfill node instances at a time. ○ If you do not select Parallelism, only one data backfill node instance is generated. The data backfill node instance is run multiple times in sequence based on the data timestamp. ○ If you select Parallelism, you can choose to use no more than 10 data backfill node instances to backfill data for the node at the same time. The data backfill node instances are run in parallel based on the data timestamp. ■ If the number of days in the data timestamp is less than the number of parallel instances, the data backfill node instances are run in parallel. For example, the data timestamp is from January 11 to January 13, and you set Number of Concurrent Nodes to 4. In this case, three data backfill node instances are generated for each day within the data timestamp and are run in parallel. ■ If the number of days in the data timestamp is greater than the number of parallel instances, some instances may be run multiple times in sequence whereas others are run in parallel. For example, the data timestamp is from January 11 to January 13, and you set Number of Concurrent Nodes to 2. In this case, two data backfill node instances are generated. They are run in parallel for once, and one of them must be run for the second time.

3. Click **OK**.

Backfill data for the current node and its descendant nodes

1. Find the required node and choose **Patch Data > Current and Descendent Nodes Retroactively** in the Actions column.
2. In the **Patch Data** dialog box, configure the parameters.

Parameter	Description
Retroactive Instance Name	DataWorks automatically generates a data backfill node instance name for your node. You can modify the name.
Data Timestamp	The data timestamp of the data backfill node instance.
Parallelism	Specifies whether to generate multiple data backfill node instances at a time. ○ If you do not select Parallelism, only one data backfill node instance is generated. ○ If you select Parallelism, you can choose to use no more than 10 data backfill node instances to backfill data at the same time for the node.
Nodes	You can specify the **Node Name** and **Node Type** parameters to search for nodes for which you want to backfill data.

3. Click **OK**.

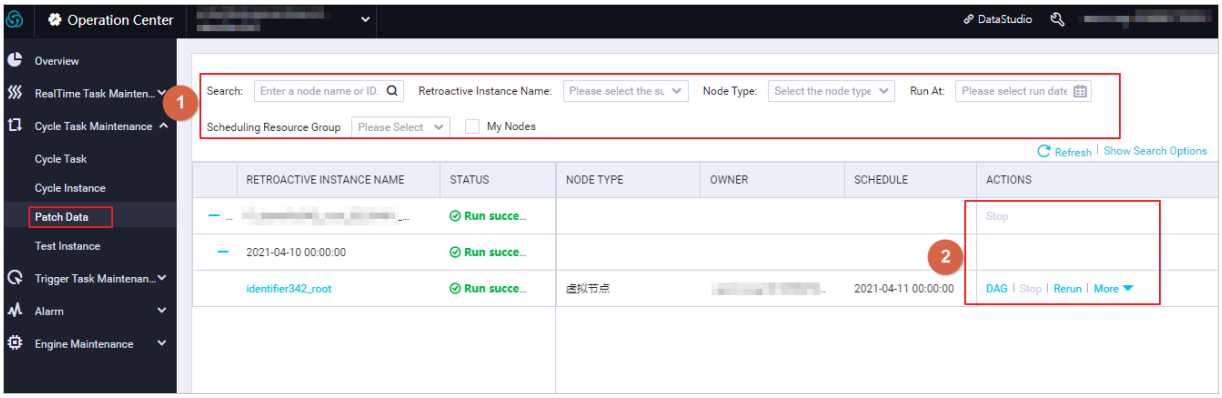
Backfill data for a large number of nodes

1. Find the required node and choose **Patch Data > Mass Nodes Retroactively** in the Actions column.
2. In the **Patch Data** dialog box, configure the parameters.

Parameter	Description
Retroactive Instance Name	DataWorks automatically generates a data backfill node instance name for your node. You can modify the name.
Data Timestamp	The data timestamp of the data backfill node instance. **Note** We recommend that you do not set this parameter to a long time range. Otherwise, the data backfill node instance may be delayed due to insufficient resources.
Nodes	◦ If you select **Current Node**, data backfill node instances are generated for the current node and its descendant nodes. ◦ If you do not select **Current Node**, a dry-run instance is generated for the current node, and data backfill node instances are generated for the descendant nodes of the node.
Workspaces	You can select workspaces from the **Available Workspaces** list and add them to the **Selected Workspaces** list. Fuzzy match is supported when you search for workspaces in the Available Workspaces list.
Node Whitelist	You can add the nodes that are not contained in the selected workspaces and for which you want to backfill data. **Note** You can search for nodes only by node ID.
Node Blacklist	You can add the nodes that are contained in the selected workspaces and for which you do not want to backfill data. **Note** You can search for nodes only by node ID.

3. Click **OK**.

Manage data backfill node instances in the instance list



Operation	Description
Filter	Allows you to search for the desired instances by specifying the filter conditions in the section marked with 1 in the preceding figure. You can search for the desired instances by node name or node ID. You can also specify the filter conditions such as **Retroactive Instance Name**, **Node Type**, **Run At**, **Scheduling Resource Group**, and **My Nodes** to perform the operation. **Note** The default data timestamp is the previous day of the current day.
DAG	Allows you to open the DAG of the current instance to view the running results of the instances.
Stop	Allows you to stop the instance. You can stop an instance only in the **Pending** or **Running** state. After you perform this operation, the instance enters the **Failed** state.
Rerun	Allows you to rerun the instance.
Rerun Descendent Nodes	Allows you to rerun the descendant instances of the current instance.
Set Status to Successful	Allows you to set the status of the instance to **Successful** and run its descendant instances that are not run. Perform this operation if an instance fails to be run.
Freeze	Allows you to freeze the instance and pause the scheduling of the instance.
Unfreeze	Allows you to resume the scheduling of the frozen instance.
View Lineage	Allows you to view the lineage of the instance.

Manage data backfill node instances in a DAG

Click the name of an instance or **DAG** in the Actions column of an instance to view the DAG of the instance. In the DAG, you can right-click the instance to perform the related operations.

Note After you click the **Refresh** icon in the upper-right corner, only the DAG of the instance is refreshed.

Operation	Description
Show Ancestor Nodes or **Show Descendant Nodes**	Allows you to view the ancestor or descendant instances of the instance at one or more levels. If a workflow contains three or more instances, the system shows only the current instance in the DAG. You can select the number of levels to view all instances at one or more levels.
View Runtime Log	Allows you to view the operational logs of the current instance that is in the states such as Running, Successful, or Failed.
View Code	Allows you to view the code of the instance.
Edit Node	Allows you to go to the **DataStudio** page to modify the node to which the instance belongs.
View Lineage	Allows you to view the lineage of the instance.
Stop	Allows you to stop the instance. You can stop an instance only in the **Pending** or **Running** state. After you perform this operation, the instance enters the **Failed** state.
Rerun	Allows you to rerun the instance if it is in the Failed or an abnormal state.
Rerun Descendent Nodes	Allows you to rerun all the descendant instances of the current instance. If the instance has multiple descendant instances, all these instances are rerun.
Set Status to Successful	Allows you to set the status of the current instance to Successful and run its pending descendant instances. Perform this operation if an instance fails to be run. **Note** Only the status of a **failed** instance can be set to successful. This operation does not apply to workflows.
Emergency Operations	Allows you to perform emergency operations in emergencies only. Emergency operations take effect only once on the current instance. Select **Delete Dependencies** to delete the dependencies of the current instance. You can perform this operation to start the current instance if the ancestor instances of the current instance fail and the current instance does not depend on the data of the ancestor instances.
Freeze	Allows you to freeze the current instance and pause the scheduling of the instance.
Unfreeze	Allows you to resume the scheduling of the frozen instance.

Instance states

No.	State	Icon
1	Succeeded	
2	Not Running	
3	Run failed	
4	Running	

No.	State	Icon
5	Waiting time	
6	Freeze	

2.1.8.4.4. 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. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Task Operation > Operation Center**.
3. In the left-side navigation pane of the Operation Center page, choose **Cycle Task Maintenance > Test Instance**. Then, you can view test instances and the directed acyclic graphs (DAGs) of the 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	Allows you to specify filter conditions to search for the desired test instances. You can search for the desired test instances by node name or node ID. You can also specify the filter conditions such as **Node Type**, **Data Timestamp**, **Scheduling Resource Group**, **My Nodes**, **Tested by Me Today**, and **Frozen Nodes** to perform the operation.
Stop	Allows you to stop the test instance. You can stop a test instance only in the **Pending** or **Running** state. After you perform this operation, the test instance enters the **Failed** state.
Rerun	Allows you to rerun the test instance. After the test instance is rerun, the descendant test instances of the test instance that are not run will be run as scheduled. Perform this operation if a test instance fails to be run or a test instance is not run as scheduled. **Note** This operation applies only to test instances in the **Pending**, **Successful**, or **Failed** state.
Set Status to Successful	Allows you to set the status of the current test instance to Successful and run its pending descendant instances. Perform this operation if a test instance fails.
Freeze	Allows you to freeze the test instance and pause the scheduling of the test instance.
Unfreeze	Allows you to resume the scheduling of the frozen test instance.
Stop, Rerun, Set Status to Successful, Freeze, Unfreeze	Allows you to perform batch operations on multiple test instances at a time. You can select multiple test instances and click **Stop**, **Rerun**, **Set Status to Successful**, **Freeze**, or **Unfreeze** to perform the related operation on the instances.

Operation	Description
More	Allows you to perform other operations on the test instance. You can click **More** in the Actions column of the test instance and select an operation. The operations include **Set Status to Successful**, **Freeze**, **Unfreeze**, **View Lineage**, and **View Runtime Log**.

Manage test instances in a DAG

Click the name of a test instance or **DAG** in the Actions column of a test instance to view the DAG of the test instance. In the DAG, you can right-click the test instance to perform the related operations.

Operation	Description
View Runtime Log	Allows you to view the operational logs of the current test instance that is in the states such as Running, Successful, or Failed.
View Code	Allows you to view the code of the test instance.
Edit Node	Allows you to go to the **DataStudio** page to modify the node to which the test instance belongs.
View Lineage	Allows you to view the lineage of the test instance.
Stop	Allows you to stop the test instance. You can stop a test instance only in the **Pending** or **Running** state. After you perform this operation, the test instance enters the **Failed** state.
Rerun	Allows you to rerun the test instance. After the test instance is rerun, the descendant test instances of the test instance that are not run will be run as scheduled. Perform this operation if a test instance fails to be run or a test instance is not run as scheduled. **Note** This operation applies only to test instances in the **Pending**, **Successful**, or **Failed** state.
Set Status to Successful	Allows you to set the status of the test instance to **Successful** and run its descendant test instances that are not run. Perform this operation if a test instance fails. **Note** This operation applies only to failed test instances.
Freeze	Allows you to freeze the test instance and pause the scheduling of the test instance.
Unfreeze	Allows you to resume the scheduling of the test instance that is paused. • If the test instance is not run, the system automatically runs this test instance after the ancestor test instances of the instance are successfully run. • If all the ancestor test instances of the test instance are successfully run, the status of the test instance is directly set to Failed. You must manually rerun the test instance.

Instance states

No.	State	Icon
1	Succeeded	

No.	State	Icon
2	Not Running	
3	Run failed	
4	Running	
5	Waiting time	
6	Freeze	

2.1.8.5. Manually triggered node O&M

2.1.8.5.1. Manage manually triggered nodes

Manually triggered nodes are the nodes that are created in a manually triggered workflow.

> **Note** After a manually triggered node is committed to the scheduling system, the node is run only after it is manually triggered.

Manage manually triggered nodes

On the Trigger Task page, all the manually triggered nodes that are committed are displayed.

Operation	Description
Filter	Allows you to specify conditions to search for the manually triggered nodes that you want to query. You can search for manually triggered nodes by node name or node ID. You can also specify the following conditions to search for your desired nodes: **Type**, **Node Type**, **Owner**, **Engine Type**, **Engine Instance**, **My Nodes**, and **Modified Today**. > **Note** When you search for nodes by node name, the search result is affected by other filter conditions that you specified. Only the nodes that meet all filter conditions are displayed.
DAG	Allows you 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	Allows you to run the node to generate manually triggered node instances.
View Instances	Allows you to go to the **Trigger Instance** page to view the manually triggered node instances generated for the node.
More	Allows you to change the owner of the node. You can choose **More > Change Owner** to perform the operation.

Operation	Description
Change Owner, Modify Scheduling Resource Group, and Undeploy	Allows you to change the owners or the resource groups for scheduling of multiple nodes or undeploy multiple nodes at a time.

Manage manually triggered nodes in a DAG

Click the name of a manually triggered 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 the related operations.

Operation	Description
View Code	Allows you to view the code of the node.
Edit Node	Allows you to go to the **DataStudio** page to modify the node.
View Instances	Allows you to view the instances of the node.
View Lineage	Allows you to view the lineage of the node.
Run	Allows you to run the node to generate manually triggered node instances.
Modify Scheduling Resource Group	Allows you to change the resource group for the node.

2.1.8.5.2. Manage manually triggered node instances

DataWorks generates manually triggered node instances from manually triggered nodes. Manually triggered nodes do not have node dependencies, and you can manually trigger these nodes as needed.

> **Notice** DataWorks generates alerts only for auto triggered node instances that fail to run.

Go to the Trigger Instance page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Task Operation > Operation Center**.
3. In the left-side navigation pane of the Operation Center page, choose **Trigger Task Maintenance > Trigger Instance**. Then, you can view the list and directed acyclic graphs (DAGs) of manually triggered workflows or manually triggered node instances.

Manage a manually triggered node instance in a manually triggered workflow

Operation	Description

Operation	Description
Filter	Allows you to specify filter conditions to search for the workflow to which the instance you want to manage belongs. You can specify a workflow name and one or more of the following parameters to perform the operation: **Type**, **Owner**, **Data Timestamp**, and **Run At**.
DAG	Allows you to view the instance in the **DAG** of the workflow. You can view the running result of the instance.
Stop	Allows you to stop the instance if it is in the Running state.
Rerun	Allows you to rerun the instance.
Stop	Allows you to stop multiple instances at a time. You can select multiple manually triggered workflows and click **Stop** in the lower part of the page to stop the instances in the selected workflows at a time.

Manage a manually triggered node instance in the DAG of a manually triggered workflow

Click the name of a manually triggered workflow or **DAG** in the Actions column of a manually triggered workflow to view the manually triggered node instance in the workflow. In the DAG, you can right-click the instance and select the desired operation.

> **Note** A manually triggered node in a manually triggered workflow does not have node dependencies. Therefore, only the instance that is generated from this node appears in the DAG of the workflow.

Operation	Description
View Runtime Log	Allows you to view the operational logs of the instance in a state such as Running, Successful, or Failed.
View Code	Allows you to view the code of the instance.
Edit Node	Allows you to go to the **DataStudio** page to modify the instance.
View Lineage	Allows you to view the lineage of the instance.
Stop	Allows you to stop the instance.
Rerun	Allows you to rerun the instance if it is in an abnormal state such as Failed.

2.1.8.6. MaxCompute engine O&M

DataWorks Operation Center provides the MaxCompute engine O&M feature. You can use this feature to view the jobs, quota groups, and projects of MaxCompute.

Prerequisites

A MaxCompute compute engine instance is added on the **Project Management** page.

Go to the MaxCompute page

1. Log on to the DataWorks console.

2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Task Operation > Operation Center**.
3. In the left-side navigation pane of the Operation Center page, choose **Engine Maintenance > MaxCompute**. The **Job Queue** tab appears.

   You can view the jobs, quota groups, and projects of MaxCompute.

View jobs

Click the **Job Queue** tab. Then, you can view the total number of jobs and numbers of jobs in the **Running**, **Waiting for Resources**, and **Initializing** states.

You can also configure the **Quota Group** and **Project** parameters to search for jobs. You can view the information indicated by the following parameters for 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 information indicated by the following parameters for 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 information indicated by the following parameters for each project: **Project**, **Owner**, **Cluster**, **Quota Group**, **Storage Used**, **Storage Quota**, **Storage Usage**, and **Files**.

2.1.8.7. Monitor

2.1.8.7.1. 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.

2.1.8.7.2. Feature description

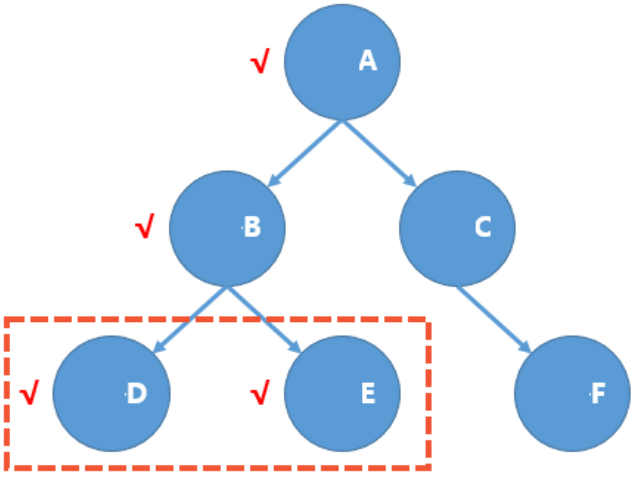
2.1.8.7.2.1. Baseline alert and event alert

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.

2.1.8.7.2.2. 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.

2.1.8.7.3. Instructions

2.1.8.7.3.1. Manage baselines

You can manage baselines on the Baseline Instance page.

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Operation Center**. The **Operation Center** page appears.
3. In the left-side navigation pane, choose **Alarm > Baseline Management**. On the **Baseline Management** page, click Create Baseline to create a baseline. For more information, see Create a baseline.

   > **Note** If you have created a baseline, skip this step.

4. In the left-side navigation pane, choose **Alarm > Baseline Instance**. The **Baseline Instance** page appears. You can search for baselines by condition, such as Data Timestamp, Owner, Event ID, Workspace, and Baseline Name. You can also click **View Details**, **Handle**, and **View Gantt Chart** in the Actions column that corresponds to a baseline to perform the related operations on the baseline.

   > **Note** After a baseline is created, you must enable the baseline so that instances can be generated for the baseline.

   A baseline can be in one of the following states:

   - **Normal**: indicates that the running of all the nodes associated with the baseline is complete before the alerting time.

- **Alerting**: indicates that the running of one or more nodes associated with the baseline is not complete after the alerting time but the committed time does not arrive.
- **Overtime**: indicates that the running of one or more nodes associated with the baseline is not complete after the committed time.
- **Others**: All nodes associated with the baseline are paused, or no nodes are associated with the baseline.

You can click **View Details**, **Handle**, or **View Gantt Chart** in the Actions column that corresponds to a baseline to perform the related operation on the baseline.

- **View Details**: Click **View Details** to view the details about the baseline on the **Baseline Instance Details** page.

  On the Baseline Instance Details page, you can view the information in the **General**, **Critical Path**, **Baseline Instance**, **History**, and **Events** sections.

  > **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** for the baseline. The Cycle parameter appears as the advanced setting of the **Committed Time** parameter only after you select **By the Hour Interval**.

- **Handle**: Click Handle to handle the alert that is generated for the baseline. During this period, alerting is paused.
- **View Gantt Chart**: Click **View Gantt Chart** to view the critical paths of the nodes associated with the baseline.

2.1.8.7.3.2. Manage baselines

You can create and define baselines on the Baseline Management page of Operation Center.

Create a baseline

1. Log on to the DataWorks console.
2. On the DataStudio page, click the 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 of the Operation Center page, choose **Alarm > Baseline Management**.
4. On the **Baseline Management** page, click **Create Baseline** in the upper-right corner.

   > **Note** Only workspace administrators can create baselines.

5. In the **Create Baseline** dialog box, configure the parameters and click **OK**.

Parameter	Description
Baseline Name	The name of the baseline.
Workspace	The workspace of the nodes that you want to associate with the baseline.
Owner	The owner of the baseline. You can search for an owner by owner name or ID.

Parameter	Description
Recurrence	Specifies whether the baseline detects nodes by day or hour. Valid values: ○ **By the Day Interval**: Select this option for nodes that are run on a daily basis. ○ **By the Hour Interval**: Select this option for nodes that are run on an hourly basis.
Node	○ **Node**: the node that you want to associate with the baseline. Enter the name or ID of a node and click the icon on the right to add the node. You can add one or more nodes. ○ **Workflow**: the workflow that you want to associate with the baseline. Enter the name or ID of a workflow and click the icon on the right to add the workflow. We recommend that you add only the last node instead of all nodes of a workflow.
Priority	The priority of the baseline. A baseline that has a higher priority is scheduled first.
Estimated Completion Time	The estimated completion time of the node. The time is estimated based on the average running duration 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 committed completion time of the node. An alert is triggered if the node is still running until the time obtained by subtracting the alert margin threshold from the committed completion time.
Margin Threshold	The interval before an alert is triggered. For example, if you set Committed Time to 3:30 and Margin Threshold to 10 minutes, an alert is triggered if the node is still running at 3:20. If the average running duration of the node is 30 minutes and the node is not started at 2:50, an alert is triggered. **Note** The average running duration of a node can be calculated based on the data of the last 15 days.

6. To enable the baseline that you created, you can click **Enable** in the Actions column of the baseline.

   You can click **View Details**, **Change**, **Enable**, **Disable**, or **Delete** in the Actions column of the baseline to perform the related operations on the baseline.

   - **View Details**: Click **View Details** to view the basic information of the baseline.
   - **Change**: Click **Change** to modify the baseline.
   - **Enable** or **Disable**: Click Enable or Disable to enable or disable the baseline. You must enable a baseline before instances can be generated for the baseline.
   - **Delete**: Click **Delete** to delete the baseline.

Associate a node with a baseline

By default, all nodes in the production environment are associated with the **default workspace baseline**. After you create a baseline, the nodes that you specify are disassociated from the default workspace baseline and are associated with the baseline that you create.

> **Note** Each node must be associated with a baseline. If you want to disassociate a node from the default workspace baseline, you must create a baseline and associate the node with the new baseline. You can delete a baseline that you created. If you delete such a baseline, the nodes that are associated with the baseline are disassociated from it and are associated with the default workspace baseline again.

To change the baseline with which a node is associated, perform one of the following operations:

- On the **Baseline Management** page, click **Create Baseline** in the upper-right corner to create a baseline and associate the node with the baseline.
- In the left-side navigation pane, choose Cycle Task Maintenance > Cycle Task. On the **Cycle Task** page, find the node and choose **More > Add to Baseline** in the Actions column of the node to associate the node with another baseline.

2.1.8.7.3.3. Manage events

You can manage all the events that are related to slowdown or errors on the Event Management page.

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Task Operation > Operation Center**.
3. In the left-side navigation pane of the Operation Center page, choose **Alarm > Event Management**. The **Event Management** page appears.

   You can search for events by condition, such as the event owner, time at which an event is 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. You can view the baseline with the minimum margin among the baselines affected by an event in the Worst Baseline column.

   - If you click **View Details** in the Actions column that corresponds to an event, you can view the time at which the event occurred, alerting time, the time at which the event was processed, historical operational logs of the node, and detailed log information.

     You can assign an alert recipient. After you click **View Alerts**, the alert details page that corresponds to the event appears. You can view all the descendant baselines that are affected by the node associated with the event in the Affected Baselines section. You can observe the descendant baselines and the impact on these baselines and analyze node logs to identify the cause of the event.
   - If you click **Handle**, the event was handled. DataWorks records this operation and does not report the event during the handling process.
   - If you click **Ignore**, the event is ignored. DataWorks records this operation and no longer reports the event.

2.1.8.7.3.4. Create a custom alert rule

This topic describes how to create a custom alert rule on the Rule Management page.

Create a custom alert rule

1. Log on to the DataWorks console.
2. On the DataStudio page, click the ☰ icon in the upper-left corner and choose **All Products > Task Operation > Operation Center**.
3. In the left-side navigation pane of the Operation Center page, choose **Alarm > Rule Management**.
4. On the **Rule Management** page, click **Create Custom Rule**.
5. In the **Create Custom Rule** dialog box, configure the parameters.

Parameter	Description
Rule Name	The name of the custom alert rule.
Object Type	The type of object that you want to monitor. You can set this parameter to **Node**.
Object	The object to which the custom alert rule applies. You can enter an object name or ID and click Add next to the Object parameter to add the object.
Trigger Condition	The condition for triggering an alert. Valid values: **Completed**, **Uncompleted**, **Error**, **Uncompleted in Cycle**, and **Overtime**.
Maximum Alerts	The maximum number of times an alert is reported. If the number of times an alert is reported exceeds the specified threshold, the alert is no longer reported.
Minimum Alert Interval	The minimum interval at which an alert is reported.
Quiet Hours	The specified period during which no alerts are reported.
Notification Method	The method used to send alert notifications. Valid values: **Email**, **SMS**, DingTalk Chatbot, and WebHook.
Recipient	The object who receives alert notifications. Valid values: Node Owner and Others.
DingTalk Chatbot	You can use a DingTalk chatbot to receive alert notifications.

6. Click **OK**. A custom alert rule is created.

   On the **Rule Management** page, find the newly created custom alert rule and click **View Details** in the Actions column. Then, you can view the details of the custom alert rule.

Add a DingTalk chatbot and obtain a webhook URL

1. Go to the DingTalk group to which you want to send alert notifications and click the **Group Settings** icon in the upper-right corner.
2. Click **Group Assistant**.
3. In the **Group Assistant** panel, click **Add Robot**.
4. In the **ChatBot** dialog box, click the + icon.
5. In the **Please choose which robot to add** section, click **Custom**.
6. In the **Robot details** message, click **Add**.
7. In the **Add Robot** dialog box, configure the parameters that are described in the following table.

Parameter	Description
Chatbot name	The name of the custom chatbot.
Add to Group	The DingTalk group to which the chatbot is added. This group cannot be changed.

Parameter	Description
Custom Keywords	After you configure custom keywords, messages are sent only when these messages contain the specified keywords. We recommend that you specify the keyword DataWorks. **Note** You can configure a maximum of 10 keywords. A message can be sent only if it contains at least one of the specified keywords.

8. Read the terms of service, select **I have read and accepted <<DingTalk Custom Robot Service Terms of Service>>**, and then click **Finished**.
9. After you complete the security settings, copy the webhook URL of the chatbot and click **Finished**.

Notice

- Save the copied webhook URL and paste it in the **Webhook Address** field when you create an alert rule.
- Keep the webhook URL confidential. If the webhook URL is leaked, your business is at risk.

2.1.8.7.3.5. View alerts

You can view all alerts on the Alert Management page.

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Task Operation > Operation Center**.
3. In the left-side navigation pane of the Operation Center page, choose **Alarm > Alert Management**. The **Alert Management** page appears.

   You can search for alerts by condition, such as **Rule ID/Name**, **Recipient**, **Alert Time**, **Notification Method**, and **Rule Type**.

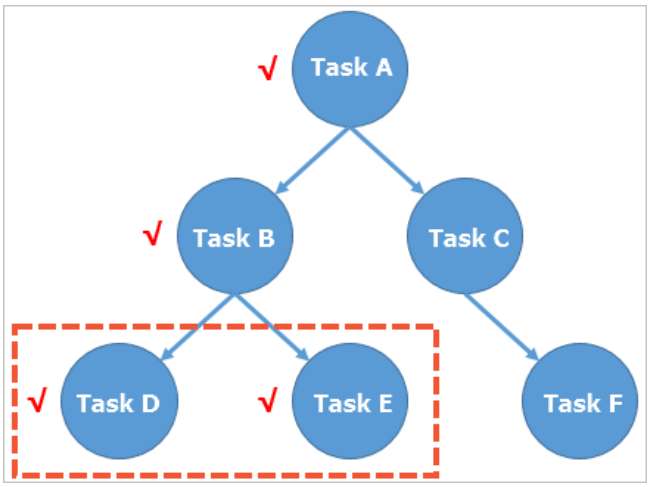
   You can also view alert details, such as the notification method and status. To view details about an alert, click **View Details** in the Actions column that corresponds to the alert.

2.1.8.7.4. 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.

2.1.9. Security Center (earlier version)

2.1.9.1. 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.

2.1.9.2. Permissions

On the My Permissions page, you can view your table or field permissions in a workspace, and request or revoke table or field permissions.

View table or field permissions

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > Security Center**. The **My Permissions** page appears.
3. On the Table tab of the **My Permissions** page, you can select a workspace in standard mode and an environment to view all the tables of the workspace in the specified environment. You can also enter a table name in the search box for fuzzy match.

   You can view the names, display names, and owners of tables in your workspace, view your permissions on the tables, and request or revoke table or field permissions.

Request table or field permissions

1. Select the tables or fields on which you want to request permissions.
   - Request permissions on a table or specific 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 or specific fields in the tables

     Select all the required tables or fields and click **Request Permission**.

     > **Note** You can also click **Request Permission** without selecting any tables or fields to go to the **Table Permission Request** page.

2. On the **Table Permission Request** page, configure the parameters as required.

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.

Parameter	Description
Grant To	The account for which you request permissions. You can request permissions for the current account or a production account of another workspace that 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 complete, click **Submit**. If you do not want to request permissions, click **Cancel**.

2.1.9.3. 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.

2.1.9.4. 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.

2.1.9.5. 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.

2.1.10. Security Center (new version)

2.1.10.1. Overview

DataWorks Security Center helps you build a security system that can secure data and personal privacy. Security Center can meet various security requirements, such as auditing, in high-risk scenarios. You can use Security Center without the need to perform additional configurations.

Security Center provides security capabilities for big data systems in the cloud throughout the entire data security lifecycle. It also provides best practices for various security diagnostic scenarios based on security specifications. Security Center provides the following features:

- Data permission management

  Security Center supports fine-grained permission requesting, request processing, and permission auditing. This allows you to manage permissions based on the principle of least privilege. In addition, Security Center allows you to view the request processing progress and follow up request processing in real time. For more information, see Data access control.

- Security diagnosis

  Security Center provides features such as platform security diagnosis and data usage diagnosis. It also provides best practices for various security diagnosis scenarios based on security specifications. These features ensure that your business is run more effectively in a secure environment. For more information, see Platform security diagnosis.

2.1.10.2. Platform security diagnosis

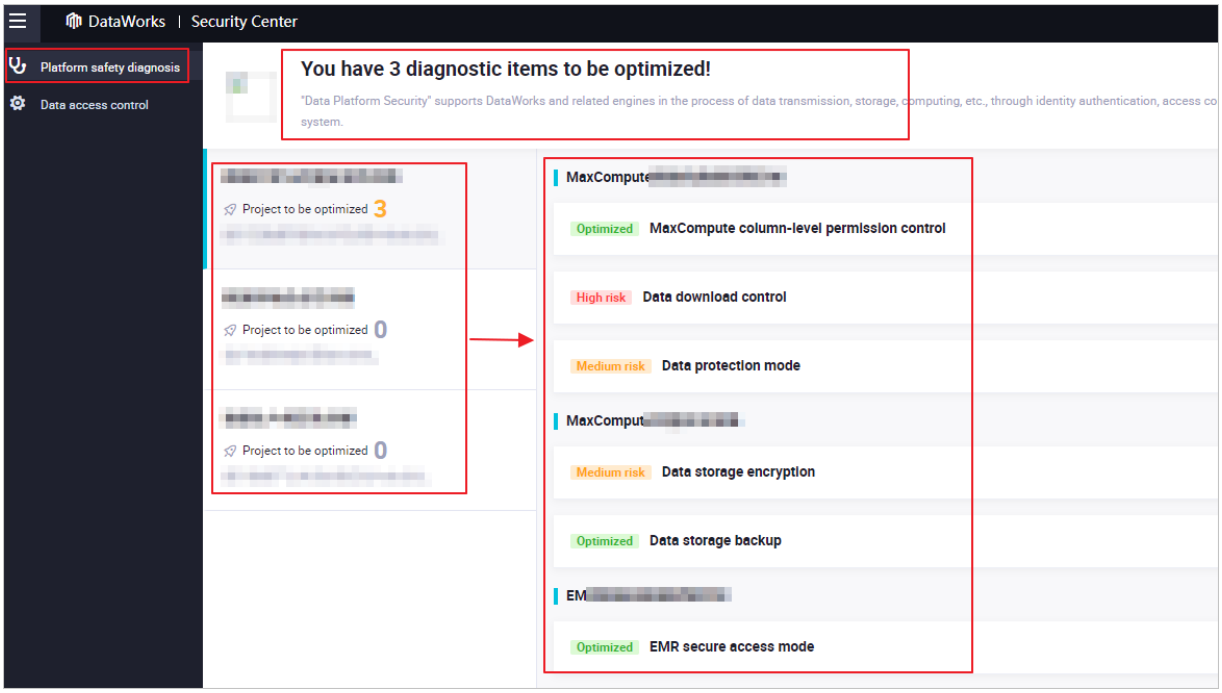
The platform security diagnosis feature of DataWorks provides security capabilities for features, such as identity authentication, access control, and development mode, during data transmission, storage, and computing on the nodes in the current DataWorks workspace and the associated compute engine. In addition, best practices are provided for security diagnosis. The platform security diagnosis feature helps you identify the security risks of your platform at the earliest opportunity and build a basic security system before you perform related transactions.

Go to the Platform safety diagnosis page

1. Log on to the DataWorks console.
2. In the upper-left corner of the page that appears, click the icon and choose **All Products > Data governance > Security Center**. The **Data access control** page appears.
3. In the left-side navigation pane, click **Platform safety diagnosis** to go to the **Platform safety diagnosis** page.

View diagnosis results

The **Platform safety diagnosis** page displays the security risks that are detected during business interactions between the current workspace and the associated compute engine instance based on the best practices for security risks. You can identify risk categories and levels based on the diagnosis results, view risk details, and process the items that can be optimized to ensure secure and reliable business interactions.



The following types of diagnostic items are provided:

- **Data calculation and storage**

  Diagnoses security issues for features such as data permission management, data storage encryption, and data storage backup, and identifies potential security risks at the earliest opportunity to improve security during data storage and access.

- **Data transmission security diagnosis**

  Diagnoses security issues for features such as the access control of data sources and the isolation of data sources in the production and development environments. In addition, this diagnostic item identifies security risks during data transmission so that you can manage these risks at the earliest opportunity. This diagnostic item ensures a secure and reliable environment for data transmission.

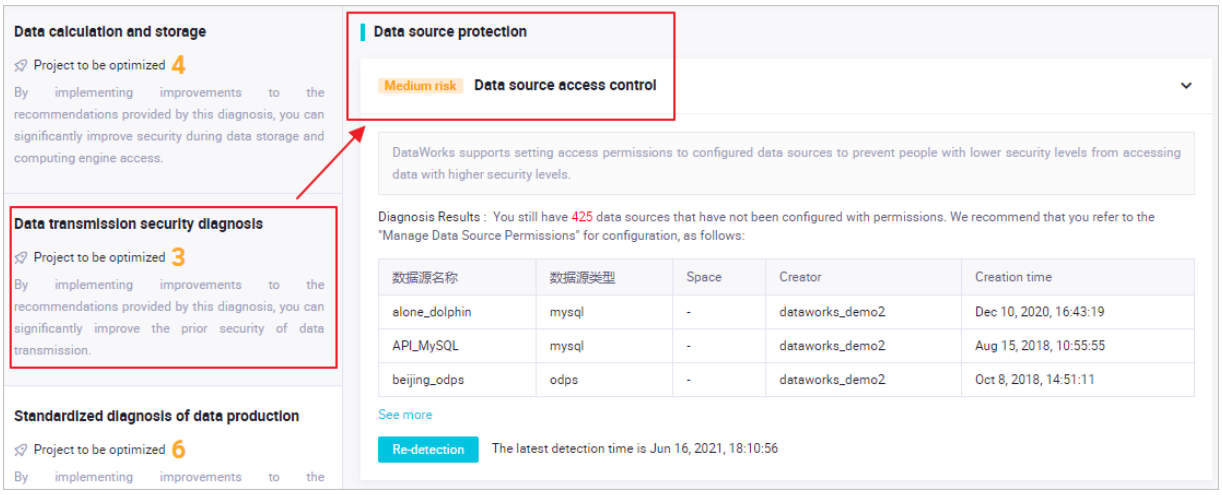
- **Standardized diagnosis of data production**

  Diagnoses security issues for production processes, such as the rationality of the roles, number of administrators, and deployment personnel within the current workspace, and allows you to identify and process security risks at the earliest opportunity. This diagnostic item improves the reliability and security of the data output system.

- **Platform security configuration diagnosis**

  Diagnoses security issues for features, such as auditing of operations on DataWorks, to improve the overall data security.

Potential security risks are classified as **medium** and **high** risks. You can click a security risk to view its details and manage the security risk at the earliest opportunity. The following figure shows the details about a **medium-level risk** of **data source access control**.



- Security risk

  Permissions are not configured on the data sources. This way, users of low security levels can access data of high security levels. This leads to insecure access to the data sources.

- Suggestion

  You can configure permissions for the data sources based on the provided suggestion to improve access security for the data sources.

2.1.10.3. Data access control

The data access control feature provides a visual interface that allows you to request permissions, process requests, view request processing progress, follow up request processing, and audit and manage permissions.

Limits

You can use the **data access control** feature to request only permissions on MaxCompute tables.

Usage note

The **Data access control** page displays the access control platform of the new version. If you want to use the access control platform of the earlier version, click **Return to old version** in the top navigation bar of the page. For more information about the access control platform of the earlier version, see Overview.

Go to the Data access control page

1. Log on to the DataWorks console.
2. In the upper-left corner of the page that appears, click the ☰ icon and choose **All Products > Data governance > Security Center**. The **Data access control** page appears.

Request permissions

1. Go to the **Permission application** tab.
2. Select the tables on which you want to request permissions.

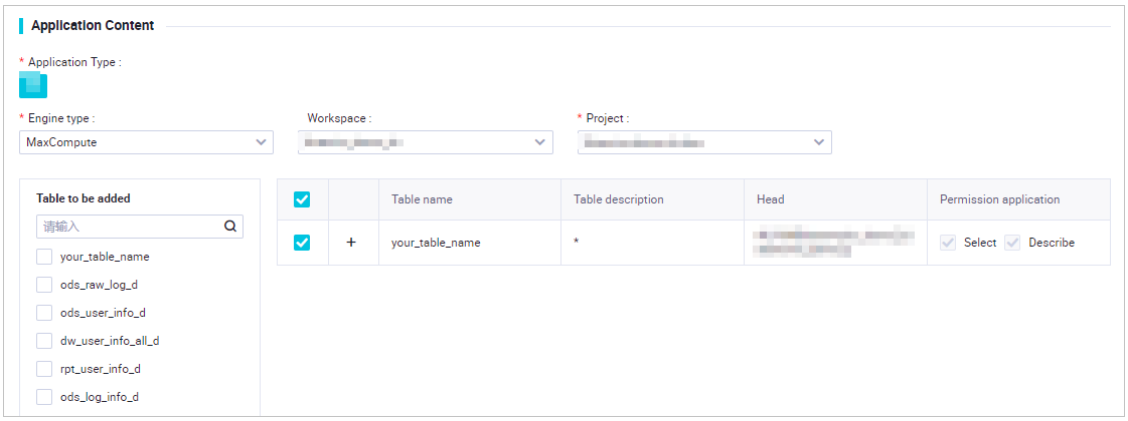
i. In the **Application Content** section, set the **Workspace** and the **Project** parameters.

You can use the **data access control** feature to request only permissions on MaxCompute tables.

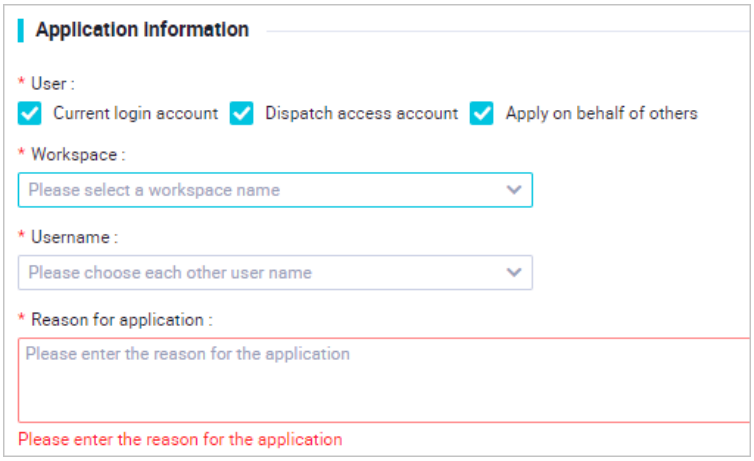
The default value of the **Application Type** parameter is **Table** and that of the **Engine type** parameter is **MaxCompute**.

ii. Select the tables on which you want to request permissions in the **Table to be added** section.

After you select tables, the information of the tables is displayed on the right. You can click the + icon on the left side of a table name to view all the fields in the table. You can request the permissions on some or all fields. By default, the permissions on all fields are requested.



3. In the **Application information** section, set the parameters.



Parameter	Description
User	◦ **Current login account**: Request the permissions on the tables for the account that is used to log on to the current workspace. ◦ **Dispatch access account**: Request the permissions on the tables for the account that has a scheduling access identity. If you select this option, you must set the **Workspace** parameter. ◦ **Apply on Behalf of others**: Request the permissions on the tables for an account that is not used to log on to the current workspace. If you select this option, you must set the **Username** parameter.

Parameter	Description
Workspace	The account that has a scheduling access identity.
Username	The username of the account that is not used to log on to the current workspace.
Reason for application	The reason why you want to request the permissions.

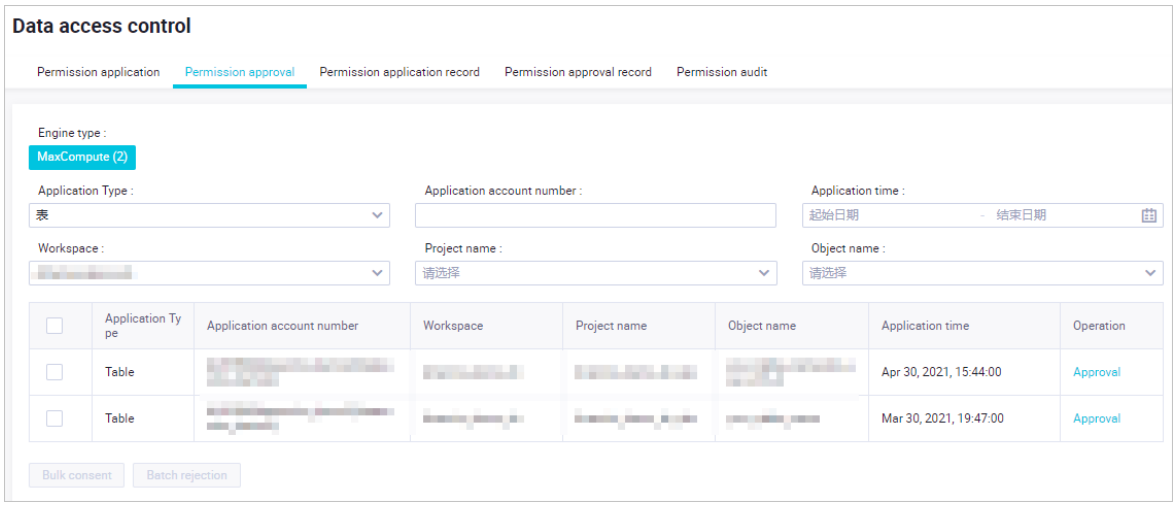
4. Click **Apply for permission** to submit the request.

   You can view the processing details and record of the current request on the **Permission application record** tab.

Process requests

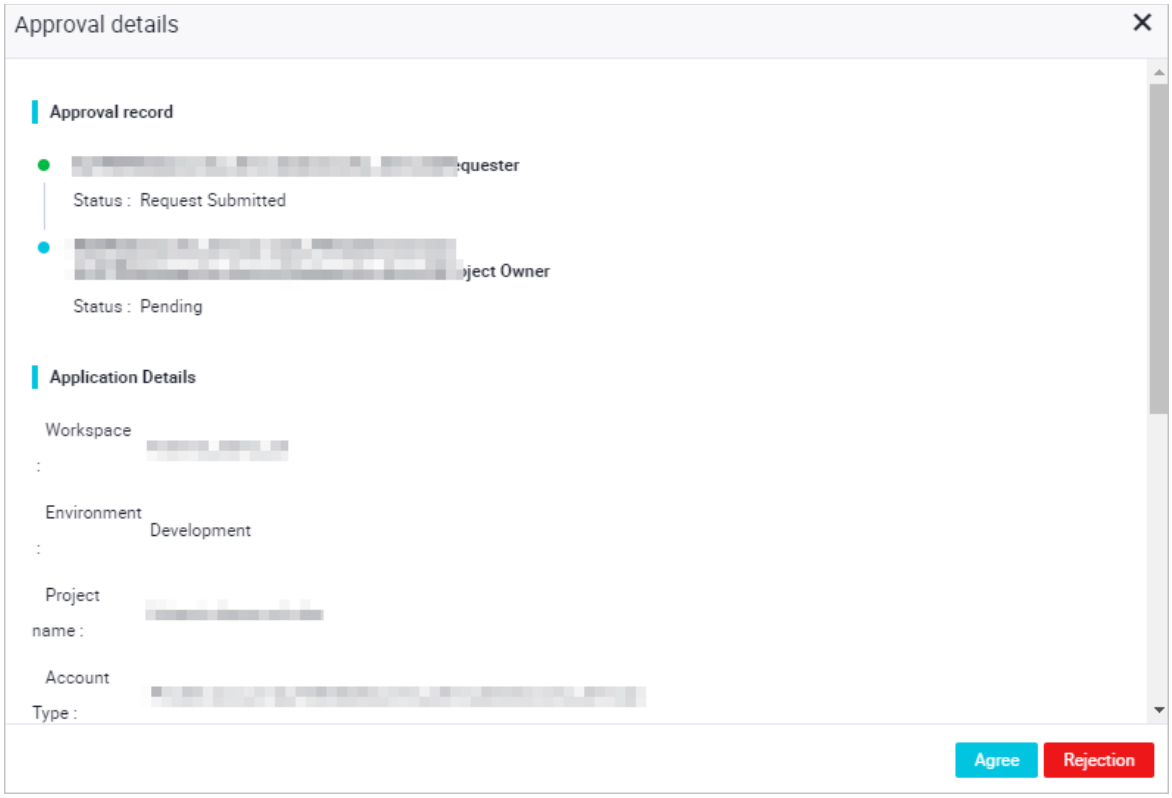
1. View the information about the pending requests.

   Go to the **Permission approval** tab. You can use the following parameters to find the pending requests within the current Apsara Stack tenant account: **Application account number**, **Application time**, **Workspace**, **Project name**, and **Object name**.



2. View the details about a request.

   Find the request and click **Approval** in the **Operation** column. Then, you can view the details and processing record of the request in the **Approval details** dialog box.



3. Process requests.

   To process a single request, enter your comments and click **Agree** or **Rejection** based on your business requirements.

   To process multiple requests at a time, you can select all the requests that you want to process on the **Permission approval** tab, click **Bulk consent** or **Batch rejection**, and then enter your comments.

View historical permission requests and their processing records

- Go to the **Permission application record** tab. Then, you can use the following parameters to find the historical permission requests within the current Apsara Stack tenant account: **Approval status**, **Application time**, **Workspace**, **Project name**, and **Table name**.

  You can click **View details** in the **Operation** column that corresponds to a request to view the details about the request. In addition, you can continue to process the requests whose approval states are **In approval**.

- Go to the **Permission approval record** tab. Then, you can use the following parameters to find the request processing records within the current Apsara Stack tenant account: **Application account number**, **Approval Results**, **Workspace**, **Project name**, **Object name**, and **Application time**.

  You can click **View details** in the **Operation** column that corresponds to a request to view the details about the request.

Audit permissions

Go to the **Permission audit** tab. Then, you can use the following parameters to find the permission requests that are processed for the workspace, project, or object in Security Center: **Workspace**, **Project name**, and **Object name**.

2.1.11. Data Quality

2.1.11.1. 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.

2.1.11.2. Features

2.1.11.2.1. 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.

2.1.11.2.2. 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.
- 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.

2.1.11.2.3. 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.

Parameter	Description
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.
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**.

Parameter	Description
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**.

2.1.11.2.4. 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**.
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.

2.1.11.2.5. 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
Basic settings	**Name**	The name of the report template.
	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.

Section	Parameter	Description
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.
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.

2.1.11.2.6. 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.

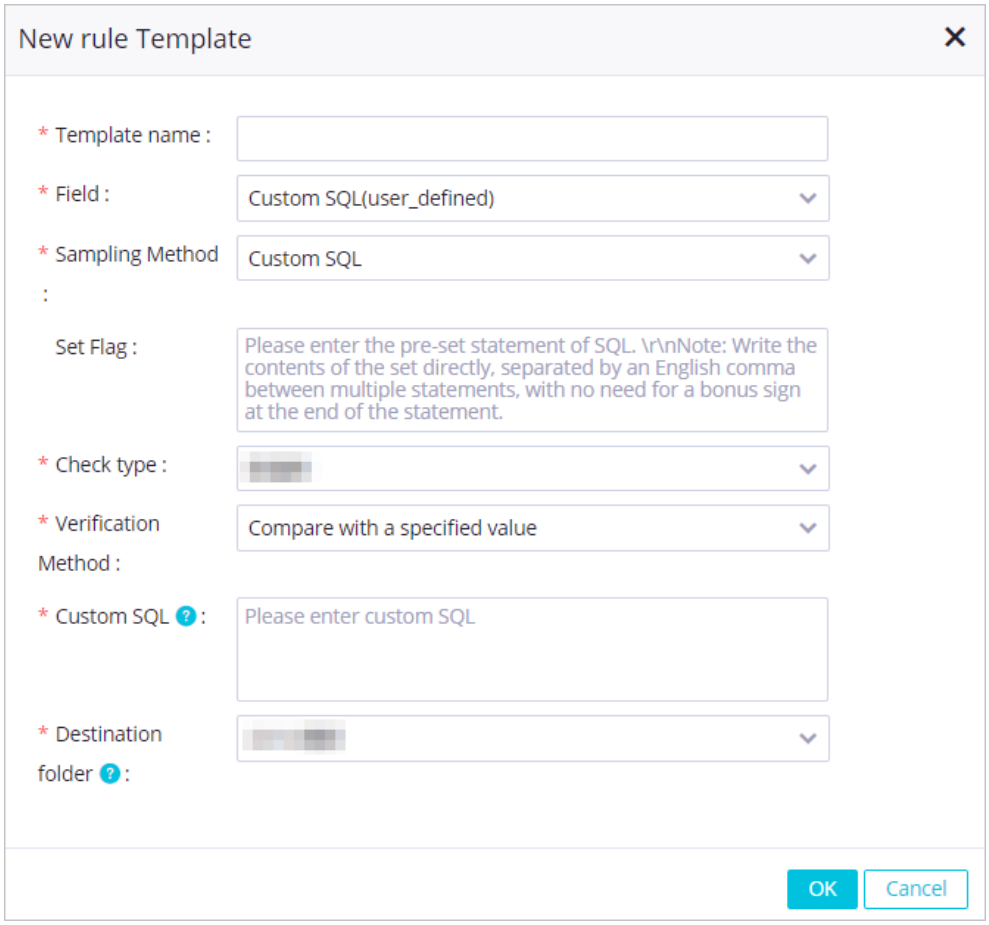
Context

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**.

Parameter	Description
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.

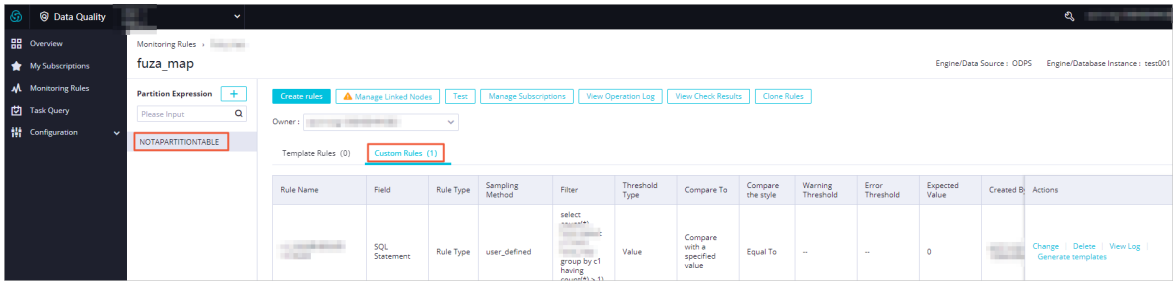
8. Click **OK**.

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.

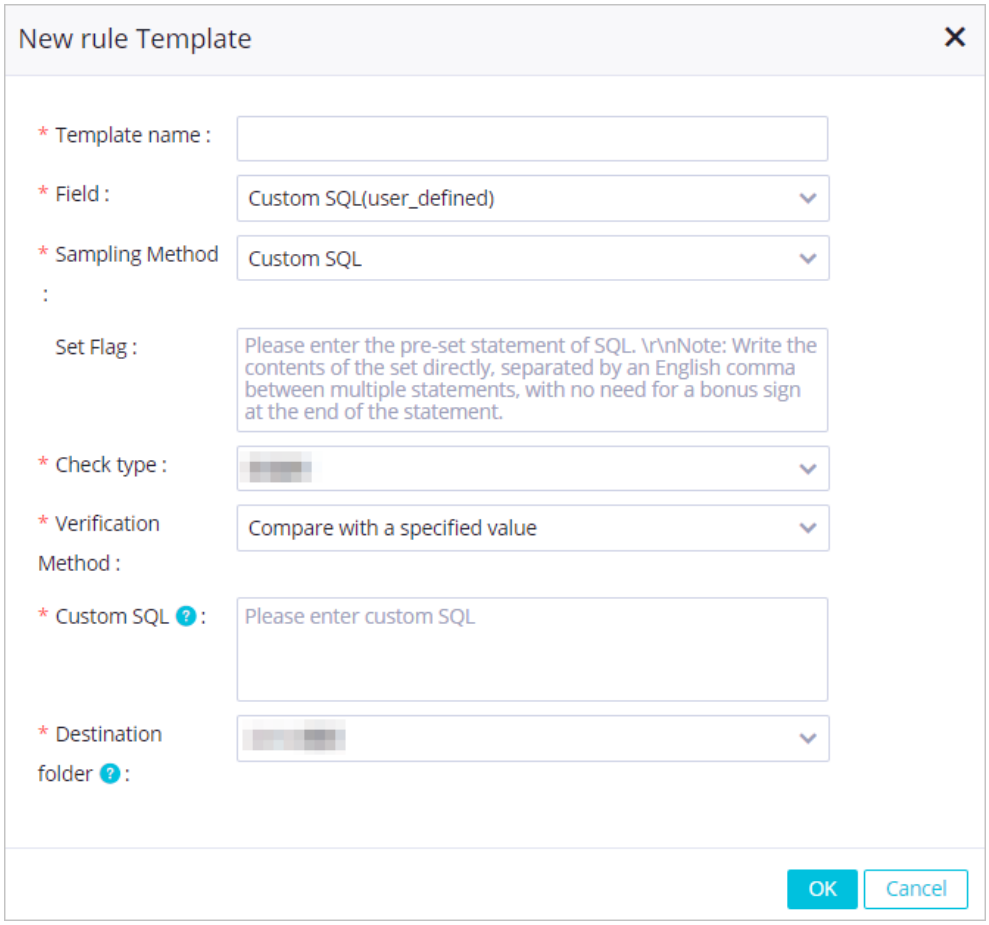
> **Note** This topic uses a MaxCompute table as an example.

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.



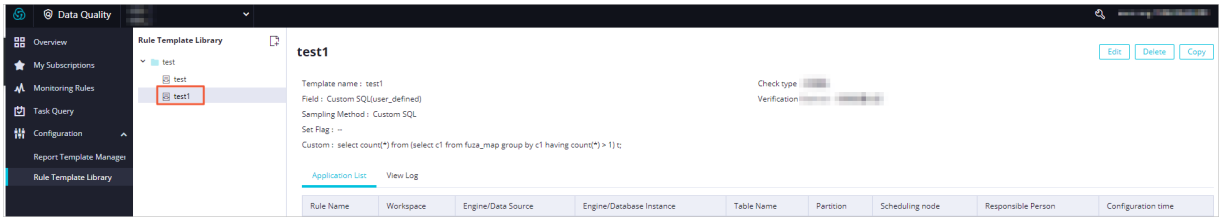
Parameter	Description
Template Name	The name of the rule template.
Field	The fields to be monitored and the statistical function. You can only set the two parameters to **Custom SQL**.
Sampling Method	
Set Flag	The `SET` clause of the SQL statement for querying the field to be monitored. **Note** Separate 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**.

Parameter	Description
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
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**.

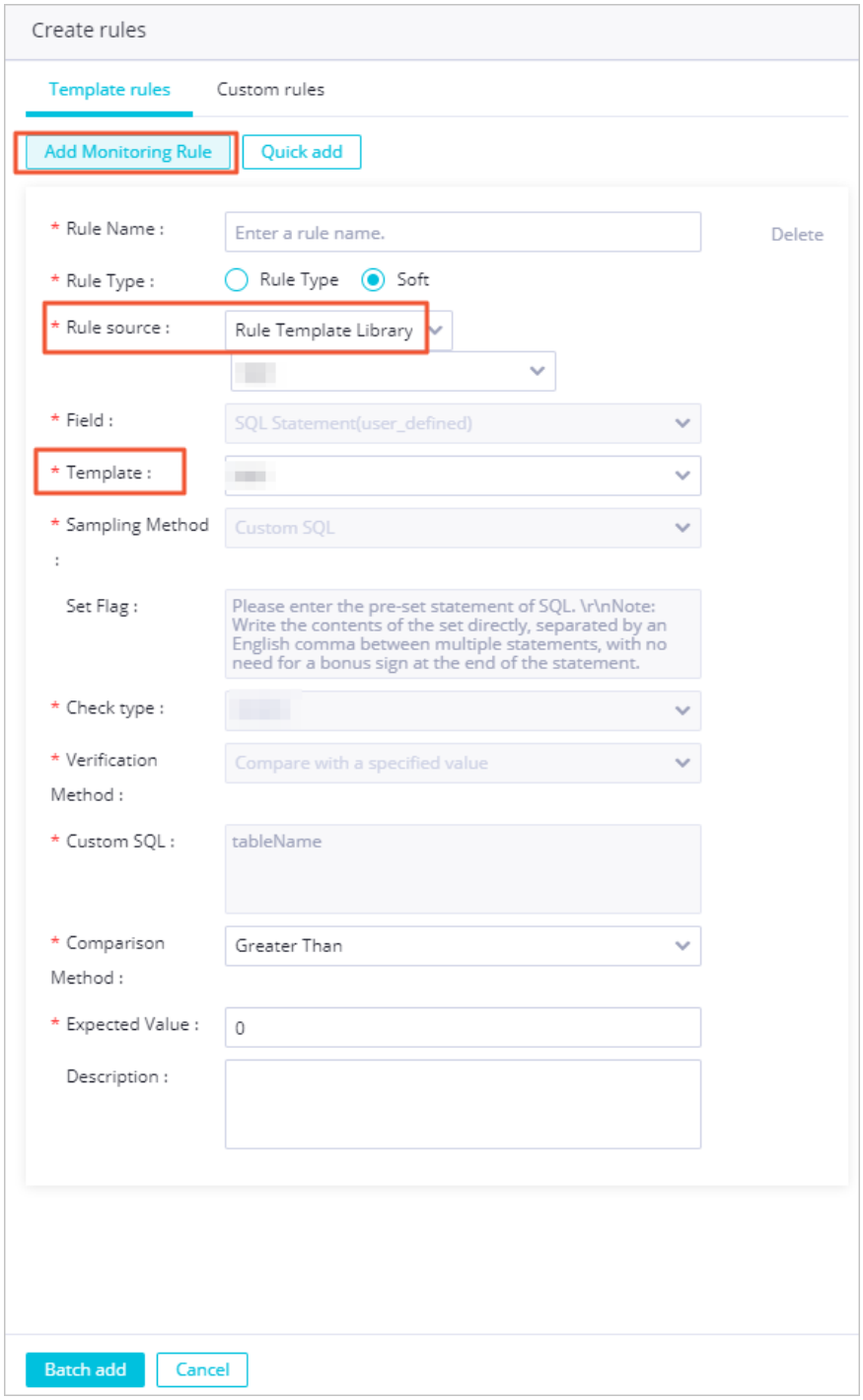
Use a rule template

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**.

2.1.11.3. User guide

2.1.11.3.1. 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.

Partition filter expression	Description
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
Add Monitoring Rule	**Rule Name**	The name of the 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.

Creation method	Parameter	Description
	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.
Quick Create	**Rule Name**	The name of the rule.
	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

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**.

2.1.11.3.2. 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 (EMR), 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 data source.
   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 Data Integration page, choose **Data Source > Data Stores**.
   iv. On the Data Source page, click **Add data source** in the upper-right corner to add a DataHub data source. For more information, see Configure monitoring rules for DataHub.
2. Select the DataHub data source.
   i. On the current page, click the icon in the upper-left corner and choose **All Products > Data governance > Data Quality**.
   ii. On the page that appears, click **Monitoring Rules** in the left-side navigation pane.

iii. On the Monitoring Rules page, select **Datahub** from the **Engine/Data Source** drop-down list and select the newly added DataHub data source from the Engine/Database Instance drop-down list. All the topics in the selected DataHub data source are displayed.

Parameter	Description
Configure Flink/SLS Resources	After you add a data source, click Configure Flink/SLS Resources to configure Realtime Compute and Log Service resources related to the data source.
Topics	The Topics tab lists all the topics in the DataHub data source. 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. ■ **Manage Subscriptions**: Click it to view and modify subscribers to the topic, and change the notification method. You can use a DingTalk chatbot to receive notifications. The changed notification method takes effect for all subscribers to the topic.
Dimension Tables	When you create custom rules for a topic, you can create and 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 RDS, Tablestore, Taobao Distributed Data Layer (TDDL), and MaxCompute. Flink SQL does not design the data definition language (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 for each primary key of the tables.

iv. Click the **Topics** tab. Find the topic for which you want to configure monitoring rules and click **View Monitoring Rules** in the Actions column.

3. On the rule configuration page of the topic, click **Create Rule**.
4. Create a monitoring rule.

   In Data Quality, you can create template rules and custom rules.

   - On the Template Rules tab of the Create rules panel, click **Create Template Rule**. Two templates are available: **Data Delay** and **Stream Discontinuity**.

     For example, you can select **Data Delay** for the Template Type parameter.

Parameter	Description
Rule Name	The name of the rule. The name can be a maximum of 255 characters in length.
Field Type	The fields to be monitored. By default, this parameter is set to 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 is displayed only when you select Data Delay for the Template Type parameter.
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 is displayed only when you select Data Delay for the Template Type parameter.
Alert Frequency	The interval at which alerts are reported. 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 create a custom rule. On the Custom Rules tab of the Create rules panel, click **Create Custom Rule**.

Note
- The field in the SELECT clause must be a column. Make sure that you can compare the field values with the warning threshold and error threshold.
- The FROM clause must include the current topic and all its columns.

Parameter	Description
Rule Name	The name of the rule. The name must be unique in the topic and can be a maximum of 20 characters in length.

Parameter	Description

Script	The custom SQL script that 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 at which alerts are reported, in minutes.
Description	The description of the rule.

5. Click **Batch Create**. After rules are created for the topic, you can perform the following operations:
   - **View Log**: Click it to view the operational logs of the rules.
   - **Manage Subscriptions**: Click it to view and modify subscribers to the rules, and change the notification method. The changed notification method takes effect for all the subscribers of the rules.

     Data Quality supports the following methods: **Email**, **Email and SMS**, **DingTalk Chatbot**, and **DingTalk Chatbot @ALL**.

     > **Note** Add a DingTalk chatbot and obtain the webhook address of the chatbot. Then, copy the webhook address to the Manage Subscriptions dialog box.

2.1.12. Data Map

2.1.12.1. Overview

Data Map is developed based on Data Management and 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.

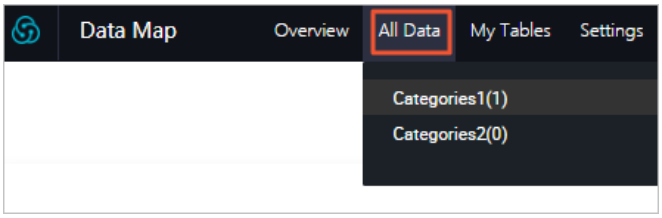
On the homepage of Data Map, you can enter keywords to search for tables by name. You can also click a table in one of the following sections to view the table data: **Recently Viewed Tables**, **Recently Read Tables**, **Most Viewed Tables**, or **Most Read Tables**.

- If you prefer a powerful search engine, go to the homepage to search for data.

  > **Note** The homepage appears when you go to the **Data Map** page. To return to the homepage from other pages, click **Data Map** in the upper-left corner.

- If you want to find tables by workspace or cluster, click **All Data** in the top navigation bar. On the page that appears, you can view tables on different tabs, such as **MaxCompute** or **EMR**. You can also find a table and perform the following operations on it: **Add to Favorites**, **Apply for Permission**, **View Lineage**, or **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 of and permissions on MaxCompute tables.

- If you want 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 want to modify tables that are owned by your account, click **My Data** in the top navigation bar. For more information, see View and manage tables and data permissions.
- If you are a category administrator or workspace administrator and want to modify the workspace configurations or global categories, click **Configuration Management** in the top navigation bar. For more information, see Manage categories of and permissions on MaxCompute tables.

2.1.12.2. Configure whitelists and category management permissions

If you want to view MaxCompute table data or collect metadata in Data Map, you must configure a whitelist for your MaxCompute project or the desired data source. Then, add the Classless Inter-Domain Routing (CIDR) blocks of the region where your DataWorks workspace resides to the whitelist. If you want to manage categories in Data Map, you must grant the related permissions to the account you use. This topic describes how to configure the whitelists and grant category management permissions.

Context

- Data Map provides a platform to manage both metadata and the data assets of enterprises. This platform allows you to search global data, view metadata details, preview data, view data lineages, and manage data categories. Data Map helps you search for, understand, and use data. If you want to use Data Map to view the table data in a MaxCompute project, check whether a whitelist is configured for the project. If a whitelist is configured, make sure that the CIDR blocks of the region where your DataWorks workspace resides are in the whitelist. Otherwise, you cannot view the table data in Data Map. Therefore, to ensure that Data Map can access MaxCompute projects, you must configure whitelists for the projects in advance.

  > **Note** Only MaxCompute requires whitelist configuration. For other products, you can directly view data in Data Map.

- The metadata collection feature allows you to collect metadata from different data sources. This way, you can manage the metadata in a centralized manner. After the metadata of a data source is collected, you can view the metadata in Data Map. Before you collect metadata from the data source, check whether a whitelist is configured for the data source. If a whitelist is configured, make sure that the CIDR blocks of the region where your DataWorks workspace resides are in the whitelist.
- The category management feature allows you to effectively organize and manage tables by category. For more information, see Manage categories of and permissions on MaxCompute tables. Before you use this feature, make sure that you have the required permissions.
  - If you use an Alibaba Cloud account to manage categories, you have the permissions by default.
  - If you use a RAM user to manage categories, you must attach the **AliyunDataWorksFullAccess** policy to the RAM user.

Configure a whitelist for a MaxCompute project to allow Data Map to access the project

1. Check whether a whitelist is configured for your MaxCompute project. For more information, see *View an IP address whitelist* in the MaxCompute documentation.

   If a whitelist is not configured for the project, Data Map can access the table data in the project. If a whitelist is configured, proceed to the next step.

2. Configure the whitelist.

   Add the desired CIDR blocks of the region where your DataWorks workspace resides to the whitelist. The following table lists the CIDR blocks of each region. For more information about how to configure a whitelist, see *Configure IP address whitelists*.

Region	CIDR block or IP address
China (Hangzhou)	100.64.0.0/10,11.193.102.0/24,11.193.215.0/24,11.194.110.0/24,11.194.73.0/24,118.31.157.0/24,47.97.53.0/24,11.196.23.0/24,47.99.12.0/24,47.99.13.0/24,114.55.197.0/24,11.197.246.0/24,11.197.247.0/24
China (Shanghai)	11.193.109.0/24,11.193.252.0/24,47.101.107.0/24,47.100.129.0/24,106.15.14.0/24,10.117.28.203,10.143.32.0/24,10.152.69.0/24,10.153.136.0/24,10.27.63.15,10.27.63.38,10.27.63.41,10.27.63.60,10.46.64.81,10.46.67.156,11.192.97.0/24,11.192.98.0/24,11.193.102.0/24,11.218.89.0/24,11.218.96.0/24,11.219.217.0/24,11.219.218.0/24,11.219.219.0/24,11.219.233.0/24,11.219.234.0/24,118.178.142.154,118.178.56.228,118.178.59.233,118.178.84.74,120.27.160.26,120.27.160.81,121.43.110.160,121.43.112.137,100.64.0.0/10,10.117.39.238
China (Shenzhen)	100.106.46.0/24,100.106.49.0/24,10.152.27.0/24,10.152.28.0/24,11.192.91.0/24,11.192.96.0/24,11.193.103.0/24,100.64.0.0/10,120.76.104.0/24,120.76.91.0/24,120.78.45.0/24,47.106.63.0/26,47.106.63.128/26,47.106.63.192/26,47.106.63.64/26
China (Chengdu)	11.195.52.0/24,11.195.55.0/24,47.108.22.0/24,100.64.0.0/10
China (Zhangjiakou)	11.193.235.0/24,47.92.22.0/24,100.64.0.0/10
China (Hong Kong)	10.152.162.0/24,11.192.196.0/24,11.193.11.0/24,100.64.0.0/10,47.89.61.0/24,47.91.171.0/24,11.193.118.0/24,47.75.228.0/24,47.56.45.0/25,47.244.92.128/25,47.101.109.0/24
Singapore (Singapore)	100.106.10.0/24,100.106.35.0/24,10.151.234.0/24,10.151.238.0/24,10.152.248.0/24,11.192.153.0/24,11.192.40.0/24,11.193.8.0/24,100.64.0.0/10,47.88.147.0/24,47.88.235.0/24,11.193.162.0/24,11.193.163.0/24,11.193.220.0/24,11.193.158.0/24,47.74.162.0/24,47.74.203.0/24,47.74.161.0/24,11.197.188.0/24

Region	CIDR block or IP address
Australia (Sydney)	11.192.100.0/24,11.192.134.0/24,11.192.135.0/24,11.192.184.0/24,11.192.99.0/24,100.64.0.0/10,47.91.49.0/24,47.91.50.0/24,11.193.165.0/24,47.91.60.0/24
China (Beijing)	100.106.48.0/24,10.152.167.0/24,10.152.168.0/24,11.193.50.0/24,11.193.75.0/24,11.193.82.0/24,11.193.99.0/24,100.64.0.0/10,47.93.110.0/24,47.94.185.0/24,47.95.63.0/24,11.197.231.0/24,11.195.172.0/24,47.94.49.0/24,182.92.144.0/24,39.99.77.0/26,39.99.77.64/26,39.99.77.128/26,39.104.220.192/26,39.107.7.0/26,39.107.7.64/26,182.92.32.128/26,182.92.32.192/26
US (Silicon Valley)	10.152.160.0/24,100.64.0.0/10,47.89.224.0/24,11.193.216.0/24,47.88.108.0/24
US (Virginia)	47.88.98.0/26,47.88.98.64/26,47.88.98.128/26,47.88.98.192/26,47.252.91.0/26,47.252.91.64/26,47.252.91.128/26,47.252.91.192/26,10.128.134.0/24,11.193.203.0/24,11.194.68.0/24,11.194.69.0/24,100.64.0.0/10
Malaysia (Kuala Lumpur)	11.193.188.0/24,11.221.205.0/24,11.221.206.0/24,11.221.207.0/24,100.64.0.0/10,11.214.81.0/24,47.254.212.0/24,11.193.189.0/24
Germany (Frankfurt)	11.192.116.0/24,11.192.168.0/24,11.192.169.0/24,11.192.170.0/24,11.193.106.0/24,100.64.0.0/10,11.192.116.14,11.192.116.142,11.192.116.160,11.192.116.75,11.192.170.27,47.91.82.22,47.91.83.74,47.91.83.93,47.91.84.11,47.91.84.110,47.91.84.82,11.193.167.0/24,47.254.138.0/24
Japan (Tokyo)	100.105.55.0/24,11.192.147.0/24,11.192.148.0/24,11.192.149.0/24,100.64.0.0/10,47.91.12.0/24,47.91.13.0/24,47.91.9.0/24,11.199.250.0/24,47.91.27.0/24,11.59.59.0/24,47.245.51.128/26,47.245.51.192/26,47.91.0.128/26,47.91.0.192/26
UAE (Dubai)	11.192.107.0/24,11.192.127.0/24,11.192.88.0/24,11.193.246.0/24,47.91.116.0/24,100.64.0.0/10
India (Mumbai)	11.194.10.0/24,11.246.70.0/24,11.246.71.0/24,11.246.73.0/24,11.246.74.0/24,100.64.0.0/10,149.129.164.0/24,11.194.11.0/24,11.59.62.0/24,147.139.23.0/26,147.139.23.128/26,147.139.23.64/26,149.129.165.192/26
UK (London)	11.199.93.0/24,100.64.0.0/10
Indonesia (Jakarta)	11.194.49.0/24,11.200.93.0/24,11.200.95.0/24,11.200.97.0/24,100.64.0.0/10,149.129.228.0/24,10.143.32.0/24,11.194.50.0/24,11.59.135.0/24,147.139.156.0/26,147.139.156.128/26,147.139.156.64/26,149.129.230.192/26
China North 2 Ali Gov 1	11.194.116.0/24,100.64.0.0/10,39.107.188.202 If the preceding CIDR blocks cannot be added, add the following information: 11.194.116.160,11.194.116.161,11.194.116.162,11.194.116.163,11.194.116.164,11.194.116.165,11.194.116.167,11.194.116.169,11.194.116.170,11.194.116.171,11.194.116.172,11.194.116.173,11.194.116.174,11.194.116.175,39.107.188.0/24.
China East 2 Finance	140.205.46.128/25,140.205.48.0/25,140.205.48.128/25,140.205.49.0/25,140.205.49.128/25,11.192.156.0/25,11.192.157.0/25,11.192.164.0/25,11.192.165.0/25,11.192.166.0/25,11.192.167.0/25,106.11.245.0/26,106.11.245.128/26,106.11.245.192/26,106.11.245.64/26,140.205.39.0/24,106.11.225.0/24,106.11.226.0/24,106.11.227.0/24,106.11.242.0/24,100.104.8.0/24

Configure a whitelist for metadata collection from a data source

1. Check whether a whitelist is configured for the data source.

   Data Map allows you to collect metadata from the following types of data sources:

- Collect metadata from an EMR data source
- Collect metadata from an AnalyticDB for PostgreSQL data source
- Collect metadata from a MySQL data source
- Collect metadata from a PostgreSQL data source
- Collect metadata from an SQL Server data source
- Collect metadata from an Oracle data source
- Collect metadata from an AnalyticDB for MySQL 2.0 data source
- Collect metadata from an AnalyticDB for MySQL 3.0 data source
- Collect metadata from a Hologres data source

The method used to check whether a whitelist is configured varies based on the data source type. You can consult technical support personnel.

If a whitelist is not configured for the data source, you can directly use Data Map to collect metadata from the data source. If a whitelist is configured, proceed to the next step.

2. Configure the whitelist.

   Add the desired CIDR blocks of the region where your DataWorks workspace resides to the whitelist. The following table lists the CIDR blocks of each region. The position where the CIDR blocks are added varies based on the data source type. You can consult technical support personnel.

Region	CIDR block or IP address
China (Shanghai)	100.104.189.64/26,11.115.110.10/24,11.115.109.9/24,47.102.181.128/26,47.102.181.192/26,47.102.234.0/26,47.102.234.64/26,100.104.38.192/26
China (Hangzhou)	100.104.135.128/26,11.193.215.233/24,11.194.73.32/24,118.31.243.0/26,118.31.243.64/26,118.31.243.128/26,118.31.243.192/26,100.104.242.0/26
China (Shenzhen)	100.104.46.128/26,11.192.91.119/24,120.77.195.128/26,120.77.195.192/26,120.77.195.64/26,47.112.86.0/26,100.104.138.128/26
China (Beijing)	100.104.37.128/26,11.193.82.20/24,11.197.254.171/24,39.107.223.0/26,39.107.223.64/26,39.107.223.128/26,39.107.223.192/26,100.104.152.128/26
China (Chengdu)	100.104.88.64/26,11.195.57.28/24,47.108.46.0/26,47.108.46.64/26,47.108.46.128/26,47.108.46.192/26,100.104.248.128/26
China (Zhangjiakou)	100.104.197.0/26,11.193.236.121/24,47.92.185.0/26,47.92.185.64/26,47.92.185.128/26,47.92.185.192/26,100.104.75.64/26

Configure category management permissions for a RAM user

If you use a RAM user to manage categories, you must attach the **AliyunDataWorksFullAccess** policy to the RAM user.

What's next

After the whitelists and category management permissions are configured, you can view MaxCompute table data, collect metadata, or manage categories in Data Map.

2.1.12.3. 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. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
3. In the top navigation bar, click **Overview**.

   The **Overview** page displays the offline statistics of the current tenant.

   > **Note** The data on the Overview page is generated on the previous day.

Section	Description
Total Number of Projects	The total number of projects of the specified type for the current tenant.
Total Number of Tables	The total number of tables in the destination project or destination database for the current tenant.
Storage	The total storage space that is occupied by all the tables in the destination MaxCompute project for the current tenant.
Storage trend chart	The offline statistics on the trend of storage usage.
Top Projects by Table Storage	The top projects that occupy the most storage space for the current tenant.
Top Tables by Occupied Storage	The top tables that occupy the most storage space for the current tenant. You can click a table name to go to the details page of the table. **Note** The logical storage space that is occupied by projects and tables is calculated 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 not only the table storage volume but also the 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. You are charged for the logical storage volume of a table rather than the physical storage volume of a table.
Most Frequently Used Tables	The most frequently referenced tables for the current tenant. You can click a table name to go to the details page of the table.

2.1.12.4. View and manage tables and data permissions

This topic describes how to view and manage tables on the Owned by Me, Managed by Me, Managed by Tenant Account, and My Favorites pages. This topic also describes how to view and manage data permissions.

Context

Data Map updates data one day after the data is generated. If you want to query real-time data, we recommend that you use SQL statements.

Go to the Owned by Me page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data**

governance > DataMap(Data Management).

3. In the top navigation bar, click **My Data**. The **Owned by Me** page appears.

View and manage tables on the Owned by Me page

On the **Owned by Me** page, you can search for your desired table by keyword, environment, project or data source, and visibility. You can also view the details about a table and perform operations on the table.

Parameter	Description
Table Name	The name of the table. You can click a table name to go to the details page of the table.
Display Name	The display name of the table. You can click the icon in the **Display Name** column of a table to modify the display name of the table.
Project/Data Store	The name of the project or data source to which the table belongs. The suffix of the name varies based on the environment where the table resides. For example, **_dev** indicates that the table resides in the development environment.
Hide or Show	You can click the icon in the Hide or Show column of a table to display or hide the table. Valid values: **Show**, **Hide**, and **Within the Project Only**.
TTL (Days)	The TTL of the table. The value is the same as that you set when you created the table.
Environment	The environment where the table resides. Valid values: **Development** and **Production**.
Storage	The volume 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.
Edit, Change Owner, Delete, and Change Category	The operations that you can perform on multiple tables at a time. You can select tables and click **Edit**, **Change Owner**, **Delete**, or **Change Category** to modify the tables, change the owners of the tables, delete the tables, or change the categories of the tables.

View and manage tables on the Managed by Me page

In the left-side navigation pane, click **Managed by Me**. On the page that appears, you can search for your desired table by keyword, project or data source, and environment. You can also view the details about a table and perform operations on the table.

Parameter	Description
Table Name	The name of the table. You can click a table name to go to the details page of the table.

Parameter	Description
Display Name	The display name of the table. You can click the icon in the **Display Name** column of a table to modify the display name of the table.
Project/Data Store	The name of the project or data source to which the table belongs. The suffix of the name varies based on the environment where the table resides. For example, **_dev** indicates that the table resides in the development environment.
TTL (Days)	The TTL of the table. The value is the same as that you set when you created the table.
Environment	The environment where the table resides. Valid values: **Development** and **Production**.
Storage	The volume 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.
Edit, Change Owner, Delete, and Change Category	The operations that you can perform on multiple tables at a time. You can select tables and click **Edit**, **Change Owner**, **Delete**, or **Change Category** to modify the tables, change the owners of the tables, delete the tables, or change the categories of the tables.

View and manage tables 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 your desired table by keyword and project or data source and view the details about a table.

Parameter	Description
Table Name	The name of the table. You can click a table name to go to the details page of the table.
Display Name	The display name of the table.
Project/Data Store	The name of the project or data source to which the table belongs. You can click a project or data source name in the Project/Data Store column of a table to go to the details page of the project or data source.
TTL (Days)	The TTL of the table. The value is the same as that you set when you created the table.
Environment	The environment where the table resides. Valid values: **Development** and **Production**.
Storage	The volume of data that is stored in the table.
Favorites	The number of times that users add the table to favorites.

Parameter	Description
Views in Last 30 Days	The number of times that users view the table in the last 30 days. You can click the icon to arrange the tables by the number of views in ascending or descending order.
Created At	The time when the table was created.

View tables on the Tables of Other Tenants page

In the left-side navigation pane, click **Tables of Other Tenants**. On the page that appears, you can search for your desired table by table name and project or data source name and view the details about a table.

Parameter	Description
Table Name	The name of the table. You can click a table name to go to the details page of the table.
Display Name	The display name of the table.
Project/Data Store	The name of the project or data source to which the table belongs. You can click a project or data source name in the Project/Data Store column of a table to go to the details page of the project or data source.
Physical Storage	The volume of data that is stored in the table.
TTL (Days)	The TTL of the table. The value is the same as that you set when you created the table.
Created At	The time when the table was created.

View and manage tables on the My Favorites page

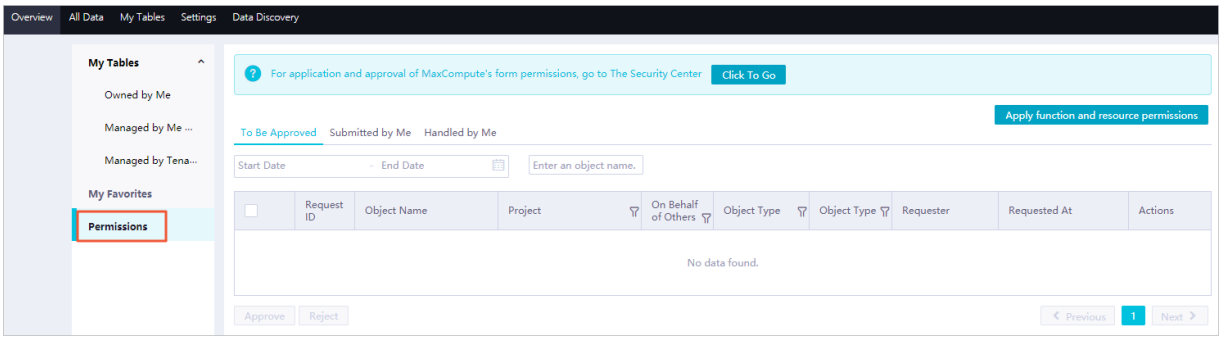
In the left-side navigation pane, click **My Favorites**. On the page that appears, you can view the tables that you have added to favorites. You can specify the following conditions to search for the tables that you have added to favorites: **Data Type**, **Project/Data Store**, and **Table Name**.

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 **Permission Management**. On the page that appears, you can view and manage data permissions.

You can click **Apply for Function and Resource Permissions** in the upper-right corner of the **Permission Management** page to request permissions. You can also view permission request details on the **To Be Approved**, **Submitted by Me**, and **Handled by Me** tabs.



- **Apply for Function and Resource Permissions**
  i. On the Permission Management page, click **Apply for Function and Resource Permissions** in the upper-right corner.
  ii. In the **Apply for Function and Resource Permissions** dialog box, configure the parameters. The following table describes the parameters.

Parameter	Description
Object Type	The type of the object on which you want to request permissions. Valid values: **Functions** and **Resources**.
Grant To	The account to which permissions will be granted. Valid values: **Current Account** and **Specified Account**. ■ If you select **Current Account**, permissions will be granted to you after the request is approved. ■ If you select **Specified Account**, you must also set the **Username** parameter. Permissions will be granted to the specified account after the request is approved.
Project Name	The name of the MaxCompute project that contains the function or resource on which you want to request permissions. Fuzzy match is supported.
Function Name	The 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, such as 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 is exceeded, 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 view and approve the requests for permissions on all objects such as tables, resources, and functions in the workspace on the **To Be Approved** tab.

- **Submitted by Me**

  On the **Permission Management** page, click the **Submitted by Me** tab.

  On the **Submitted by Me** tab, you can view the permission requests that you have submitted.

- **Handled by Me**

  On the **Permission Management** page, click the **Handled by Me** tab.

  If you are the workspace administrator, you can view the permission requests that you have handled for all objects such as tables, resources, and functions in the workspace on the **Handled by Me** tab.

Manually synchronize a table

1. In the left-side navigation pane, choose **My Tools > Manually Sync Table** to go to the **Manually Sync Table** page.
2. Enter a table GUID in the **Table GUID** field. Then, click **Manually Sync Table** to synchronize the table.

2.1.12.5. Manage categories of and permissions on MaxCompute tables

This topic describes how to manage categories of and permissions on MaxCompute tables that are in your owned or managed workspaces on the Configuration Management page of Data Map.

Go to the Configuration Management page

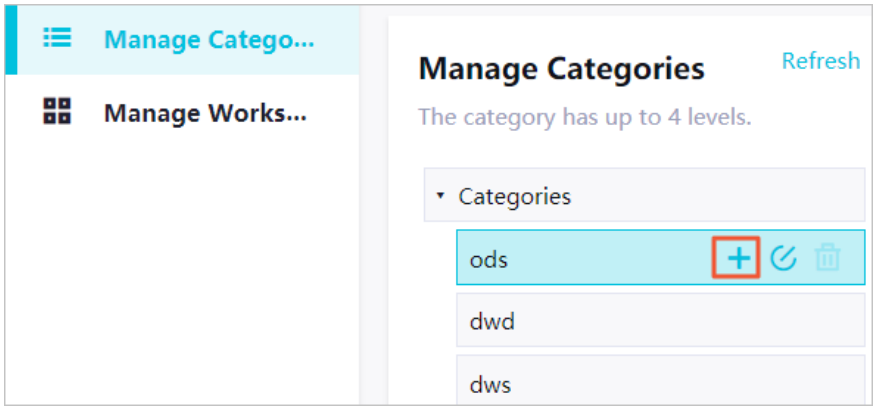
1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
3. On the Data Map page, click **Configuration Management** in the top navigation bar. The **Manage Categories** tab is displayed.

   The Configuration Management page allows you to manage categories and permissions on MaxCompute tables in a workspace.

Manage categories

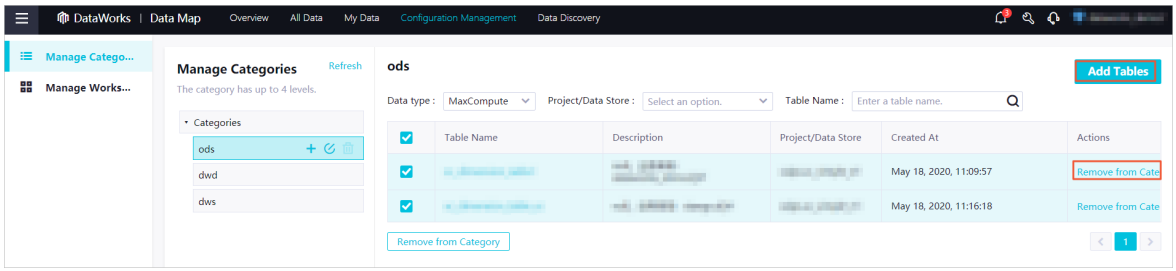
On the **Manage Categories** page, you can perform the following steps to create a category and add tables to and remove tables from the category.

1. On the **Manage Categories** page, move the pointer over **Categories** and click the + icon. In the field that appears, enter a category name and press Enter to create a level-1 category.
2. Move the pointer over the level-1 category and click the + icon. In the field that appears, enter a category name and press Enter to create a level-2 category.

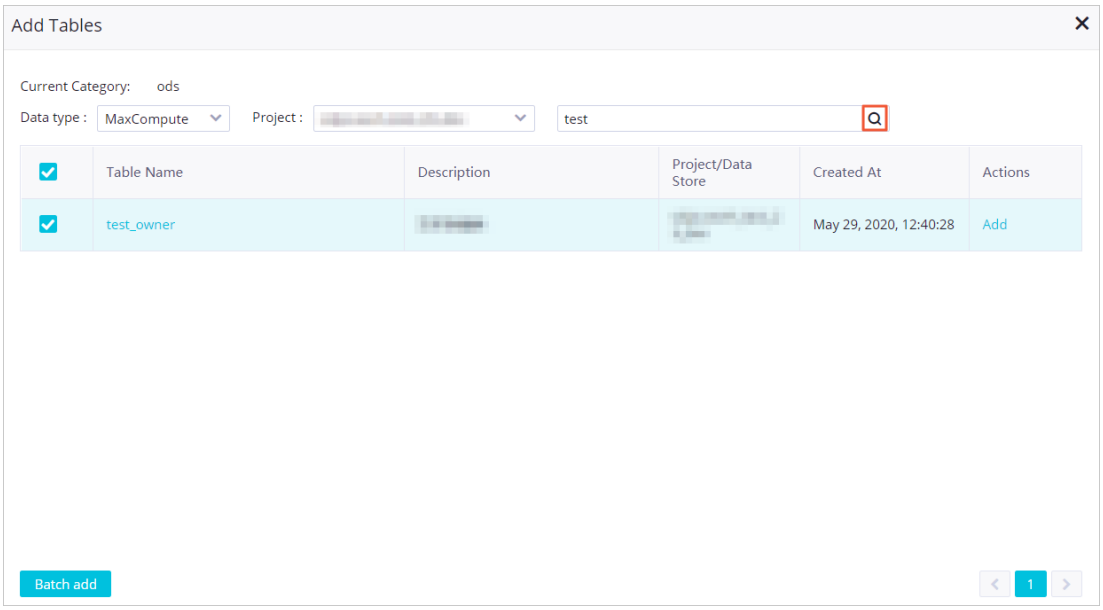
   Use the same method to create more categories. DataWorks allows you to create a maximum of four levels of categories. You can click the icon to edit a category or click the icon to delete a category.

3. Add tables to and remove tables from a category.

   - Add tables to a category
     a. Select the category and click **Add Tables** in the upper-right corner.

b. In the **Add Tables** dialog box, specify the table type and project, enter a table name or keyword, and then click the icon to search for tables.



c. If you want to add a table to the category, find the table and click **Add** in the Actions column.

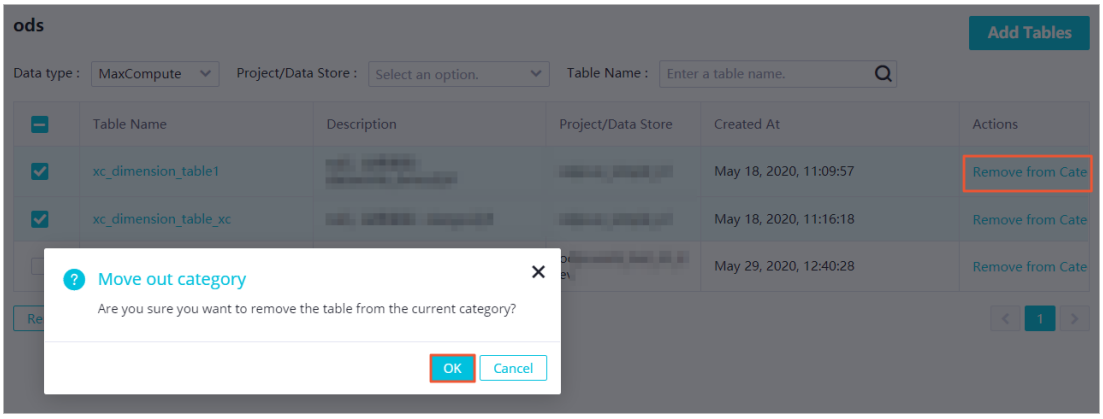
If you want to add multiple tables at a time, select the tables and click **Batch add**.

- Remove tables from a category

a. Select the category. If you want to remove a table from the category, find the table and click **Remove** in the Actions column.

If you want to remove multiple tables at a time, select the tables and click **Remove from Category**.

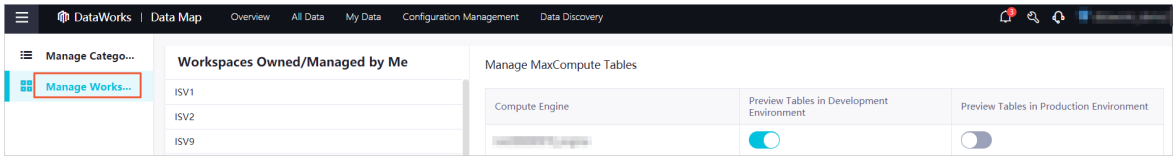
b. In the **Move out category** message, click **OK**.



Manage permissions on MaxCompute tables

On the **Manage Workspaces** page, you can specify whether MaxCompute tables can be previewed in a compute engine in the development and production environments.

1. In the left-side navigation pane, click **Manage Workspaces**.
2. In the **Workspaces Owned/Managed by Me** section, click the workspace for which you want to manage permissions on MaxCompute tables.
3. In the **Manage MaxCompute Tables** section, turn on or off the switch in the **Preview Tables in Development Environment** or **Preview Tables in Production Environment** column.



4. Turn on the switch in the **Preview Tables in Production Environment** column. In the **Attention** message, click **I already know that I am sure to open it**.

> **Note**
> - If you use a workspace in basic mode, you can turn on or off only the switch in the **Preview Tables in Production Environment** column.
> - After the switch in the **Preview Tables in Production Environment** column is turned on, all members of the workspace can preview MaxCompute tables in the production environment without requesting permissions. This may cause the leak of sensitive data. Therefore, exercise caution before you turn on the switch.

2.1.12.6. Table details

2.1.12.6.1. 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. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
3. In the top navigation bar, click **All Data**.
4. On the All Data page, click a tab as required, such as MaxCompute.
5. On the tab that appears, click the name of the table that you want to view.

   On the details page that appears, you can view the basic information, business information, permission information, technical information, detailed information, 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 times that the table is read, the table is added to favorites, and the table is viewed. You can also view the number and code of the output nodes, MaxCompute project name, region where the current workspace resides, region to which the engine belongs, owner, creation time, time-to-live (TTL), storage capacity, description, and tags of the table. You can also check whether the table is a partitioned table.

You can perform the following operations in the **Basic Information** section:

- View the code of an output node of the table: Click **View Code** next to **Output Node**. On the **Operation Center** page, view the node code.
- View the details of a MaxCompute project: Click the MaxCompute project name. On the page that appears, view the details of the MaxCompute project to which the table belongs.
- Edit the description of the table: Click the icon next to **Description**, edit the description, and then click the icon.

- Add a tag to the table: Click the + icon next to **Tags**, enter a tag name, and then press **Enter**.

  To remove a tag from the table, move the pointer over the tag and click the x icon.

View business information

In the **Business Information** section, you can view the DataWorks workspace name, environment type, category, and display name of the table.

You can perform the following operations in the **Business Information** section:

- View the details about the workspace: Click the DataWorks workspace name. On the page that appears, view the details about the DataWorks workspace to which the table belongs.
- Edit the display name of the table: Click the icon next to **Display Name**, edit the display name, and then click the icon.

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 **View More** in the upper-right corner of the **Permission Information** section. Then, the Permission application tab of the Data access control page appears.
2. On the **Permission application** tab, specify **User**, **Application duration**, and **Reason for application** in the Application Information section.

   > **Note** If you do not specify **Application duration**, the permissions that you request will be permanently valid after your request is approved.

3. Click **Apply for permission**.

View technical information

In the **Technical Information** section, you can view the technical type, time when the data definition language (DDL) statement was last modified, time when data was last modified, time when data was last viewed, and compute engine information.

In the **Technical Information** section, you can click **View** next to **Compute Engine Information**. In the **Compute Engine Information** dialog box, you can view or copy the information about the compute engine.

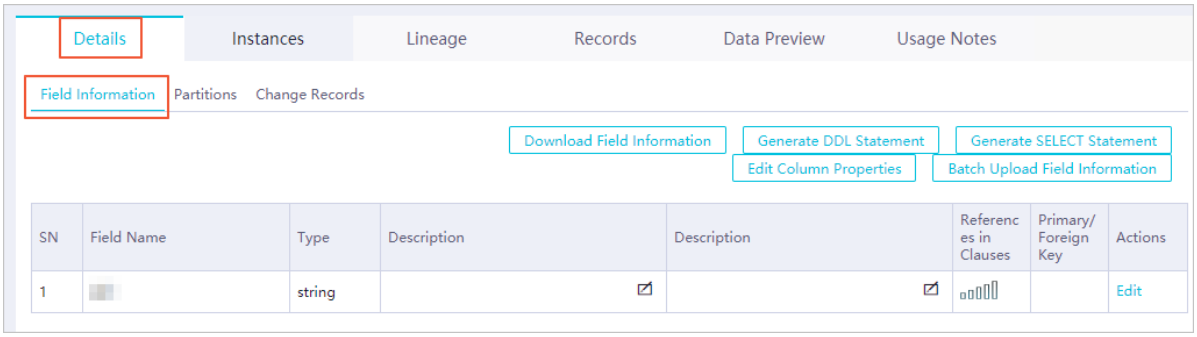
> **Note** By default, the time format yyyy-MM-dd HH:mm:ss is used to describe the compute engine.

View detailed information

The **Details** tab contains the following subtabs: **Field Information**, **Partitions**, and **Change Records**.

- **Field Information** subtab

On the **Field Information** subtab, 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 a foreign key.



Button	Description
Edit	Click this button, modify the field description and business description, determine whether to select the Primary Key check box for the field, and then click **Saved** or **Cancel**.
Upload	Click this button and drag the file that you want to upload from your on-premises machine to the **Batch Upload Field Information** dialog box.
Download	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 used to query data in the current table.

- **Partitions** subtab

  On the **Partitions** subtab, 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** subtab

  On the **Change Records** subtab, you can view the description, type, granularity, time, and operator of changes performed on the current table.

  On this subtab, 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**, **Create Partition**, **Delete Partition**, **Change Owner**, and **Change TTL**.

View output information

If the table data periodically changes with the related node, you can view the change status and data that is continuously updated on the **Instances** tab.

On this tab, you can also click **View Code** or **View** 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 subtabs: **Table Lineage**, **Field Lineage**, and **Impact Analysis**.

- The **Table Lineage** subtab consists of the **Graph Analysis** and **View by Level** parts.

- **Graph Analysis**: displays the ancestor and descendant tables of a specific number of levels for the current table and the number of ancestor and descendant tables for each table.
- **View by Level**: displays the ancestor and descendant tables of a single level for the current table by default. You can search for the ancestor and descendant tables based on the globally unique identifier (GUID).

- On the **Field Lineage** subtab, you can select a field from the **Field Name** drop-down list to view the lineage information of the field.
- On the **Impact Analysis** subtab, you can query the node that generates a lineage and the full link of the lineage based on information such as the lineage level, field, node type, table name, workspace name, and table owner.

  You can click **Manual update** to rerun the impact analysis. You can also download the impact analysis result or send the impact analysis result to the owners of descendant tables of the current table by email.

View reference records

The **Records** tab contains the following subtabs: **Foreign Key References** and **Access Statistics**.

- **Foreign Key References** subtab: On this subtab, you can check the number of users who reference the current table.
- **Access Statistics** subtab: On this subtab, 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 **Usage Notes** tab, you can edit usage notes, check the historical versions of the usage notes, and view the business description of data.

2.1.12.6.2. Request permissions on tables

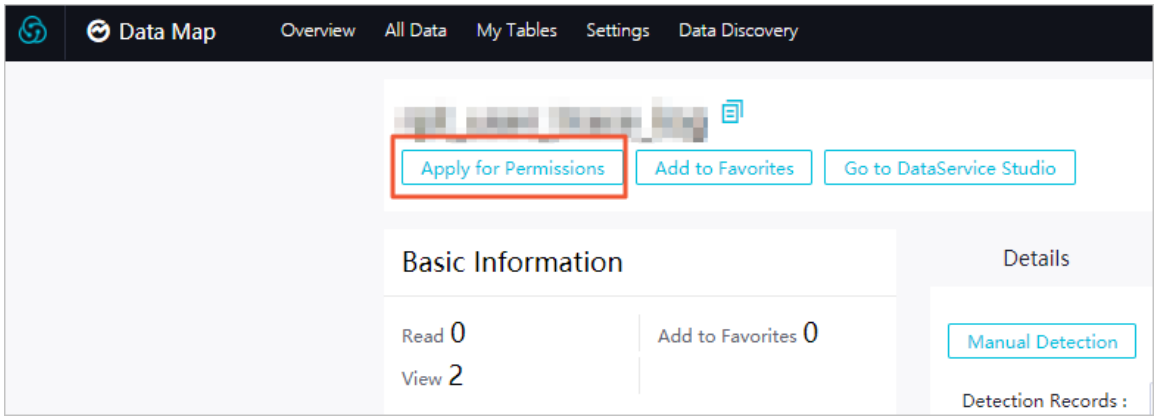
This topic describes how to request permissions on tables in Security Center or Data Map.

Go to the details page of a table

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
3. In the top navigation bar, click **All Data**.
4. On the All Data page, click a tab based on your business requirements, such as MaxCompute.
5. On the tab that appears, click the name of the table on which you want to request permissions.

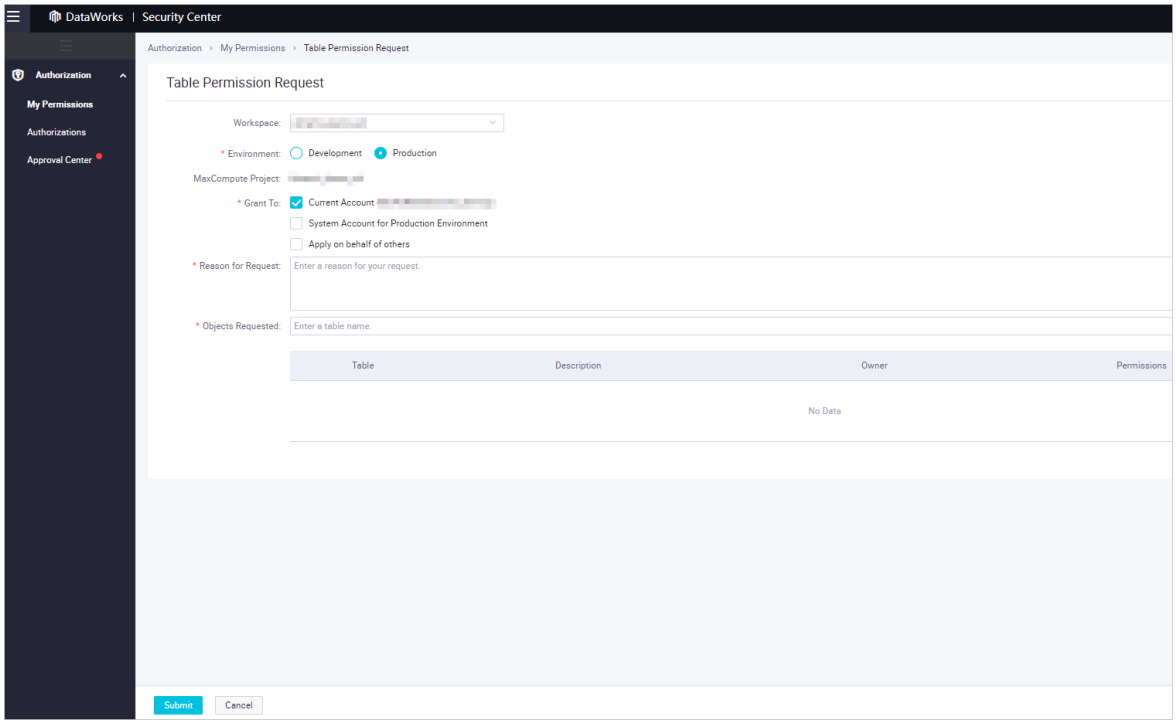
Request permissions on tables in Security Center

1. On the table details page, click **Apply for Permission**.



> **Note** If the table is hidden, the **Apply for Permission** button does not appear on the details page of the table.

2. On the **Table Permission Request** page, configure the parameters.



Parameter	Description
Workspace	The workspace to which the table belongs.
Environment	The environment to which the table belongs. For a workspace in standard mode, you can request permissions on the table in both the development environment and production environment. For a workspace in simple mode, you can request permissions on the table only in the production environment.
MaxCompute Project	The name of the MaxCompute project that is associated with the DataWorks workspace you selected. The value is automatically generated and cannot be changed.
Grant To	The account for which you request permissions on the table. Valid values: **Current Account** and **System Account for Production Environment**.

Parameter	Description
Valid Until	The validity period of the permissions on the table. Valid values: **1 Month**, **3 Months**, **6 Months**, **1 Year**, **Permanent**, and **Others**.
Reason for Request	The reason why you want to request permissions on the table. Enter a reason for faster approval.
Objects Requested	The name of the table.

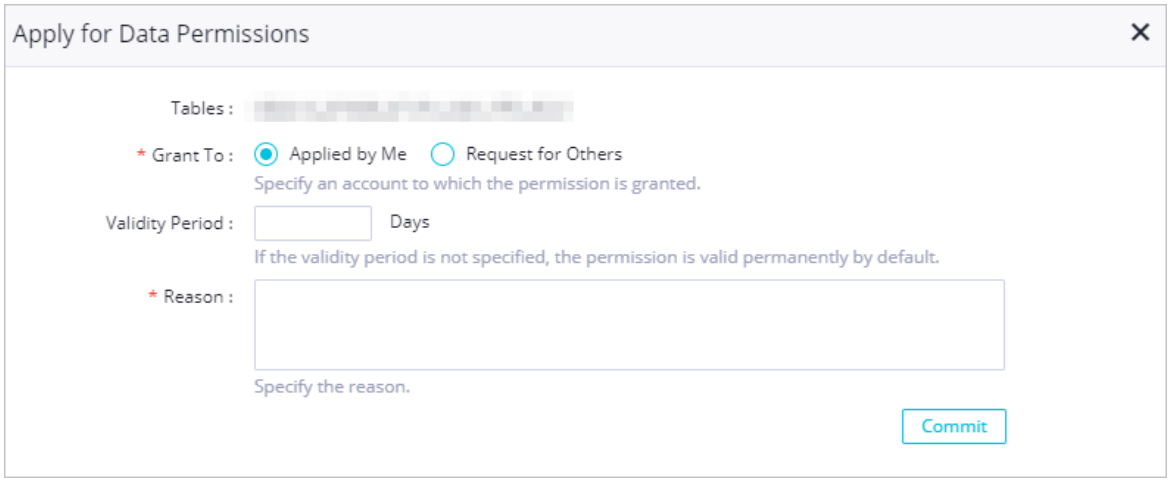
3. Click **Submit**.

Request permissions on tables in Data Map

1. On the table details page, click **Apply for Permission**.

> **Note** If the table is hidden, the **Apply for Permission** button does not appear on the details page of the table.

2. In the **Apply for Permission** dialog box, configure the parameters.



Parameter	Description
Table	The name of the table on which you want to request permissions. The value is automatically generated and cannot be changed.
Grant To	Specifies whether you want to request permissions on the table for the current account or another account. Valid values: **Current Account** and **Specified Account**.
Username	The username of the account for which you request permissions on the table. **Note** This parameter is available only when you set the **Grant To** parameter to **Specified Account**.
Validity Period	The validity period of the permissions on the table. If you do not set this parameter, the permissions are permanently valid.
Reason	The reason why you want to request permissions on the table. Enter a reason for faster approval.

3. Click **Submit**.

View the request status

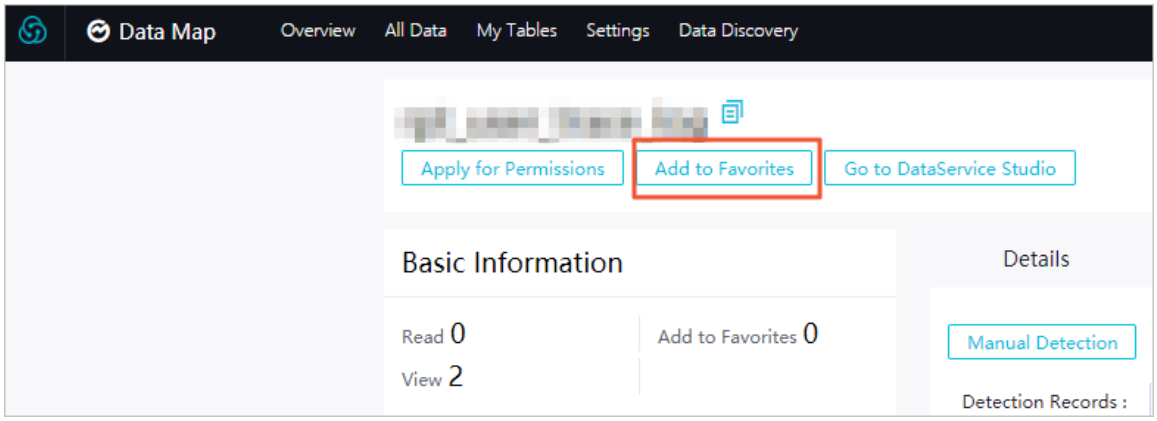
1. In the top navigation bar, click **My Data**.
2. On the My Data page, click **Permission Management** in the left-side navigation pane.
3. On the page that appears, click the **Submitted by Me** tab.
4. Find the required request record and click **View** in the Actions column to view the request status.

2.1.12.6.3. Add a table to favorites

This topic describes how to add a table to favorites, remove a table from favorites, and view the tables added to favorites.

Procedure

1. Go to the details page of a table.
   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 governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **All Data**.
   iv. On the **All Data** page, click a tab based on your business requirements, such as MaxCompute.
   v. On the tab that appears, click the name of the table that you want to add to favorites.
2. On the table details page, click **Add to Favorites**.



3. In the top navigation bar, click **My Data**.
4. On the **My Data** 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 that you want to remove and click **Remove from Favorites** in the Actions column.

2.1.12.6.4. Go to DataService Studio to create an API

This topic describes how to go to DataService Studio from the details page of a table to create an API.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data**

governance > DataMap(Data Management).

3. In the top navigation bar, click **All Data**.
4. On the **All Data** page, click a tab based on your business requirements, such as MaxCompute.
5. On the details page of the table, click **Create API in DataService Studio**.
6. On the **DataService Studio** page, create an API based on tables or register the existing API as required. For more information, see Overview.

2.1.12.7. Data discovery

2.1.12.7.1. Collect metadata from an EMR data source

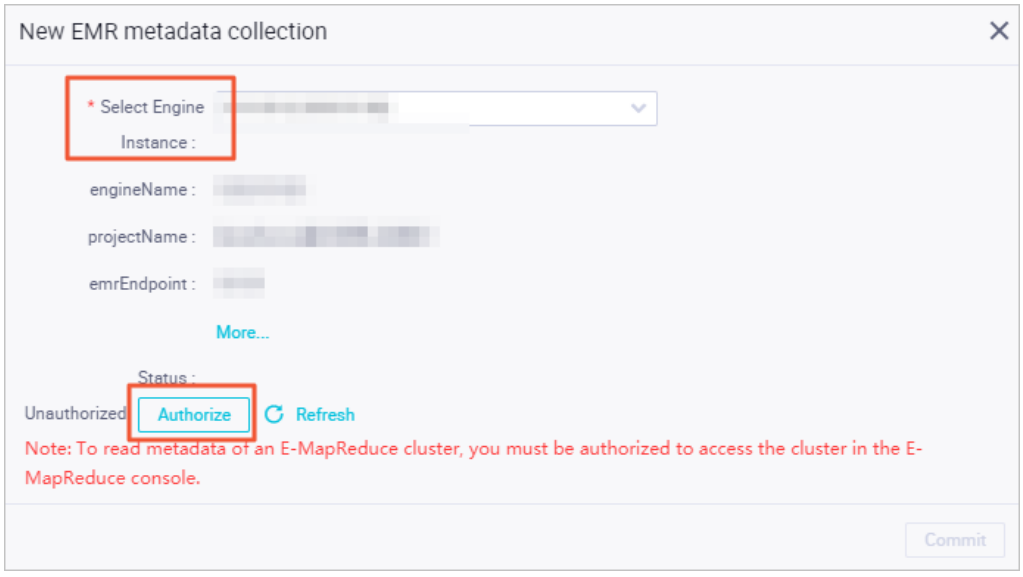
This topic describes how to create a crawler to collect metadata from an E-MapReduce (EMR) data source to DataWorks. You can view the collected metadata in Data Map.

Prerequisites

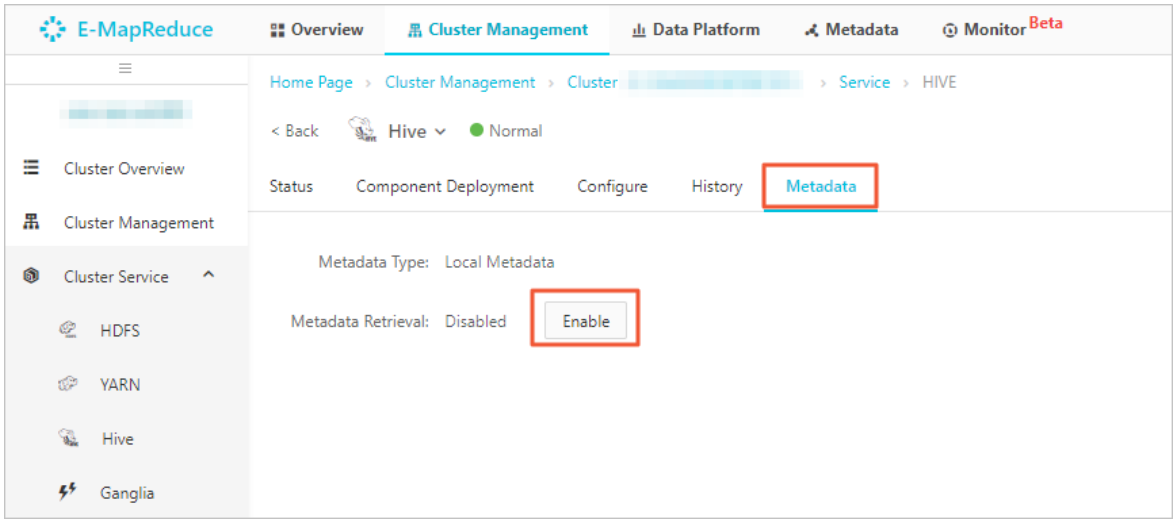
An EMR cluster is associated with your DataWorks workspace.

Procedure

1. Go to the **Data Discovery** 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 governance > DataMap(Data Management)**.
   iii. On the Data Map page, click **Data Discovery** in the top navigation bar.
2. On the **E-MapReduce Metadata Crawler** page, click **Create Crawler**.
3. In the **Create Crawler** dialog box, select the associated EMR cluster from the **Select a cluster** drop-down list and click **Authorize**.



4. On the page that appears, click the **Metadata** tab and click **Enable**.



5. In the **Confirm Operation** message, click **OK**.
6. Return to the **Create Crawler** dialog box on the **E-MapReduce Metadata Crawler** page and click **Refresh**.
7. After the authorization status changes to **Authorized**, click **Commit**.
8. On the **E-MapReduce Metadata Crawler** page, find the newly created crawler and click **Obtain All** in the Actions column.

   Click **Refresh** in the upper-right corner of the page and verify that the value in the **Running Status** column of the created crawler changes to **Collected**.

   > **Note** After full metadata from the EMR data source is collected, the system automatically synchronizes new metadata from the data source.

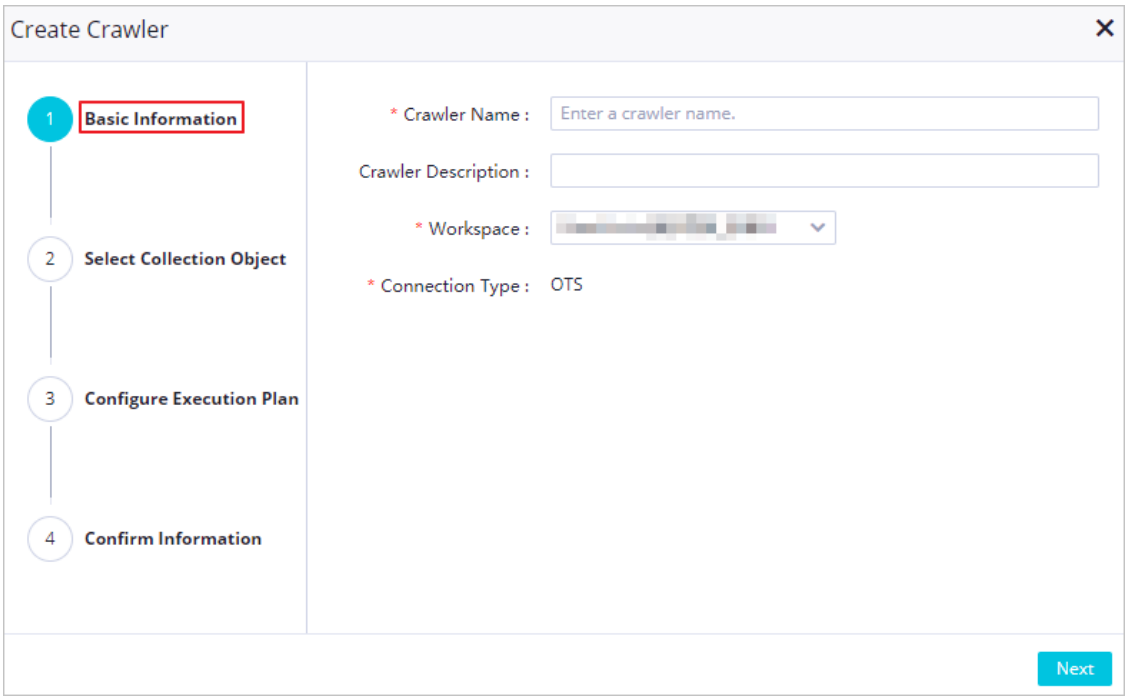
   If you want to delete the created crawler, click **Delete** in the Actions column. In the **Delete Instance** message, click **OK**.
9. View the metadata collected from the EMR data source.
   i. In the top navigation bar, click **All Data**.
   ii. Click the **E-MapReduce** tab.
   iii. On the **E-MapReduce** tab, click the name of the table that stores the collected metadata and view the table details.

2.1.12.7.2. Collect metadata from a Tablestore data source

You can collect information about the schema and lineage of a table to Data Map. This way, the inner structure and association relationships of the table can be clearly displayed. This topic describes how to create a crawler and collect metadata from a Tablestore data source to DataWorks. You can view the collected metadata in Data Map.

1. Go to the **Data Discovery** 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 governance > DataMap(Data Management)**.
   iii. On the Data Map page, click **Data Discovery** in the top navigation bar.
2. In the left-side navigation pane, click **OTS**.
3. On the **OTSMetadata Crawler** page, click **Create Crawler**.

4. In the **Create Crawler** dialog box, perform the following steps:
   i. In the **Basic Information** step, configure the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must specify a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of data source from which you want to collect metadata. The default value is **OTS** and cannot be changed.

   ii. Click **Next**.
   iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

   If no data source is available, click **Create** to go to the **Data Source** page and add a Tablestore data source. For more information, see Configure a Tablestore connection.
   iv. Click Start Testing next to **Test Crawler Connectivity**.
   v. If the message **The connectivity test is successful** appears, click **Next**.

   If the message **The connectivity test failed** appears, check whether you have configured a valid data source.
   vi. In the **Configure Execution Plan** step, specify **Execution Plan**.

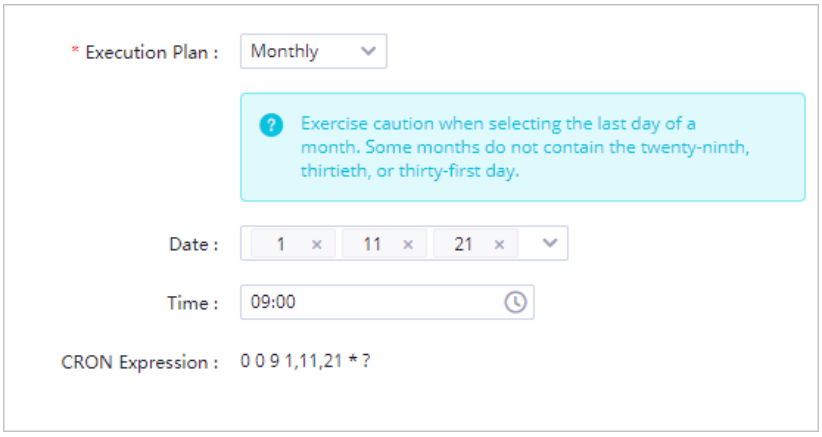
   Valid values of the **Execution Plan** parameter: **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**. The running plan that is generated varies based on the running cycle. The system collects metadata from the Tablestore data source based on the running cycle that you specify. The following descriptions explain each value and provide examples:

   - On-demand Execution: The system collects metadata from the Tablestore data source based on your business requirements.

- Monthly: The system automatically collects metadata from the Tablestore data source once at a specific time on several specific days of each month.

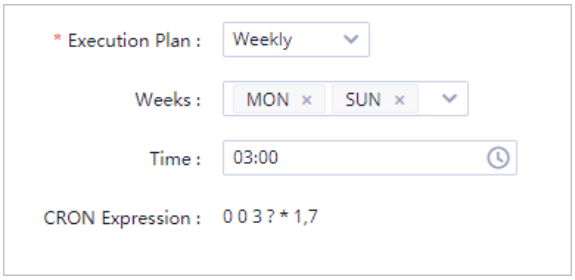
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In this case, the system does not collect metadata from the Tablestore data source on these dates. We recommend that you do not select the last day of a month.

  The following figure shows that the system automatically collects metadata from the Tablestore data source once at 09:00 on the 1st, 11th, and 21th days of each month. **Cron Expression** is automatically generated based on the values of Date and Time.

- Weekly: The system automatically collects metadata from the Tablestore data source once at a specific time on several specific days of each week.

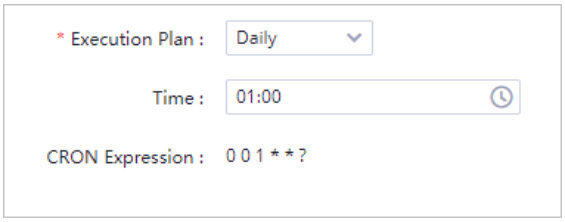
  The following figure shows that the system automatically collects metadata from the Tablestore data source once at 03:00 on Sunday and Monday of each week.

  If the **Time** parameter is not specified, the system automatically collects Tablestore metadata once at 00:00:00 on the specific days of each week.

- Daily: The system automatically collects metadata from the Tablestore data source once at a specific time of each day.

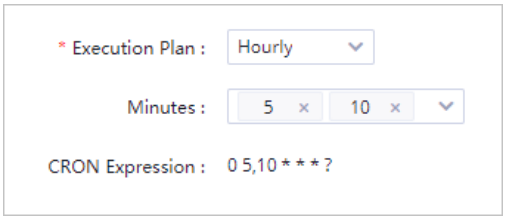
  The following figure shows that the system automatically collects metadata from the Tablestore data source once at 01:00 each day.

- Hourly: The system automatically collects metadata from the Tablestore data source once from the `N × 5` th minute of each hour.

  > **Note** For a Tablestore metadata collection task that runs each hour, you can configure the time to a multiple of 5 minutes.

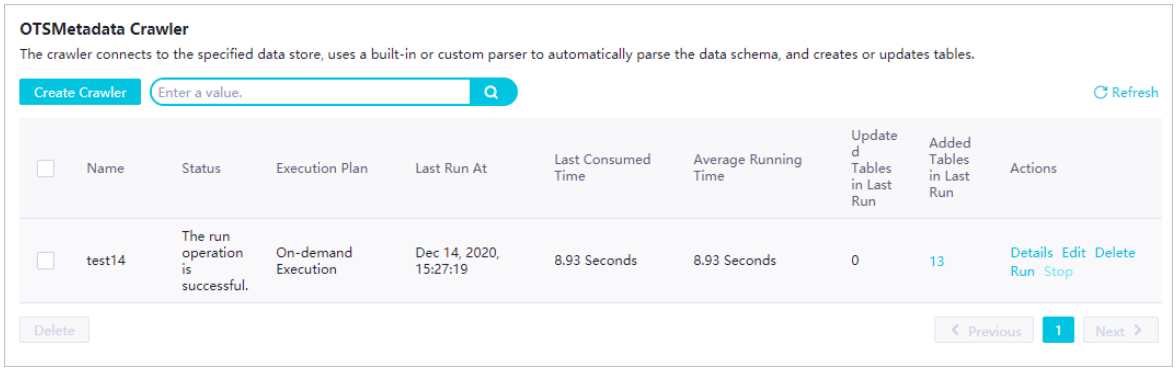
  The following figure shows that the system automatically collects metadata from the Tablestore data source from the 5th and 10th minutes of each hour.

vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

5. On the **OTSMetadata Crawler** page, you can view the information about your crawler and manage your crawler.



The following descriptions show the information that you can view and the operations that you can perform:

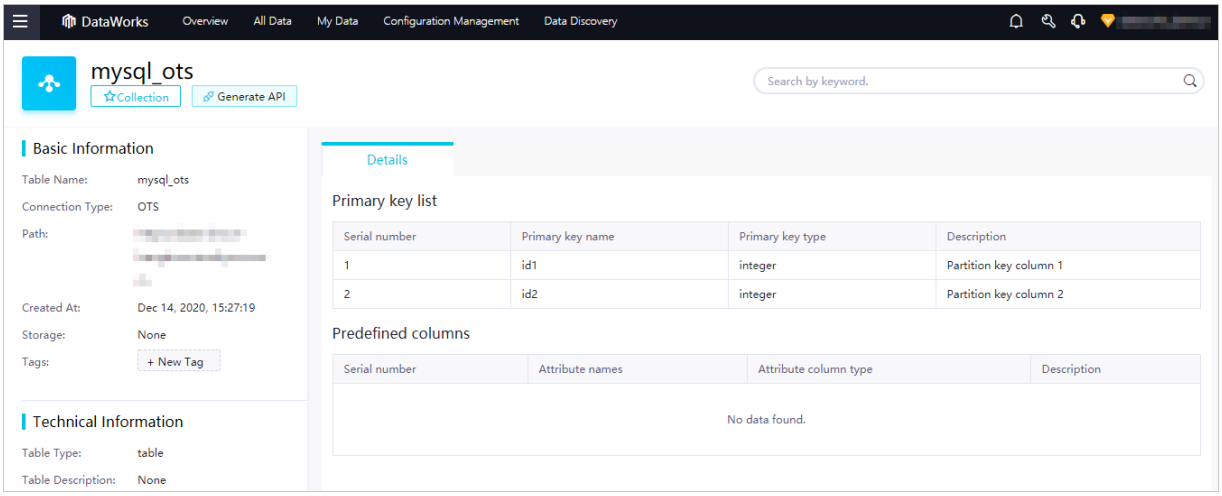
- You can view **Status**, **Execution Plan**, **Last Run At**, **Last Consumed Time**, **Average Running Time**, Updated Tables in Last Run, and Added Tables in Last Run of your crawler.
- You can click **Details**, **Edit**, **Delete**, **Run**, or **Stop** in the **Actions** column to perform the desired operation.
  - **Details**: View **Crawler Name**, **Data Source Type**, and **Execution Plan** configured for the crawler.
  - **Edit**: Modify the configurations of the crawler.
  - **Delete**: Delete the crawler.
  - **Run**: Run the task to collect metadata from the Tablestore data source. The **Run** entry point is available only when **Execution Plan** is set to **On-demand Execution**.
  - **Stop**: Stop running the crawler.

Result

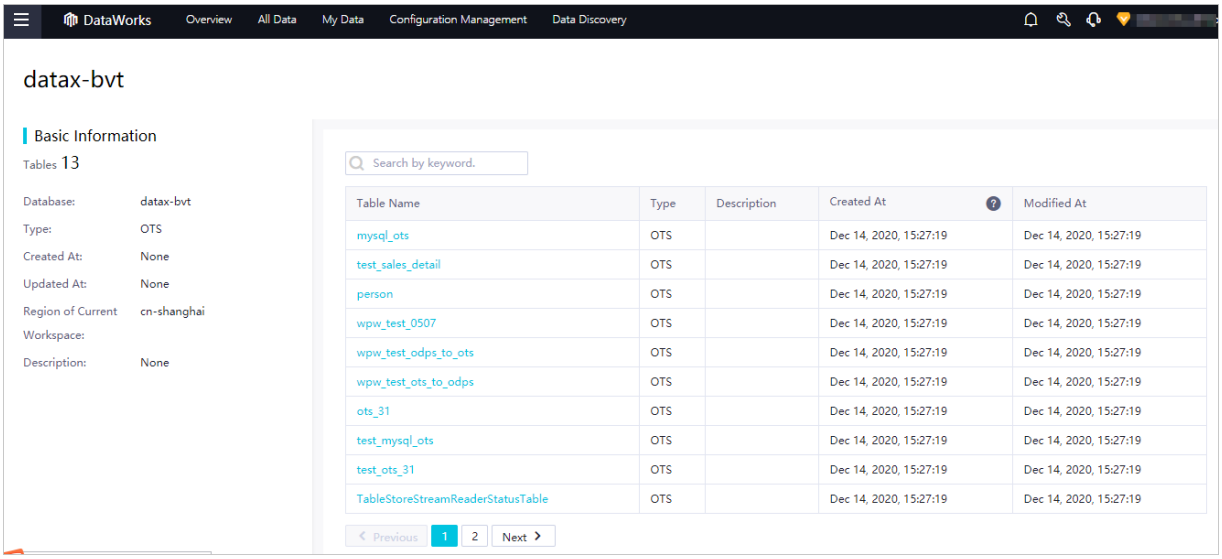
After the metadata in the Tablestore data source is collected, switch back to the previous page and click **All Data** in the top navigation bar. On the page that appears, click the **OTS** tab in the upper part. On the OTS tab, you can view the table that stores the collected Tablestore metadata.

Click the **table name**, **workspace**, or **database** to view the related details.

Example 1: View the details of the *mysql_ots* table.



Example 2: View all tables in the *datax-bvt* database.



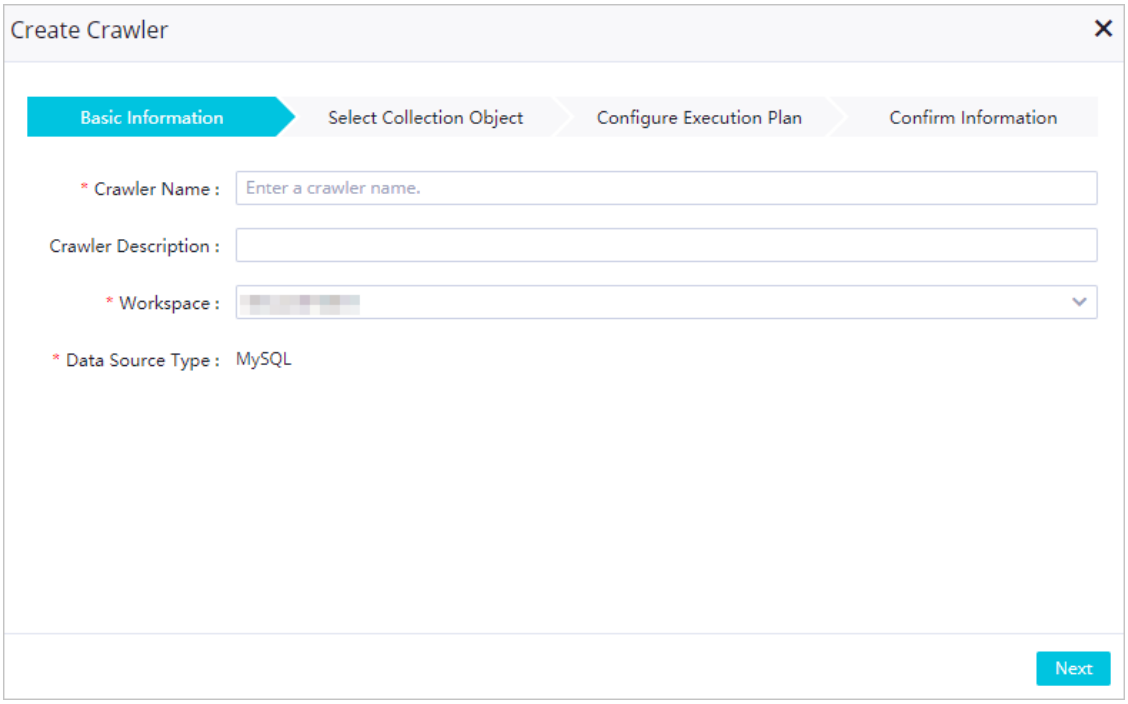
2.1.12.7.3. Collect metadata from a MySQL data source

This topic describes how to create a crawler to collect metadata from a MySQL data source to DataWorks. You can view the collected metadata on the Data Map page.

Procedure

1. Go to the **Data Discovery** 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 governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **MySQL**.
3. On the **MySQLMetadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.

i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **MySQL** and cannot be changed.

ii. Click **Next**.

iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

If no data sources are available, click **Create** to go to the **Data Source** page and add a MySQL data source. For more information, see Configure a MySQL connection.

> **Note** You can select an ApsaraDB RDS for MySQL instance or a MySQL data source that is accessible from the Internet by using a Java Database Connectivity (JDBC) connection string.

iv. Click Start Testing next to **Test Crawler Connectivity**.

v. If the message **The connectivity test is successful** appears, click **Next**.

vi. In the **Configure Execution Plan** step, configure an execution plan.

Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**.

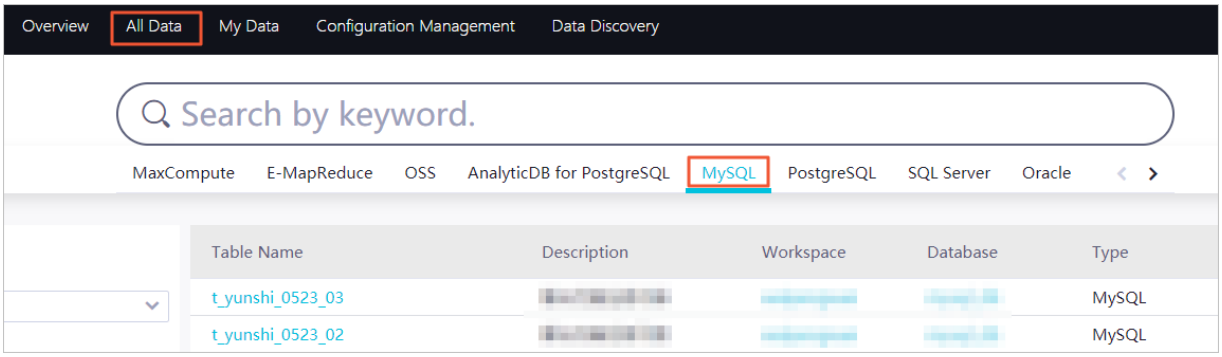
vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

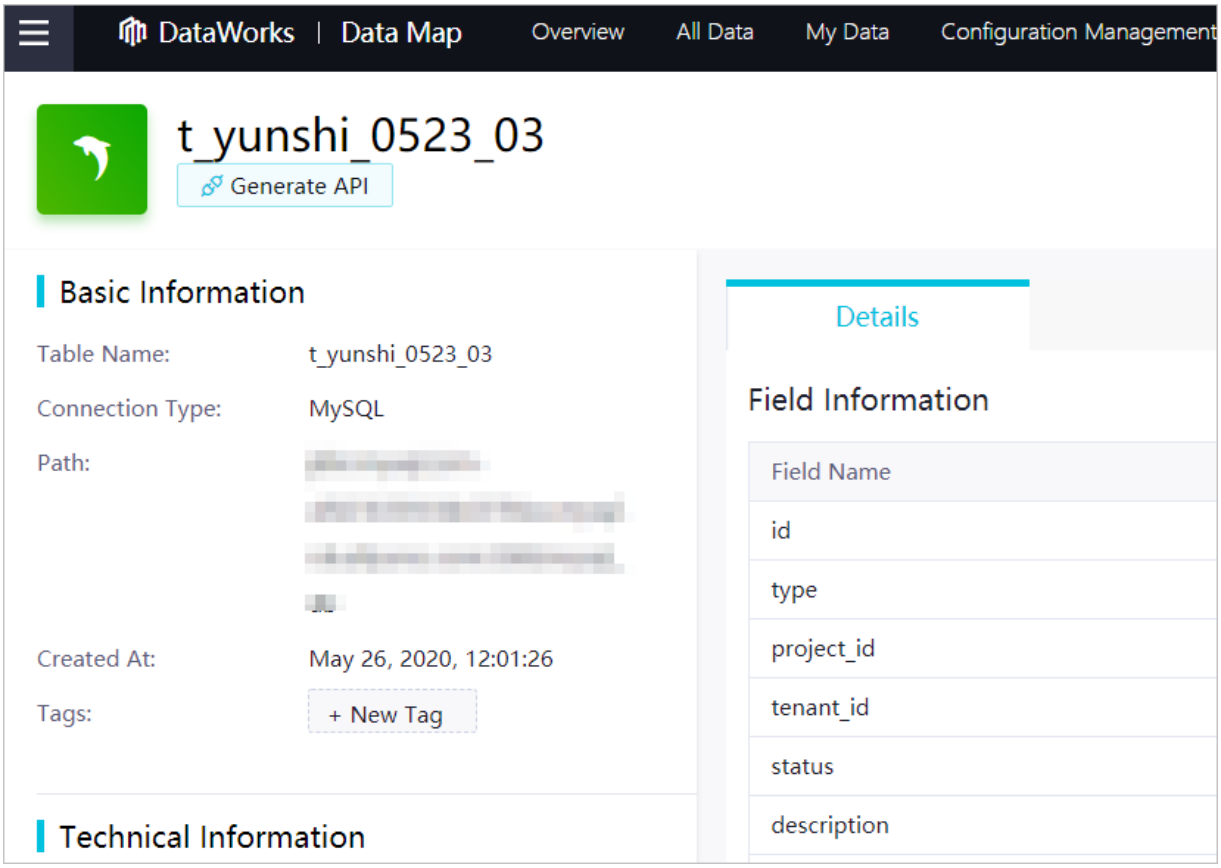
5. On the **MySQLMetadata Crawler** page, find the created crawler and click **Run** in the Actions column.

Result

After the metadata in the MySQL data source is collected, click **All Data** in the top navigation bar. Select **MySQL** from the options in the upper part of the page. You can view the tables that store the collected MySQL metadata.



Click the name of the table to view the table details.



2.1.12.7.4. Collect metadata from an SQL Server data source

This topic describes how to create a crawler to collect metadata from an SQL Server data source to DataWorks. You can view the collected metadata on the Data Map page.

Procedure

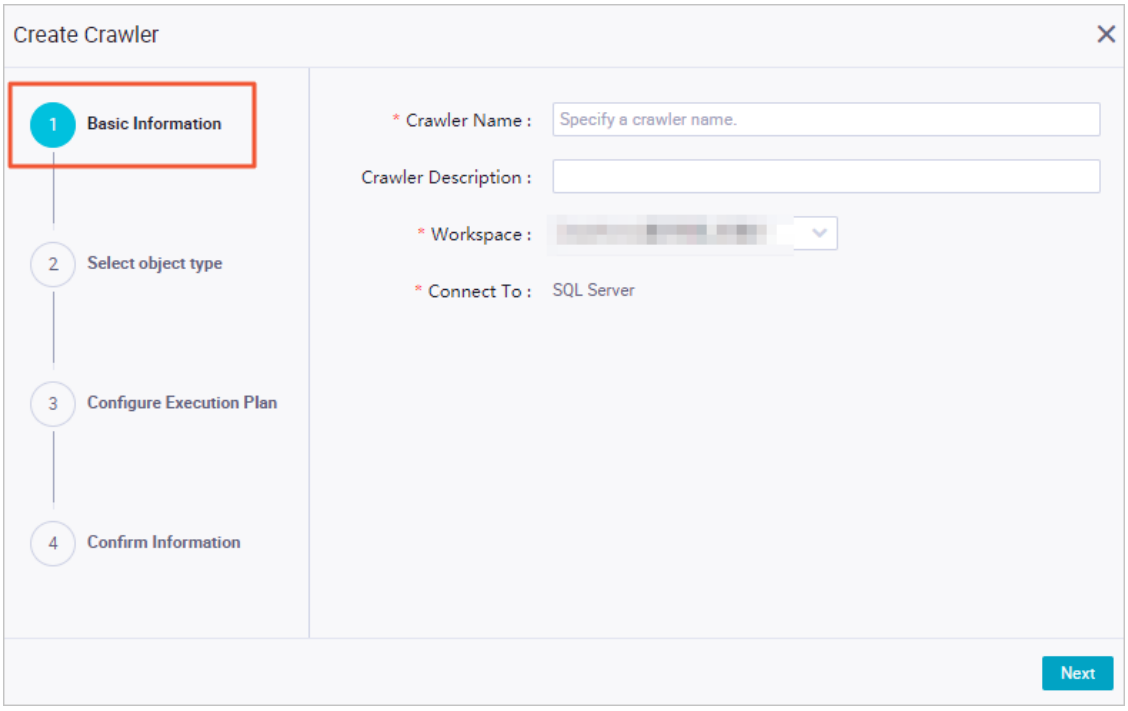
1. Go to the **Data Discovery** page.
   i. Log on to the DataWorks console.

ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.

iii. In the top navigation bar, click **Data Discovery**.

2. In the left-side navigation pane, click **SQL Server**.
3. On the **SQL ServerMetadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.

i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **SQL Server**.

ii. Click **Next**.

iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

If no data sources are available, click **Create** to go to the **Data Source** page and add an SQL Server data source. For more information, see Configure an SQL Server data source.

iv. Click Start Testing next to **Test Crawler Connectivity**.

v. If the message **The connectivity test is successful** appears, click **Next**.

If the message **The connectivity test of the data source failed, and the data source cannot be connected to the resource group** appears, check whether you have configured a valid data source.

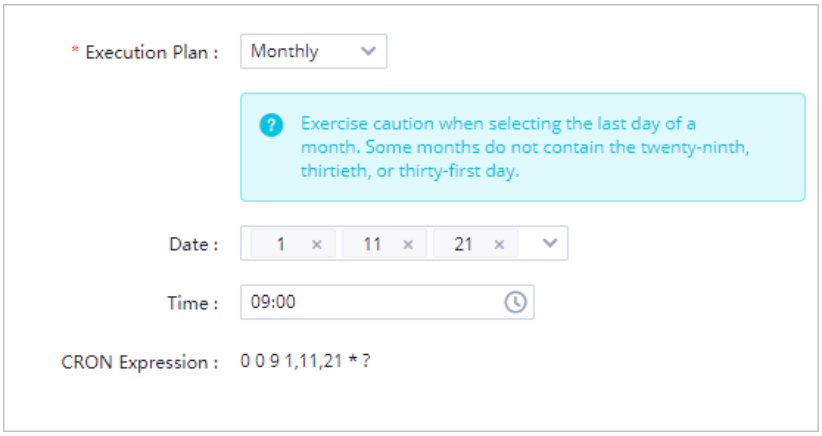
vi. In the **Configure Execution Plan** step, configure an execution plan.

Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the SQL Server data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: The system collects metadata from the SQL Server data source based on your business requirements.
- Monthly: The system automatically collects metadata from the SQL Server data source once at a specific time on several specific days of each month.

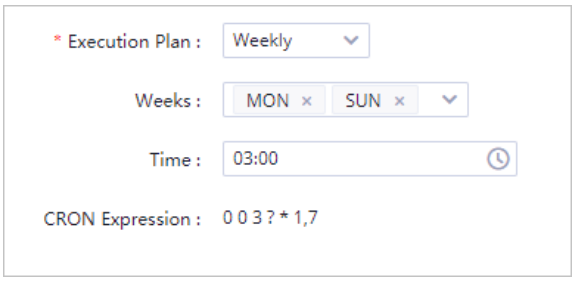
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the SQL Server data source on these dates. We recommend that you do not select the last days of a month.

  The following figure shows that the system automatically collects metadata from the SQL Server data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Weekly: The system automatically collects metadata from the SQL Server data source once at a specific time on several specific days of each week.

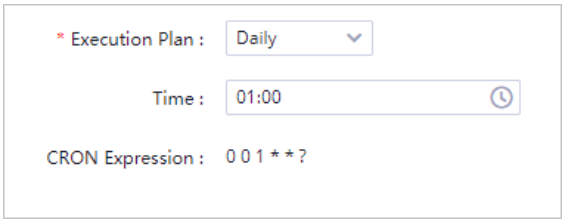
  The following figure shows that the system automatically collects metadata from the SQL Server data source once at 03:00 on Sunday and Monday of each week.

  If the **Time** parameter is not set, the system automatically collects metadata from the SQL Server data source once at 00:00:00 on the specific days of each week.

- Daily: The system automatically collects metadata from the SQL Server data source once at a specific time of each day.

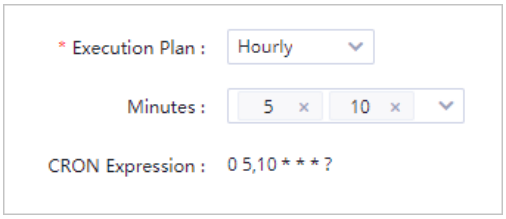
  The following figure shows that the system automatically collects metadata from the SQL Server data source once at 01:00 each day.

- Hourly: The system automatically collects metadata from the SQL Server data source once from the `N × 5` th minute of each hour.

  > **Note** For an SQL Server metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the SQL Server data source from the 5th and 10th minutes of each hour.

vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

5. On the **SQL ServerMetadata Crawler** page, find the created crawler and click **Run** in the Actions column.

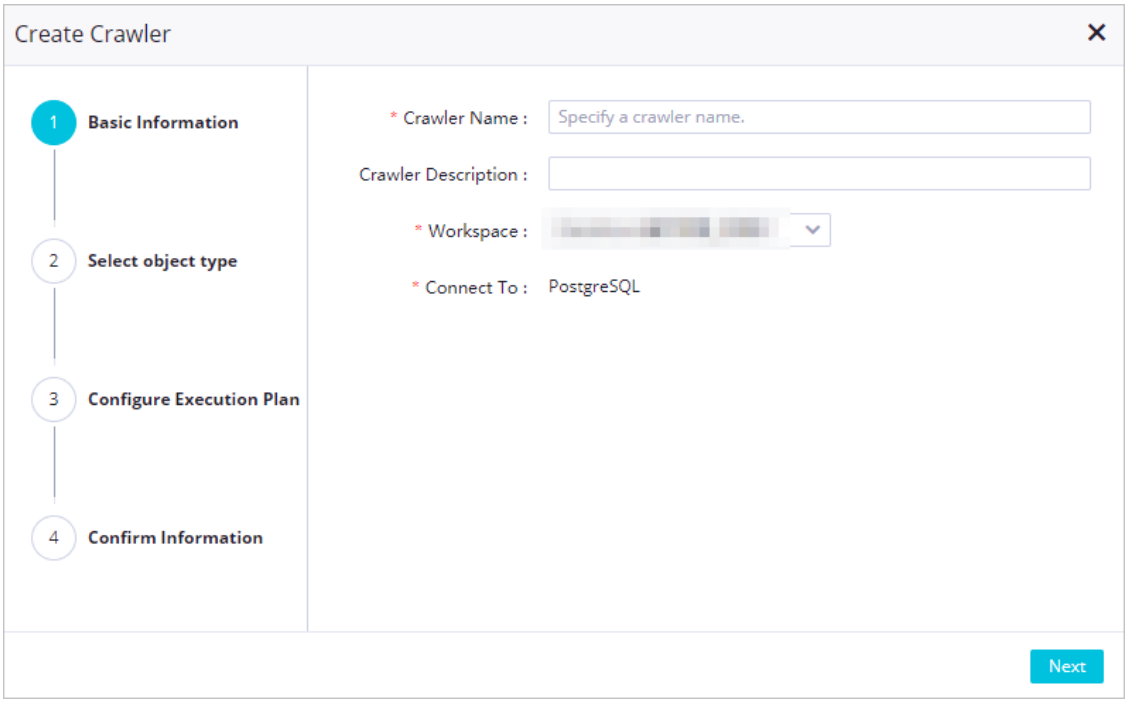
2.1.12.7.5. Collect metadata from a PostgreSQL data source

This topic describes how to create a crawler to collect metadata from a PostgreSQL data source to DataWorks. You can view the collected metadata on the Data Map page.

Procedure

1. Go to the **Data Discovery** 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 governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **PostgreSQL**.
3. On the **PostgreSQLMetadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.

i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **PostgreSQL** and cannot be changed.

ii. Click **Next**.

iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

If no data sources are available, click **Create** to go to the **Data Source** page and add a PostgreSQL data source. For more information, see Configure a PostgreSQL connection.

iv. Click Start Testing next to **Test Crawler Connectivity**.

v. If the message **The connectivity test is successful** appears, click **Next**.

If the message **The connectivity test of the data source failed, and the data source cannot be connected to the resource group** appears, check whether you have configured a valid data source.

vi. In the **Configure Execution Plan** step, configure an execution plan.

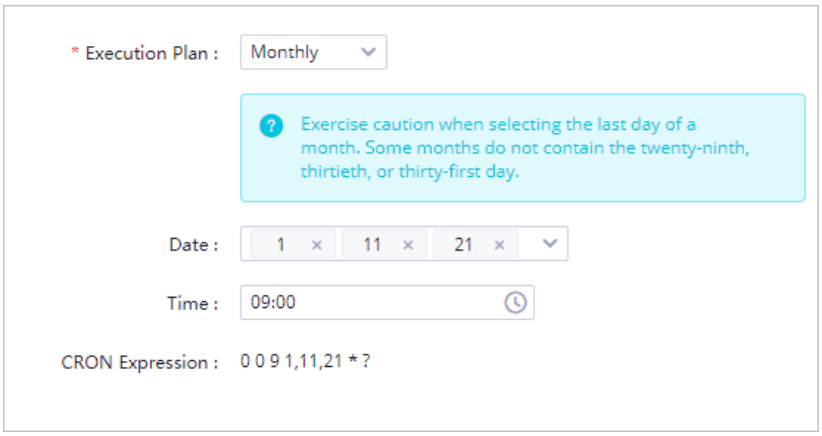
Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the PostgreSQL data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: The system collects metadata from the PostgreSQL data source based on your business requirements.

- Monthly: The system automatically collects metadata from the PostgreSQL data source once at a specific time on several specific days of each month.

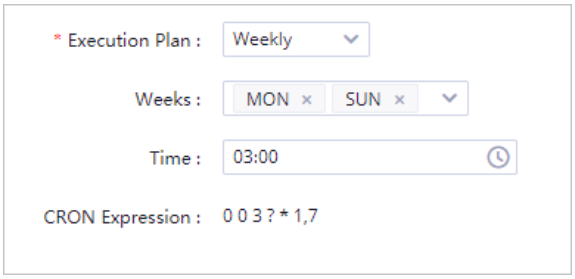
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the PostgreSQL data source on these dates. We recommend that you do not select the last days of a month.

  The following figure shows that the system automatically collects metadata from the PostgreSQL data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Weekly: The system automatically collects metadata from the PostgreSQL data source once at a specific time on several specific days of each week.

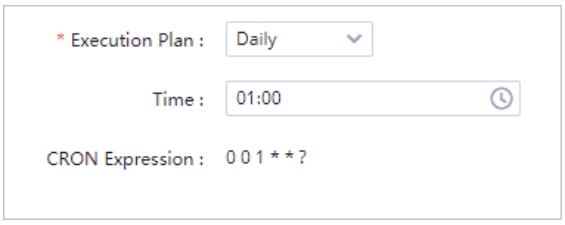
  The following figure shows that the system automatically collects metadata from the PostgreSQL data source once at 03:00 on Sunday and Monday of each week.

  If the **Time** parameter is not set, the system automatically collects metadata from the PostgreSQL data source once at 00:00:00 on the specific days of each week.

- Daily: The system automatically collects metadata from the PostgreSQL data source once at a specific time of each day.

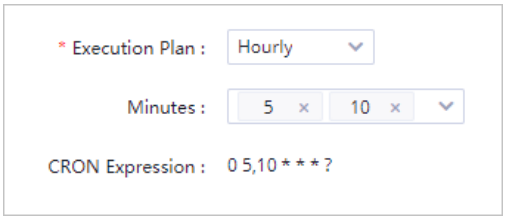
  The following figure shows that the system automatically collects metadata from the PostgreSQL data source once at 01:00 each day.

- Hourly: The system automatically collects metadata from the PostgreSQL data source once from the `N × 5` th minute of each hour.

  > **Note** For a PostgreSQL metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the PostgreSQL data source from the 5th and 10th minutes of each hour.

vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

5. On the **PostgreSQLMetadata Crawler** page, find the created crawler and click **Run** in the Actions column.

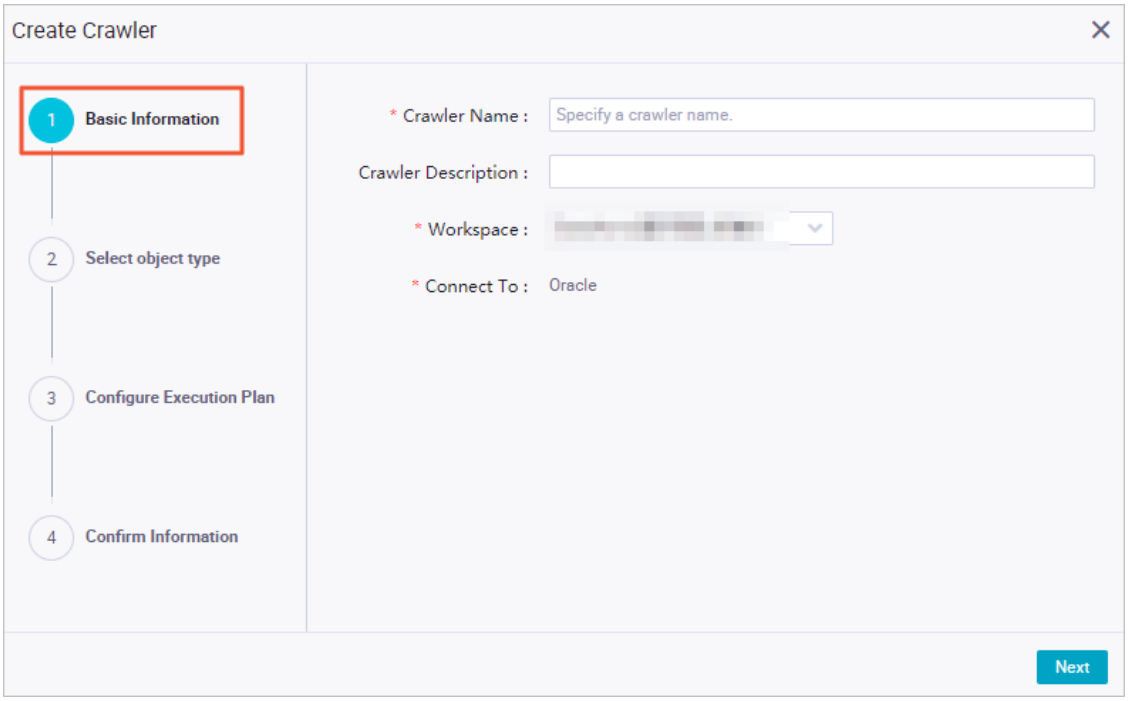
2.1.12.7.6. Collect metadata from an Oracle data source

This topic describes how to create a crawler to collect metadata from an Oracle data source to DataWorks. You can view the collected metadata on the Data Map page.

Procedure

1. Go to the **Data Discovery** 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 governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **Oracle**.
3. On the **OracleMetadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.

i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **Oracle** and cannot be changed.

ii. Click **Next**.

iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

If no data sources are available, click **Create** to go to the **Data Source** page and add an Oracle data source. For more information, see Configure an Oracle connection.

iv. Click Start Testing next to **Test Crawler Connectivity**.

v. If the message **The connectivity test is successful** appears, click **Next**.

If the message **The connectivity test of the data source failed, and the data source cannot be connected to the resource group** appears, check whether you have configured a valid data source.

vi. In the **Configure Execution Plan** step, configure an execution plan.

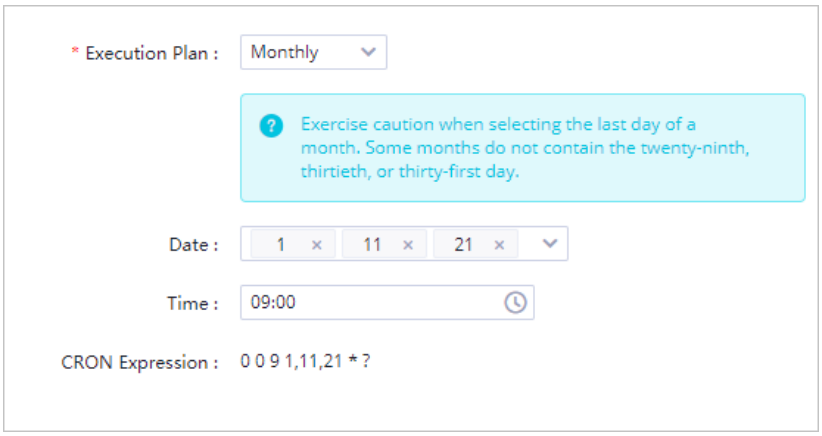
Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the Oracle data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: The system collects metadata from the Oracle data source based on your business requirements.

- Monthly: The system automatically collects metadata from the Oracle data source once at a specific time on several specific days of each month.

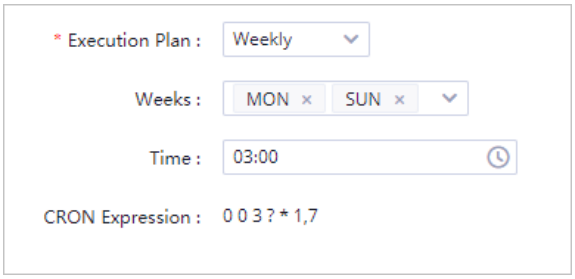
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the Oracle data source on these dates. We recommend that you do not select the last days of a month.

  The following figure shows that the system automatically collects metadata from the Oracle data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Weekly: The system automatically collects metadata from the Oracle data source once at a specific time on several specific days of each week.

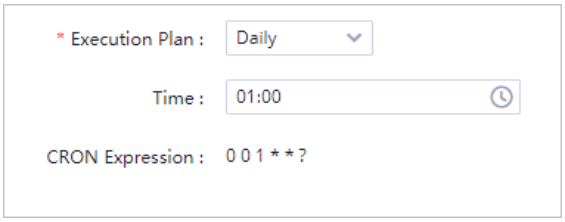
  The following figure shows that the system automatically collects metadata from the Oracle data source once at 03:00 on Sunday and Monday of each week.

  If the **Time** parameter is not specified, the system automatically collects Oracle metadata once at 00:00:00 on the specific days of each week.

- Daily: The system automatically collects metadata from the Oracle data source once at a specific time of each day.

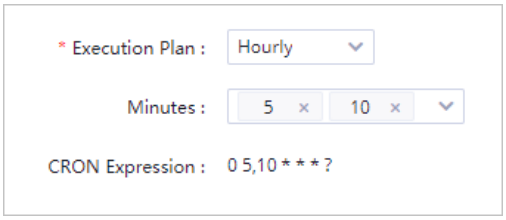
  The following figure shows that the system automatically collects metadata from the Oracle data source once at 01:00 each day.

- Hourly: The system automatically collects metadata from the Oracle data source once from the `N × 5` th minute of each hour.

  > **Note** For an Oracle metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the Oracle data source from the 5th and 10th minutes of each hour.

vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

5. On the **OracleMetadata Crawler** page, find the created crawler and click **Run** in the Actions column.

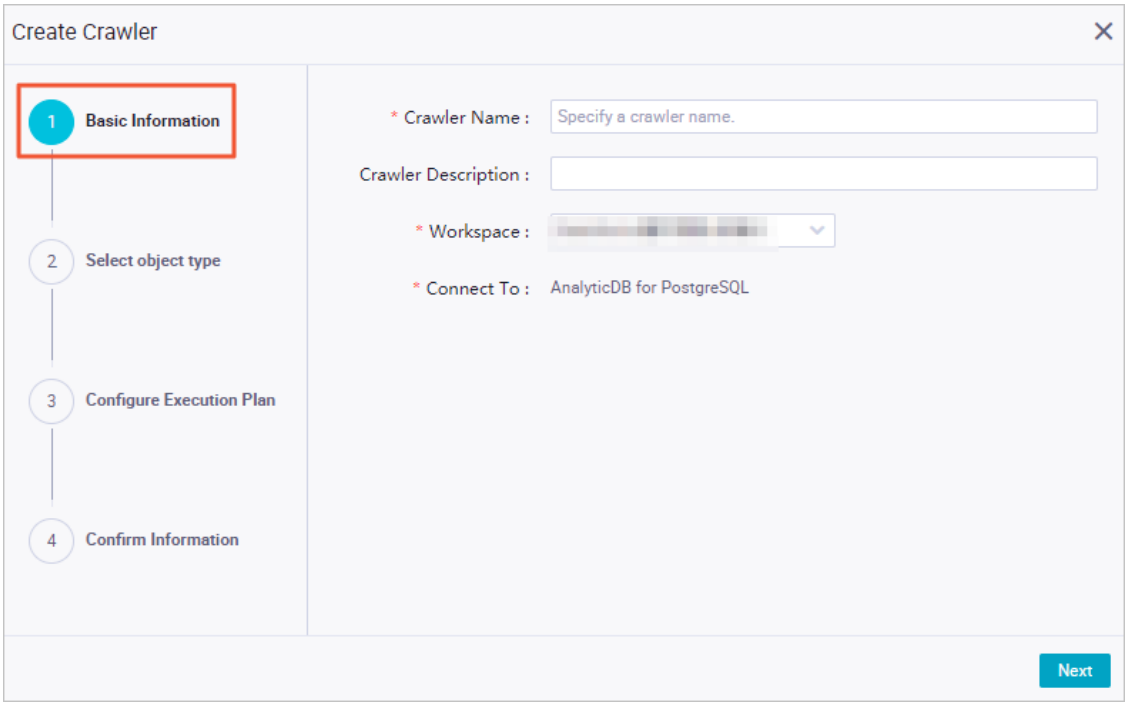
2.1.12.7.7. Collect metadata from an AnalyticDB for PostgreSQL data source

This topic describes how to create a crawler to collect metadata from an AnalyticDB for PostgreSQL data source to DataWorks. You can view the collected metadata in Data Map.

Procedure

1. Go to the **Data Discovery** 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 governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **AnalyticDB for PostgreSQL**.
3. On the **AnalyticDB for PostgreSQLMetadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.

i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **AnalyticDB for PostgreSQL** and cannot be changed.

ii. Click **Next**.

iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

If no data sources are available, click **Create** to go to the **Data Source** page and add an AnalyticDB for PostgreSQL data source.

iv. Click Start Testing next to **Test Crawler Connectivity**.

v. If the message **The connectivity test is successful** appears, click **Next**.

If the message **The connectivity test of the data source failed, and the data source cannot be connected to the resource group** appears, check whether you have configured a valid data source.

5. On the **AnalyticDB for PostgreSQLMetadata Crawler** page, find the created crawler and click **Run** in the Actions column.

After the crawler is run, click the number in the **Tables found during Last Run** column to view the details of the updated or created tables.

> **Notice** The **Run** button is available only in the Actions column of a crawler that needs to be manually triggered.

You can also perform the following operations on the AnalyticDB for PostgreSQLMetadata Crawler page:

- Click **Details** in the Actions column that corresponds to a crawler. In the **Crawler Details** dialog box, view the detailed information about the crawler.
- Click **Edit** in the Actions column that corresponds to a crawler. In the **Edit Crawler** dialog box, modify the configurations of the crawler.
- Click **Delete** in the Actions column that corresponds to a crawler. In the **Confirm** message, click **Ok** to delete the crawler.
- Find a crawler that is running and click **Stop** in the **Actions** column to stop the crawler.

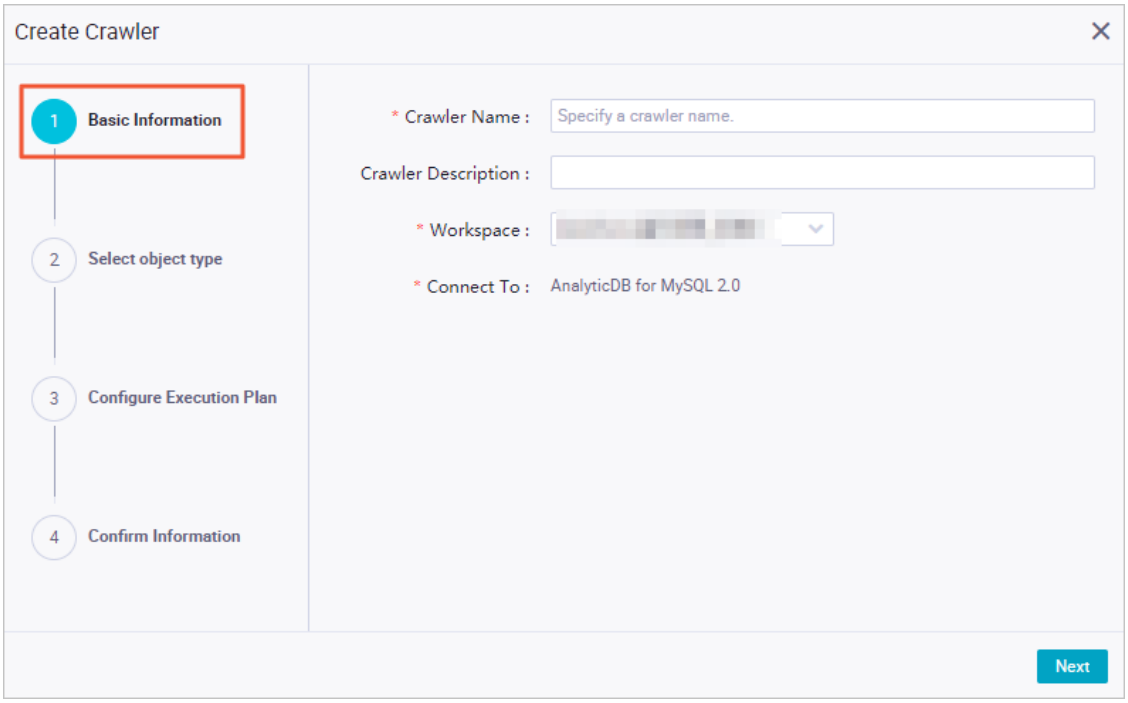
2.1.12.7.8. Collect metadata from an AnalyticDB for MySQL 2.0 data source

This topic describes how to create a crawler to collect metadata from an AnalyticDB for MySQL 2.0 data source to DataWorks. You can view the collected metadata on the Data Map page.

Procedure

1. Go to the **Data Discovery** 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 governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **AnalyticDB for MySQL 2.0**.
3. On the **AnalyticDB for MySQL 2.0Metadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.

i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **AnalyticDB for MySQL 2.0** and cannot be changed.

ii. Click **Next**.

iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

If no data sources are available, click **Create** to go to the **Data Source** page and add an AnalyticDB for MySQL V2.0 data source.

iv. Click Start Testing next to **Test Crawler Connectivity**.

v. If the message **The connectivity test is successful** appears, click **Next**.

If the message **The connectivity test of the data source failed, and the data source cannot be connected to the resource group** appears, check whether you have configured a valid data source.

vi. In the **Configure Execution Plan** step, configure an execution plan.

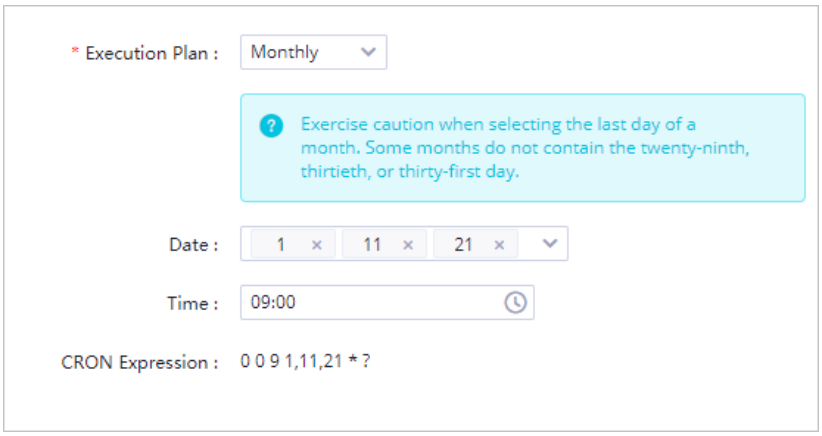
Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the AnalyticDB for MySQL V2.0 data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: The system collects metadata from the AnalyticDB for MySQL V2.0 data source based on your business requirements.

- Monthly: The system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source once at a specific time on several specific days of each month.

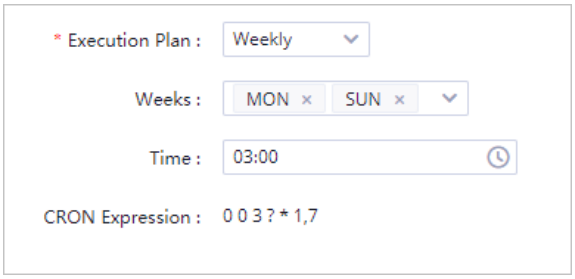
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the AnalyticDB for MySQL V2.0 data source on these dates. We recommend that you do not select the last days of a month.

  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Weekly: The system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source once at a specific time on several specific days of each week.

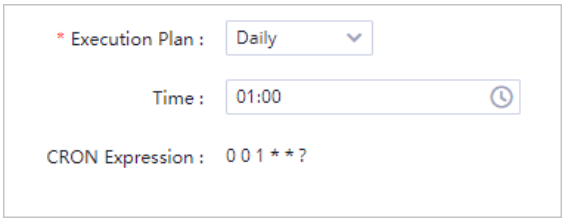
  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source once at 03:00 on Sunday and Monday of each week.

  If the **Time** parameter is not specified, the system automatically collects AnalyticDB for MySQL V2.0 metadata once at 00:00:00 on the specific days of each week.

- Daily: The system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source once at a specific time of each day.

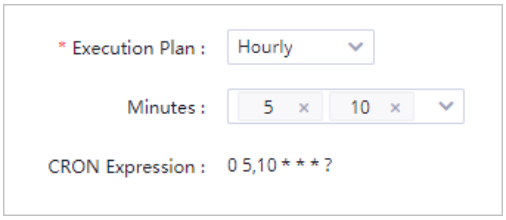
  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source once at 01:00 each day.

- Hourly: The system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source once from the `N × 5` th minute of each hour.

  > **Note** For an AnalyticDB for MySQL V2.0 metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source from the 5th and 10th minutes of each hour.

vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

5. On the **AnalyticDB for MySQL 2.0Metadata Crawler** page, find the created crawler and click **Run** in the Actions column.

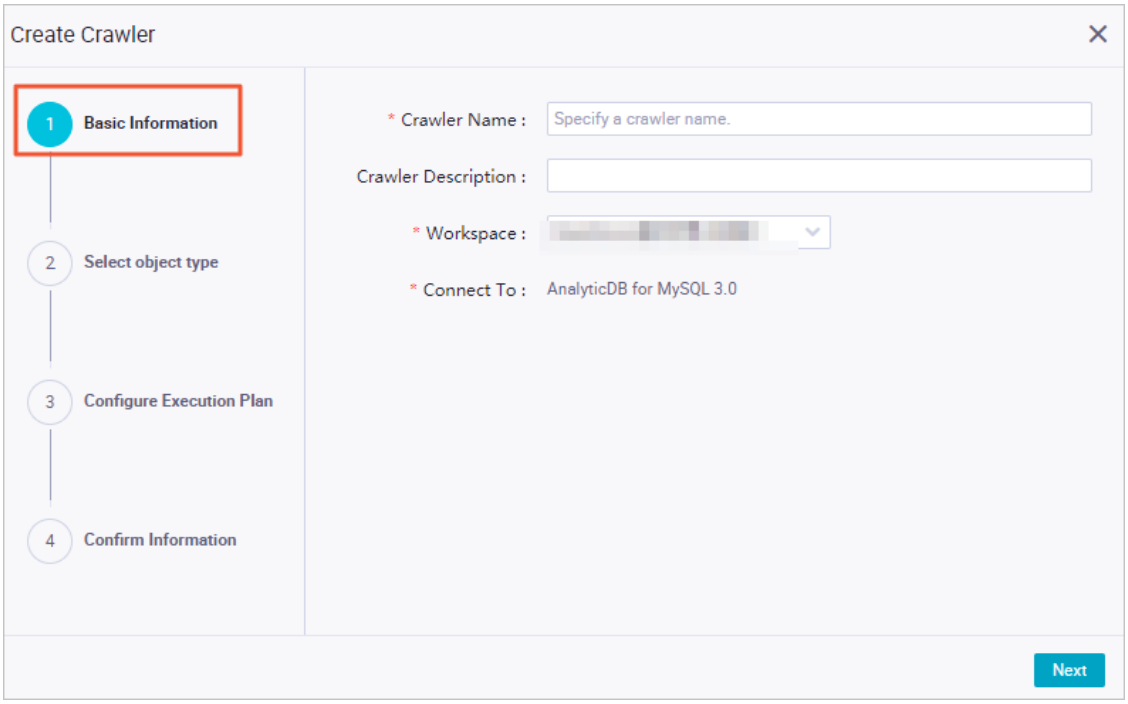
2.1.12.7.9. Collect metadata from an AnalyticDB for MySQL 3.0 data source

This topic describes how to create a crawler to collect metadata from an AnalyticDB for MySQL 3.0 data source to DataWorks. You can view the collected metadata on the Data Map page.

Procedure

1. Go to the **Data Discovery** 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 governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **AnalyticDB for MySQL 3.0**.
3. On the **AnalyticDB for MySQL 3.0Metadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.

i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **AnalyticDB for MySQL 3.0** and cannot be changed.

ii. Click **Next**.

iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

If no data sources are available, click **Create** to go to the **Data Source** page and add an AnalyticDB for MySQL V3.0 data source.

iv. Click Start Testing next to **Test Crawler Connectivity**.

v. If the message **The connectivity test is successful** appears, click **Next**.

If the message **The connectivity test of the data source failed, and the data source cannot be connected to the resource group** appears, check whether you have configured a valid data source.

vi. In the **Configure Execution Plan** step, configure an execution plan.

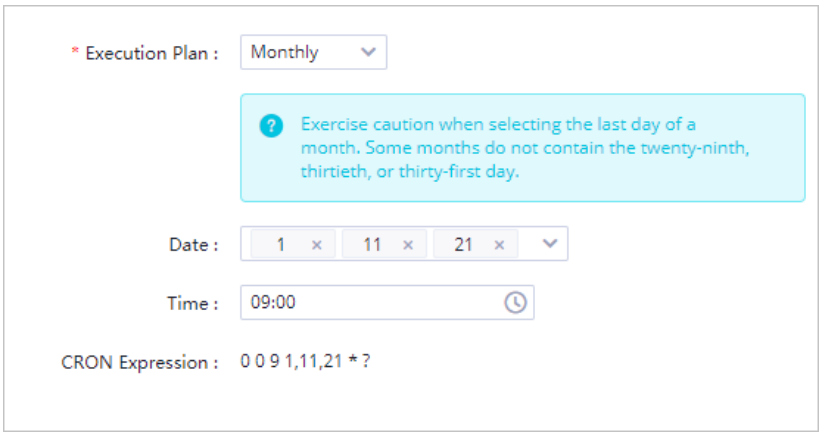
Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the AnalyticDB for MySQL 3.0 data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: The system collects metadata from the AnalyticDB for MySQL 3.0 data source based on your business requirements.

- Monthly: The system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source once at a specific time on several specific days of each month.

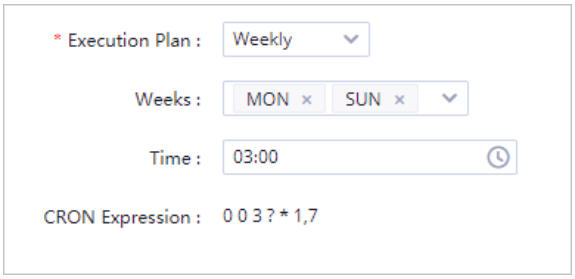
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the AnalyticDB for MySQL 3.0 data source on these dates. We recommend that you do not select the last days of a month.

  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Weekly: The system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source once at a specific time on several specific days of each week.

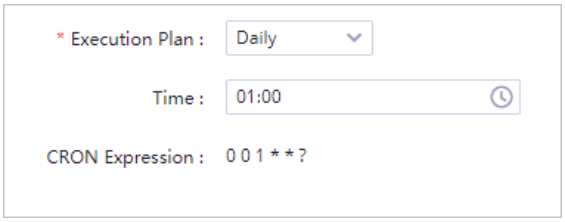
  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source once at 03:00 on Sunday and Monday of each week.

  If the **Time** parameter is not specified, the system automatically collects AnalyticDB for MySQL 3.0 metadata once at 00:00:00 on the specific days of each week.

- Daily: The system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source once at a specific time of each day.

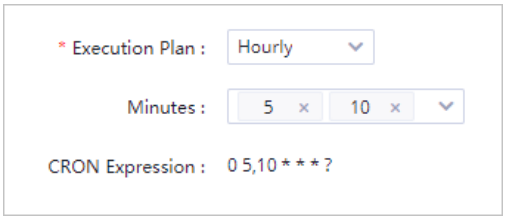
  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source once at 01:00 each day.

- Hourly: The system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source once from the `N × 5` th minute of each hour.

  > **Note** For an AnalyticDB for MySQL 3.0 metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source from the 5th and 10th minutes of each hour.

vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

5. On the **AnalyticDB for MySQL 3.0Metadata Crawler** page, find the created crawler and click **Run** in the Actions column.

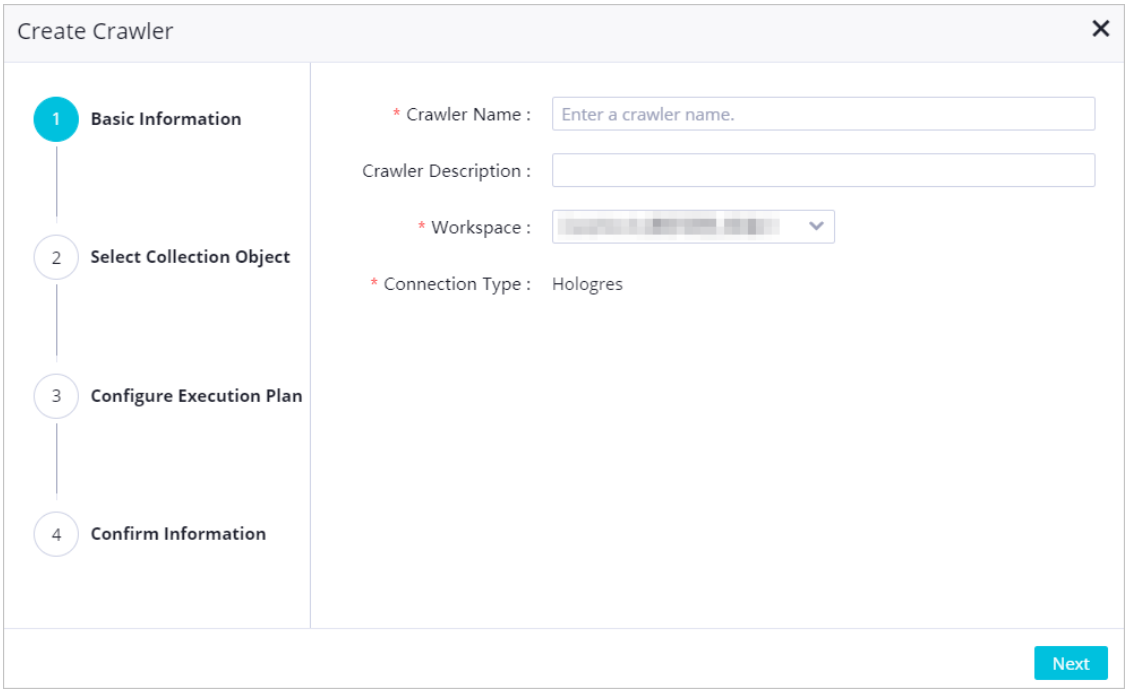
2.1.12.7.10. Collect metadata from a Hologres data source

This topic describes how to create a crawler to collect metadata from a Hologres data source to DataWorks. You can view the collected metadata on the Data Map page.

Procedure

1. Go to the **Data Discovery** 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 governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **Hologres**.
3. On the **HologresMetadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.

i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **Hologres** and cannot be changed.

ii. Click **Next**.

iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

iv. Click Start Testing next to **Test Crawler Connectivity**.

v. If the message **The connectivity test is successful** appears, click **Next**.

If the message **The connectivity test of the data source failed, and the data source cannot be connected to the resource group** appears, check whether you have configured a valid data source.

vi. In the **Configure Execution Plan** step, configure an execution plan.

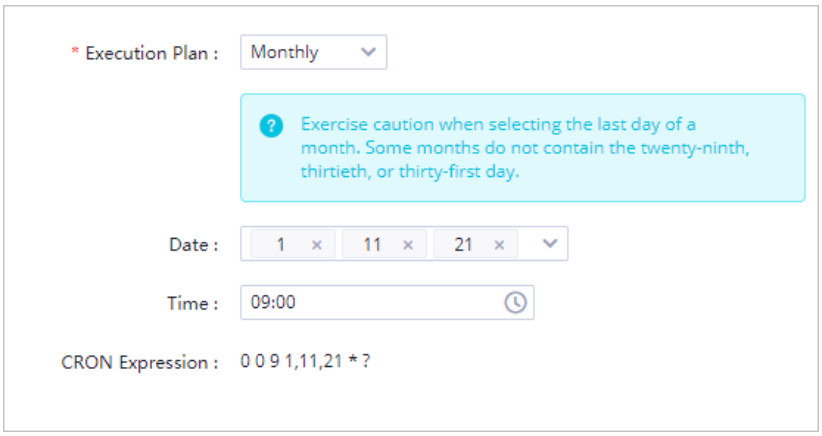
Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the Hologres data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: The system collects metadata from the Hologres data source based on your business requirements.

- Monthly: The system automatically collects metadata from the Hologres data source once at a specific time on several specific days of each month.

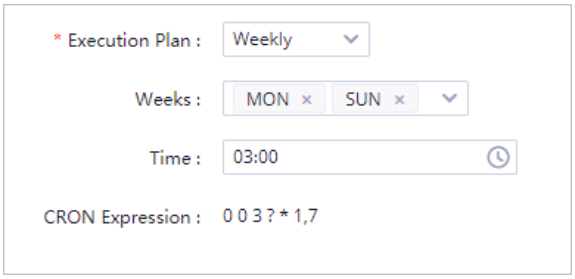
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the Hologres data source on these dates. We recommend that you do not select the last days of a month.

  The following figure shows that the system automatically collects metadata from the Hologres data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Weekly: The system automatically collects metadata from the Hologres data source once at a specific time on several specific days of each week.

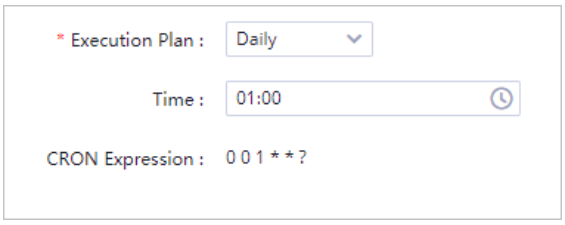
  The following figure shows that the system automatically collects metadata from the Hologres data source once at 03:00 on Sunday and Monday of each week.

  If the **Time** parameter is not specified, the system automatically collects Hologres metadata once at 00:00:00 on the specific days of each week.

- Daily: The system automatically collects metadata from the Hologres data source once at a specific time of each day.

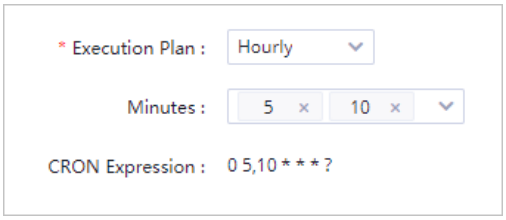
  The following figure shows that the system automatically collects metadata from the Hologres data source once at 01:00 each day.

- Hourly: The system automatically collects metadata from the Hologres data source once from the N × 5 th minute of each hour.

  > **Note** For a Hologres metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the Hologres data source from the 5th and 10th minutes of each hour.

vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

5. On the **HologresMetadata Crawler** page, find the created crawler and click **Run** in the Actions column.

2.1.12.7.11. Collect metadata from a CDH Hive data source

DataWorks allows you to collect metadata that includes the table schema and the linage information of tables in Data Map. You can view the schema of a table and the relationships between tables. This topic describes how to create a crawler to collect metadata from a CDH Hive data source to DataWorks. You can view the collected metadata on the Data Map page.

Prerequisites

A CDH cluster is associated with the current DataWorks workspace. For more information, see Associate a CDH cluster with a workspace.

Limits

You cannot collect metadata across regions. You must create a crawler in the region where the source metadata resides to collect the metadata.

Create a crawler

1. Go to the **Data Discovery** 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 governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. Create a crawler.
   i. In the left-side navigation pane, click **CDH Hive**.
   ii. On the **CDH Hive Metadata Crawler** page, click **Create Crawler**.
3. Configure the crawler.
   i. Select a CDH cluster.

   In the **Create Crawler** dialog box, select the cluster from which you want to collect metadata from the drop-down list.

   ii. Configure Execution Plan

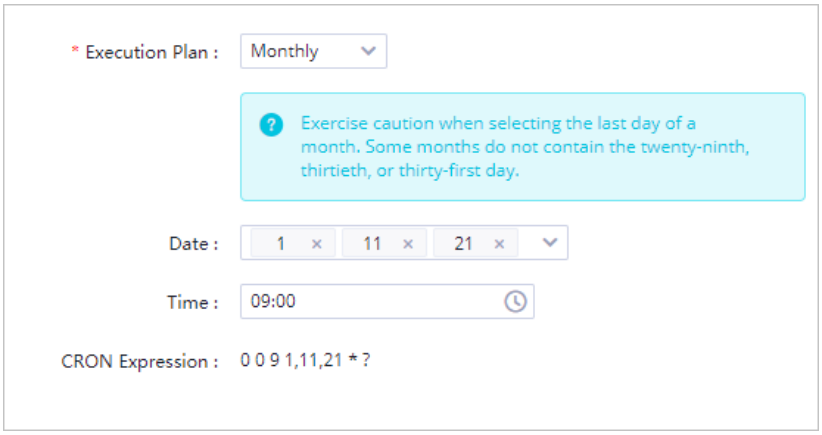
In the **Create Crawler** dialog box, specify a value for the **Execution Plan** parameter from the drop-down list.

Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, **Hourly**, and **Customize**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the CDH Hive data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: You need to manually run the crawler. The system collects metadata from the CDH Hive data source based on your business requirements.
- Monthly: The system automatically collects metadata from the CDH Hive data source once at a specific time on several specific days of each month.

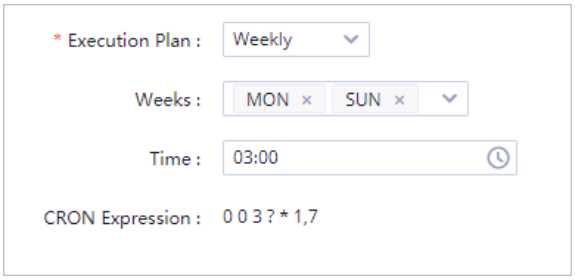
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the CDH Hive data source on these dates. We recommend that you do not select the last days of a month.

  The following figure shows that the system automatically collects metadata from the CDH Hive data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Weekly: The system automatically collects metadata from the CDH Hive data source once at a specific time on several specific days of each week.

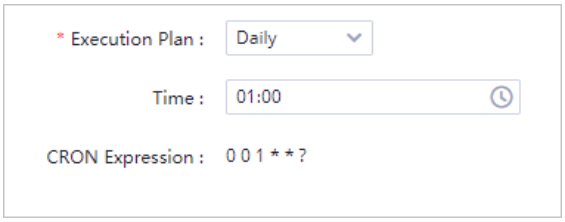
  The following figure shows that the system automatically collects metadata from the CDH Hive data source once at 03:00 on Sunday and Monday of each week. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

  If the **Time** parameter is not set, the system automatically collects metadata from the CDH Hive data source once at `00:00:00` on the specific days of each week.

- Daily: The system automatically collects metadata from the CDH Hive data source once at a specific time of each day.

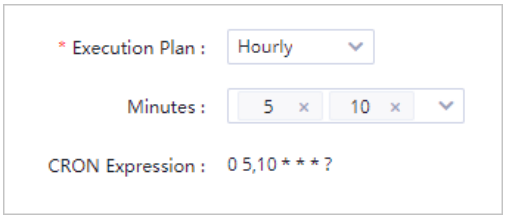
  The following figure shows that the system automatically collects metadata from the CDH Hive data source once at 01:00 each day. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Hourly: The system automatically collects metadata from the CDH Hive data source once from the `N × 5` th minute of each hour.

  > **Note** For a CDH Hive metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the CDH Hive data source from the 5th and 10th minutes of each hour. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Customize: You can enter a cron expression in the field of the Cron Expression parameter. The system automatically collects metadata based on the time configuration that matches the corn expression.

iii. Click **OK**.

Manage the crawler

On the **CDH Hive Metadata Crawler** page, you can view, edit, and delete the created crawler.

Area No.	Description
1	In this area, you can enter the name of a crawler to find the crawler. > **Note** The fuzzy match is supported. If you enter keywords in the search box, crawlers whose names contain the keywords are displayed.

Area No.	Description
2	In this area, you can view the details of a crawler in the **Status**, **Execution Plan**, **Last Run At**, **Last Consumed Time**, and **Average Running Time** columns. You can also perform the following operations on the crawler: • **Details**: View the CDH cluster and the execution plan that are configured for the crawler. • **Edit**: Modify the CDH cluster and the execution plan that are configured for the crawler. • **Delete**: Delete the crawler. • **Run**: Run the crawler to collect metadata based on the configurations. • **Stop**: Stop the crawler. **Note** **Run** and **Stop** are displayed in the **Actions** column only if the **Execution Plan** parameter is set to **On-demand Execution**.

What's next

After the metadata is collected, you can view the details of the collected metadata on the **All Data** page of Data Map.

2.1.12.7.12. Collect metadata from an HBase data source

DataWorks allows you to collect metadata that includes the table schema and the linage information of tables in Data Map. You can view the schema of a table and the relationships between tables. This topic describes how to create a crawler to collect metadata from an HBase data source to DataWorks. You can view the collected metadata on the Data Map page.

Prerequisites

A Cloudera Distribution Hadoop (CDH) cluster is associated with the current DataWorks workspace. For more information, see Associate a CDH cluster with a workspace.

Limits

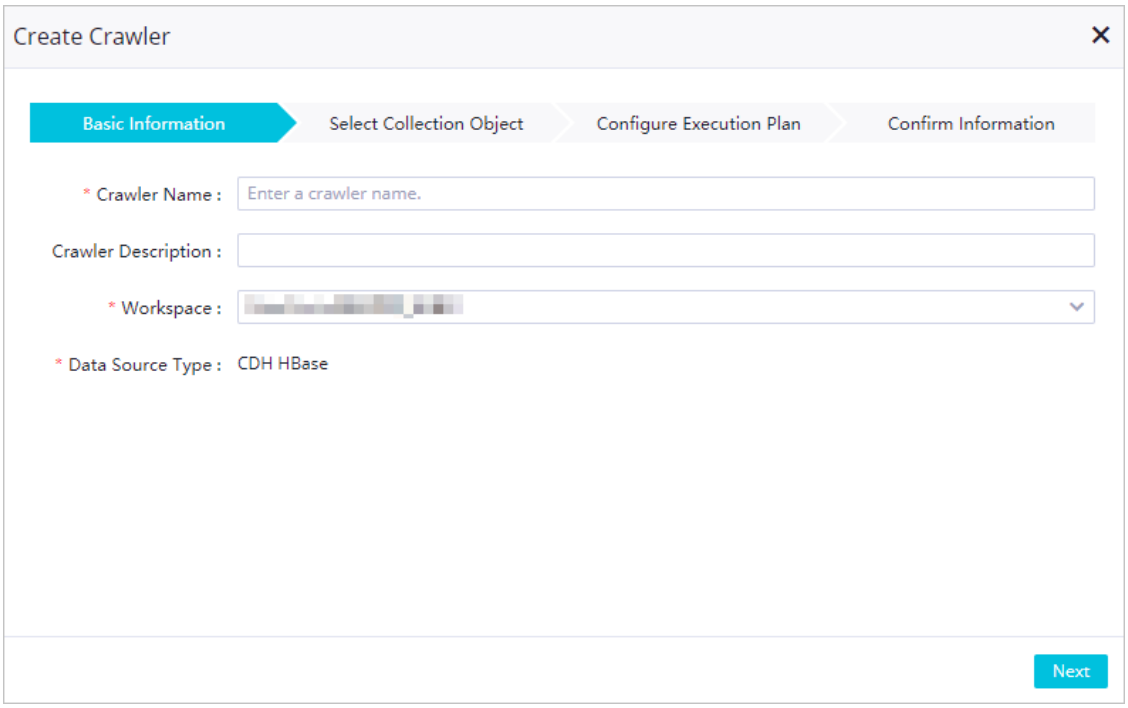
You cannot collect metadata across regions. You must create a crawler in the region where the source metadata resides to collect the metadata.

Create a crawler

1. Go to the **Data Discovery** 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 governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. Create a crawler.
   i. In the left-side navigation pane, click **CDH HBase**.
   ii. On the **CDH HBaseMetadata Crawler** page, click **Create Crawler**.
3. Configure the crawler.

i. Configure the basic information.

In the **Basic Information** step of the **Create Crawler** dialog box, set the parameters.



Parameter	Description
Crawler Name	The name of the crawler.
Crawler Description	The description of the crawler.
Workspace	The workspace in which the crawler resides.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is CDH HBase.

ii. Click **Next**.

iii. Select an HBase data source.

In the **Select Collection Object** step, select a data source and a resource group from the drop-down lists, and click **Start Testing** to test the connectivity between the data source and the resource group.

> **Note** If no data sources are available, click **Create** to add an HBase data source on the Data Source page.

iv. Configure an execution plan.

In the **Configure Execution Plan** step, configure an execution plan.

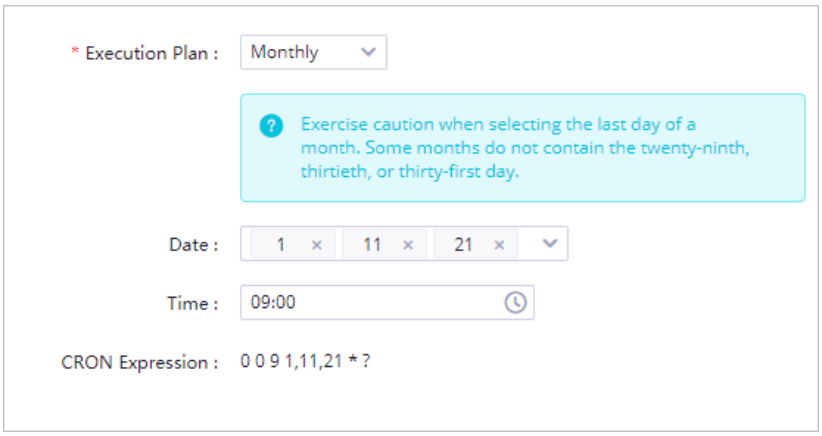
Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, **Hourly**, and **Customize**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the HBase data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: You need to manually run the crawler. The system collects metadata from the HBase data source based on your business requirements.

- Monthly: The system automatically collects metadata from the HBase data source once at a specific time on several specific days of each month.

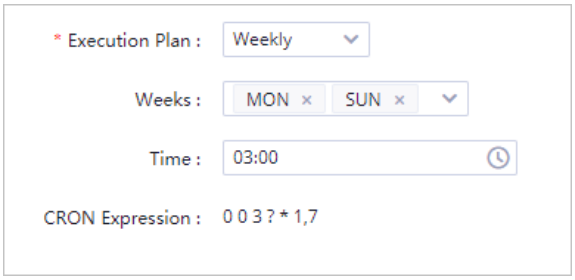
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the HBase data source on these dates. We recommend that you do not select the last days of a month.

  The following figure shows that the system automatically collects metadata from the HBase data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Weekly: The system automatically collects metadata from the HBase data source once at a specific time on several specific days of each week.

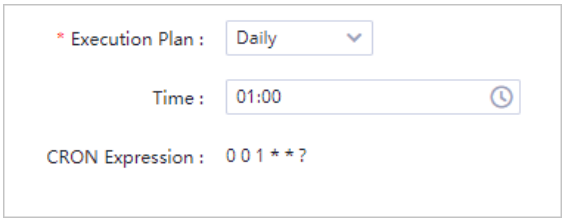
  The following figure shows that the system automatically collects metadata from the HBase data source once at 03:00 on Sunday and Monday of each week. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

  If the **Time** parameter is not set, the system automatically collects metadata from the HBase data source once at `00:00:00` on the specific days of each week.

- Daily: The system automatically collects metadata from the HBase data source once at a specific time of each day.

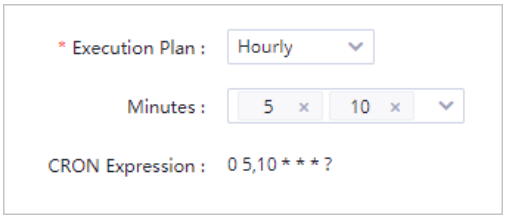
  The following figure shows that the system automatically collects metadata from the HBase data source once at 01:00 each day. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Hourly: The system automatically collects metadata from the HBase data source once from the `N ×` `5` th minute of each hour.

  > **Note** For an HBase metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the HBase data source from the 5th and 10th minutes of each hour. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Customize: You can enter a cron expression in the field of the Cron Expression parameter. The system automatically collects metadata based on the time configuration that matches the corn expression.

v. Click **Next**.

vi. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

Manage the crawler

On the **CDH HBaseMetadata Crawler** page, you can view, edit, and delete the created crawler.

Area No.	Description
1	In this area, you can enter the name of the crawler or the name of the data source in the search boxes to find the crawler. > **Note** The fuzzy match is supported. If you enter keywords in the search boxes, crawlers whose names or data source names contain the keywords are displayed.
2	In this area, you can view the details of the crawler in the **Status**, **Environment**, **Network Connectivity**, **Execution Plan**, **Last Run At**, **Last Consumed Time**, and **Average Running Time** columns. You can also perform the following operations on the crawler: • **Details**: View the configurations of the crawler. • **Edit**: Modify the configurations of the crawler. For example, you can modify the resource group and the execution plan of the crawler. • **Delete**: Delete the crawler. • **Run**: Run the crawler.

What's next

After the metadata is collected, you can view the details of the collected metadata on the **All Data** page of Data Map.

2.1.12.7.13. Collect metadata from a Kudu data source

DataWorks allows you to collect metadata that includes the table schema and the linage information of tables in Data Map. You can view the schema of a table and the relationships between tables. This topic describes how to create a crawler to collect metadata from a Kudu data source to DataWorks. You can view the collected metadata on the Data Map page.

Prerequisites

A CDH cluster is associated with the current DataWorks workspace. For more information, see Associate a CDH cluster with a workspace.

Limits

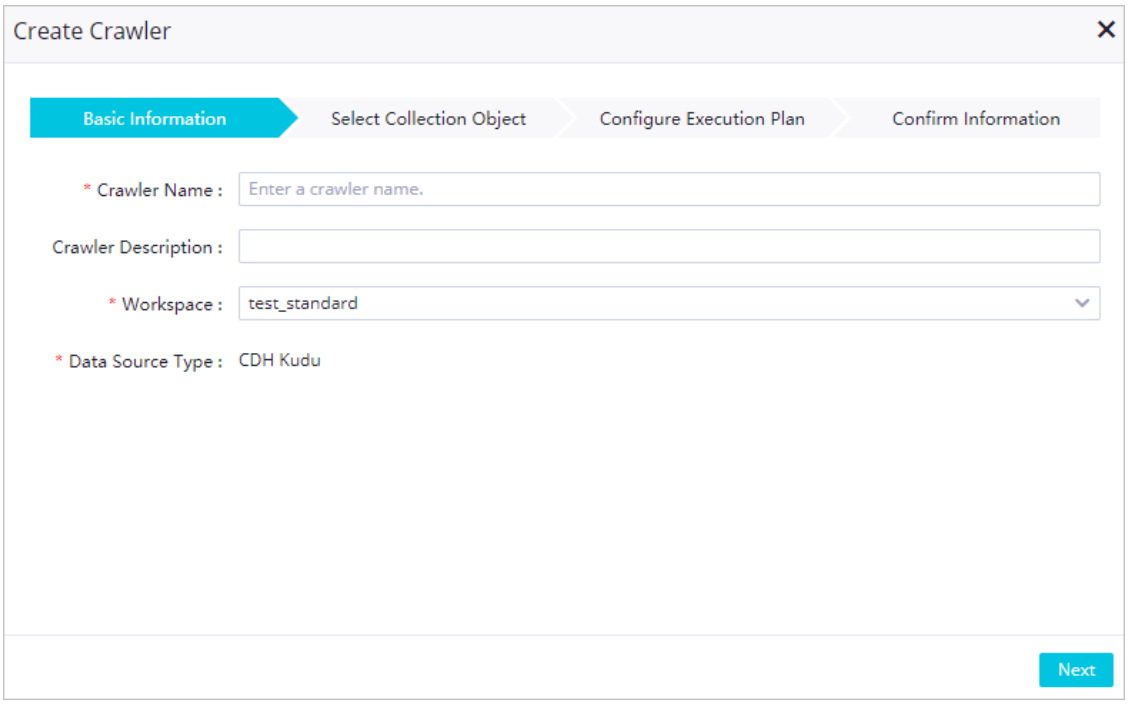
You cannot collect metadata across regions. You must create a crawler in the region where the source metadata resides to collect the metadata.

Create a crawler

1. Go to the **Data Discovery** 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 governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. Create a crawler.
   i. In the left-side navigation pane, click **CDH Kudu**.
   ii. On the **CDH KuduMetadata Crawler** page, click **Create Crawler**.
3. Configure the crawler.

i. Configure the basic information.

In the **Basic Information** step of the **Create Crawler** dialog box, set the parameters.



Parameter	Description
Crawler Name	The name of the crawler.
Crawler Description	The description of the crawler.
Workspace	The workspace in which the crawler resides.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is CDH Kudu.

ii. Click **Next**.

iii. Select a Kudu data source.

In the **Select Collection Object** step, select a data source and a resource group from the drop-down lists, and click **Start Testing** to test the connectivity between the data source and the resource group.

> **Note** If no data sources are available, click **Create** to add a Kudu data source on the Data Source page.

iv. Configure an execution plan.

In the **Configure Execution Plan** step, configure an execution plan.

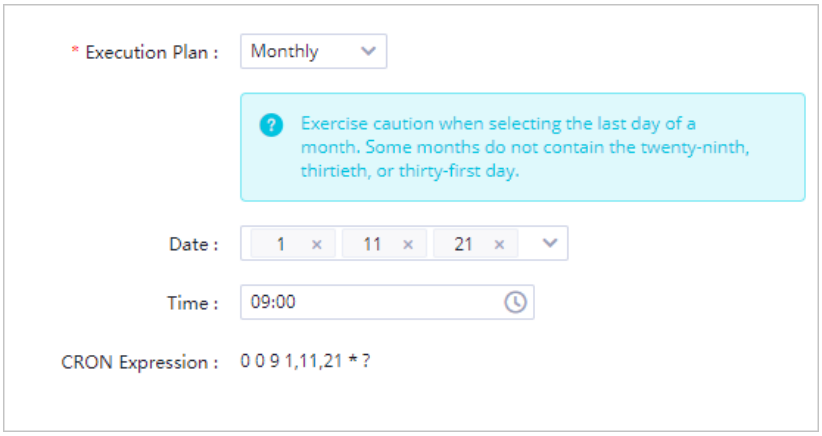
Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, **Hourly**, and **Customize**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the Kudu data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: You need to manually run the crawler. The system collects metadata from the Kudu data source based on your business requirements.

- Monthly: The system automatically collects metadata from the Kudu data source once at a specific time on several specific days of each month.

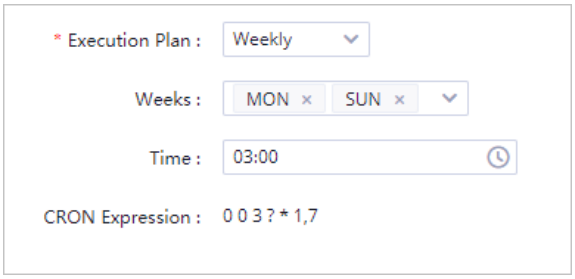
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the Kudu data source on these dates. We recommend that you do not select the last days of a month.

  The following figure shows that the system automatically collects metadata from the Kudu data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Weekly: The system automatically collects metadata from the Kudu data source once at a specific time on several specific days of each week.

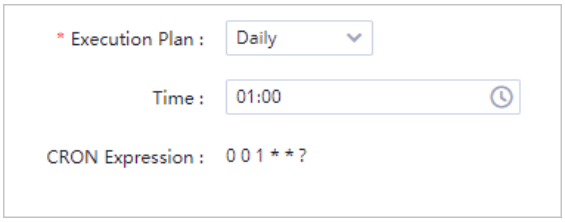
  The following figure shows that the system automatically collects metadata from the Kudu data source once at 03:00 on Sunday and Monday of each week. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

  If the **Time** parameter is not set, the system automatically collects metadata from the Kudu data source once at `00:00:00` on the specific days of each week.

- Daily: The system automatically collects metadata from the Kudu data source once at a specific time of each day.

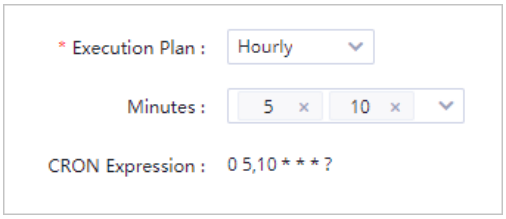
  The following figure shows that the system automatically collects metadata from the Kudu data source once at 01:00 each day. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Hourly: The system automatically collects metadata from the Kudu data source once from the `N` × `5` th minute of each hour.

  > **Note** For a Kudu metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the Kudu data source from the 5th and 10th minutes of each hour. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

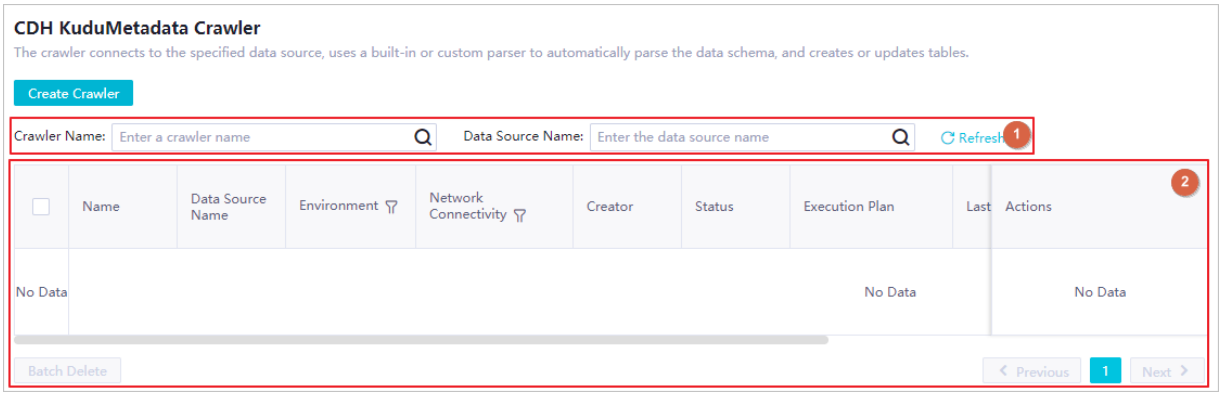
- Customize: You can enter a cron expression in the field of the Cron Expression parameter. The system automatically collects metadata based on the time configuration that matches the corn expression.

v. Click **Next**.

vi. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

Manage the crawler

On the **CDH KuduMetadata Crawler** page, you can view, edit, and delete the created crawler.



Area No.	Description
1	In this area, you can enter the name of the crawler or the name of the data source in the search boxes to find the crawler. **Note** The fuzzy match is supported. If you enter keywords in the search boxes, crawlers whose names or data source names contain the keywords are displayed.

Area No.	Description
2	In this area, you can view the details of the crawler in the **Status**, **Environment**, **Network Connectivity**, **Execution Plan**, **Last Run At**, **Last Consumed Time**, and **Average Running Time** columns. You can also perform the following operations on the crawler: • **Details**: View the configurations of the crawler. • **Edit**: Modify the configurations of the crawler. For example, you can modify the resource group and the execution plan of the crawler. • **Delete**: Delete the crawler. • **Run**: Run the crawler.

What's next

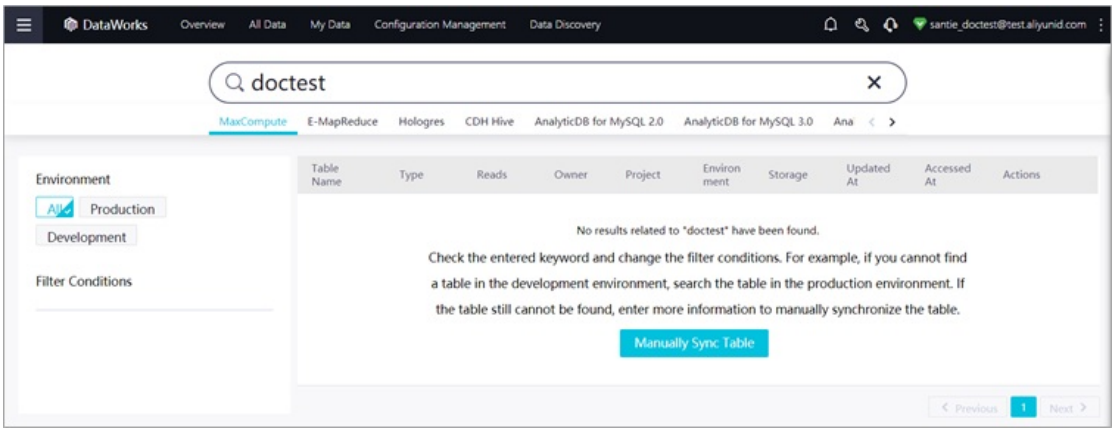
After the metadata is collected, you can view the details of the collected metadata on the **All Data** page of Data Map.

2.1.12.8. What do I do if no search results are returned when I query a newly created table in the Data Map service of DataWorks?

The Data Map service of DataWorks allows you to query tables on the homepage of Data Map by using a keyword. It also allows you to view the tables that you have recently browsed or read. If no search results are returned when you query a newly created table in the Data Map service of DataWorks, the possible cause is a latency of DataWorks in obtaining metadata. This topic describes how to address this issue.

Problem description

When I query a newly created table on the **All Data** page of Data Map by using a keyword, the **No data found.** message is displayed.



Possible causes

Possible causes:

- Possible cause 1: You entered an invalid keyword and no tables that match the keyword can be found.
- Possible cause 2: The table that you want to query has just been created, and no search results can be returned because of a latency of DataWorks in obtaining metadata.

Solution

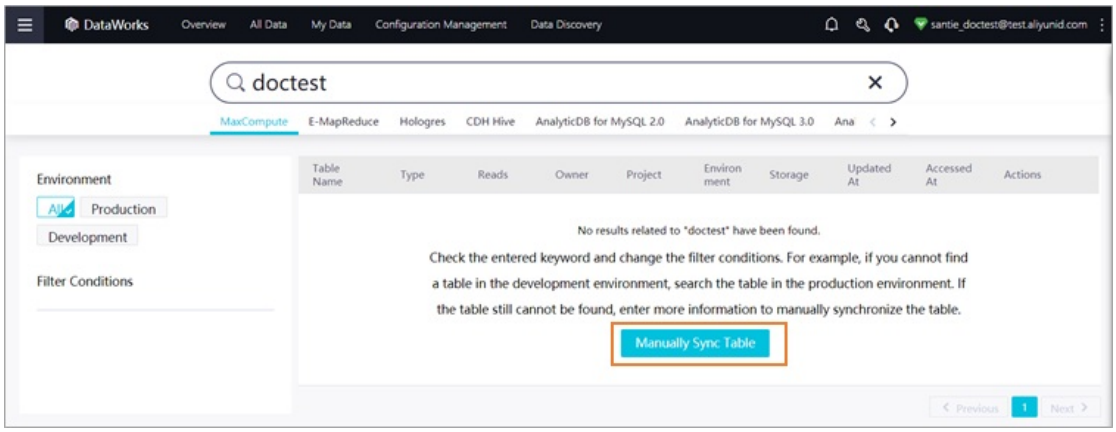
1. Check the keyword that you have entered.

   Make sure that the keyword that you have entered is valid and matches the table that you want to query.

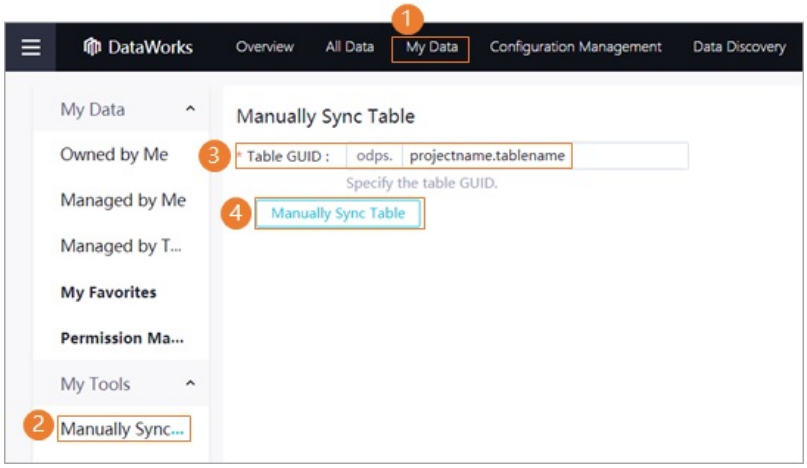
2. Manually synchronize the table.

   If the table that you want to query exists and the keyword that you have entered is correct, the possible cause is that the metadata of the newly created table has not been synchronized. In this case, you need to manually synchronize the table. Use one of the following methods to synchronize the table:

   - Click **Manually Sync Table** on the query result page.



   - Choose **My Data > Manually Sync Table** in Data Map of the DataWorks console. On the page that appears, `set Table GUID to a value in the format of Project name.Table name, and click Manually Sync Table.`



After you complete the preceding steps, query the table on the All Data page of Data Map by using the keyword again.

2.1.13. Data Asset Management

2.1.13.1. 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.

2.1.13.2. 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.

2.1.13.3. 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.

2.1.13.4. 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.

Parameter	Description
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.

2.1.13.5. 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.

2.1.13.6. Perform cross-tenant authorization

This topic describes how to perform cross-tenant authorization.

Authorization logic

DataWorks performs authorization within a tenant and cross-tenant authorization based on the following logic:

- Authorization within a tenant: An access control list (ACL) is used for authorization. You must 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 must 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.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > Data Asset Management**.
3. On the **Asset Portal** tab, enter the name of the table that belongs to another tenant in the search box and click **Search**.
4. 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.
5. Click **Request Permission** in the upper-right corner. The **Table Permission Request** page appears.
6. Set parameters on the **Table Permission Request** page.

Parameter	Description
Target Workspace	The workspace where the table is to be used.

Parameter	Description
Environment	The type of the environment where the table is to be used. If the selected workspace is in standard mode, you must 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 to which the permissions are granted. Valid values: **Current Account** and **System Account for Production Environment**.
Requested Period	The period in which the requested permissions are valid.
Reason for Request	The reason why you request the permissions. We recommend that you describe the reason in detail for faster approval.
Objects Requested	The tables you want to use. By default, the current table is selected. You can select other tables.

7. After the configuration is complete, 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 need to revoke a request, click **Revoke** in the Actions column of the request.
   - If you need to submit a revoked or rejected request again, click **Reapply** in the Actions column of the request.

8. 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 you click **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.

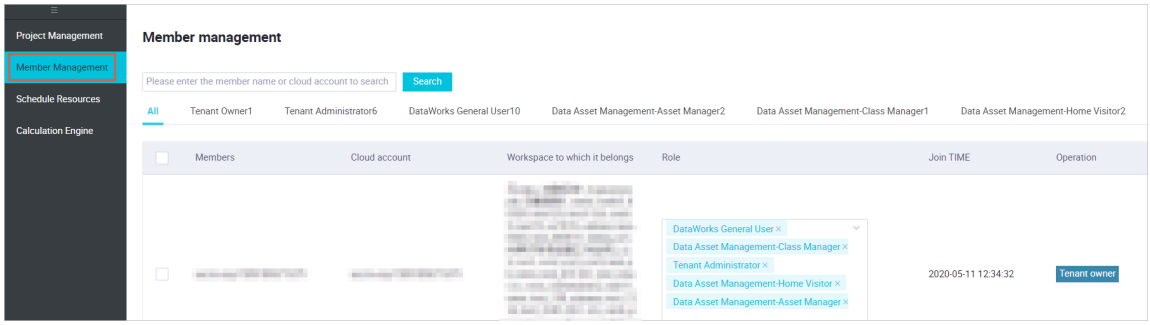
2.1.14. Organization management

2.1.14.1. Manage members

On the Member Management page of a workspace, you can manage and configure members in the workspace.

1. Log on to the DataWorks console as a workspace administrator.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > Project Management**.
3. In the left-side navigation pane, click **Member Management**.
4. Enter a member name or logon name in the search box to search for a member. You can assign a role to the member found or remove the member from the current workspace.
   - Assign a role to a member

To assign a role to a member, find the member in the member list and select the role from the drop-down list in the **Roles** column.



To revoke a role from a member, click **x** next to the role in the Roles column.

- Remove a member from the workspace

  To remove a member from the current workspace, find the member in the member list, click **Delete** in the Actions column, and then click **OK** in the **Remove from Tenant** message.

2.1.14.2. Resource groups

2.1.14.2.1. Go to the page for managing scheduling resources

Scheduling resources are managed at the tenant level. An exclusive scheduling resource can consist of multiple physical machines for running nodes.

1. Log on to the DataWorks console.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > Project Management**.
3. In the left-side navigation pane, click **Schedule Resources**. On the Schedule Resources page, manage scheduling resources as required.

   > **Note** If you are the tenant administrator, you can modify existing scheduling resources on the **Schedule Resources** page.

2.1.14.2.2. Change the workspace to which a scheduling resource belongs

After you create and configure an exclusive scheduling resource, you can change the workspace to which the resource belongs.

Procedure

1. Log on to the DataWorks console as the tenant administrator.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > Project Management**.
3. On the page that appears, click **Schedule Resources** in the left-side navigation pane. On the **Schedule Resources** page, enter a keyword in the search box in the upper-left corner. DataWorks searches for the scheduling resources whose names contain the keyword in fuzzy match mode.
4. Find the target scheduling resource and click **Modify attribution**.
5. In the **Modify attribution** dialog box, select the workspace to which the exclusive scheduling resource will be bound.
6. Click **OK**.

2.1.14.3. Configure the compute engine

DataWorks supports only MaxCompute as its compute engine. All workflows and nodes in a workspace are run on the MaxCompute compute engine associated with the workspace.

Background information

You can modify the compute engine that has been configured for a workspace as the tenant administrator. You can modify the following settings of the compute engine: description, whether to use the account to which the MaxCompute project belongs to run nodes, account that is used to run nodes, and AccessKey pair of the account.

For example, if the owner of a MaxCompute project resigns, but Run Nodes Using MaxCompute Owner Account is not selected for the corresponding nodes, you must specify another account and the corresponding AccessKey pair for running those nodes in time.

Procedure

To change the account for running nodes and its AccessKey pair, perform the following steps:

1. Log on to the DataWorks console as the tenant administrator.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > Project Management**.
3. On the page that appears, click **Calculation Engine** in the left-side navigation pane. On the **Calculation Engine** page, enter a keyword in the search box in the upper-left corner. DataWorks searches for the compute engines whose names contain the keyword in fuzzy match mode.
4. Find the target compute engine and click **Configuration** in the Operation column.
5. In the **Configure MaxCompute Compute Engine** dialog box, set the Tenant Account for Running Nodes, Access Id, and Access Key parameters as required.

   > **Note** As the tenant administrator, you can select Run Nodes Using MaxCompute Owner Account or create an account and set the Tenant Account for Running Nodes parameter to this account.

6. Click **OK**.

2.1.15. Data Service

2.1.15.1. 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.

2.1.15.2. 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.

2.1.15.3. 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.

2.1.15.4. Manage business processes and objects under business processes

2.1.15.4.1. Manage business processes

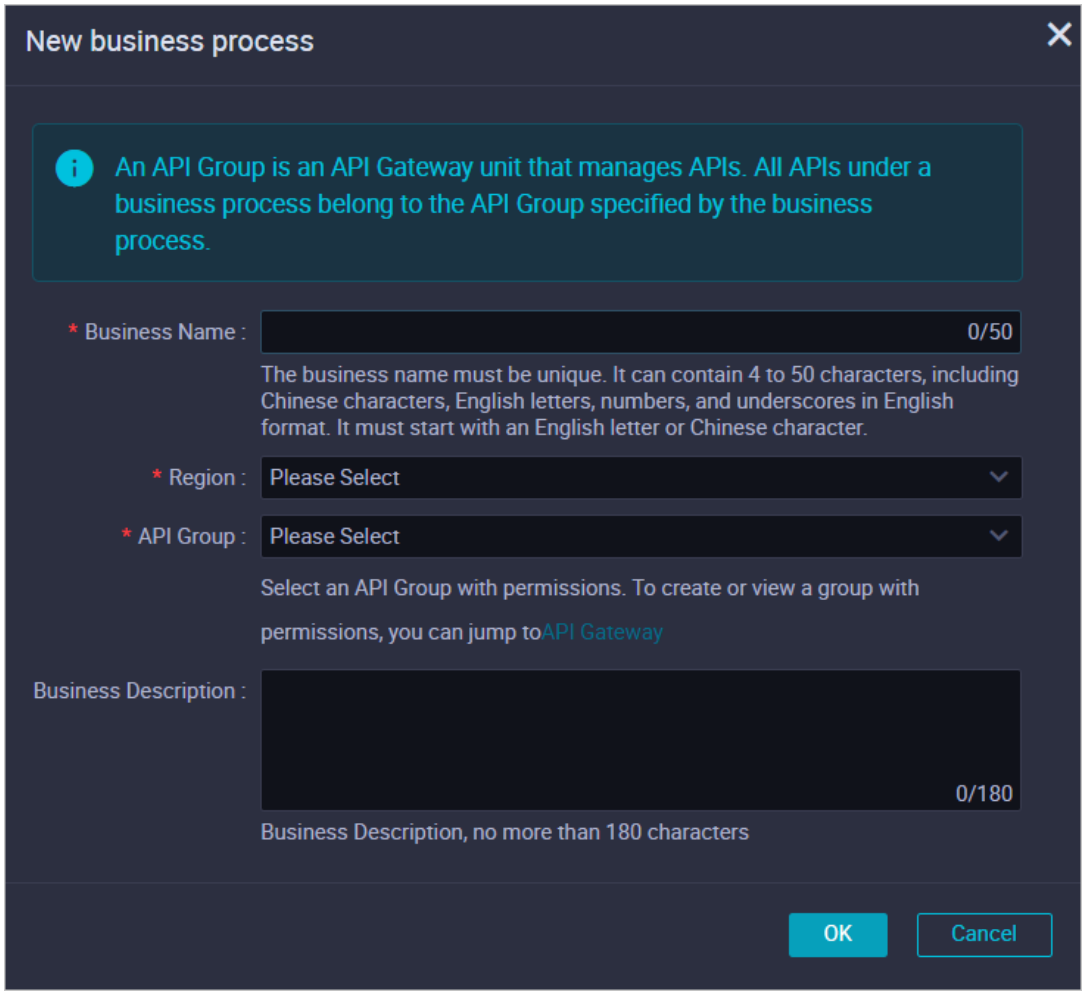
This topic describes how to create, modify, and delete a business process.

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 the following categories: API, function, and workflow.

Create a business process

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Development > Data Service**.
3. On the **Service Development** page, move the pointer over the icon and select **Workflows**.
4. In the **Create Workflow** dialog box, configure the parameters.



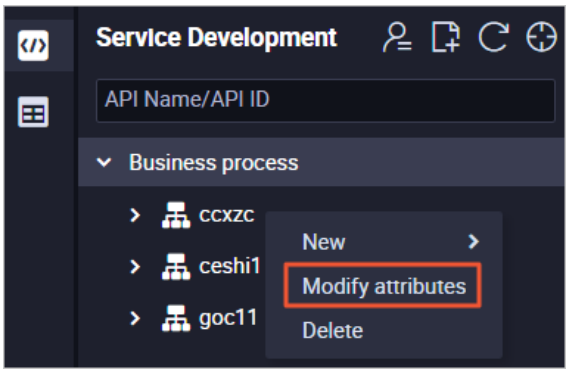
Parameter	Description
Business Name	The name of the business process. ○ The name can contain letters, digits, and underscores (_). ○ The name must start with a letter. ○ The name must be 4 to 50 characters in length. ○ The name must be unique in the workspace to which the business process belongs.
Region	The region. After you select a region, you can publish APIs to this region. You can set this parameter to a region that is associated with the level-1 organization to which your account belongs in the Apsara Uni-manager Management Console.
API Group	The API group to which the APIs in the business process belong. An API group is the API management unit of API Gateway. You can select an API group from the **API Group** drop-down list. We recommend that you select an API group on which you have permissions in the Apsara Uni-manager Management Console. If you want to view existing API groups or create an API group, go to the API Gateway console.
Business Description	The description of the business process, which cannot exceed 180 characters in length.

5. Click **OK**.

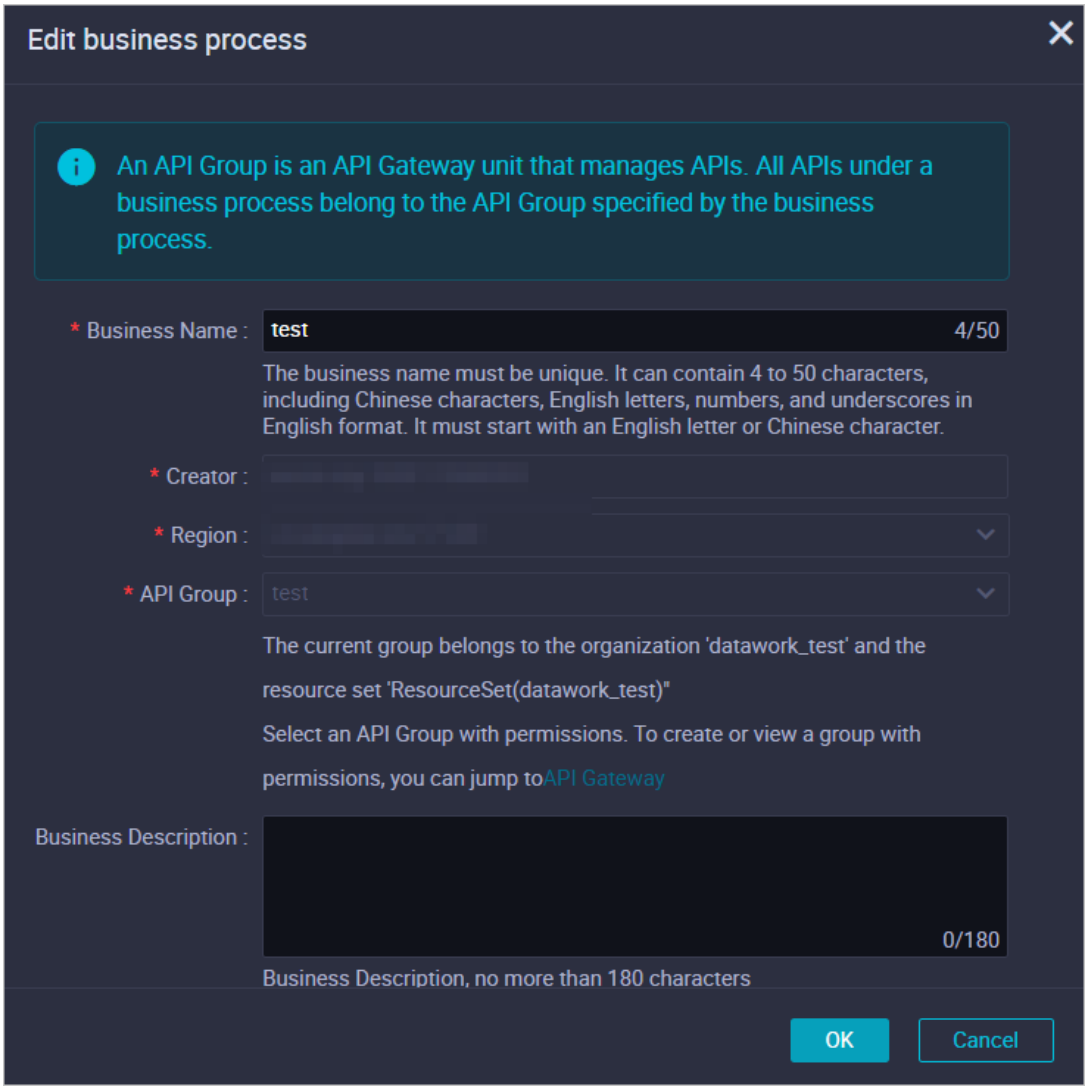
   After the business process is created, you can view it in the business process list.

Modify a business process

1. On the **Service Development** page, right-click the name of the business process that you want to modify and select **Modify attributes**.



2. In the **Edit business process** dialog box, modify the **Business Name** and **Business Description** parameters as required.

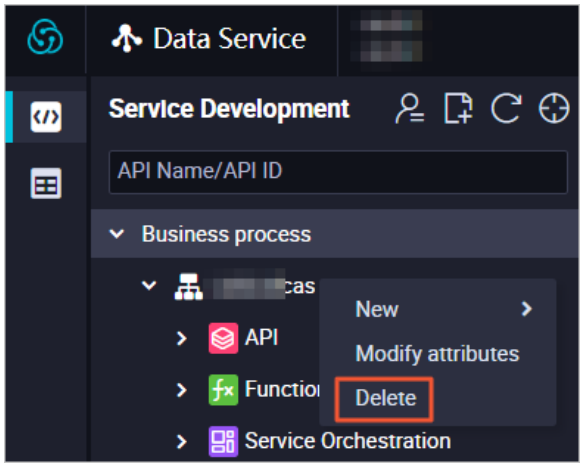


> **Note** You cannot modify the **Creator** or **API Group** parameter of a business process.

3. Click **OK**.

Delete a business process

1. On the **Service Development** page, right-click the name of the business process that you want to delete and select **Delete**.



2. In the Notes message, click **OK**.

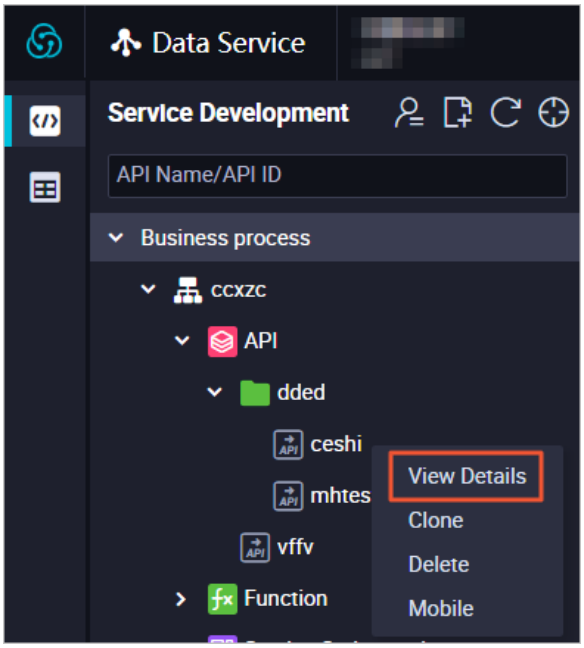
> **Note**
> - You can delete only business processes that do not contain objects such as folders, APIs, functions, or workflows.
> - If you want to delete a business process that contains such objects, delete the objects before you delete the business process.

2.1.15.4.2. 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**.

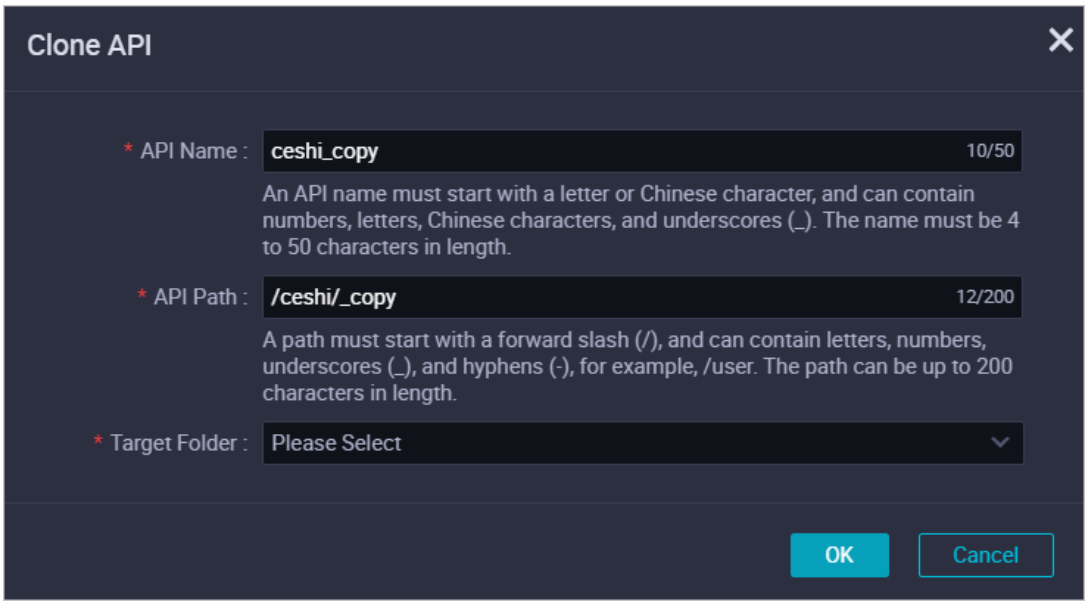


Note The **View Details** option appears only in the shortcut menu of an API that has been published. If an API has not been published, double-click the API to go to the configuration tab of the API. Then, click **Properties** in the right-side navigation pane to view its basic information.

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.

Parameter	Description
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**.

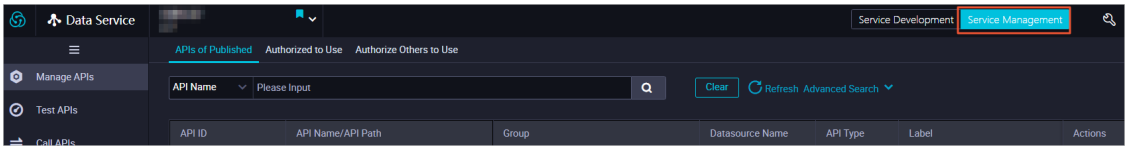
Delete an API

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.

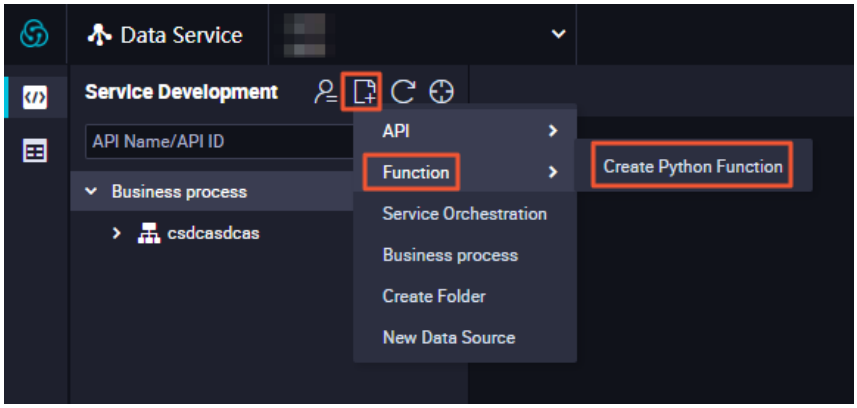
1. On the **Service Development** tab, right-click the name of the target API and select **Mobile**.
2. In the **Modify file path** dialog box, set the **Target Folder** parameter.
3. Click **OK**.

2.1.15.4.3. Manage functions

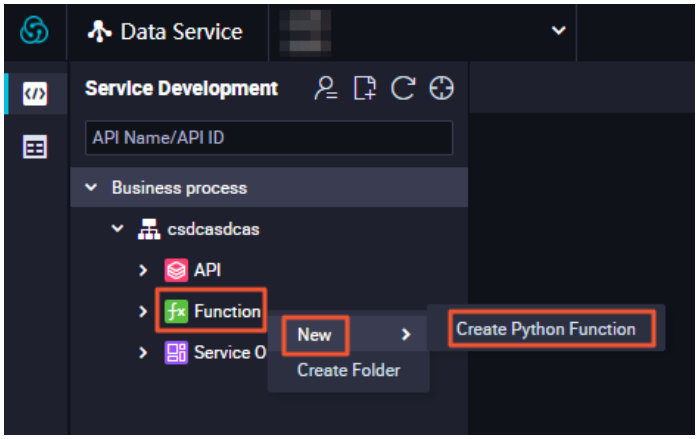
This topic describes how to create, clone, delete, and move Python functions.

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**.

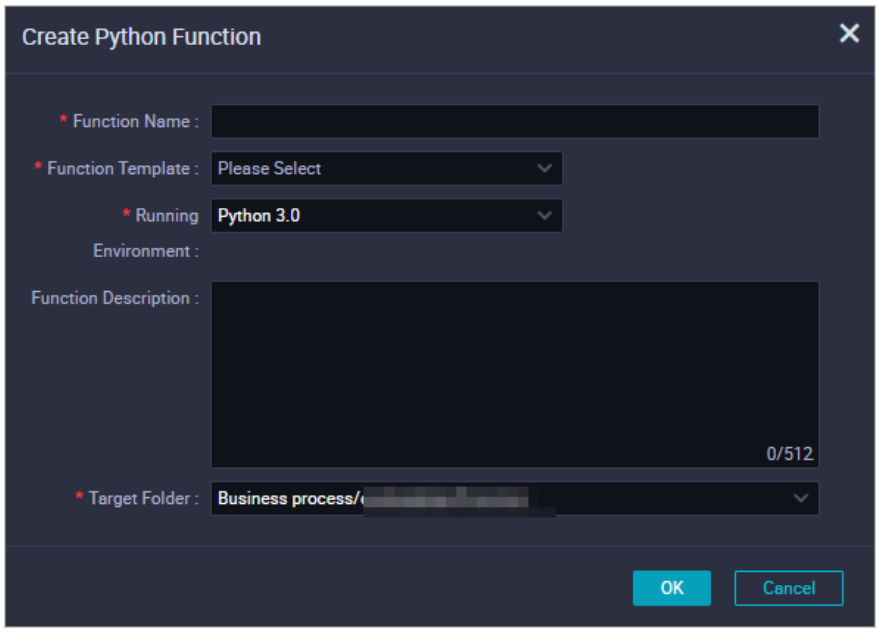


You can also click a business process, right-click **Function**, and then choose **New > Create Python Function**.



> **Notice** DataService Studio allows you to create only Python functions.

4. In the **Create Python Function** dialog box, set the parameters as required.

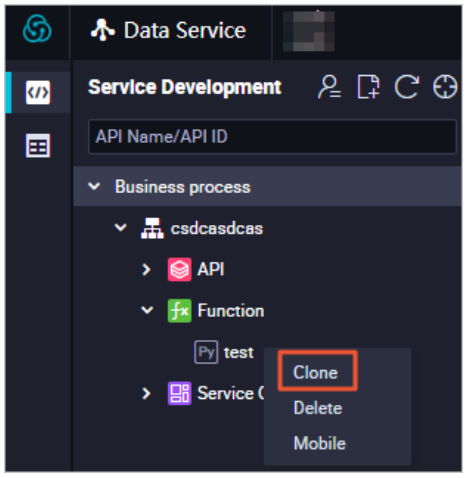


Parameter	Description
Function Name	The name of the function to create, which can be up to 256 characters in length.
Function Template	The template used to create the function. Set the value to Python3 Standard v1.
Running Environment	The runtime environment of the function. Set the value to Python 3.0.
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.

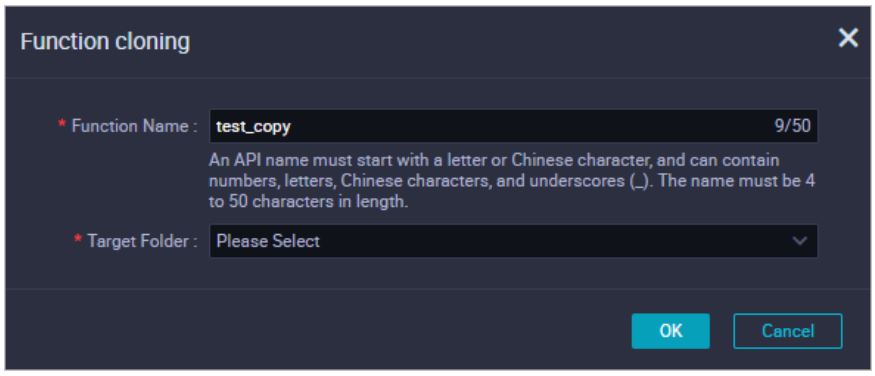
5. Click **OK**.
6. On the configuration tab of the function, configure the function.
   i. In the **Edit Code** section, enter the function code.
   ii. In the **Environment Configuration** section, set the **Memory** and **Function Timeout** parameters.
7. Click Save icon in the toolbar.

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.

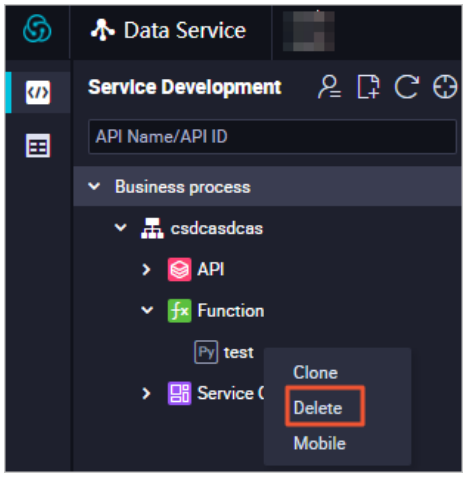


> **Note** The name of the function must be 4 to 50 characters in length and can contain letters, digits, and underscores (_). It must start with a letter.

3. Click **OK**.

Delete a function

1. On the **Service Development** tab, right-click the name of the target function and select **Delete**.

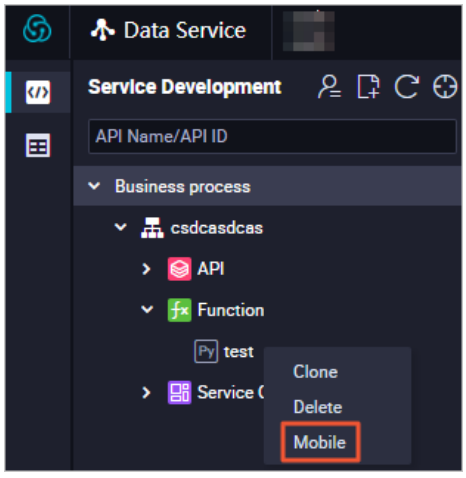


2. In the message that appears, click **OK**.

> **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**.

2.1.15.4.4. Manage workflows

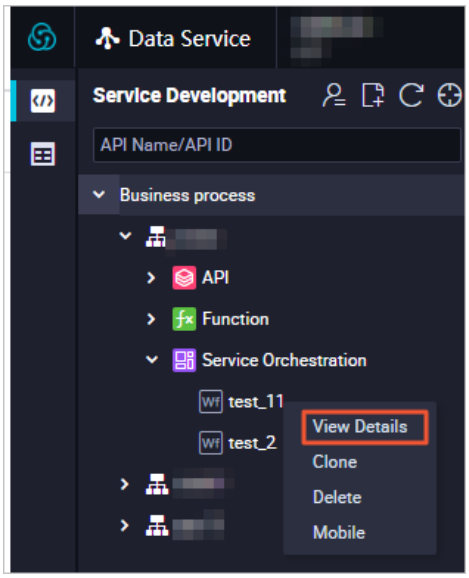
This topic describes how to view, clone, delete, and move a workflow.

Prerequisites

Workflows are created and published. For more information, see Use workflows.

View a workflow

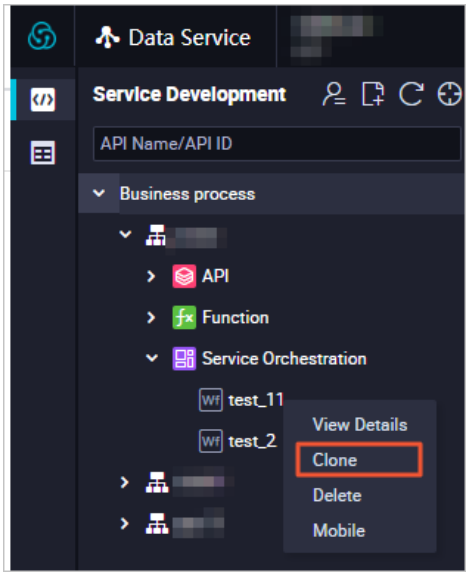
1. Log on to the DataWorks console.
2. Click ☰ in the upper-left corner and choose **All Products > Data Service**.
3. On the Service Development tab, right-click the name of the target workflow and select **View Details**.



> **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**.

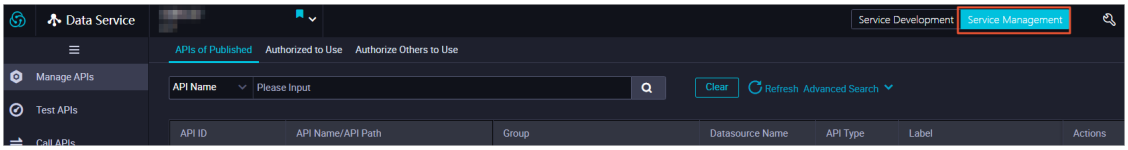
Delete a workflow

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**.

2.1.15.5. 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
	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

2.1.15.5.1. Configure a data source

You can configure data sources for DataService Studio to read table schemas and process API query requests.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
3. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.

   You can configure a data source on this page. The following table lists the supported data sources.

Data source	Generate an API in wizard mode	Generate an API in script mode
PolarDB-X	Supported	Supported
MySQL	Supported	Supported
PostgreSQL	Supported	Supported
SQL Server	Supported	Supported

Data source	Generate an API in wizard mode	Generate an API in script mode
Oracle	Supported	Supported
AnalyticDB for MySQL 2.0	Supported	Supported
AnalyticDB for PostgreSQL	Supported	Supported
Tablestore	Supported	Not supported
MongoDB	Supported	Not supported
Hologres	Supported	Supported

> **Note** DataService Studio cannot directly read data from MaxCompute. You can add a Hologres data source and use the accelerated query feature of Hologres to allow DataService Studio to query data in MaxCompute.
>
> Hologres is compatible with PostgreSQL. PostgreSQL does not support the DATETIME data type. If your MaxCompute project contains data of the DATETIME type, Hologres converts the data from the DATETIME type to the TIMESTAMP type that PostgreSQL supports for DataService Studio to query.

2.1.15.5.2. 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.

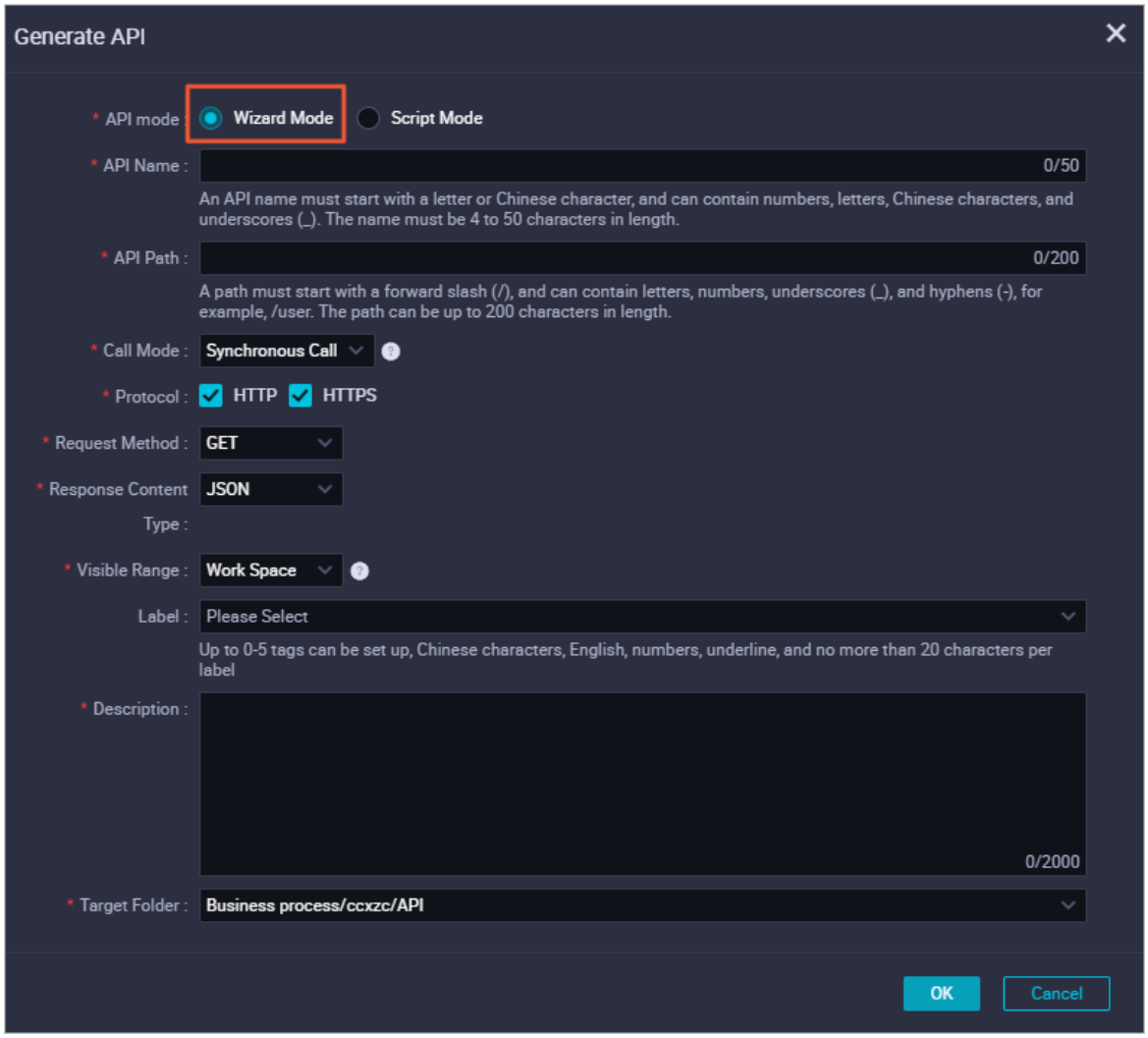
Prerequisites

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**.

Parameter	Description
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**.

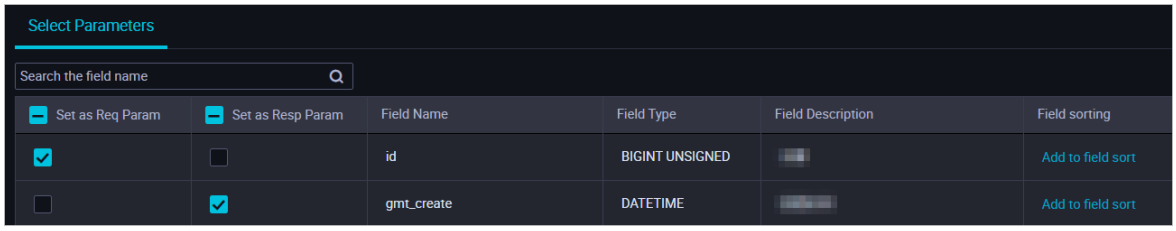
Configure the API

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.

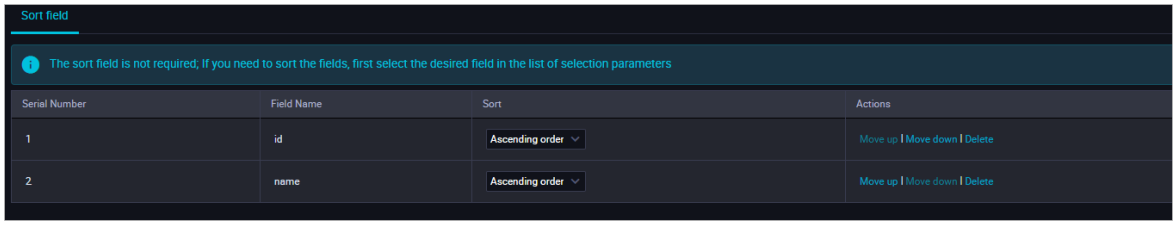
   Note
   - Before you select a table for an API, you must configure a connection in Data Integration. You can enter a table name in the Table Name field to search for the desired table.
   - After you create an API, the table configuration tab automatically appears for you to select a table for the API.

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.

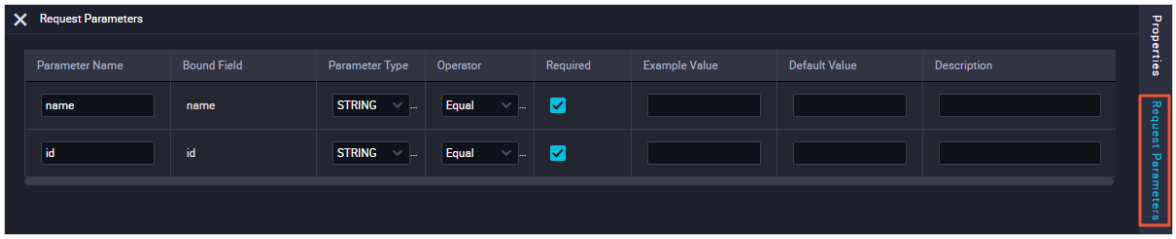


To sort the data returned by the API based on a field, click **Add to field sort** in the Actions column of the field to add it to the **Sort field** section.



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.



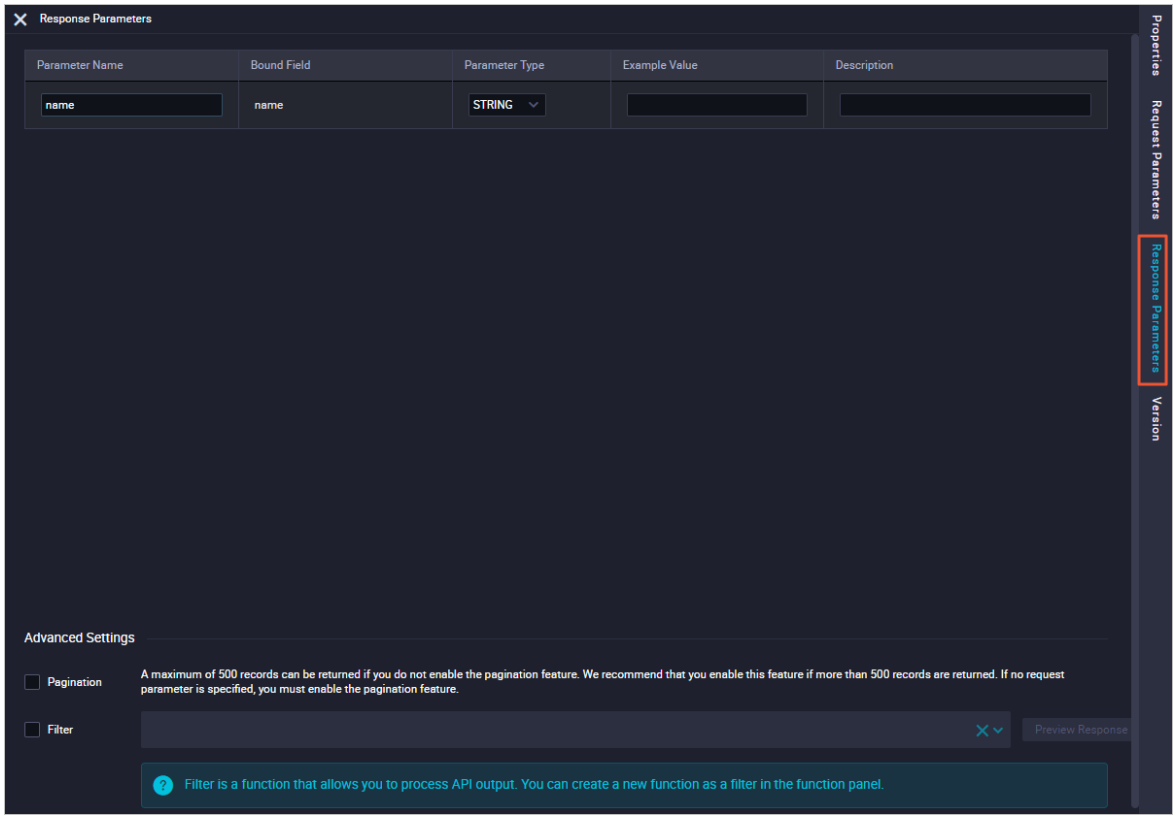
Parameter	Description
Parameter Name	The name of the request parameter. The name can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter and can be up to 64 characters in length.
Bound Field	The field to be bound to the request parameter. You cannot change the value.
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.

Parameter	Description
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	Description
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.

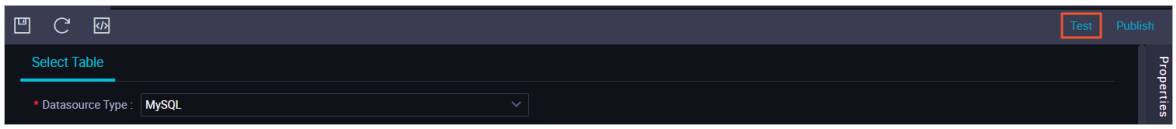
If you need to process the query results returned by the API, select **Filter**.

> **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.

2.1.15.5.3. Create an API in the code editor

To meet the requirements of advanced data queries, 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

Data sources are configured on the **Data Source** page in **Workspace Management**. For more information about how to configure a data source, see Configure a data source.

Create an API

1. Log on to the DataWorks console.
2. On the DataStudio page, click the  icon in the upper-left corner and choose **All Products > Data Development > Data Service**.
3. On the **Service Development** page, move the pointer over the  icon and choose **API > Generate API**.

   You can also select a required business process, right-click **API**, and then choose **New > Generate API**.
4. In the **Generate API** dialog box, configure the parameters as required.

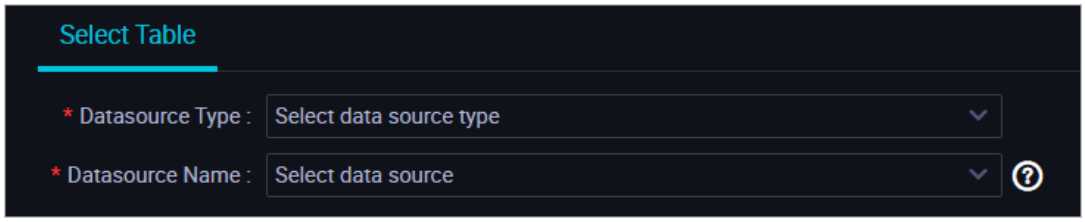
Parameter	Description
API mode	The mode used to create an API. Valid values: **Wizard Mode** and **Script Mode**. In this example, select **Script Mode**.
SQL Mode	The SQL mode. Valid values: **Basic SQL** and **Advanced SQL**. ○ **Basic SQL**: Use basic SQL statements to implement the query logic. This mode provides the SQL capability the same as that in earlier versions. ○ **Advanced SQL**: Use SQL statements with MyBatis tags to implement the query logic. This mode supports the following tag types: if, choose, when, otherwise, trim, foreach, and where.
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 in which the API is stored, such as */user*.
Call Mode	The mode used to call the API. Valid values: **Synchronous Call** and **Asynchronous Call**. ○ If you set this parameter to **Synchronous Call**, the API returns results immediately after it is called. The synchronous mode is most commonly used. ○ If you set this parameter to **Asynchronous Call**, the API returns the request ID 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. Valid values: **HTTP** and **HTTPS**.
Request Method	The request method. Valid values: **GET** and **POST**.
Response Content Type	The format of the data returned by the API. Set the value to **JSON**.
Visible Range	The range of users to whom the API is visible. 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 members. **Note** If you set this parameter to Private, other members in the workspace cannot view the API in the API list.
Label	The tag of the API. Select one or more tags from the drop-down list. For more information about tags, see Manage tags. **Note** You can set a maximum of five tags for an API.
Description	The description of the API, which cannot exceed 2,000 characters in length.
Target Folder	The folder that stores the API.

5. Click **OK**.

Configure the API

1. Double-click the API in the API list. On the tab that appears, specify **Datasource Type** and **Datasource**

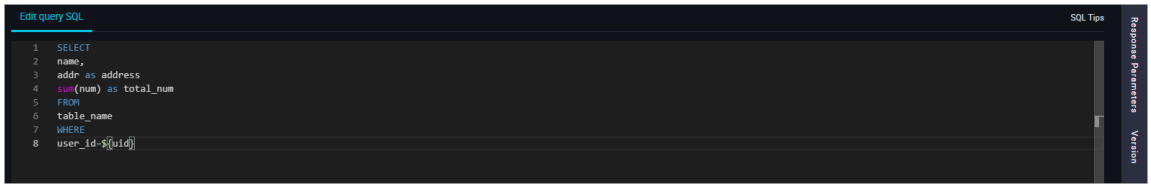
Name in the **Select Table** section.



> **Note** A data source must be selected first, and only table join queries in the same data source are supported.

2. In the **Environment Configuration** section, specify **Memory** and **Function Timeout**.

3. In the **Edit query SQL** section, enter an SQL statement for querying data.
   - If you set the SQL Mode parameter to **Basic SQL**, you can enter only a basic SQL statement.



> **Note** The SELECT statement 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 an SQL statement:

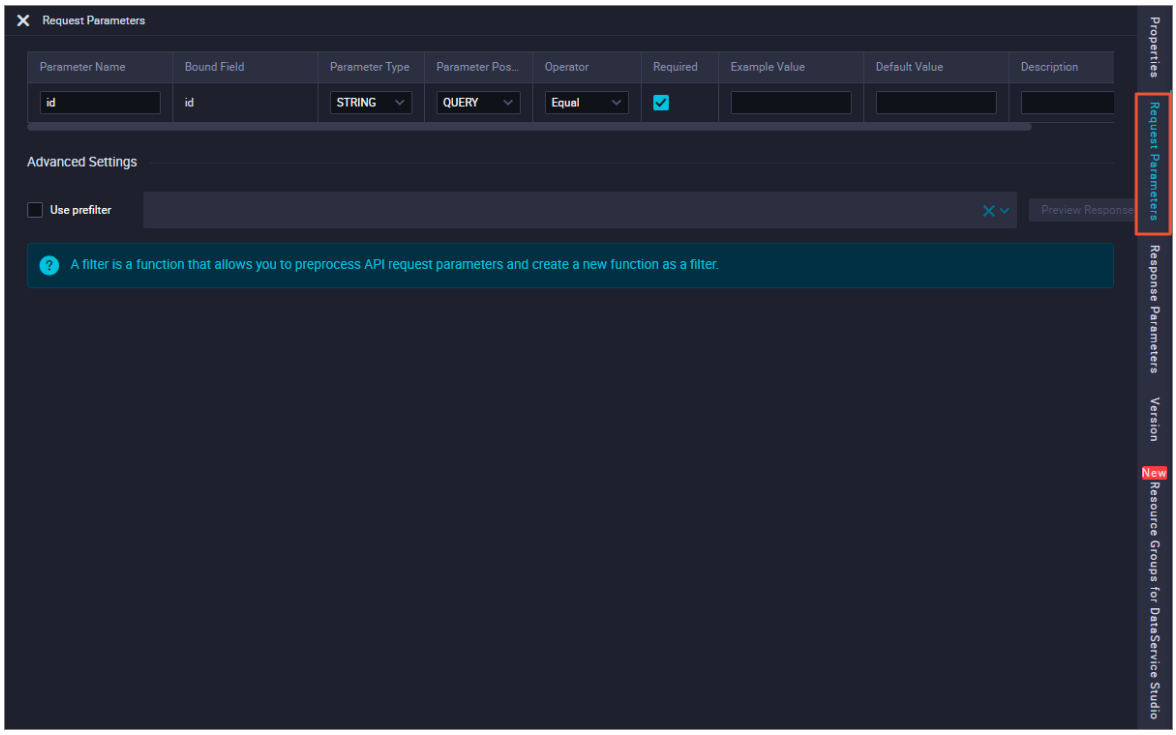
- Single-table queries, table join queries, and nested queries in the same data source are supported.
- Only a SELECT statement is supported. You cannot add comments before the SELECT statement. Other statements such as INSERT, UPDATE, and DELETE are not supported.
- `SELECT *` is not supported. You must specify the columns to be queried.
- If the name of a column to be queried is prefixed with a table name (example: t.name) or you use an aggregate function, such as min, max, sum, or count in the column name, you must create an alias for the corresponding response parameter. Examples: t.name as name and sum(num) as total_num.
- The ${param} variable in an SQL statement is regarded as a request parameter and is replaced with an actual value. The variable cannot be replaced with a column name.
- ${param} cannot be enclosed in single quotation marks (' '). For example, `'${id}'` and `'abc${xyz}123'` are not allowed. If necessary, you can use `concat('abc', ${xyz}, '123')` instead.
- ${param} is not allowed in comments. For example, --${id} is not allowed.
- If a request parameter is set as optional and is not specified when you call an API, a null value is used for this parameter in an SQL statement that is executed.

   - If you set the SQL Mode parameter to **Advanced SQL**, you can enter an SQL statement with MyBatis tags.

     This mode supports the following MyBatis tag types: if, choose, when, otherwise, trim, foreach, and where.

4. In the right-side navigation pane, click **Request Parameters**. In the Request Parameters pane, configure the parameters as required.

   If you set the SQL Mode parameter to **Advanced SQL**, you must manually add all request parameters that are specified in the SQL statement to the parameter list. This ensures that the parameters that are described in the API details are consistent with the parameters that are actually used.



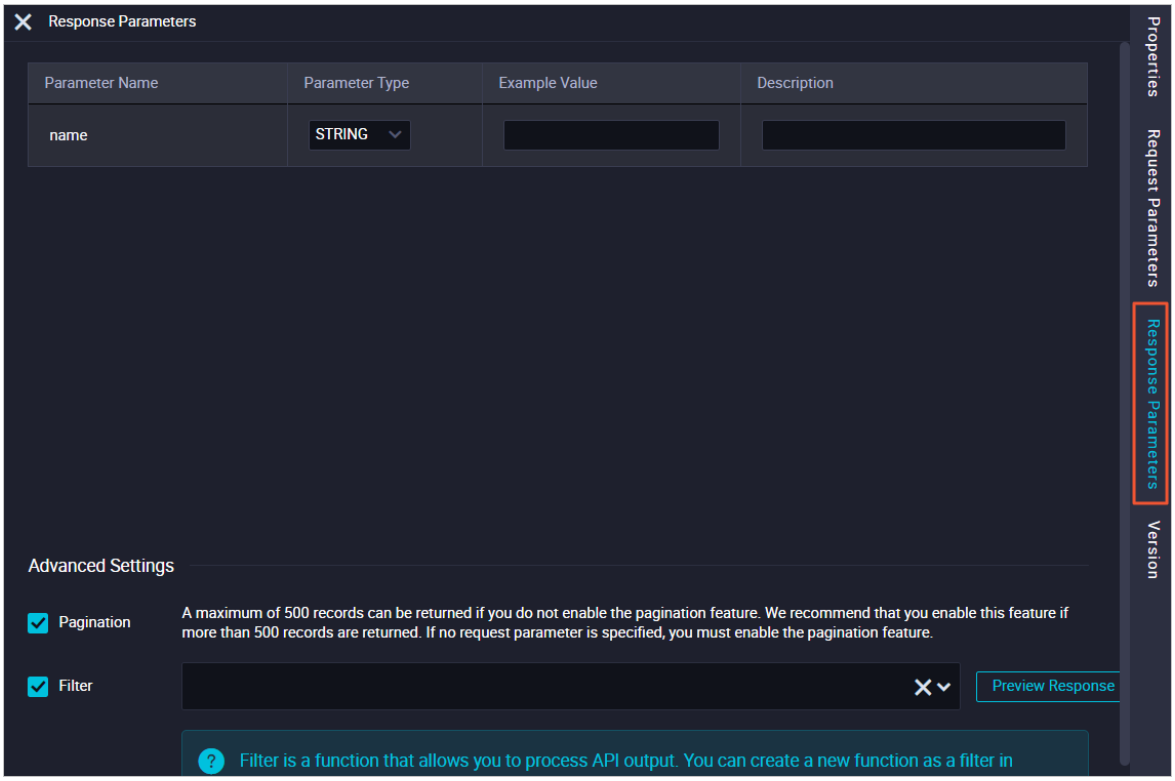
Parameter	Description
Parameter Name	The name of the request parameter. The name cannot exceed 64 characters in length, and can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter.
Parameter Type	The type of the request parameter. Valid values: **STRING**, **INT**, **LONG**, **FLOAT**, **DOUBLE**, and **BOOLEAN**.
Required	Specifies whether the request parameter is required.
Example Value	The example 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 match 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 example 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, configure the parameters as required.

   If you set the SQL Mode parameter to **Advanced SQL**, you must manually add all response parameters that are specified in the SQL statement to the parameter list. This ensures that the parameters that are described in the API details are consistent with the parameters that are actually used.



Parameter	Description
Parameter Name	The name of the response parameter. The name cannot exceed 64 characters in length, and can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter.
Parameter Type	The type of the response parameter. Valid values: **STRING**, **INT**, **LONG**, **FLOAT**, **DOUBLE**, and **BOOLEAN**.
Example Value	The example 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.
- If the API may return more than 2,000 records, we recommend that you select **Pagination**.

The following common parameters are available when you select Pagination:

- 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.

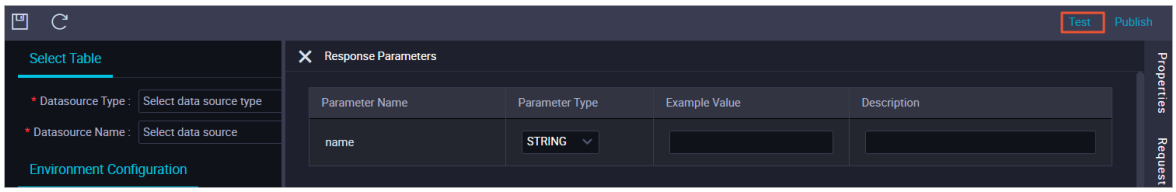
If you want to process the query results returned by the API, select **Filter**.

> **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 request parameters for an API, you must select **Pagination**.

6. Click 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 to pass 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 must be configured because it is an important reference for API callers.
> - The API call latency is the latency of the current API request. The latency is used to evaluate the API performance. If the latency is long, consider whether to optimize the database.

3. After the API is tested, close the **Test APIs** dialog box and click **Publish** in the upper-right corner.

2.1.15.5.4. Use filters

2.1.15.5.4.1. 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
```

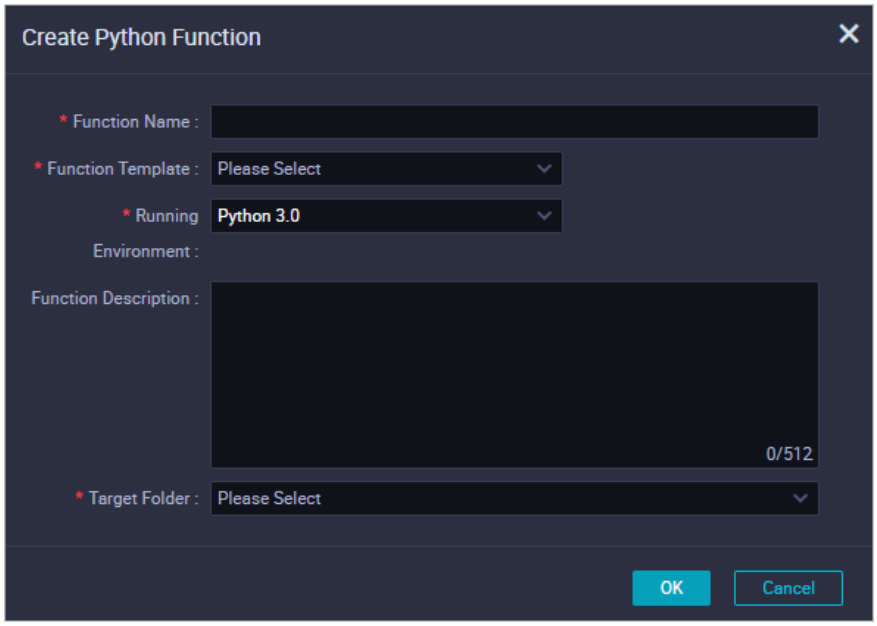
You can modify the function template to write your own function. You can modify the names of the input parameters as needed.

```
Parameter 1 [context]: the context of calling APIs. The value is of the STRING type. This parameter
is not in use and is left empty.
Parameter 2 [event]: the result data returned by APIs or the preceding filter. The value is of the S
TRING 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.

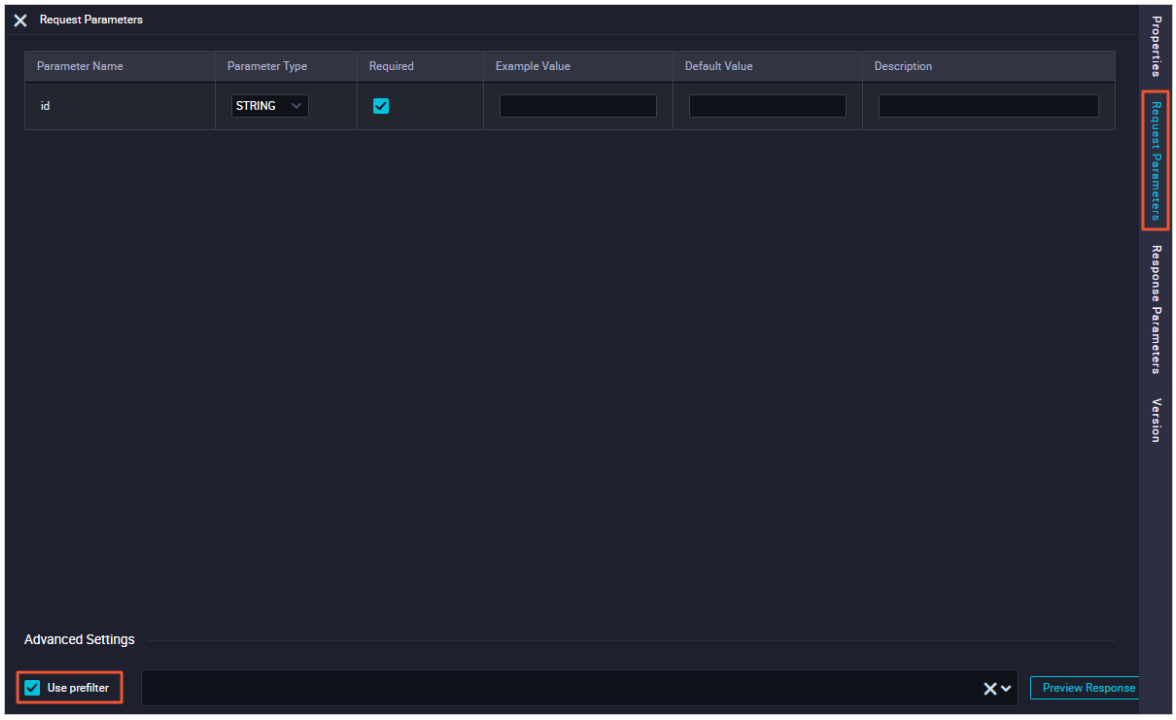


Parameter	Description
Function Name	The name of the function to create, which can be up to 256 characters in length.
Function Template	The template used to create the function. Set the value to Python3 Standard v1.
Running Environment	The runtime environment of the function. Set the value to Python 3.0.
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.

2.1.15.5.4.2. 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
```

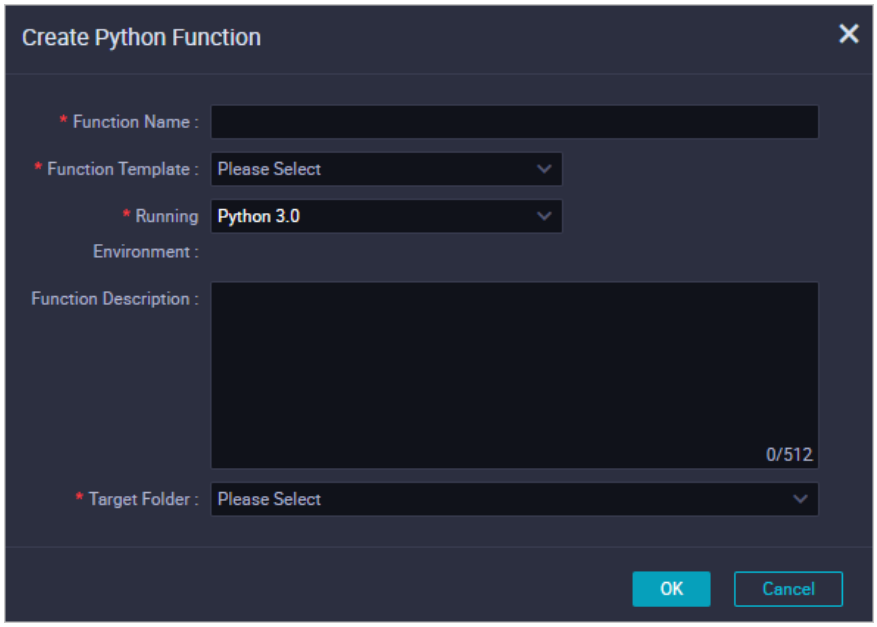
You can modify the function template to write your own function. You can modify the names of the input parameters as needed.

```
Parameter 1 [context]: the context of calling APIs. The value is of the STRING type. This parameter
is not in use and is left empty.
Parameter 2 [event]: the result data returned by APIs or the preceding filter. The value is of the S
TRING 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.



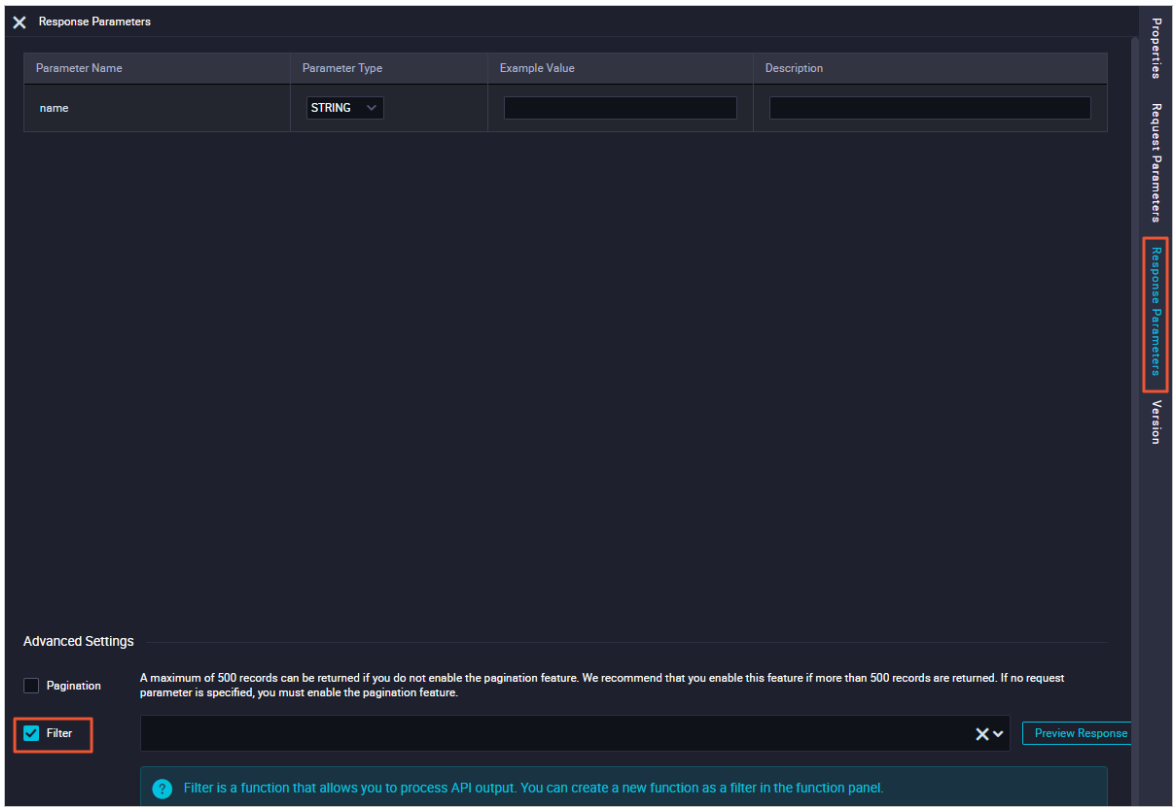
Parameter	Description

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.

2.1.15.6. 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.
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.

2.1.15.7. 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.

2.1.15.8. 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.

2.1.15.9. Call an API

This topic describes how to call an API after the API is published to API Gateway.

Prerequisites

An API is published to API Gateway. For more information, see Publish APIs.

The following conditions are met:

- The parameter definition of the API is obtained.
- 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.
- The app is authorized to call the API.

Context

API Gateway allows you to use SDKs to authorize apps to call APIs. You can authorize your own account, a user in your enterprise, or a third party to call APIs.

Procedure

1. Obtain the API documentation.

   The method of obtaining the API documentation varies based on how you obtain an API. You can use one of the following methods to obtain an API:

   - Purchase the API in Alibaba Cloud Marketplace.
   - Obtain the API authorization from the API provider.

2. Create an app in API Gateway.

   In API Gateway, apps define the identities that you use to call APIs. Each app has a key pair that consists of AppKey and AppSecret, which are equivalent to an account and its password.

3. Obtain the permissions to call the API.

   Authorization is to grant an app the permissions to call an API. Your app must be authorized before it can be used to call an API. The authorization method varies based on how you obtain an API.

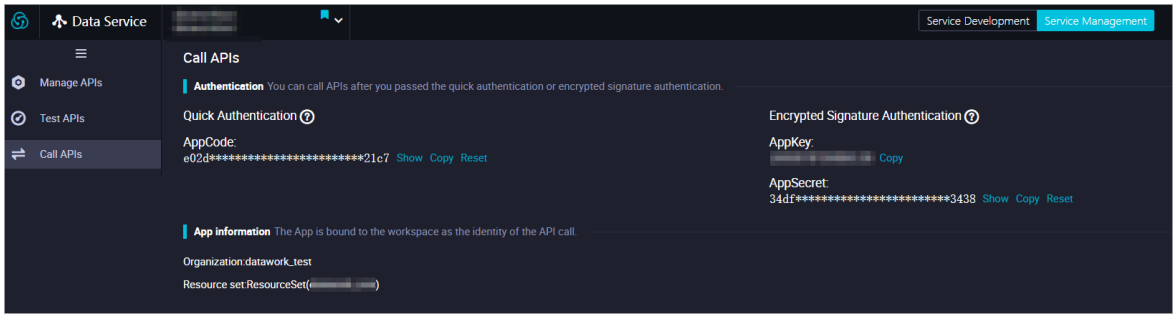
4. Call the API.

   You can send an HTTP or HTTPS request to call the API. Before you call the API, you can test the call by using the API calling examples provided in the API Gateway console. The examples are provided in multiple programming languages.

View the authentication information for calling APIs

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Development > Data Service**.
3. On the **Service Development** page, click **Service Management** in the upper-right corner.
4. In the left-side navigation pane, click **Call APIs**.

   On the **Call APIs** page, you can view or copy the authentication information that is required to call APIs. You can also view the information about the apps that are bound to the current workspace. This helps you find the apps in the API Gateway console.



2.1.15.10. 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.

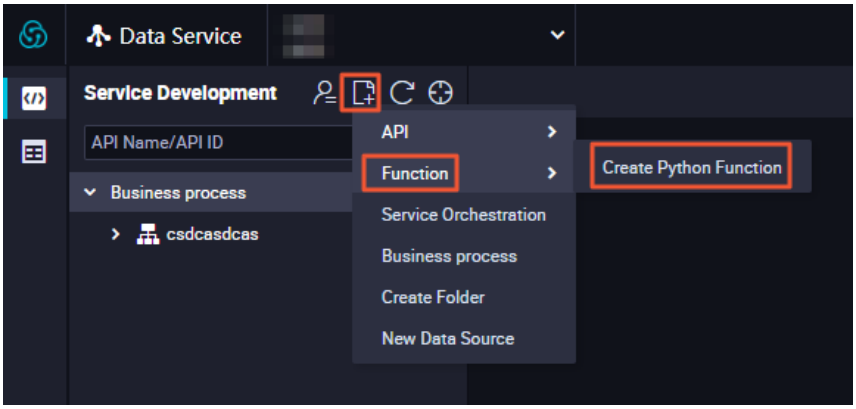
1. Register an API.

   In this example, create an API by using the registration method. For more information, see Register an API.

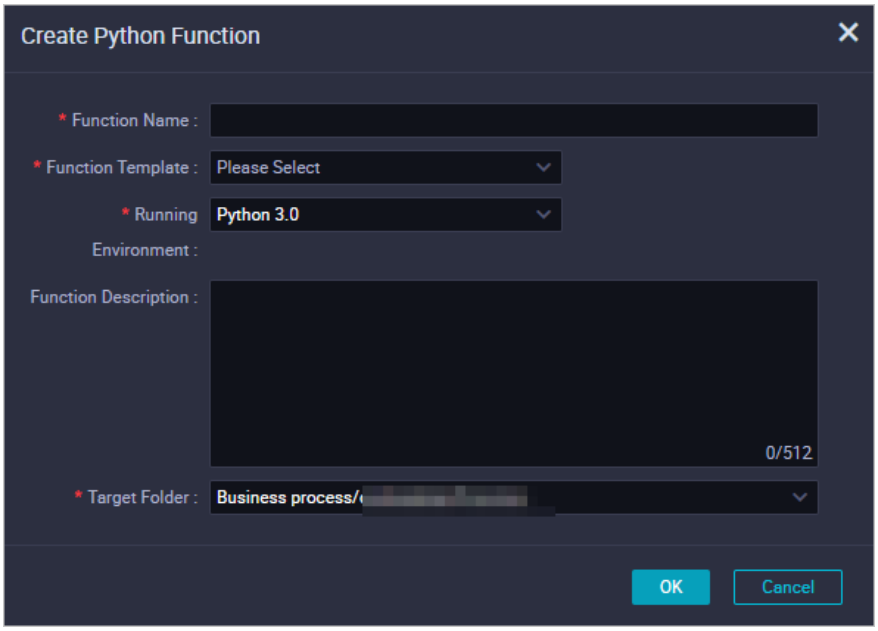
2. Register a function. For more information, see Create a function.

   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
```

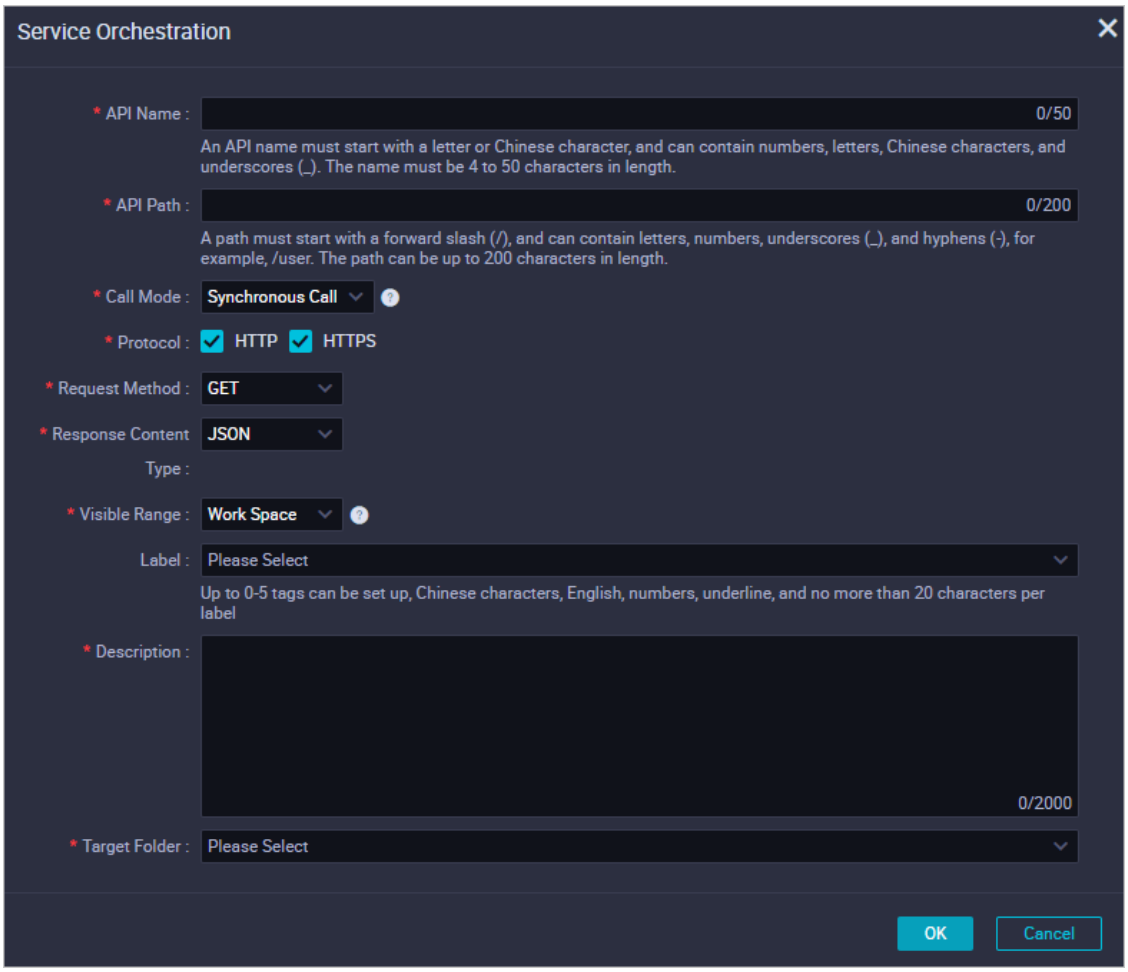
b. In the **Environment Configuration** section, set the **Memory** and **Function Timeout** parameters.

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.

2.1.15.11. 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.
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.

2.1.15.12. 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.

2.1.15.13. Appendix: DataService Studio error codes

After DataService Studio receives an API request, it returns a response that contains an error code. You can locate and troubleshoot issues based on the error code. This topic describes common error codes that are returned by DataService Studio.

Error code	Error message	Description
0	success	The error message returned because the data query has succeeded.
1108110583	query timeout	The error message returned because the query has timed out. The timeout occurs because the total runtime of the API in DataService Studio and the database exceeds the timeout period that is configured for the API.
1108110519	param miss	The error message returned because specific required request parameters are not specified.

Error code	Error message	Description
1108110584	api context failed	The error message returned because the system has failed to obtain context information based on the third party. The information includes the connection information of the data source, AccessKey information of the data source, and tenant information.
1108110622	datasource query error	The error message returned because the system has failed to query the data source. The failure may occur because the SQL syntax is invalid, the data source does not respond within the configured timeout period (10s), or the number of connections to the data source exceeds the upper limit.
1108110703	database connection error	The error message returned because the data source has failed to be connected.
Other error codes	Other error messages	If the response contains an error code other than the preceding error codes, you can consult technical support personnel.

2.1.16. Stream Studio

2.1.16.1. Overview

Stream Studio is a one-stop platform provided by DataWorks for developing real-time computing nodes. This topic describes the features and development process of Stream Studio.

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 when you develop nodes.

Stream Studio has the following features:

- Allows you to develop nodes in DAG mode. You can drag components to configure real-time computing nodes.
- Allows you to develop nodes in SQL mode. You can edit Flink SQL code to configure real-time computing nodes.
- Allows you to switch between the DAG mode and the SQL mode. You can check SQL operators with ease.
- Allows you use Function Studio to create and deploy user-defined functions (UDFs) online in exclusive mode.
- Supports smart diagnosis for real-time computing nodes to facilitate online troubleshooting.

Development and maintenance of real-time computing nodes

To develop and manage a real-time computing node, perform the following steps:

1. Bind a Realtime Compute project

   Before you can properly use Stream Studio, you must activate Realtime Compute, create a project, and then bind the project to a DataWorks workspace.

2. Collect data

   Data must be collected before it can be processed in DataWorks. Realtime Compute is integrated with various streaming storage systems that can store source tables. This frees you from manual data collection.

3. Create a real-time computing node

   After data is collected, you can create a real-time computing node.

4. Develop the real-time computing node

   After you create a real-time computing node, you can develop the node for data analytics in Stream Studio.

5. Manage the real-time computing node

   After you develop and deploy the real-time computing node, you can click **OAM** on the **Stream Studio** page to manage the node. The node O&M feature of Stream Studio is provided by Realtime Compute.

2.1.16.2. Bind a Realtime Compute project

This topic describes how to bind a Realtime Compute project to a DataWorks workspace on 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, click the **Real-time computing** tab in the **Computing Engine information** section.
4. Click **Add engine service**.
5. In the **New Blink engine** dialog box, set the **Enter the Realtime Compute project name to add** parameter.
6. Click **Binding**.

2.1.16.3. 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

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 directed acyclic graph (DAG) mode on the **Components** page.

   The Components page consists of the following four sections:

   - Component list section: In this section, you can view the list of available components. You can click **Components** in the left-side navigation pane 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-compile, 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 the operation on the selected component. Available operations include **Rename**, **View schema**, **Delete node**, **View error message**, **New component group**, and **Copy**.

2.1.16.4. 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 tag` 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
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*.
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.

2.1.16.5. Configure components

2.1.16.5.1. Source tables

2.1.16.5.1.1. 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

Parameter	Description	Remarks
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
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.

2.1.16.5.1.2. 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

Parameter	Description	Remarks
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
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"}]
```

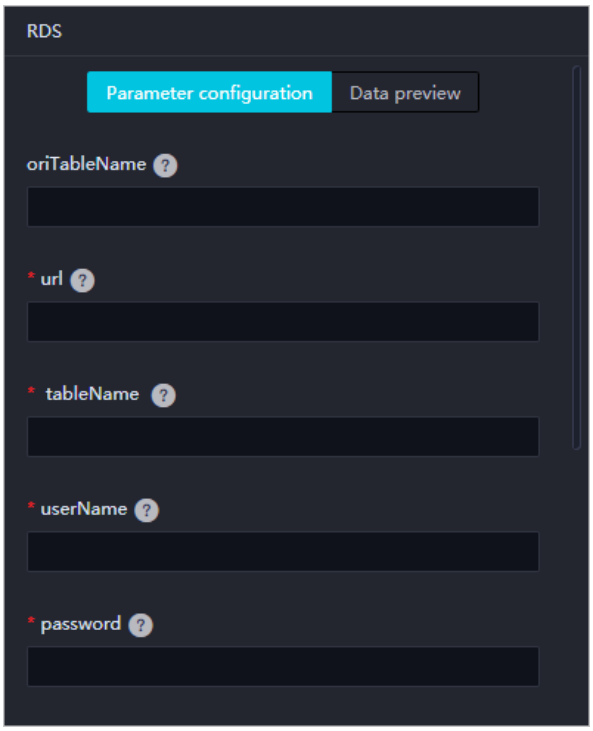
- The batchGetSize value must not exceed 1,000. Otherwise, an error occurs.
- The batchGetSize parameter specifies the number of log items read at a time in a log group. If both the size of a single log item and the batchGetSize value are too large, frequent garbage collection (GC) may be triggered. To avoid this, you must set batchGetSize parameter to a smaller value.

2.1.16.5.2. Dimension tables

2.1.16.5.2.1. ApsaraDB RDS

ApsaraDB RDS is a stable, reliable, and scalable cloud database service.

Parameter configuration



Parameter	Description	Remarks
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.	None

Parameter	Description	Remarks
url	The URL of your ApsaraDB RDS instance.	None
tableName	The name of the table in your database.	None
userName	The username that is used to connect to the database.	None
password	The password that is used to connect to the database.	None
Output Field	The fields that you want to return to the descendant component.	None
maxRetryTimes	The maximum number of retries for reading data from the table.	None
Cache Policy	The policy that is used to cache data.	Valid values: *None*, *LRU*, and *ALL*.
primaryKey	The primary key field in the output fields.	• You must specify a primary key when you declare a dimension table. • When you join a dimension table with another table, the ON clause must contain the equivalent (=) conditions for all the primary key fields. • The primary key in ApsaraDB RDS or PolarDB-X is the primary key or unique index column of an ApsaraDB RDS or PolarDB-X dimension table.

Additional information

- ApsaraDB RDS and PolarDB-X provide the following cache policies:
  - **None**: indicates that data is not cached.
  - **LRU**: indicates that only the recently used data is cached.

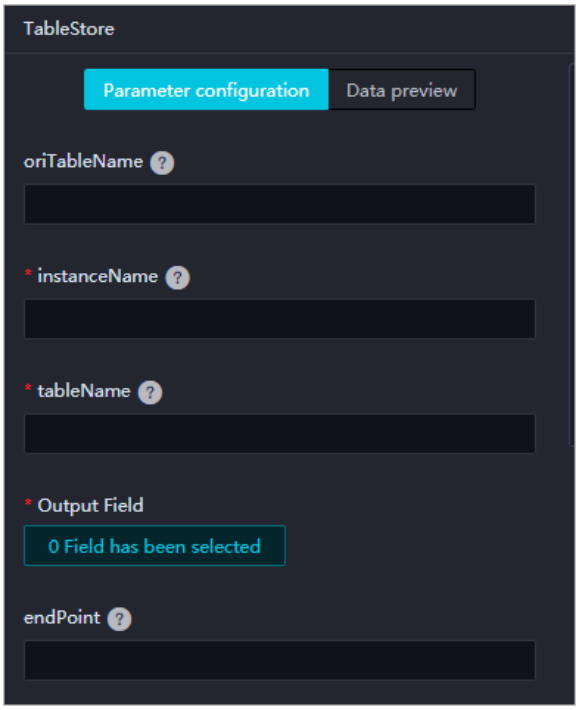
    If this cache policy is selected, you must specify the cacheSize and cacheTTLMs parameters.
  - **ALL**: indicates that all data is cached.

  Before the system runs a node, it loads all the data in the remote table to the memory. Then, the system searches the cache for data in all dimension table queries. If a cache miss occurs, all data is cached again after the cache times out. The ALL cache policy applies to scenarios where the remote table is small but a large number of missing keys exist. If this cache policy is selected, you must specify the cacheTTLMs and cacheReloadTimeBlackList parameters.
- If the ALL cache policy is used, the system 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.
- If the ALL cache policy is used, pay special attention to the memory of the JOIN operator to prevent out of memory (OOM) errors.

2.1.16.5.2.2. Tablestore

Tablestore is a distributed NoSQL database service that is developed based on the Apsara distributed operating system. It features 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 name of the instance.
tableName	The name of the table.
Output Field	The fields that you want to return to the descendant component.
endPoint	The endpoint of the instance. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID of the account that you use to read data from the table.
accessKey	The AccessKey secret of the account that you use to read data from the table.
Cache Policy	The policy used to cache data. Valid values: **None** and **LRU**.
primaryKey	The primary key field in the output fields.

2.1.16.5.2.3. 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
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.

2.1.16.5.3. Data operators

2.1.16.5.3.1. 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'`.

2.1.16.5.3.2. 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.

2.1.16.5.3.3. 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.

2.1.16.5.3.4. 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.

2.1.16.5.3.5. 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.

2.1.16.5.3.6. UnionAll

The UnionAll component corresponds to the UNION ALL clause in SQL statements.

Parameter configuration

No parameter configuration is required.

2.1.16.5.3.7. 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.

2.1.16.5.3.8. 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.
Add column	The fields that store the split data. Specify a key and a sequence number for each field. An alias is allowed.

2.1.16.5.3.9. 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.

Parameter	Description	Remarks
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.

2.1.16.5.4. Result tables

2.1.16.5.4.1. 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.

Parameter	Description
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

2.1.16.5.4.2. 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

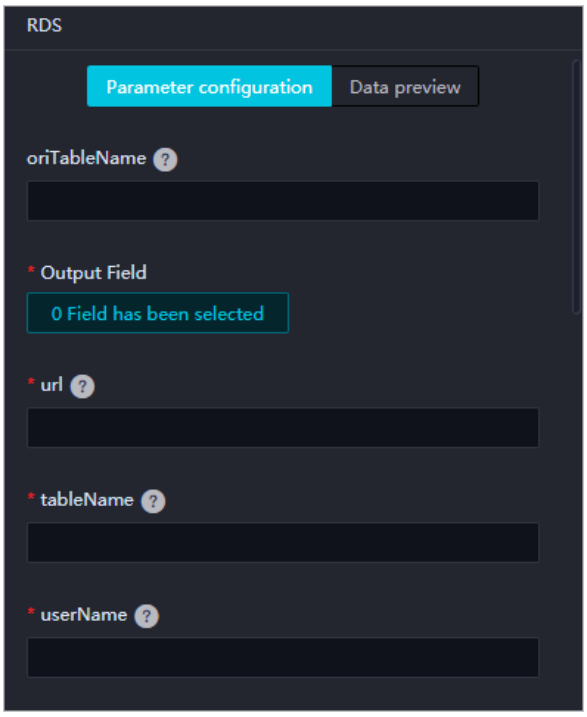
Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.
Output Field	The fields to be returned.
endPoint	The endpoint used to access Log Service. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.
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.

Parameter	Description
accessId	The AccessKey ID used to access Log Service.
accessKey	The AccessKey secret used to access Log Service.
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.

2.1.16.5.4.3. ApsaraDB RDS

ApsaraDB RDS is a stable, reliable, and scalable cloud database service.

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 that you want to write to the table in your ApsaraDB RDS database.
url	The URL used to access your ApsaraDB RDS instance. This parameter corresponds to the url parameter in the WITH clause of the CREATE TABLE statement.

Parameter	Description
tableName	The name of the table to which you want to write data. This parameter corresponds to the tableName parameter in the WITH clause of the CREATE TABLE statement.
userName	The username that is used to access your ApsaraDB RDS database. This parameter corresponds to the userName parameter in the WITH clause of the CREATE TABLE statement.
password	The password that is used to access your ApsaraDB RDS database. This parameter corresponds to the password parameter in the WITH clause of the CREATE TABLE statement.
maxRetryTimes	The maximum number of retries for writing data to the table. This parameter corresponds to the maxRetryTimes parameter in the WITH clause of the CREATE TABLE statement.
batchSize	The number of data records that can be written at a time. This parameter corresponds to the batchSize parameter in the WITH clause of the CREATE TABLE statement.
bufferSize	The maximum number of data records that can be stored in the buffer before deduplication is triggered. This parameter corresponds to the bufferSize parameter in the WITH clause of the CREATE TABLE statement. You can use this parameter only after the primaryKey parameter is specified.
flushIntervalMs	The time interval at which the buffer is cleared. Unit: milliseconds. This parameter corresponds to the flushIntervalMs parameter in the WITH clause of the CREATE TABLE statement.
excludeUpdateColumns	The fields that are not updated when Realtime Compute updates data records with the same primary key value. This parameter corresponds to the excludeUpdateColumns parameter in the WITH clause of the CREATE TABLE statement.
ignoreDelete	Specifies whether to skip delete operations. This parameter corresponds to the ignoreDelete parameter in the WITH clause of the CREATE TABLE statement.
partitionBy	Specifies the partitioning rule for the result table. Before Realtime Compute writes data to the sink node, Realtime Compute performs hash partitioning based on the value of this parameter. The data then flows to the relevant hash node. This parameter corresponds to the partitionBy parameter in the WITH clause of the CREATE TABLE statement.
primaryKey	The primary key field in the output fields.

Data type mapping

ApsaraDB RDS data type	Realtime Compute data type
TEXT	VARCHAR
BYTE	VARCHAR
INTEGER	INT
LONG	BIGINT
DOUBLE	DOUBLE
DATE	VARCHAR
DATETIME	VARCHAR
TIMESTAMP	VARCHAR
TIME	VARCHAR
YEAR	VARCHAR
FLOAT	FLOAT
DECIMAL	DECIMAL
CHAR	VARCHAR

JDBC parameters

Parameter	Description	Default value	Required JDBC version
useUnicode	Specifies whether to use the Unicode character set. This parameter must be set to true if you set the characterEncoding parameter to gb2312 or gbk.	*false*	1.1g
characterEncoding	The character encoding format. This parameter must be set if 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 if the connection to the database is unexpectedly interrupted.	*false*	1.1
autoReconnectForPools	Specifies whether to apply the reconnection policy to a database connection pool.	*false*	3.1.3

Parameter	Description	Default value	Required JDBC version
failOverReadOnly	Specifies whether to set the connection to read-only after the database is automatically reconnected.	*true*	3.0.12
maxReconnects	The maximum number of reconnection attempts allowed. This parameter must be set if the autoReconnect parameter is set to true.	*3*	1.1
initialTimeout	The interval between two reconnection attempts. Unit: seconds. This parameter must be set if the autoReconnect parameter is set to true.	*2*	1.1
connectTimeout	The timeout period when you use a socket connection to access the database server. Unit: milliseconds. Default value: 0. This value indicates that the connection never times out. This parameter applies to JDK 1.4 and later.	*0*	3.0.1
socketTimeout	The timeout period for read and write operations on a socket connection. Unit: milliseconds. Default value: 0. This value indicates that read or write operations never time out.	*0*	3.0.1

FAQ

- Q: When the output data of Realtime Compute is written to an ApsaraDB RDS table, is the result table updated based on the primary key or is a new data record generated in the table?

  A: The processing method depends on whether the primary key is defined in the DDL statement.

  - If a primary key is defined in the DDL statement, the result table is updated by using `insert into on duplicate key update`. For a data record, if the primary key does not exist, the record is inserted into the table as a new row. If the value of the primary key field exists, the original row in the table is updated.
  - If no primary key is defined in the DDL statement, new data records are inserted into the table by using `insert into`.

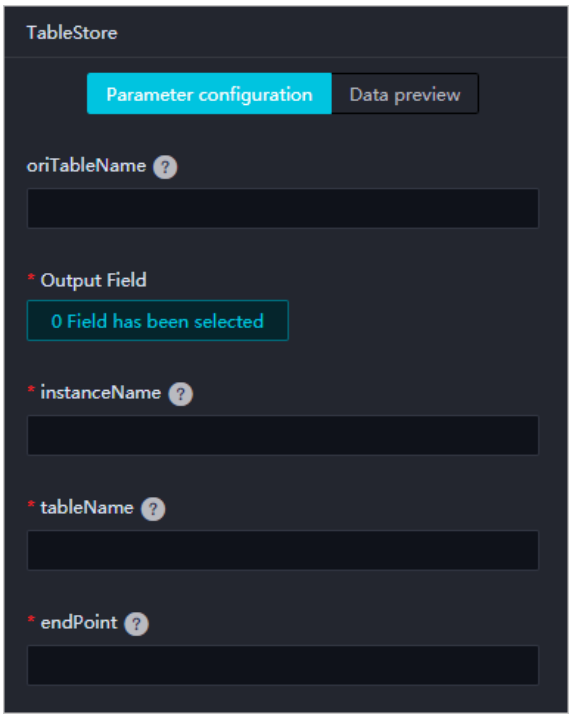
- Q: What do I need to pay attention to when I perform GROUP BY operations based on the unique index of an ApsaraDB RDS table?

A: An ApsaraDB RDS table has only one auto-increment primary key. Therefore, this auto-increment primary key cannot be declared as the primary key in a Realtime Compute job. If you want to perform GROUP BY operations based on the unique index of the table, declare the unique index as the primary key in the job.

2.1.16.5.4.4. TableStore

Tablestore is a NoSQL database service that is built on the Apsara distributed operating system. Tablestore adopts data sharding and load balancing technologies to scale out and handle concurrent transactions. You can use Tablestore to store and access large volumes of structured data in real time.

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 that you want to write to the Tablestore table.
instanceName	The name of the Tablestore instance. This parameter corresponds to the instanceName parameter in the WITH clause of the CREATE TABLE statement.
tableName	The name of the table to which you want to write data. This parameter corresponds to the tableName parameter in the WITH clause of the CREATE TABLE statement.
endPoint	The endpoint of the Tablestore instance. This parameter corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.

Parameter	Description
accessId	The AccessKey ID of the account that is used to access the Tablestore instance. This parameter corresponds to the accessId parameter in the WITH clause of the CREATE TABLE statement.
accessKey	The AccessKey secret of the account that is used to access the Tablestore instance. This parameter corresponds to the accessKey parameter in the WITH clause of the CREATE TABLE statement.
valueColumns	The names of the fields that you want to insert into the result table. Separate multiple names with commas (,).
bufferSize	The maximum number of data records that can be stored in the buffer before deduplication is triggered. This parameter corresponds to the bufferSize parameter in the WITH clause of the CREATE TABLE statement.
batchWriteTimeoutMs	The write timeout period. This parameter corresponds to the batchWriteTimeoutMs parameter in the WITH clause of the CREATE TABLE statement.
maxRetryTimes	The maximum number of retries for writing data to the table. This parameter corresponds to the maxRetryTimes parameter in the WITH clause of the CREATE TABLE statement.
retryIntervalMs	The retry interval. This parameter corresponds to the retryIntervalMs parameter in the WITH clause of the CREATE TABLE statement.
ignoreDelete	Specifies whether to skip delete operations. This parameter corresponds to the ignoreDelete parameter in the WITH clause of the CREATE TABLE statement.
primaryKey	The primary key field in the output fields.

Data type mapping

Tablestore data type	Realtime Compute for Apache Flink data type
integer	bigint
string	varchar
boolean	boolean
double	double

2.1.16.5.4.5. 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.
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

MaxCompute data type	Realtime Compute data type
FLOAT	FLOAT
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.

2.1.16.5.5. 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.

2.1.17. Data Protection

2.1.17.1. 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

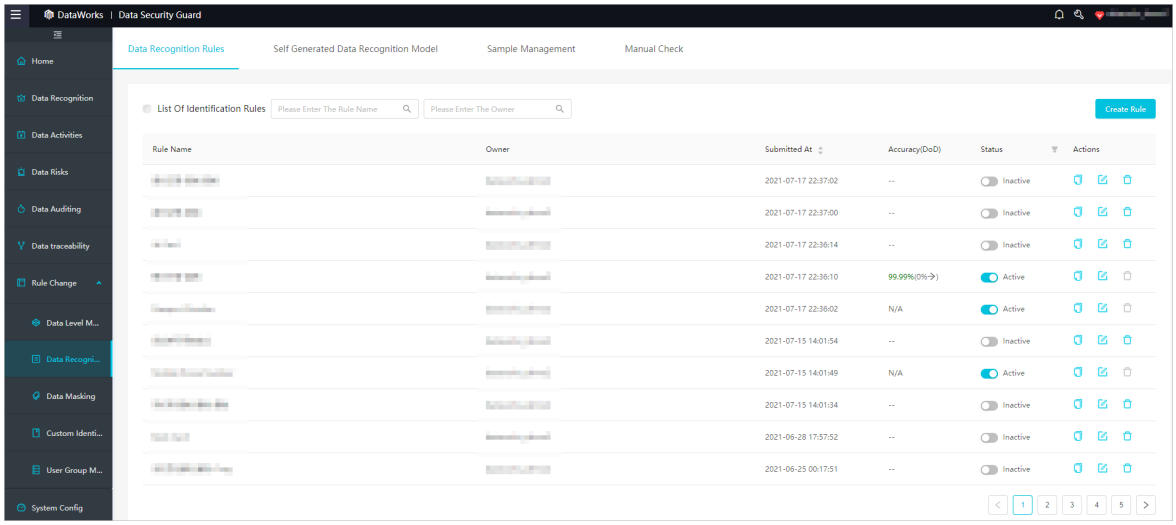
  Data Protection uses various correlation analysis algorithms to detect risky behavior. It also provides alerts and supports visualized auditing for detected risks.

2.1.17.2. Configure rules for defining sensitive data

This topic describes how to configure rules for defining sensitive data.

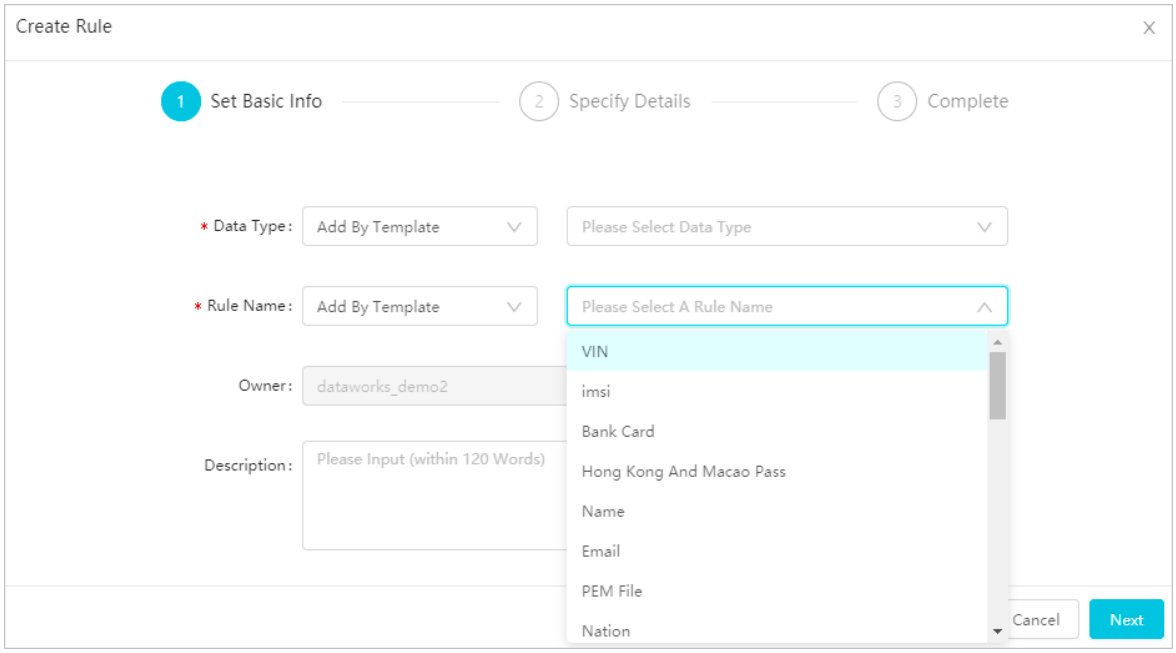
Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > Data Protection**.
3. Click **Try now** to go to the **Data Security Guard** page.
4. In the left-side navigation pane, choose **Rule Change > Data Recognition Rules**. On the Data Recognition Rules tab, click **Create Rule**.



5. In the Create Rule dialog box, configure the parameters in the **Set Basic Info** step.

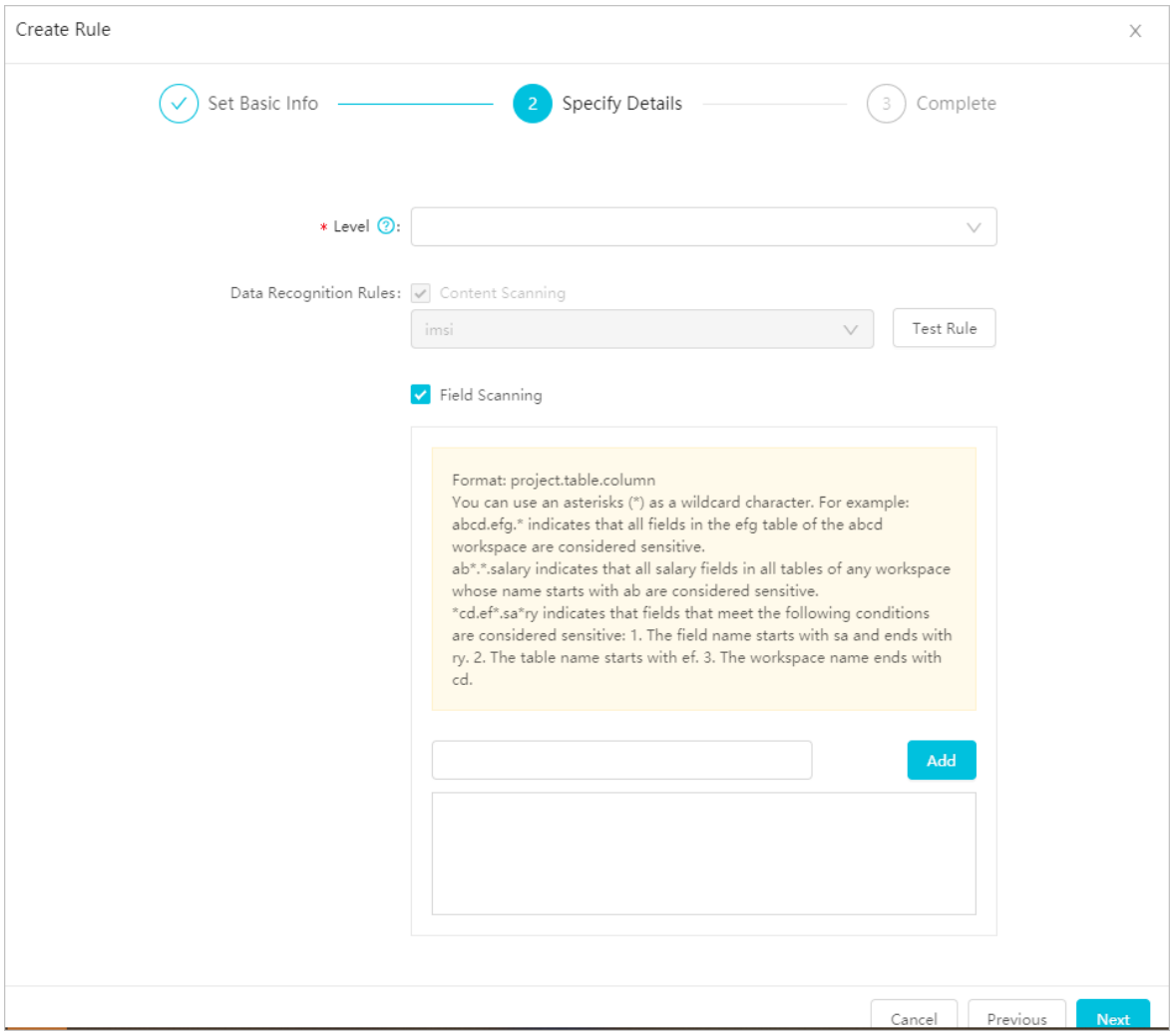
   You can create a template-based rule or a custom rule.



Parameter	Description
Data Type	The category of the rule. You can select Add By Template or Custom from the Data Type drop-down list. ○ If you select **Add By Template**, you can further select **Personal Information**, **Merchant Information**, or **Company Information** from the drop-down list on the right. ○ If you select **Custom**, you can enter a rule category.

Parameter	Description
Rule Name	◦ If you select **Add By Template** from the Rule Name drop-down list, you can select a built-in identification rule template from the drop-down list on the right, such as **Email**, **Seat Number**, **Mobile Phone Number**, **IP**, **Mac Address**, **Car No**, **Post Code**, **Id Card**, or **Bank Card**. ◦ If you select **Custom** from the Rule Name drop-down list, you can enter a rule name.
Owner	The owner of the rule.
Description	The description of the rule.

6. Click **Next**. In the Specify Details step, set the Level and Data Recognition Rules 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 business requirements, choose Rule Change > Data Level Management in the left-side navigation pane to add levels.

Parameter	Description
Content Scanning	Specifies whether to enable content scanning. This option is selected by default for all the built-in sensitive data identification rule templates. ○ If you set Rule Name to Add By Template, you cannot modify the Data Recognition Rules parameter, but you can click Test Rule to verify the accuracy of the identification rule. ○ If you select Regex Express, you can customize the identification rule.
Field Scanning	Specifies whether to enable field scanning. You can use exact match or fuzzy match to specify one or more field names to be identified by the rule. The rule is applied if data matches one of the specified field names.

7. Click **Next**. In the Complete step, confirm the configurations, and click **Save**.

> **Note** When you create a rule to define sensitive data, note the following points:
> ○ The rule name must be unique.
> ○ The content scanning and field scanning configurations 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.

2.1.17.3. 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.

2.1.17.4. 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.

2.1.17.5. 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.

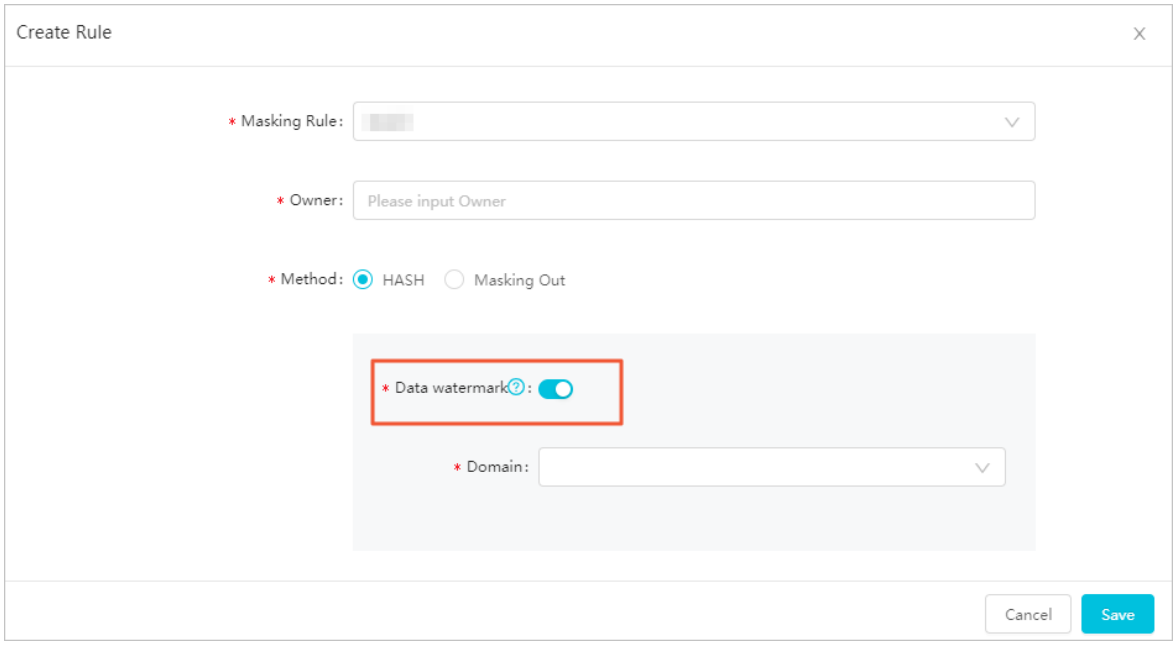
2.1.17.6. Track data

You can create a data tracking task to track the source of leaked data. You can also download a data tracking file and delete a data tracking task.

Configure a data watermark

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > Data Protection**.
3. On the page that appears, click **Try now**.
4. On the Data Protection page, choose **Rule Change > Data Masking** in the left-side navigation pane.
5. On the Data Masking page, click Create Rule to create a data masking rule. For more information, see Customize de-identification rules.

   When you configure a data masking rule, you can turn on **Data watermark**. This way, a data watermark can be embedded into data when data masking is performed on the data, which facilitates the tracking of the leaked data.



> **Note** A data watermark cannot be added to Chinese data.

Create a data tracking task

1. In the left-side navigation pane, click **Data traceability**.
2. On the **Data traceability** page, click **New data tracing task**.
3. Drag the file in which data is leaked to the middle part of the **Traceability tasks** dialog box or click **Upload** to upload the file.

> **Note** You can upload a CSV file whose size is no more than 200 MB.

4. Click **Start tracing** to track the source of the leaked data.

> **Note** After the tracking computing starts, you can close the dialog box, and the computing is not affected.

5. After the computing is complete, click the icon that corresponds to your data tracking task and view the tracking result.

   You can also enter a file name in the search box of the Data traceability page to view the historical data tracking task.

> **Note** Tracking results are for reference only. A data tracking task may not generate results or generate one or more results.

Download a data tracking file

On the **Data traceability** page, find a data tracking file that you want to download in the Traceability file column and click the icon to download the file.

Delete a data tracking task

1. On the **Data traceability** page, find a data tracking task that you want to delete and click the icon.
2. In the message that appears, click **Confirm**.

2.1.17.7. Manage a self-generated data recognition model

This topic describes how to train a model that can be used to summarize the data characteristics of specific columns in a table based on these columns. It also describes how to use the trained model to identify sensitive data.

Go to the Self Generated Data Recognition Model tab

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > Data Protection**.
3. On the page that appears, click **Try now**.
4. On the Data Protection page, choose **Rule Change > Data Recognition Rules** in the left-side navigation pane.
5. Then, click **Self Generated Data Recognition Model**.

Create a self-generated data recognition model

1. On the **Self Generated Data Recognition Model** tab, click **Add Model**.
2. In the **Add Model** dialog box, configure the parameters.

Parameter	Description
Model Name	The name of the model that you want to create. The name can contain letters, digits, and underscores (_).

Parameter	Description
Owner	The name of the owner for the model. The name of the model can contain letters, digits, spaces, and underscores (_).
Selected Samples	You can select one or more columns from an existing MaxCompute table.

3. Click **Next** to go to the **Model Training** step.
4. Select **I accept data umbrella sampling for model training** and click **Start Training**.

   > **Note** The column that you select must contain more than 10 rows. Otherwise, the system cannot start to train the model.

   After the training starts, you can close the dialog box and view the training progress in the recognition model list.
5. View the status of the model in the recognition model list. After the training is complete, the status of the model becomes **Training Completed**. Click the icon in the Actions column. Then, the Assess step of the Edit Model dialog box appears.
6. View the recognition results in the Assess step. In this step, a maximum of 10 results are displayed. You can also adjust the results based on your business requirements. If the accuracy of the recognition can meet your requirements, click **Create**. Then, the model is created. If excessive mismatches exist, you can click **Retrain** to retrain the model after you adjust the recognition results.

   > **Note** In most cases, you must perform two to three training to optimize the model.

Retrain the self-generated data recognition model

1. Click **Retrain** to retrain the model if the recognition effect presented by the recognition results in the Assess step is not satisfactory.
2. Navigate to the Select Samples step. The system automatically places the recognition results in the Assess step into the Sample Field and Exclude Field sections based on the recognition results. Click **Next** to continue the training.
3. View the recognition results after the training process is complete.
4. Click **Create** if the recognition results are satisfactory to finish creating the model. If the recognition results are still not satisfactory, click **Retrain** to continue retraining the model.

Use the self-generated data recognition model

1. After the model is created, click **Go To Data Recognition Rules Online Page**.
2. In the **Create Rule** dialog box, Configure the parameters.

Parameter	Description
Data Type	You can set this parameter to **Custom** or **Add By Template**.
Rule Name	The value cannot be changed.
Owner	The value cannot be changed.

Parameter	Description
Description	The description of the data recognition rule. The description can be a maximum of 120 characters in length and cannot contain special characters.

3. In the **Specify Details** step, specify **Level** and select **Field Scanning**. Select a trained model from the **Data Recognition Rules** drop-down list and click **Next**.
4. In the Complete step, click **Save**.

Stop training a self-generated data recognition model

1. Find a model that is being trained and click the icon in the Actions column to stop training the model.
2. Find the model whose training is stopped and click the icon to continue the training.

Delete a self-generated data recognition model

Find a model that is not being trained and click the **Delete** icon in the Actions column to delete the model.

> **Note**
> - A model that is being trained cannot be deleted. If you want to delete such a model, you can stop training the model and delete it.
> - A model that is in use cannot be deleted. If you want to delete such a model, you can delete the data recognition rules configured for the model and delete the model.

2.1.17.8. 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**.

Operation	Description
Modify the priority of a security level	Find the target security level. Drag and drop the  icon in the Actions column.

2.1.17.9. 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**.

2.1.17.10. 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.

2.1.17.11. 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.

2.1.17.12. Automatically mark security levels for sensitive data

After you turn on Open Marking in DataWorks, DataWorks can identify sensitive data, automatically mark the security level of the sensitive data, and then display the security level as a label for the sensitive data that belongs to a MaxCompute project. The security level of the sensitive data is displayed in Data Map of DataWorks.

Precautions

If you turn on Open Marking in DataWorks, Data Protection automatically configures security levels for the columns that contain sensitive data. This configuration affects your access permissions on sensitive data. Fully evaluate the impact before you turn on Open Marking in DataWorks.

Turn on Open Marking in DataWorks

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > Data Protection**.
3. Click **Try now** to go to the **Data Security Guard** page.
4. In the left-side navigation pane, click **System Config**.
5. On the System Config page, turn on **Open Marking**. **MarkingOpen** is displayed on the right of the switch.

2.1.17.13. Mask the underlying data of a MaxCompute project

This topic describes how to mask the underlying data of a MaxCompute project on the Data Masking page. After the data is masked, the data queried from each of the MaxCompute query entries is masked.

Prerequisites

- The data masking function base_meta.masking_v2 is available. If the function is unavailable, contact O&M personnel to publish the function.
- A network whitelist is enabled and SQL properties are configured for the MaxCompute projects whose underlying data needs to be masked. If no network whitelist is enabled, contact the O&M personnel.
- The rules for identifying sensitive data are created.

Go to the Data Masking page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > Data Protection**.
3. Click **Try now** to go to the **Data Security Guard** page.

4. In the left-side navigation pane, choose **Rule Change > Data Masking**.

Configure the rules for masking the underlying data of a MaxCompute project

1. On the **Data Masking** page, set **Masking Scene** to **MaxCompute Config(maxcompute_desense_code)**.
2. Select a MaxCompute project whose underlying data needs to be masked.
   i. Click **Select Desensitization Project**.
   ii. In the **Authorize The Desensitization Of Account** dialog box, select the name of the MaxCompute project whose underlying data needs to be masked in the **Not Desensitized Project** section, and click the icon to add it to the **Desensitized project** section.
   iii. Select **I agree to authorize data protection umbrella to desensitize the maxcompute underlying layer of the above projects.**
   iv. Click **OK**.
3. In the left-side navigation pane, choose Rule Change > Custom Identification Rules. On the **Rule Settings** tab of the page that appears, you can create, edit, or delete a data masking rule.

> **Note** The data masking rules that are configured for MaxCompute projects take effect only in underlying data masking scenarios, and can be edited or deleted only in such scenarios.

What's next

You can go to the MaxCompute project for which underlying data masking is enabled to check whether the data is masked.

2.1.18. App Studio

2.1.18.1. 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.

2.1.18.2. 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**.
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**

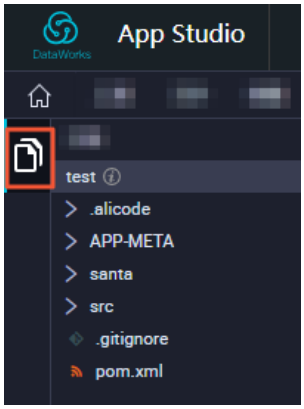
  You can choose **Template > Manage Templates** to go to the **My Templates** page to manage templates.

- Left-side navigation pane
  - Entry

    Click the icon framed in red. The project section appears.

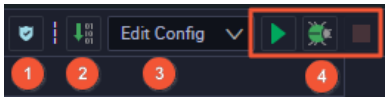
- Edit section

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.

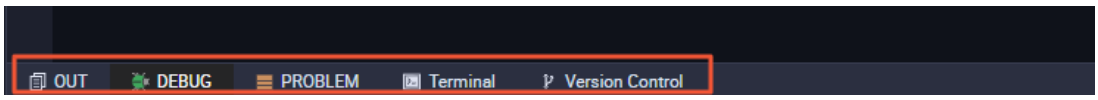
- Icons in the upper-right corner



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.

No.	Feature
4	Operations on the project, including running, debugging, or stopping the project.

- Bottom bar



  - OUT tab

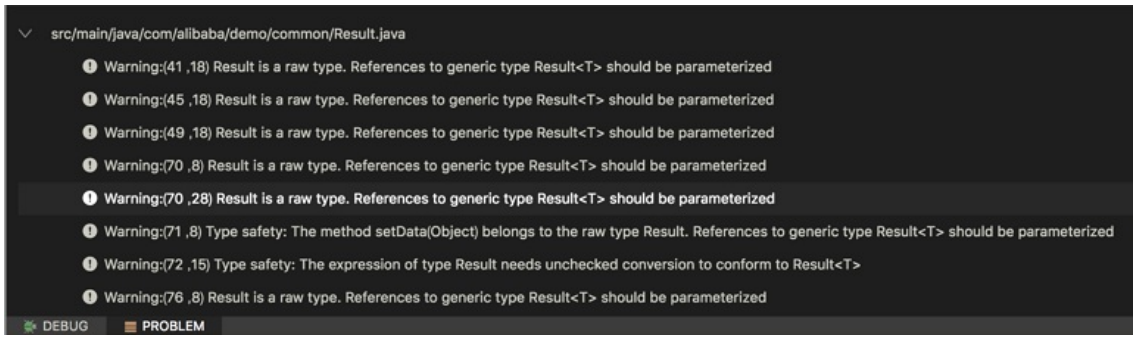
    You can click the OUT tab to view the output.

  - 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.

2.1.18.3. Navigation pane

2.1.18.3.1. 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**.

2.1.18.3.2. 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.

2.1.18.4. Manage projects

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, set the **Name** and **Description** parameters, and select a template.

   > 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.
   > - 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, set the **Name** and **Description** parameters, and select a runtime environment.

   > **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.

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** for 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.

2.1.18.5. Code editing

2.1.18.5.1. 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

Basic feature	Java	Python	JavaScript and TypeScript
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
Generate	Constructor Override Getter and Setter Implement	Not supported	Not supported

2.1.18.5.2. 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.

2.1.18.5.3. 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.

2.1.18.5.4. 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.

2.1.18.6. Debugging

2.1.18.6.1. 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.
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.

2.1.18.6.2. 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 backend Java code.

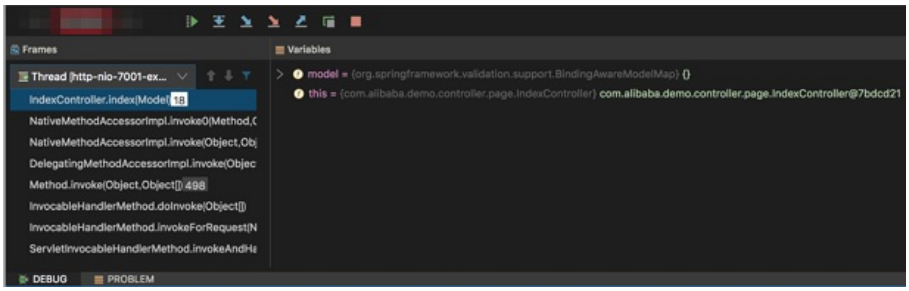
Panel introduction

- 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

The Call Stack panel displays the thread list of your app, stack information, variables of the current stack, and observed variables at the current breakpoint. You can double-click a variable in the list and modify the value of the variable to debug your app.

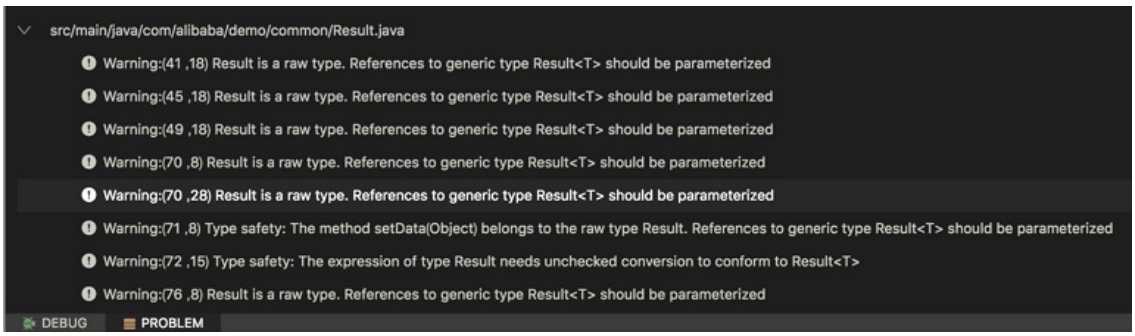


- Breakpoint

  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.



2.1.18.6.3. 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

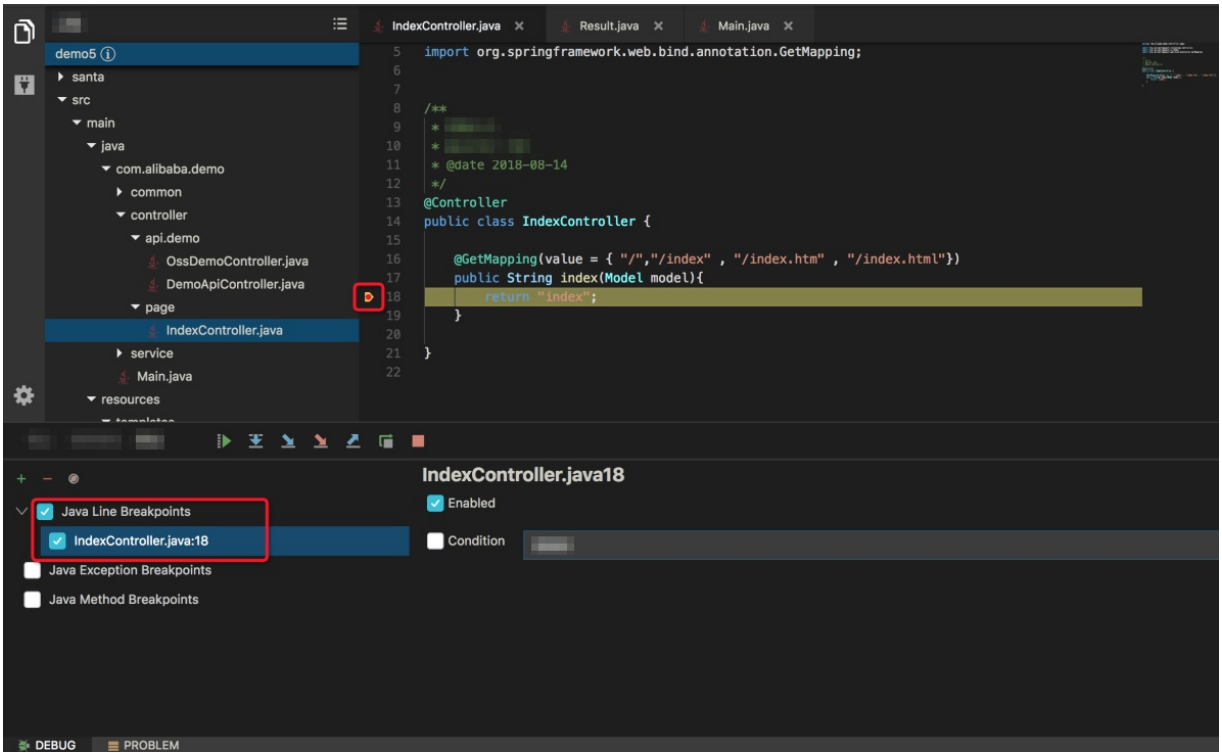
Different from a line breakpoint or an exception breakpoint, a method breakpoint triggers two events, namely, entry and exit. You can manually add a method breakpoint, or set a breakpoint at the place where the method is defined.

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.



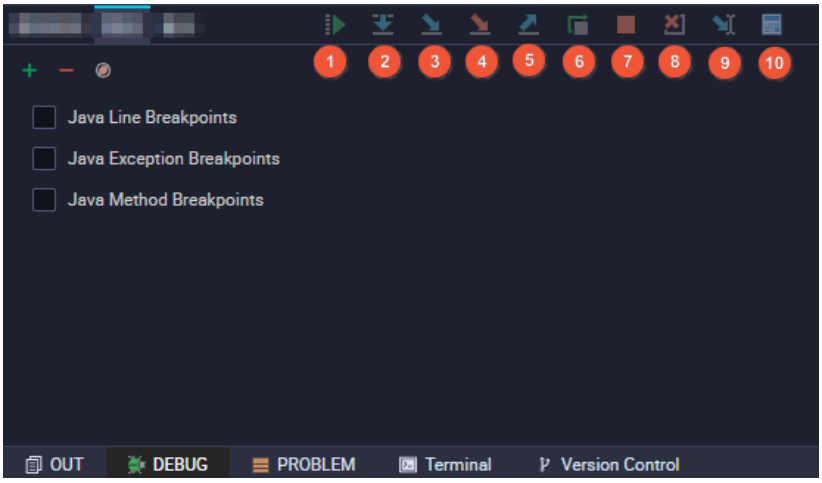
2.1.18.6.4. Breakpoint operations

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

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.

2.1.18.6.5. 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**.

2.1.18.6.6. 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.

2.1.18.7. WYSIWYG designer

2.1.18.7.1. 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.

2.1.18.7.2. 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.

2.1.18.7.3. 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

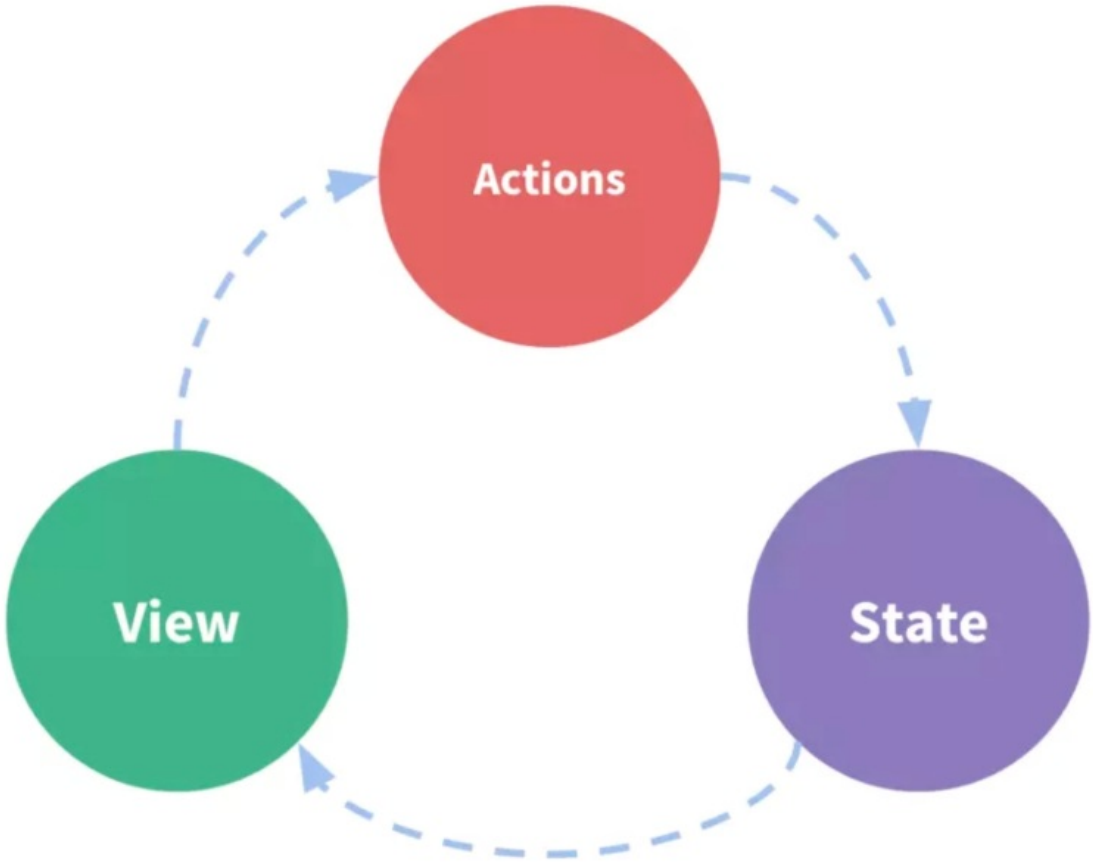
```
{ var a = 1 } // The value assignment statement.
{ aaa * 111; 2} // Multiple statements separated with semicolons (;).
```

2.1.18.7.4. 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.

2.1.18.7.5. 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.

2.1.18.7.6. 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.

2.1.19. Migration Assistant

2.1.19.1. Overview

The Migration Assistant service of DataWorks allows you to migrate data objects across different DataWorks versions, Alibaba Cloud accounts, regions, and workspaces.

Migration Assistant allows you to export data objects in your workspace, including auto triggered nodes, manually triggered nodes, resources, functions, data sources, table metadata, ad hoc queries, and components. You can create full export tasks, incremental export tasks, or custom export tasks to export your data objects in DataWorks based on your business requirements.

> **Notice** To create export or import tasks, you must use an Alibaba Cloud account or be the workspace administrator. If you use a Resource Access Management (RAM) user that is not assigned the administrator role, you can only view export and import tasks.

Scenarios

- Back up node code

  You can use Migration Assistant to periodically back up your node code to prevent data from being deleted by mistake. In this case, we recommend that you create a full export task.

- Export a common workflow for replication

  You can use Migration Assistant to export a common workflow that can be replicated in other workspaces. In this case, we recommend that you create a custom export task.

- Build a test environment

  You can use Migration Assistant to copy all the node code and replace the production data with test data to build a test environment. In this case, we recommend that you create a full export task or a custom export task.

- Develop data in a hybrid cloud environment

  You can use Migration Assistant to migrate node code from Alibaba Cloud public cloud to Alibaba Cloud Apsara Stack to develop data in a hybrid cloud environment. We recommend that you create a custom export task. If the difference between Alibaba Cloud public cloud and Alibaba Cloud Apsara Stack is large, a compatibility problem may occur when data objects are migrated.

- Migrate data objects between the development environment and production environment

  If a workspace consists of the production environment and development environment that are completely isolated, you can use Migration Assistant to export nodes from the development environment and import them to the production environment for deployment.

2.1.19.2. Cloud tasks

2.1.19.2.1. Export tasks from open source engines

DataWorks allows you to migrate tasks from open source scheduling engines such as Oozie and Azkaban to DataWorks. This topic describes the requirements for the files to be exported.

Context

Before you import a task of an open source scheduling engine to DataWorks, you must export the task to your on-premises machine or Object Storage Service (OSS). For more information about the import procedure, see Import tasks of open source engines.

Export a task from Oozie

Requirements and structure of the package to be exported:

- Requirements

  The package must contain XML-formatted definition files and configuration files of a flow task. The package is exported in the ZIP format.

- Structure

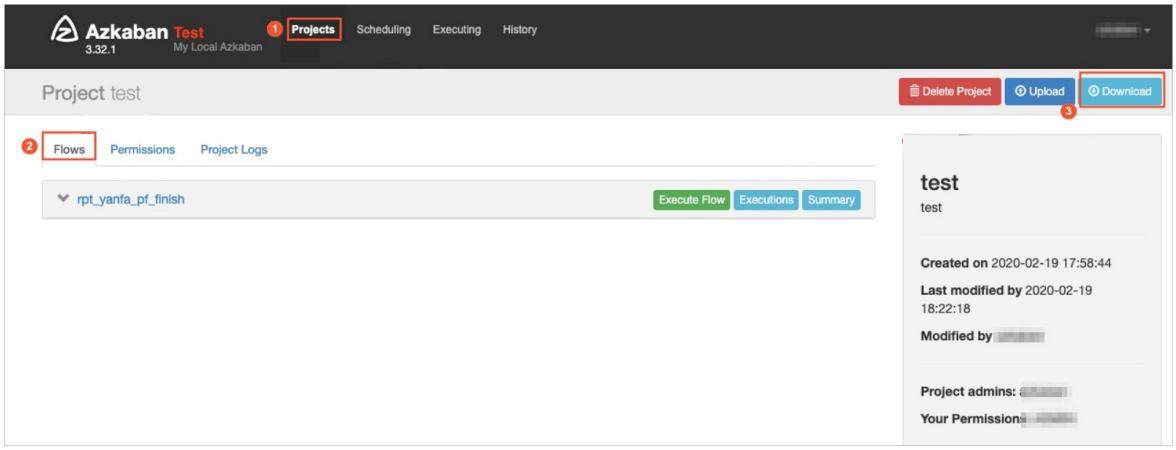
Oozie task descriptions are saved in a Hadoop Distributed File System (HDFS) directory. For example, each subdirectory under the apps directory in the Examples package at the Apache Oozie official website is a flow task of Oozie. Each subdirectory contains XML-formatted definition files and configuration files of a flow task.

```
$tree
.
├── aggregator
│   ├── coordinator-with-offset.xml
│   ├── coordinator.xml
│   ├── job.properties
│   ├── job-with-offset.properties
│   ├── lib
│   │   └── oozie-examples-4.2.0.jar
│   └── workflow.xml
├── sqoop
│   ├── db.hsqldb.properties
│   ├── db.hsqldb.script
│   ├── job.properties
│   └── workflow.xml
├── cron
│   ├── coordinator.xml
│   ├── job.properties
│   └── workflow.xml
├── cron-schedule
│   ├── coordinator.xml
│   ├── job.properties
│   └── workflow.xml
```

Export a task from Azkaban

You can download a specific flow task in the Azkaban console.

1. Log on to the Azkaban console and go to the **Projects** page.
2. Select a project whose package you want to download. On the page for the project, click **Flows** to show all flow tasks under the project.
3. Click **Download** in the upper-right corner of the page to download the package of the project.



Native Azkaban packages can be exported. No limit is imposed on the packages of Azkaban. The exported package in the ZIP format contains information about all tasks and relationships under a specific project of Azkaban.

Export tasks from other open source engines

DataWorks provides a standard template for you to export the tasks of open source engines except for Oozie and Azkaban. Before you run an export task, you must download the standard template and modify the content to be exported based on the file structure in the template. You can go to the **Open Source engine export** page to download the standard template and view the file structure.

1. Log on to the DataWorks console.
2. On the DataStudio page, click the ☰ icon in the upper-left corner and choose **All Products > Other > Migration Assistant**.
3. In the left-side navigation pane, choose **Cloud tasks > Open Source engine export** to go to the **Open Source engine export scheme selection** page.
4. Click the **Standard Template** tab.
5. On the **Standard Template** tab, click **standard format Template** to download the template.
6. Modify the content to be exported based on the template and generate a package to be exported.

2.1.19.2.2. Import tasks of open source engines

This topic describes how to import tasks that are exported from open source engines into DataWorks.

Procedure

1. Go to the **Open Source engine import** 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 > Other > Migration Assistant**.
   iii. In the left-side navigation pane, choose **Cloud tasks > Open Source engine import**.

2. Create an import task.
   i. On the **Import Tasks** page, click **Create Import Task** in the upper-right corner.
   ii. In the **Create Import Task** dialog box, configure the parameters as required.

Parameter	Description
Name	The name of the import task. The name can contain only letters, digits, underscores (_), and periods (.).
Engine type	The engine type. Valid values: **Azkaban**, **Oozie**, and **Standard format**.
Upload From	The source of the package that you want to import. Valid values: **Local** and **OSS**. ■ If you select **Local** for this parameter, perform the following steps to upload a package on your machine: a. Click **Upload File**. b. Select the package that you want to upload and click **Open**. c. Click **Check**. d. After the message **The resource package has passed the check** appears, verify that the file format and content are correct. **Note** The size of the package that you want to upload cannot exceed 30 MB. If the size of the package exceeds 30 MB, select OSS for this parameter. ■ If you select **OSS** for this parameter, enter the endpoint of an Object Storage Service (OSS) object in the **OSS Endpoint** field. Then, click **Check** and **Preview** in sequence to check and preview the package that you want to upload.
Remarks	The description of the import task.

   iii. Click **OK**. The **Edit import task** page appears.
3. Edit the import task.
   i. On the **Edit import task** page, specify **Import objects**.

      Periodic tasks is selected for **Import objects** by default. If you want to import data objects of another type, select the required value from the **Import objects** drop-down list.
   ii. (Optional)Click **Advanced Settings**. In the **Advanced Settings** dialog box, configure the mappings between the compute engine instances and the node types and click OK.

      If multiple compute engine instances are bound to the destination workspace, you must complete the settings in the **Advanced Settings** dialog box. You can configure the mappings between compute engine instances and nodes of the Shell, Hive, and Sqoop types.
   iii. On the Edit import task page, click **start import** in the upper-right corner.
4. View the import report.
   i. In the **Import progress** dialog box, confirm the import task progress.
   ii. After the import task is completed, click **Return to import task list**.
   iii. Find the task on the **Import Tasks** page and click **View Import Report** in the Actions column. On the page that appears, view the task information in the **Basic Information**, **Import Settings**, **Import results**, and **Details** sections.

2.1.19.3. Migrate data objects in DataWorks

2.1.19.3.1. Create and view export tasks

Migration Assistant allows you to export data objects in your workspace, including auto triggered nodes, manually triggered nodes, resources, functions, table metadata, data sources, components, and ad hoc queries. This topic describes how to create and view export tasks.

Prerequisites

To create export or import tasks, you must use an Alibaba Cloud account or be the workspace administrator. If you use a Resource Access Management (RAM) user that is not assigned the administrator role, you can only view export and import tasks.

Context

Migration Assistant allows you to export data objects in different modes. These modes include full export, incremental export, and custom export. You can choose an export mode that best suits your business scenario.

- Full export tasks are used to export all the data objects in a workspace. For example, you can run a full export task to back up node code or clone the workspace to a test environment. When you run a full export task, data objects of the latest version are exported.

  Only saved data objects can be exported. If a node is saved in both the development environment and production environment, the node saved in the development environment is exported.

- Incremental export tasks are used to export data objects that were modified after the specified date.

  > **Note** You cannot configure a blacklist for incremental export tasks.

- Custom export tasks are used to export data objects that you specify. For example, you can run a custom export task to extract a common workflow and clone it to other workspaces. If a workspace runs in both a production environment and a development environment that are completely isolated from each other, you can run a custom export task to export nodes from the development environment and import them to the production environment for deployment.

Go to the Migration Assistant page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Other > Migration Assistant**. The **DataWorks export** page under **DataWorks migration** appears.

Create a full export task

1. On the **Export Tasks** page, click **Create Export Task** in the upper-right corner.
2. In the **Create Export Task** dialog box, configure the parameters as required.

Parameter	Description
Name	The name of the export task. The name can contain only letters, digits, underscores (_), and periods (.).
Type	The type of the export task. Select **Full export** for this parameter. A full export task is used to export all the auto triggered nodes, manually triggered nodes, table metadata, and data sources that have been saved or committed in the current workspace.
Blacklist	Specifies whether to enable the blacklist feature for full export based on your business needs. If you select the **Add to Blacklist** check box, you can add the nodes and resources that do not need to be exported to a blacklist.

Parameter	Description
Export Version	Valid values: **Standard**, **Private Cloud(>=V3.12)**, and **Private Cloud(V3.6.1-V3.1.1)**. The DataWorks version determines the format in which data objects are exported. Check the DataWorks version of the destination workspace before you create an export task.
Remarks	The description of the export task.

3. (Optional)Click Add to Blacklist and run the export task.

   If you select **Add to Blacklist**, perform the following steps to configure the blacklist:

   i. In the **Create Export Task** dialog box, click **Add to Blacklist**.
   ii. On the **Set Blacklist** page, select the data objects that you do not want to export.
   iii. Click **Add Selected to Blacklist**.
   iv. Click **Export** in the upper-right corner.
   v. In the **Export confirmation** message, click **Confirm**.
4. (Optional)If you do not select **Add to Blacklist**, click **Export** in the Create Export Task dialog box.
5. In the **Export Progress** dialog box, view the progress of the export task. After the task succeeds, click **Back to Export Tasks**.

Create an incremental export task

1. On the **Export Tasks** page, click **Create Export Task** in the upper-right corner.
2. In the **Create Export Task** dialog box, configure the parameters as required.

Parameter	Description
Name	The name of the export task. The name can contain only letters, digits, underscores (_), and periods (.).
Type	The type of the export task. Select **Incremental** for this parameter. An incremental export task is used to export data objects that were modified after the specified date. Supported data objects include auto triggered nodes, manually triggered nodes, table metadata, and data sources that have been saved or committed in the current workspace.
Start Date	The date on which data is modified. The data generated after this date is incremental data.
Export Version	Valid values: **Standard**, **Private Cloud(>=V3.12)**, and **Private Cloud(V3.6.1-V3.1.1)**.
Remarks	The description of the export task.

3. Click **Export**.

Create a custom export task

1. On the **Export Tasks** page, click **Create Export Task** in the upper-right corner.
2. In the **Create Export Task** dialog box, configure the parameters as required.

Parameter	Description

Parameter	Description
Name	The name of the export task. The name can contain only letters, digits, underscores (_), and periods (.).
Type	The type of the export task. Select **Custom** for this parameter. A custom export task is used to export data objects that you specify. Supported data objects include auto triggered nodes, manually triggered nodes, table metadata, and data sources that have been saved or committed in the current workspace.
Export Version	Valid values: **Standard**, **Private Cloud(>=V3.12)**, and **Private Cloud(V3.6.1-V3.1.1)**.
Remarks	The description of the export task.

3. Click **Select Export Objects**.
4. On the **Export Objects** page, select a type of data object that you want to export from the **Export Object** drop-down list.

   Valid values of **Export Object**: **Table**, **Periodic tasks**, **Resources**, **Manual tasks**, **Function**, **DATA_SERVICE**, **Data source**, **Components**, and **Temporary query**.
5. Select the data objects that you want to export and click **Add Selected to Export Package**.

   You can also configure filter conditions such as **Export Object**, **Object Type**, and **Export Environment** to search for data objects. Then, click **Add All to Export Package** to add all the data objects that have been found to the package that you want to export.
6. Click **Export** in the upper-right corner.

View and manage export tasks

On the **Export Tasks** page, you can view the name, type, creator, status, update time, and description of created export tasks. The operations that you can perform on export tasks vary based on their status.

- If an export task is in the **Successful** state, you can perform the following operations on the task:
  - Click **View Export Report** in the Actions column. On the page that appears, view the task information in the **Basic Information**, **Overview**, and **Details** sections.
  - Click **Download** in the upper-right corner to download the package of the export task to a local directory.
  - Clone the export task.
    - Full export task for which the blacklist feature is not enabled: Click **Clone** in the **Actions** column. In the **Clone** dialog box, specify **Name** and click **Export**.
    - Full export task for which the blacklist feature is enabled: Click **Clone** in the **Actions** column. In the **Clone** dialog box, specify **Name** and click **Add to Blacklist**.

      On the **Set Blacklist** page, select the data objects that you do not want to export, click **Add Selected to Blacklist**, and then click **Export** in the upper-right corner.
    - Custom export task: Click **Clone** in the **Actions** column. In the **Clone** dialog box, specify **Name** and click **Select Export Objects**.

      On the **Export Objects** page, select the data objects that you want to export, click **Add Selected to Export Package**, and then click **Export** in the upper-right corner.
- If an export task is in the **Export failed** state, you can click **View Export Package**, **Download Export Package**, or **Re-export** in the Actions column as required. To retry the export task, click **Re-export**.
- If an export task is a custom export task that is in the **Editing** state, you can perform the following operations on the task:
  - Click **Edit** in the Actions column. On the Export Objects page, modify the data objects that you want to export.

- Click **View Export Package** in the Actions column. On the **Export Package Details** page, view the task information in the **Basic Information**, **Overview**, and **Details** sections.
- Click **Delete** in the Actions column. In the **Delete** message, click **Ok** to delete the task.

- If an export task is a full export task that is in the **Editing** state, you can click **Edit**, **Delete**, or **View Blacklist** in the Actions column as required. If you click View Blacklist in the Actions column, you can check the blacklist and click **Export** to run the export task or click **Close** in the dialog box that appears.

2.1.19.3.2. Create and view import tasks

After you run an export task to export data objects from a workspace, you can create an import task to import these data objects to a specified workspace.

Prerequisites

To create export or import tasks, you must use an Alibaba Cloud account or be the workspace administrator. If you use a Resource Access Management (RAM) user that is not assigned the administrator role, you can only view export and import tasks.

Go to the Migration Assistant page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the ≡ icon in the upper-left corner and choose **All Products > Other > Migration Assistant**. The **DataWorks export** page under **DataWorks migration** appears.

Create an import task

1. In the left-side navigation pane of Migration Assistant, choose **DataWorks migration > DataWorks import**.
2. On the **Import Tasks** page, click **Create Import Task** in the upper-right corner.
3. In the **Create Import Task** dialog box, configure the parameters as required.

Parameter	Description
Name	The name of the import task. The name can contain only letters, digits, underscores (_), and periods (.).
Upload From	The source of the package that you want to import. Valid values: **Local** and **OSS**. ○ If you select **Local** for this parameter, perform the following steps to upload a package on your machine: a. Click **Upload File**. b. Select the package that you want to upload and click **Open**. c. Click **Check**. d. After the message **The resource package has passed the check** appears, click **Preview**. On the page that appears, check the package that you want to import. **Note** The size of the package that you want to upload cannot exceed 30 MB. ○ If you select **OSS** for this parameter, enter the endpoint of an Object Storage Service (OSS) object in the **OSS Endpoint** field. Then, click **Check** and **Preview** in sequence to check and preview the package that you want to upload.

Parameter	Description
Remarks	The description of the import task.

4. Click **OK**. The **Import Task Settings** page appears.

   Make sure that you have checked the format and content of the package before you click **OK**.

5. Configure the import task.

   When you configure the import task, you must complete the settings in the **Engine Instance Mapping** section. The settings in other sections are optional and can be configured as required.

   (Optional)

   i. In the **Engine Instance Mapping** section, select a compute engine of the destination workspace for each compute engine that is bound to the source workspace.

   If multiple compute engines are bound to the source workspace and only one compute engine is bound to the destination workspace, the import task fails. This is because some types of nodes cannot be created in the destination workspace due to the absence of the required compute engines.

   ii. (Optional)In the **Resource Group Mapping** section, configure resource group mapping between the source and destination workspaces. This ensures that resource groups are available for running imported nodes.

   iii. (Optional)In the **Dependency Mapping** section, configure workspace mapping for relevant nodes.

   Some nodes use the name of the source workspace in their code. In this case, you must configure workspace mapping for the nodes to run properly after they are imported. Set the **New Workspace** parameter to the name of the destination workspace. The system uses this workspace name to replace the original workspace name in the node code and the names of the ancestor and descendant nodes of the current node. After the import task is complete, the original workspace name is replaced with the new workspace name.

   iv. (Optional)In the **Dry-run** section, find the destination node that you want to set as a dry-run node and click **Set up empty run** in the Actions column.

   You can also select multiple nodes and click **Batch Configure** to set these nodes as dry-run nodes.

   This configuration is used to configure the scheduling mode of auto triggered nodes. The auto triggered node that is set as a dry-run node returns a success response without running and does not generate data.

   v. (Optional)In the **Commission Rules** section, configure the commission rules for **Resources**, **Tables**, and **Functions**, and specify whether to enable **Change Owner** as required.

   > Note
   >
   > - If a data object with the same name as the data object you want to import exists in the destination workspace, the imported data object cannot be committed.
   > - If you select No for the Change Owner parameter and no owner is specified for the node you want to import, you are automatically configured as the owner of the node after it is imported.

6. Click **Import** in the upper-right corner.
7. In the **Confirm** message, click **OK**.

View and manage import tasks

On the **Import Tasks** page, the operations that you can perform on import tasks vary based on their status.

- After an import task is complete, you can view the details of the task. To view the task details, find the task on the **Import Tasks** page and click **View Import Report** in the Actions column. On the page that appears, view

the task information in the **Basic Information**, **Import Settings**, **Import results**, and **Details** sections.

- If an import task is in the **Editing** state, you can perform the following operations on the task:
  - Click **Edit** in the Actions column. On the **Import Task Settings** page, modify the task configurations.
  - Click **Preview** in the Actions column. On the page that appears, view the task information in the **Basic Information**, **Overview**, and **Details** sections.
  - Click **Delete** in the Actions column. In the message that appears, click **Ok** to delete the task.
- If an import task is in the **Import failed** state, you can click **Re-import** in the Actions column of the task. In the **Import progress** dialog box, click **Return to import task list** after the import task is complete.

2.1.20. Workspace management

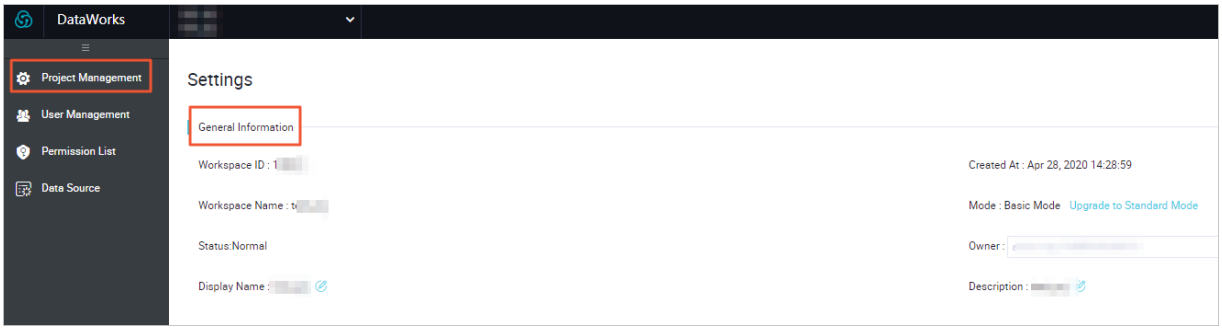
2.1.20.1. Configure a workspace

On the Workspace Management page of a workspace, you can manage and configure the workspace. DataWorks supports a variety of compute engines, such as MaxCompute, E-MapReduce (EMR), Realtime Compute for Apache Flink, Hologres, Graph Compute, AnalyticDB for PostgreSQL, and AnalyticDB for MySQL.

Go to the Workspace Management page

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click the icon in the upper-right corner.
3. On the **Project Management** page, set the parameters in the **Basic properties**, **Scheduling Properties**, **Security Settings**, and **Compute Engine Information** sections for the workspace based on your business requirements.

Configure basic properties

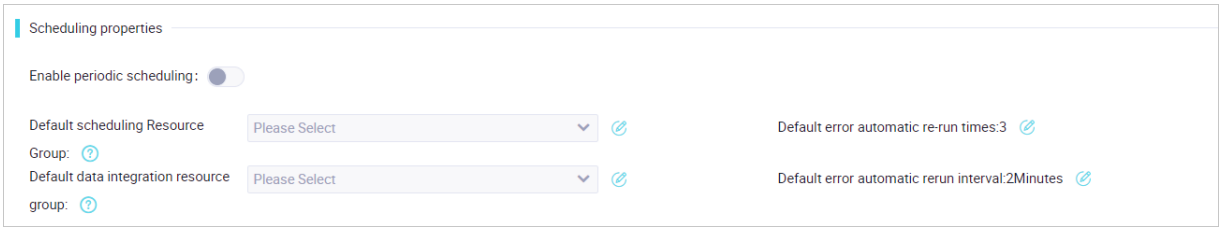


Parameter	Description
Workspace ID	The ID of the workspace, which cannot be changed.
Workspace Name	The name of the workspace. The name must start with a letter and can contain only letters and digits. It is not case-sensitive. The name uniquely identifies the workspace and cannot be changed after the workspace is created.
Status	The status of the workspace.
Display Name	The display name that is used to identify the workspace. The display name can contain only letters and digits. You can change it based on your requirements.
Creation Time	The time when the workspace was created. The value cannot be changed.

Parameter	Description
Mode	The mode of the workspace, which cannot be changed. Valid values: **Simple Mode** and **Standard**.
Owner	The owner of the workspace, which cannot be changed.
Description	The description of the workspace. You can modify the description based on your requirements. The description can be a maximum of 128 characters in length and can contain letters, special characters, and digits.

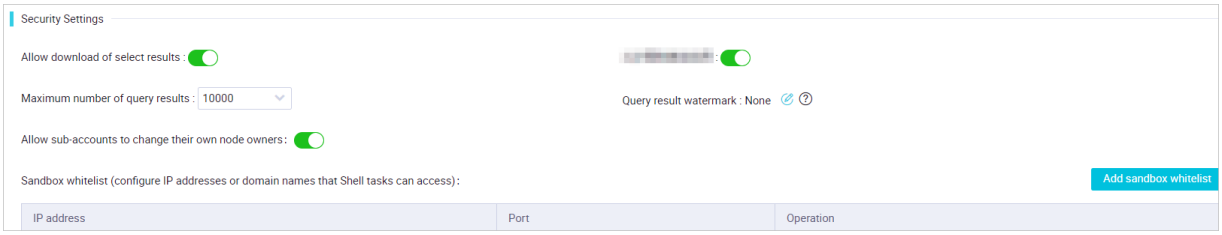
Configure scheduling properties

In the **Scheduling Properties** section, you can enable periodic scheduling for the workspace. You can also set the **Default Scheduling Resource Group**, **Default Data Integration Resource Group**, **Default Automatic Rerun Times Upon Error**, and **Default Automatic Rerun Interval Upon Error** parameters for the workspace.



Nodes can be periodically run in a workspace only after you turn on **Periodic Scheduling** for the workspace.

Configure security settings



Parameter	Description
Allow download of select results	Specifies whether the query results that are returned by SELECT statements in DataStudio can be downloaded. If you turn off this switch, the query results cannot be downloaded.
Copy Query Result	Specifies whether the query results that are returned in DataStudio can be copied.
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**. For example, 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**. Create an SQL node and execute a query statement on the node to query data in a table that has more than 1,000 data records. The number of returned records is 1,000.
Watermark for Query Results	Specifies whether watermarks for the query results are displayed.
Change Node Owner by RAM User	Specifies whether to allow RAM users to change the owners of their nodes.

Parameter	Description
Sandbox Whitelist (contains IP addresses and domain names that can be accessed by Shell nodes)	The IP addresses or domain names that can be accessed by a Shell node that runs on the default resource group.

To add an IP address or domain name to the whitelist, perform the following steps:

1. In the **Security Settings** section, click **Add**.
2. In the **Add** dialog box, enter an IP address or a domain name in the **Address** field and a port number in the **Port** field.
3. Click **Confirm**.

Associate a MaxCompute project with a workspace

In the **Computing Engine Information** section, click the **MaxCompute** tab. On this tab, you can view the settings of the **MaxCompute Project Name** and **MaxCompute Visitor Identity** parameters for an associated MaxCompute compute engine instance.

Associate an EMR cluster with a workspace

Note If Kerberos authentication is enabled for an E-MapReduce (EMR) cluster., you cannot create tables, resources, and functions in a visualized manner for this cluster.

1. In the **Compute Engine Information** section, click the **E-MapReduce** tab. On this tab, you can view the information about all the available EMR clusters in the workspace.
2. Click **Add instance**.
3. In the **New EMR cluster** dialog box, set the parameters as required.

Parameter	Description
Instance Display Name	The display name of the EMR cluster to be associated.
Region	The region of the current workspace.
Cluster ID	The ID of the user who created the EMR cluster.
Project ID	The ID of the project in the EMR cluster.
YARN resource queue	The name of the resource queue in the EMR cluster. Unless otherwise specified, set this parameter to *default*.
Endpoint	The endpoint of the EMR cluster. You can obtain the endpoint in the EMR console.

4. Click **Confirm**.

Associate a Realtime Compute for Apache Flink project with a workspace

1. In the **Compute Engine Information** section, click the **Real-time Computing** tab. On this tab, you can view the information about all the available Realtime Compute for Apache Flink compute engine instances in the workspace.
2. Click **Add Instance**.
3. In the **Add a real-time computing instance** dialog box, set the parameters as required.

Parameter	Description
Instance Display Name	The display name of the Realtime Compute for Apache Flink compute engine instance.
Select Project	The Realtime Compute for Apache Flink project that you want to associate with the workspace. Select a project from the drop-down list. If you need to create a project, click **Real-time calculation control platform**.

4. Click **Confirm**.

Associate a Graph Compute compute engine instance with a workspace

1. In the **Compute Engine Information** section, click the **GraphCompute** tab.
2. Click **Bind Graph Compute Instance**.

> **Notice** A Graph Compute instance can be associated with only one DataWorks workspace. After a Graph Compute instance is associated with a DataWorks workspace, the instance cannot be used in other DataWorks workspaces.

3. In the **Bind Graph Compute Instance** dialog box, configure the parameters as required.

Parameter	Description
Instance Display Name	The display name of the Graph Compute instance.
Graph Compute Instance Name	The name of the Graph Compute instance that you want to associate with the workspace.

4. Click **Bind**.

Associate a Hologres compute engine instance with a workspace

1. In the **Compute Engine Information** section, click the **Hologres** tab. On this tab, you can view the information about all the available Hologres compute engine instances in the workspace.
2. Click **Bind Hologres Database**.
3. In the **Bind Hologres Database** dialog box, configure the parameters.

Parameter	Description
Instance Display Name	The display name of the Hologres compute engine instance.
Access identity	The identity that is used to run the code of committed Hologres nodes. Valid values: **Alibaba Cloud primary account** and **Alibaba Cloud sub-account**.
Hologres instance name	The name of the Hologres instance that you want to associate with the workspace.
Database name	The name of the database that was created in **SQL Console**, such as testdb.
Server	The endpoint of the purchased Hologres instance. This value is automatically generated after you select the Hologres instance.
Port	The port of the purchased Hologres instance. This value is automatically generated after you select the Hologres instance.

4. Click **Test Connectivity**.

5. After the connectivity test is passed, click **Confirm**.

Associate an AnalyticDB for PostgreSQL instance with a workspace

1. In the **Computing Engine information** section, click the **AnalyticDB for PostgreSQL** tab.
2. Click **Add Instance**.
3. In the **Add AnalyticDB for PostgreSQL Instance** dialog box, configure the parameters as required.

Parameter	Description
Instance Display Name	The display name of the AnalyticDB for PostgreSQL instance. The display name must be unique.
InstanceName	The name of the AnalyticDB for PostgreSQL instance that you want to associate with the workspace.
DatabaseName	The name of the AnalyticDB for PostgreSQL database that you want to associate with the workspace.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Connectivity Test	AnalyticDB for PostgreSQL nodes must be run on exclusive resource groups. Therefore, you must specify an exclusive resource group. Click **Test connectivity** to test the connectivity between the specified exclusive resource group and AnalyticDB for PostgreSQL instance.

4. After the connectivity test is passed, click **Confirm**.

Associate an AnalyticDB for MySQL instance with a workspace

1. In the **Compute Engine Information** section, click the **AnalyticDB for MySQL** tab.
2. Click **Add Instance**.
3. In the **Add an AnalyticDB for MySQL instance** dialog box, configure the parameters.

Parameter	Description
Instance Display Name	The display name of the AnalyticDB for MySQL cluster. The display name must be unique.
InstanceName	The name of the AnalyticDB for MySQL cluster that you want to associate with the workspace.
DatabaseName	The name of the AnalyticDB for MySQL database that you want to associate with the workspace.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Connectivity Test	AnalyticDB for MySQL nodes must be run on exclusive resource groups. Therefore, you must specify an exclusive resource group. Click **Test connectivity** to test the connectivity between the specified exclusive resource group and the AnalyticDB for MySQL cluster.

4. After the connectivity test is passed, click **Confirm**.

Associate a CDH cluster with a workspace

1. In the **Compute Engine Information** section, click the **CDH** tab.
2. Click **Add Instance**.

   For a workspace in standard mode, the development environment is isolated from the production environment. If you are using a workspace in standard mode, you must add instances to both the development and production environments.
3. In the **Add CDH Compute Engine** dialog box, configure the parameters.

   You can set Access Mode to **Shortcut mode** or **Security mode**. If **Security mode** is selected, the permissions on the data of the node that is run by different Apsara Stack tenant accounts or RAM users can be isolated.

   Configure the following parameters:

Parameter	Description
Instance Display Name	The display name of the compute engine instance. The display name must be unique.
Access Mode	◦ If **Shortcut mode** is used, multiple Apsara Stack tenant accounts or RAM users map to the same CDH cluster account. These Apsara Stack tenant accounts or RAM users can access data in the same CDH cluster account. In this case, data permissions are not isolated. ◦ If **Security mode** is used, you can configure the mappings between the Apsara Stack tenant accounts or RAM users and CDH cluster accounts to isolate the permissions on the data of the node that is run by the Apsara Stack tenant accounts or RAM users.
Select Cluster	◦ If **Shortcut mode** is selected for **Access Mode**, you must select a CDH cluster whose Authentication Type is set to Kerberos Account Authentication. If you do not have a CDH cluster, create one. ◦ If **Security mode** is selected for **Access Mode**, you must select a CDH cluster whose Authentication Type is set to Kerberos Account Authentication. You can check whether Kerberos Account Authentication is enabled for the CDH cluster in the DataWorks console. On the **Workspace Management** page, click **Hadoop Config** in the left navigation pane and find the cluster of which you want to view the configuration. Then, click **Modify** to view the **Authentication Type** parameter in the **Mapping Configuration** section. If you do not have a CDH cluster, create one.

Parameter	Description
Access identity	◦ If **Shortcut mode** is selected for Access Mode, the Authentication Type parameter is set to No Authentication by default. You can use only the specified accounts, such as admin and hadoop. These accounts are used only to commit nodes. ◦ **Security mode** ▪ You can set **Account for Scheduling Nodes** based on your business requirements. This account is used to automatically schedule and run the node after the node is committed to the scheduling system. You need to configure the mappings between the Apsara Stack tenant accounts or RAM users and CDH cluster accounts. Valid values: **Task owner**, **Alibaba Cloud primary account**, and **Alibaba Cloud sub-account**. **Note** ▪ This parameter is available only for the production environment. ▪ On the DataStudio page, the identity used to run nodes is the CDH cluster account that is mapped to the logon Apsara Stack tenant account or RAM user. Therefore, you must configure the identity mappings not only for scheduling access identities, but also for the workspace developers in a project to prevent node running failures. ▪ For a CDH cluster in the development environment, the default value of this parameter is **Task Owner**.
Resource Group	Select an exclusive resource group for scheduling that connects to the DataWorks workspace. If no exclusive resource group for scheduling is available, create one. After you select an exclusive resource group for scheduling, click **Test Connectivity** to test the connection between the exclusive resource group for scheduling and the CDH cluster.

4. After the connectivity test is passed, click **Confirm**.

2.1.20.2. Manage members and roles

DataWorks provides roles that have different permissions for you to implement finer-grained permission management. You can add the required users to your workspace and assign the required roles to the users. You can also create custom roles and grant permissions to the roles based on your business requirements.

Background information

Multiple users can be added to the same DataWorks workspace. In this case, if the users have excessive permissions on the workspace, the data security of the workspace may be affected by inappropriate permission use. However, if the users have insufficient permissions on the workspace, they may be unable to use the required features. To resolve this issue, DataWorks provides identities such as members and roles. You can assign different roles to users based on their requirements on the use of workspaces.

If the default roles that are provided by DataWorks cannot meet your requirements, you can create custom roles and grant the required permissions to the roles.

DataWorks provides the following identities:

- **Member**: the Apsara Stack tenant accounts or RAM users that are added to a DataWorks workspace.
- **Cloud account**: Apsara Stack tenant accounts or RAM users.
- **Role**: the carriers that have permissions in a workspace and can be assumed by the members of the workspace. DataWorks provides the following roles:
  - **Project Manager**: the administrators that have all the permissions on the features in a workspace. For example, the workspace administrator role can be used to assign the required role to a RAM user and remove a member that is not the workspace owner from a workspace.
  - **Deploy**: the engineers that have the permissions to deploy nodes.
  - **Development**: the developers that have the permissions to develop and commit nodes.
  - **Model Developer**: the designers that have the permissions to use the data modeling feature.
  - **Visitor**: the visitors that have the read-only permissions on a DataWorks workspace.
  - **Project owner**: the owner that has the highest level of permissions on a workspace.
  - **O&M**: the engineers that have the permissions to allocate resources and deploy nodes.
  - **Security Manager**: the administrators that have the permissions to use Data Security Guard.

  For more information about the permissions of different roles, see Permission list.

Limits

- Only the **Project Manager** and the **Project owner** roles can add users, change the roles of users, and remove **users** and the added **custom roles**.
- You can use only an Apsara Stack tenant account or the RAM user whose role is an administrator or a super administrator of a MaxCompute project to map a custom DataWorks role to a role of the MaxCompute project.

Go to the User Management page

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click the icon in the upper-right corner.
3. In the left-side navigation pane, click **User Management** to go to the **User Management** page.

   You can manage members and roles on the **User Management** page.

Manages members

On the **Manage Members** tab, you can perform the following operations:

- View member information.

  You can view the accounts of members and the roles that are assigned to the members in the current workspace. You can also specify the name of the member, account, or role to search for a specific member. Then, you can view the member information and the number of members to which the role has been assigned. This allows you to realize centralized management of members and roles assigned to the members.
- Add a user.
  i. Click **Add Member** in the upper-right corner of the **Manage Members** tab to add a user to the current workspace.
  ii. In the **Add Member** dialog box, select one or more RAM users from the **Available Accounts** list.
    - **Member**: the Apsara Stack tenant accounts or RAM users that are added to a DataWorks workspace.
    - **Cloud account**: Apsara Stack tenant accounts or RAM users.

- **Role**: the carriers that have permissions in a workspace and can be assumed by the members of the workspace. DataWorks provides the following roles:
  - **Project Manager**: the administrators that have all the permissions on the features in a workspace. For example, the workspace administrator role can be used to assign the required role to a RAM user and remove a member that is not the workspace owner from a workspace.
  - **Deploy**: the engineers that have the permissions to deploy nodes.
  - **Development**: the developers that have the permissions to develop and commit nodes.
  - **Model Developer**: the designers that have the permissions to use the data modeling feature.
  - **Visitor**: the visitors that have the read-only permissions on a DataWorks workspace.
  - **Project owner**: the owner that has the highest level of permissions on a workspace.
  - **O&M**: the engineers that have the permissions to allocate resources and deploy nodes.
  - **Security Manager**: the administrators that have the permissions to use Data Security Guard.

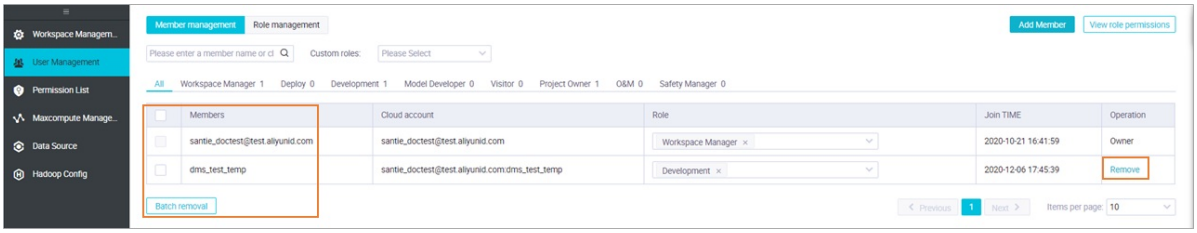
iii. Click the > icon to move the selected RAM users to the **Added Accounts** list.

iv. Select one or more roles that you want to assign to the selected RAM users.

v. Click **Confirm**.

- Remove a member.

  On the **Manage Members** tab, find the member that you want to remove from the workspace and click **Remove** in the **Actions** column to remove the member from the workspace. If you want to remove multiple members from the workspace, you can select them and click Batch removal to remove them at a time.



Manage roles

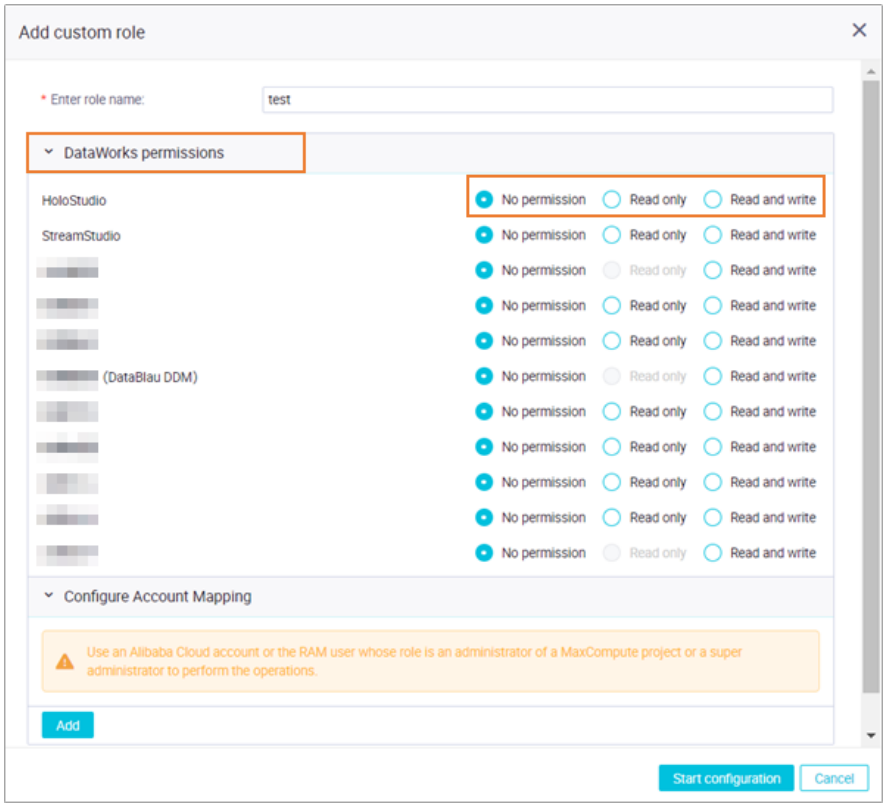
On the **Roles** tab, you can perform the following operations:

- Create a custom role.

  i. Click **Add Custom Role** in the upper-right corner of the **Roles** tab.

  ii. In the **Add Custom Role** dialog box, enter a name for your custom role, such as test.

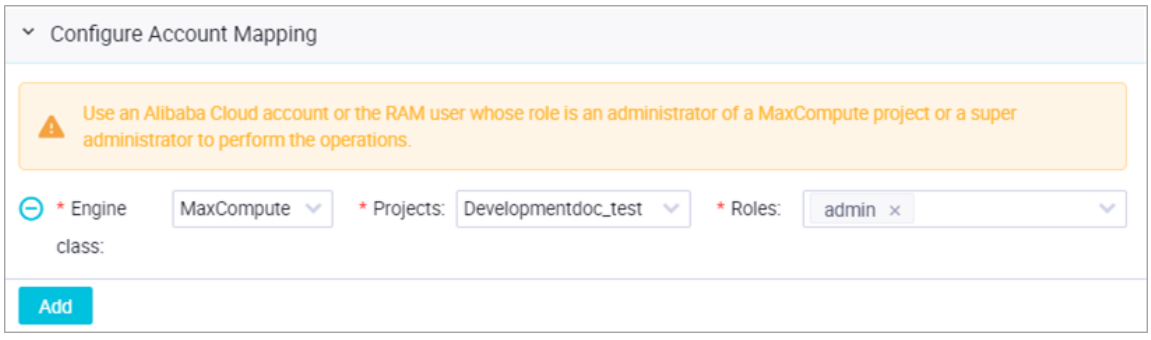
  iii. Grant permissions on the required DataWorks modules to the role.
  - **Unauthorized**: indicates that the role has no permissions on the related module.
  - **Read-only**: indicates that the role can only view the data in the related module.
  - **Read and Write**: indicates that the role can modify the data in the related module.



iv. **Map a custom role to a role of a compute engine.**

You can map a custom role to a role of a compute engine. For example, you can map the custom role test to the **Admin** role of a MaxCompute project. In this case, the Admin role is assumed by the custom role when the custom role accesses the MaxCompute project.

> **Note** You can use only an Apsara Stack tenant account or the RAM user whose role is an administrator or a super administrator of a MaxCompute project to map a custom DataWorks role to a role of the MaxCompute project.



v. Click **Configure**.

- View or edit roles.

  You can view the **preset roles** and **custom roles** that have been configured for the workspace on the **Roles** tab. You can also edit or delete **custom roles**. For more information about the permissions of **preset roles**, see Permission list.

2.1.20.3. 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
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

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
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
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

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
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

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
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

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Create baselines	Yes	Yes	Yes	No	No	No	No
Delete baselines	Yes	Yes	Yes	No	No	No	No
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
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

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
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
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

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
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
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

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
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
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
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
Create data quality rules	Yes	Yes	Yes	Yes	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
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

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
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
Delete nodes	Yes	Yes	Yes	No	No	No	No
Migrate database tables	Yes	Yes	No	No	No	No	No

2.1.20.4. 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. User Guide

3.1.1. 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. Realtime Compute shortens the data processing delay, implements real-time computational logic, and greatly reduces 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 if 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 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.

3.1.2. Quick start

3.1.2.1. Log on to the Realtime Compute console

This topic describes how to log on to the Realtime Compute console.

Prerequisites

- The URL of the Apsara Uni-manager Management Console is obtained from the deployment personnel before you log on to the Apsara Uni-manager Management Console.
- We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL of the Apsara Uni-manager Management Console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password that you can use to log on to the console from the operations administrator.

   > **Note** When you log on to the Apsara Uni-manager Management Console for the first time, you must change the password of your username. Your password must meet complexity requirements. The password must be 10 to 32 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters
   > - Digits
   > - Special characters, which include ! @ # $ %

3. Click **Log On**.
4. In the top navigation bar, move the pointer over Products and click **Realtime Compute**.
5. Specify **Organization** and **Region**.
6. Click **Blink**.

3.1.2.2. Frequently used words

3.1.2.2.1. 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.

3.1.2.2.2. Code development

This topic uses a word count job as an example to describe how to create a Realtime Compute job.

Prerequisites

Before you create 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 to go to the homepage of Realtime Compute.
2. In the top navigation bar, click **Development**.
3. Right-click the folder that you created.
4. Click the **Create File** icon.
5. In the Create File dialog box, configure the following parameters:
   - File Name: Enter `wordcount`.
   - File Type: Select 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 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 inbound 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.

3.1.2.2.3. 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.

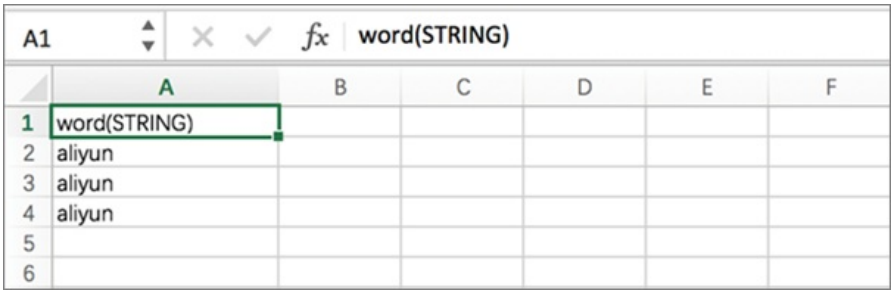
   > **Note** The file that is uploaded for debugging must meet the following requirements:
   > - The file size cannot exceed 1 MB, and the file contains maximum of 1,000 records.
   > - The file must use UTF-8 encoding.
   > - Commas (,) cannot be used in test data, because the file uses the comma-separated 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.

3.1.2.2.4. 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.

3.1.2.3. Big screen service for the Tmall Double Eleven Global Shopping Festival

3.1.2.3.1. 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.

3.1.2.3.2. 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
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.

3.1.2.3.3. 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.

3.1.2.3.4. 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, perform the following steps:

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. 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 RDS for MySQL to store the result data for data visualization, you must register an ApsaraDB RDS data store in Realtime Compute. For more information, see Register an RDS instance.

3.1.2.3.5. 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.

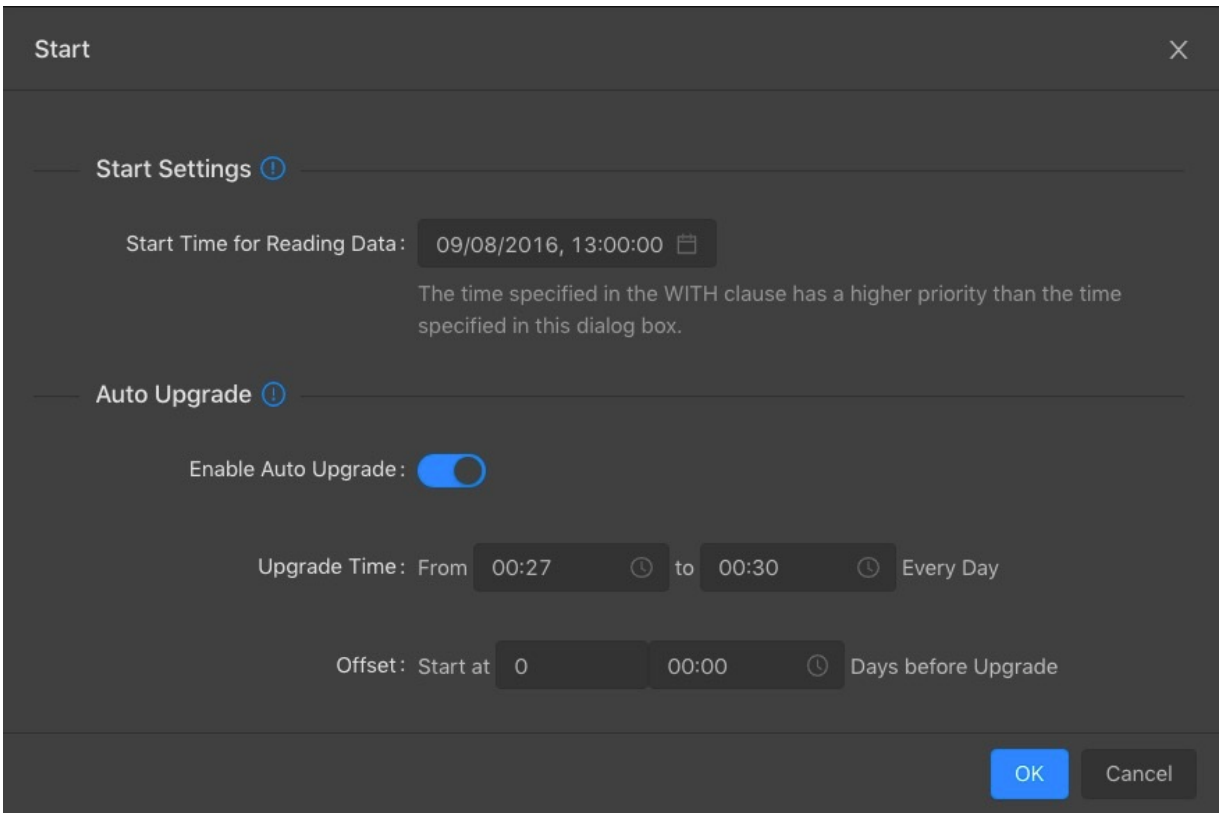
   After the computational logic has been verified in the debugging phase, click **Publish** on the **Development** page to publish the SQL file for the tmall_d11 job. Then, you can view the tmall_d11 job on the **Administration** page of the Realtime Compute console, and manage the job in the production environment, such as starting the job.

3.1.2.3.6. Operations and maintenance (O&M)

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.



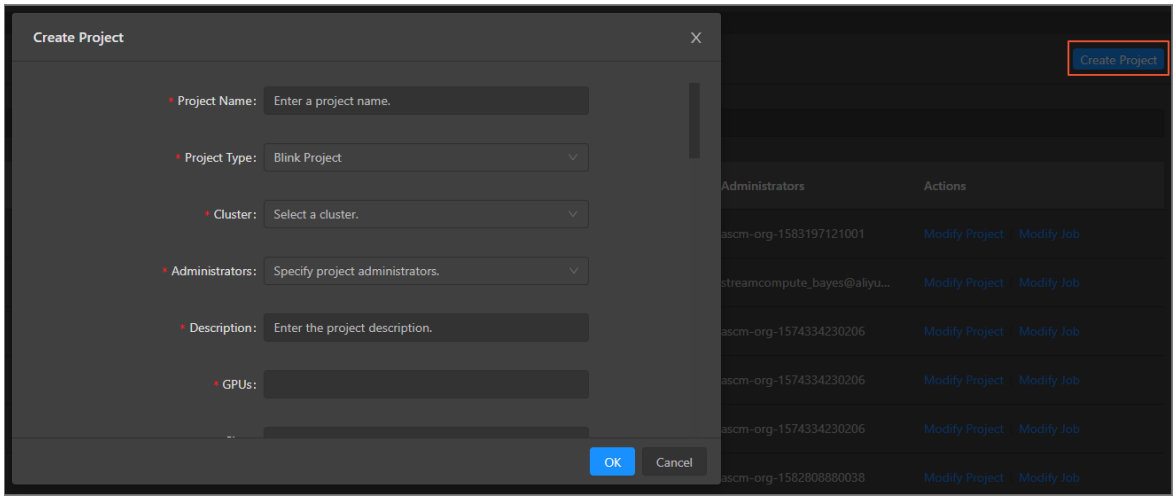
You can check the result data in the ApsaraDB RDS data store after the job runs as expected. In the result table, five transactions and a turnover of CNY 500 are displayed. This is consistent with the source data for testing. In this way, an end-to-end verification is performed to check the SQL code.

3.1.3. 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

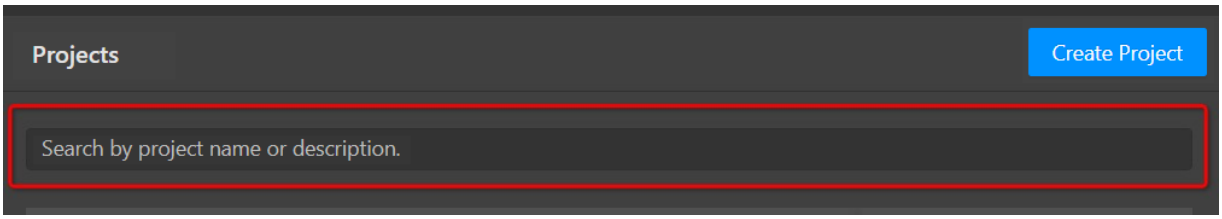
Parameter	Description
Project Name	The name of the project.
Project Type	The type of the project. **Blink Project** is selected by default.
Cluster	The cluster on which the jobs in the project run.
Administrators	The Administrators of the project.
Description	The description of the project.
GPUs	The 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.
Max Resources	The maximum number of JAR files and DICTIONARY resources that can be uploaded. You can keep the default setting.

Parameter	Description
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.



3.1.4. Data storage

3.1.4.1. Overview

This chapter describes data storage systems supported by Realtime Compute.

3.1.4.2. Authorize Realtime Compute to access a VPC

Before you use Realtime Compute to access storage resources in a virtual private cloud (VPC), you must authorize Realtime Compute to access the VPC. This topic describes how to authorize Realtime Compute to access a VPC.

Procedure

1. Log on to the Realtime Compute console.
2. Move the pointer over the username in the upper-right corner.
3. In the list that appears, click **Project Management**.
4. In the left-side navigation pane, click **VPC Access Authorization**.
5. In the upper-right corner of the **VPC Access Authorization** page, click **Add Authorization**.
6. In the **Authorize StreamCompute VPC Access** dialog box, configure the parameters as required. The following table describes the parameters.

Parameter	Description
Name	The name of the VPC.
Region	The region where the storage resource resides.
VPC ID	The ID of the VPC. To view the VPC ID of an ApsaraDB RDS instance, perform the following steps: i. Log on to the ApsaraDB RDS console. ii. In the top navigation bar, select the region where the ApsaraDB RDS instance resides. iii. On the Instances page, find the ApsaraDB RDS instance and click the instance ID in the Instance ID/Name column. iv. In the left-side navigation pane, click **Database Connection**. v. On the **Instance Connection** tab, view the VPC ID next to the value of **Network Type** in the **Database Connection** section. For example, the VPC ID is `vpc-bp1lysht98wrvl9n3****`.
Instance ID	The instance ID of the storage resource in the VPC. To view the ID of an ApsaraDB RDS instance, perform the following steps: i. Log on to the ApsaraDB RDS console. ii. In the top navigation bar, select the region where the ApsaraDB RDS instance resides. iii. On the Instances page, find the ApsaraDB RDS instance and click the instance ID in the Instance ID/Name column to go to the **Basic Information** page. iv. View **Instance ID** in the **Basic Information** section.
Instance Port	The port of the storage resource in the VPC.

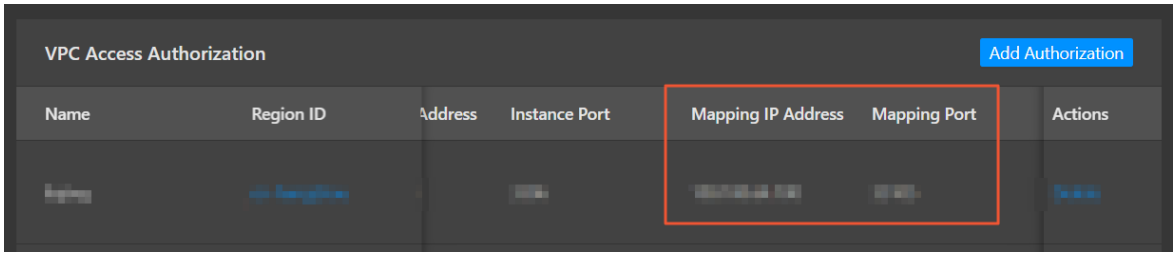
FAQ

Q: How do I set the url parameter when I use data definition language (DDL) statements to reference a storage resource in a VPC?

A: When you use DDL statements to reference storage resources in a VPC, you can set the url parameter in the WITH clause based on the **Mapping IP Address** and **Mapping Port** parameters on the **VPC Access Authorization** page. For example, you can set url='jdbc:mysql://<mappingIP>:<mappingPort>/<databaseName>'. To obtain the values of the **Mapping IP Address** and **Mapping Port** parameters, perform the following steps:

1. Log on to the Realtime Compute console.
2. Move the pointer over the username in the upper-right corner.
3. In the list that appears, click **Project Management**.
4. In the left-side navigation pane, click **VPC Access Authorization**.

5. On the **VPC Access Authorization** page, view the values of the **Mapping IP Address** and **Mapping Port** parameters.



3.1.4.3. Overview

3.1.4.3.1. Overview

To facilitate data storage management, you can register data storage resources on the Realtime Compute development platform. This enables you to use the advantages of the one-stop Realtime Compute service. In Realtime Compute, you can manage multiple data storage systems, such ApsaraDB RDS, AnalyticDB for MySQL, and Tablestore. This one-stop management service allows you use manage data stores in the cloud on the Realtime Compute development platform without the need to navigate across multiple management consoles of different storage systems.

3.1.4.3.2. 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

Support	Dimension table	Output
ApsaraDB RDS	Supported	Supported
Tablestore	Supported	Supported

3.1.4.3.3. Registration and usage

This topic describes how to register and use external data stores in Realtime Compute.

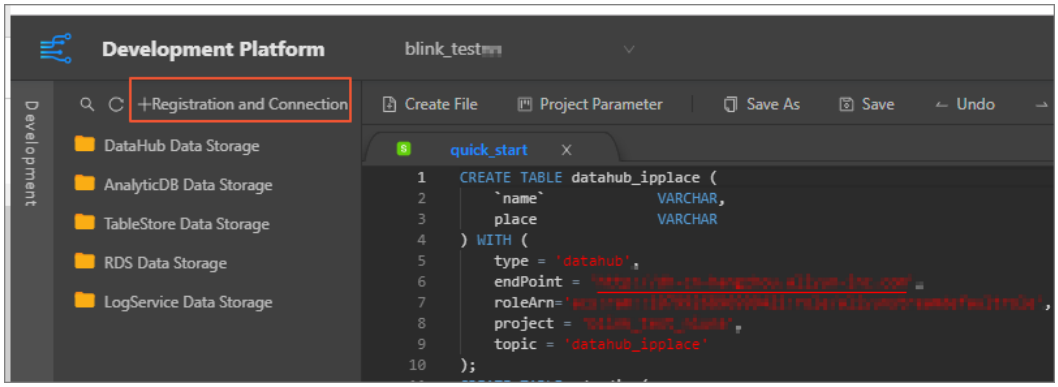
> **Note** If a job requires the use of the data stores owned by another Apsara Stack tenant account, you can write DDL statements to reference the data stores. In the DDL statements, you must specify the AccessKey ID and AccessKey secret of the account. In this scenario, you cannot use the codeless 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**.

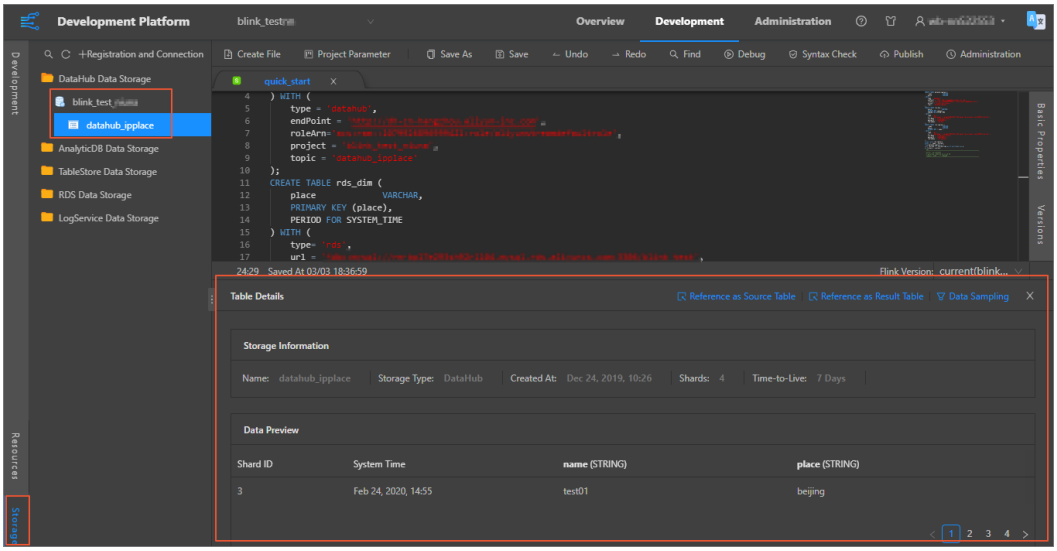
> **Note** After you enable the data store registration feature, you can only register data stores that are owned by your organization.

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

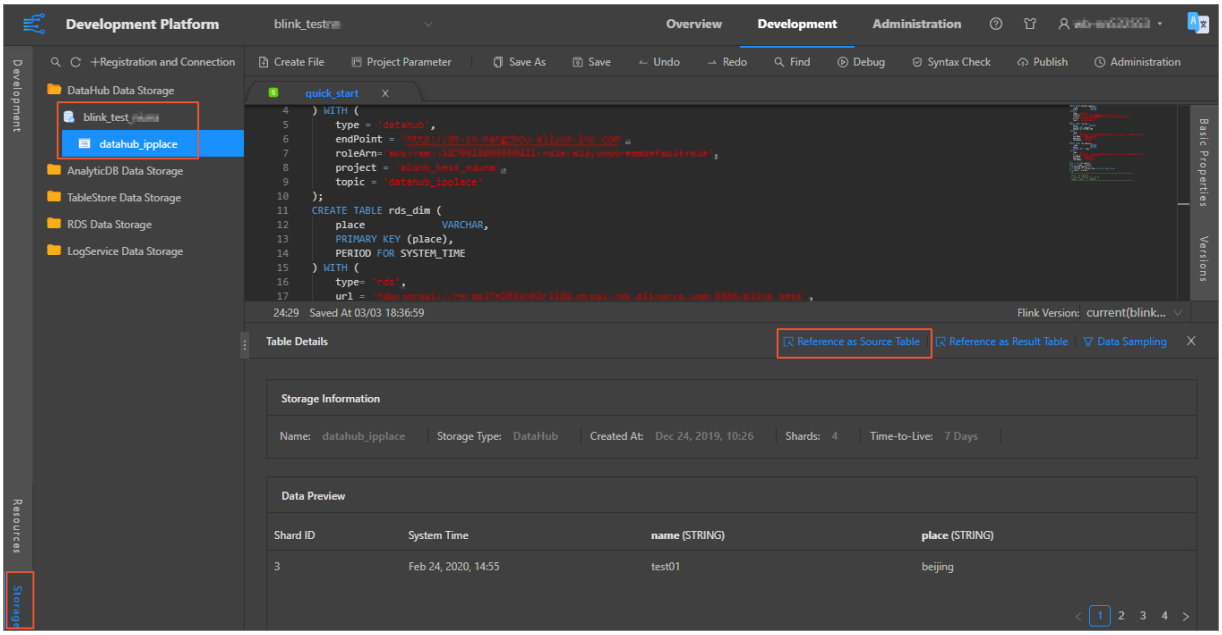
You must declare tables from external data stores before you can reference these tables. The following example shows how to reference a source table that contains streaming data:

```
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.

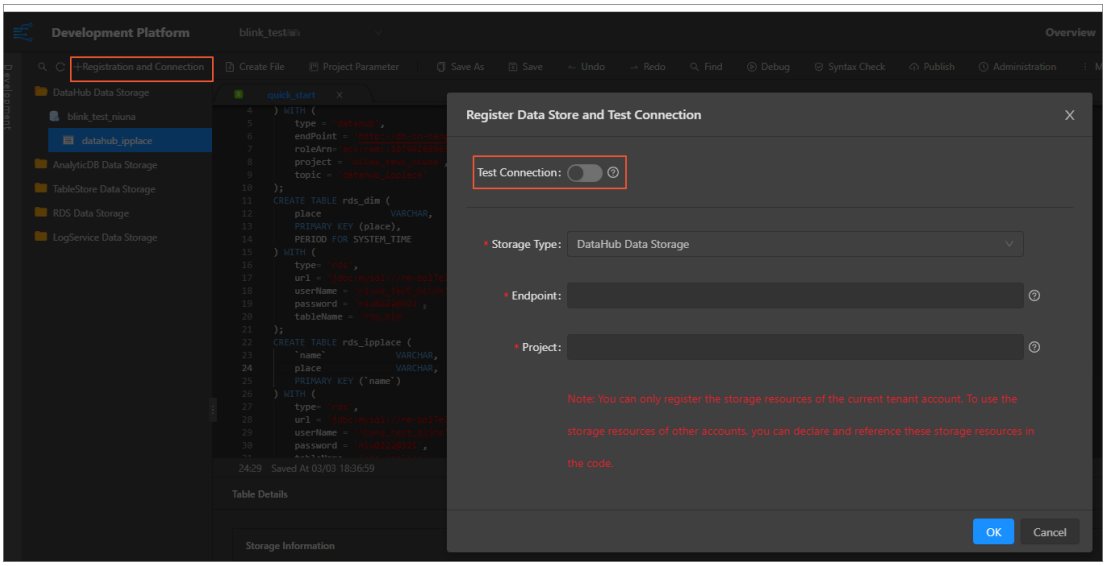
To reference a source table, log on to the Realtime Compute console and open the target SQL file on the Development page. Click the Storage tab, select the table for reference, and then click **Reference as Source Table**. The required DDL statements are displayed on the current page.



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');
```

3.1.4.4. 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.
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.

3.1.4.5. 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.

Parameter	Description
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.

3.1.4.6. 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.
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.

3.1.4.7. Register an ApsaraDB RDS data store

This topic describes how to register and use an ApsaraDB RDS data store in Realtime Compute.

Introduction to ApsaraDB RDS

ApsaraDB RDS offers a stable, reliable, and scalable online database service. Based on the Apsara distributed operating system and high performance SSD storage, ApsaraDB 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 ApsaraDB 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 ApsaraDB RDS. The performance of ApsaraDB RDS is restricted by the limits of relational models. Therefore, ApsaraDB 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, ApsaraDB 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 ApsaraDB RDS connectors are used. If Realtime Compute frequently writes data to a DRDS table or an ApsaraDB 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 ApsaraDB RDS to store the result tables of Blink jobs. To prevent deadlocks, we recommend that you use Tablestore to store result tables.

Register an ApsaraDB RDS instance

1. Log on to the Realtime Compute console.
2. In the top navigation bar, click **Development**.

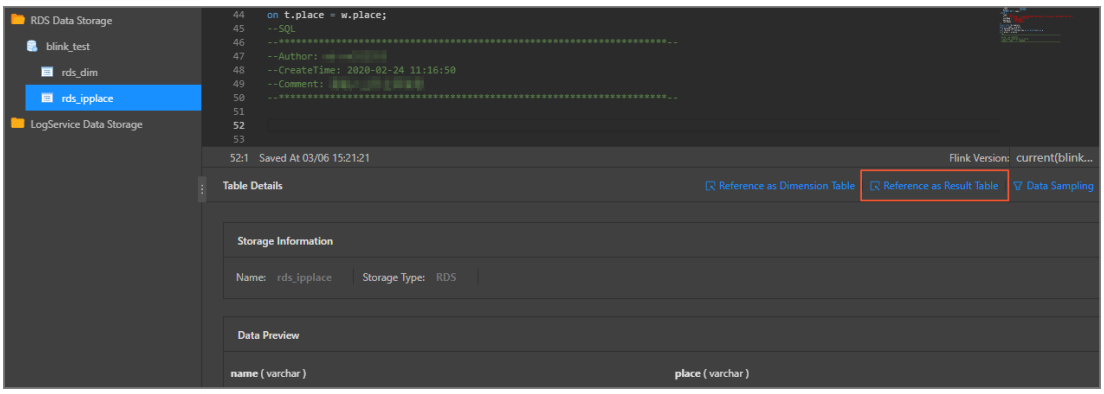
3. In the left-side navigation pane, click the **Storage** tab.
4. Right-click **RDS Data Storage**, and select **Register Data Store**. In the dialog box that appears, register an ApsaraDB RDS instance in Realtime Compute. Parameter description

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 Test Connection.
Storage Type	The type of the data store. RDS Data Storage is selected by default.
URL	The URL that is used to access the ApsaraDB RDS database.
DBName	The name of the ApsaraDB RDS database to be accessed by Realtime Compute. **Note** This parameter specifies the ApsaraDB RDS database name instead of the ApsaraDB RDS instance name. ApsaraDB 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 the whitelists of ApsaraDB RDS. Otherwise, Realtime Compute may fail to connect to ApsaraDB RDS. For more information, see Specify whitelist settings.
User Name	The username that is used to log on to the ApsaraDB RDS database.
Password	The password that is used to log on to the ApsaraDB RDS database.
Engine	The type of the ApsaraDB RDS database. Valid values: ○ **mysql** ○ **postgresql** ○ **sqlserver**

Reference an ApsaraDB RDS table as a result table

After you register an ApsaraDB RDS data store, double-click the ApsaraDB RDS database, double-click the ApsaraDB RDS table that you want to reference as a result table, and then click **Reference as Result Table**.

Reference an ApsaraDB RDS table as a result table



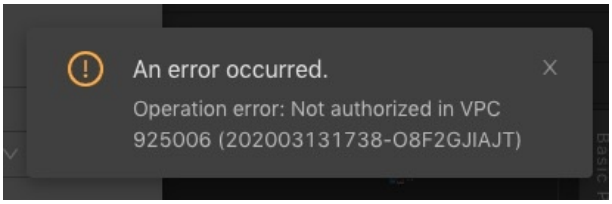
After you click Reference as Result Table, Realtime Compute automatically generates the related DDL statements on the current page.

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  );
```

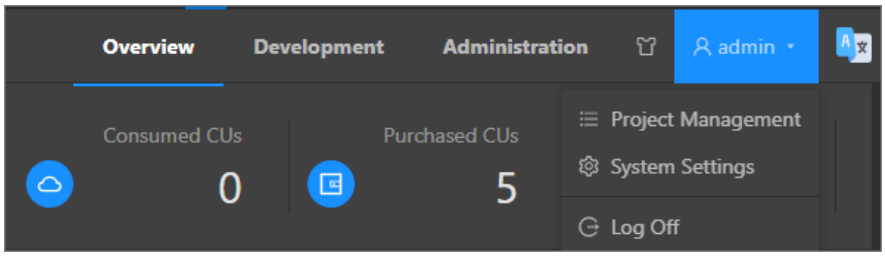
If the following error message appears, troubleshoot and rectify the fault in the following way.

Error message



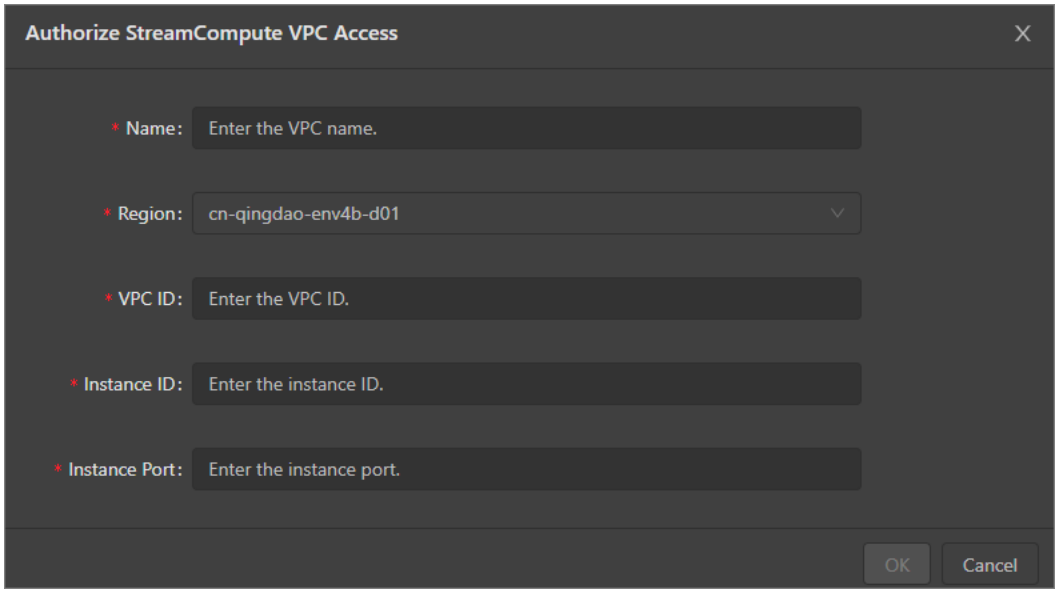
The error occurs because a VPC instead of a classic network was selected when you created the ApsaraDB RDS instance. You can perform the following steps to rectify this fault:

1. Move the pointer over the administrator icon, as shown in the following figure.

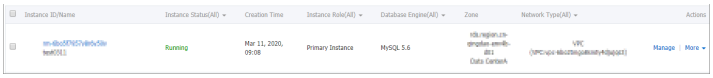


2. Click **System Settings**.
3. In the left-side navigation pane, click **VPC Access Authorization**.
4. Click **Add Authorization**. The **Authorize StreamCompute VPC Access** page appears.

Authorization



Parameter description

Parameter	Description
Name	The name of the VPC.
Region	The region where the ApsaraDB RDS instance resides.
VPC ID	The ID of the VPC.
Instance ID	The ID of the ApsaraDB RDS instance. You can log on to the ApsaraDB RDS console and view the instance ID. Instance information 
Instance Port	The port that is used to access the ApsaraDB RDS instance. To view the internal port number, log on to the ApsaraDB RDS console, click the ID or name of the instance that you want to access in the Instance ID/Name column. On the page that appears, view the internal port number in the Basic Information section.

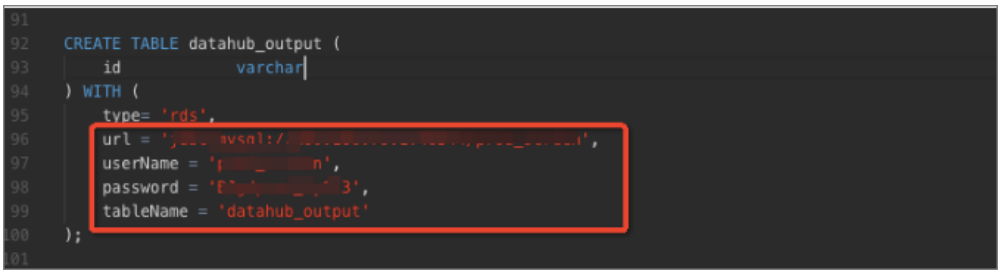
5. Register the ApsaraDB RDS instance. You must specify the required parameters during the registration.

You can register only data storage resources that are owned by your level-1 organization. For example, if ApsaraDB RDS Instance A is owned by Organization A, a user from Organization B cannot register ApsaraDB 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 ApsaraDB 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 category



To use ApsaraDB RDS storage resources by writing SQL code, the user from Organization B must specify whitelist settings.

Specify whitelist settings

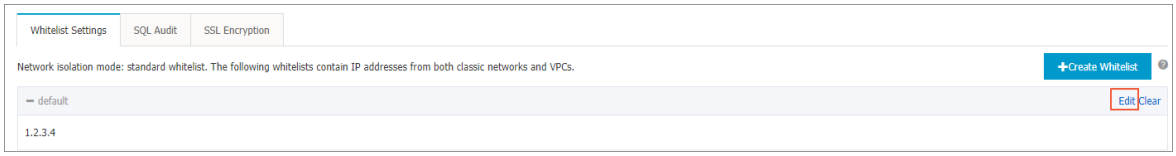
Some data stores use whitelists for access control to ensure high-level security. These data stores allow access only from the IP addresses that are added to the whitelists. This prevents unauthorized Apsara Stack services from accessing data in these data stores. For example, a newly created ApsaraDB RDS database denies all access. You must add IP addresses to a whitelist of the ApsaraDB RDS database to allow access to the database.

ApsaraDB RDS can be accessed from both external and internal networks. To enable Realtime Compute to access ApsaraDB RDS, you must add the CIDR blocks of Realtime Compute to a whitelist of the ApsaraDB RDS database.

Procedure:

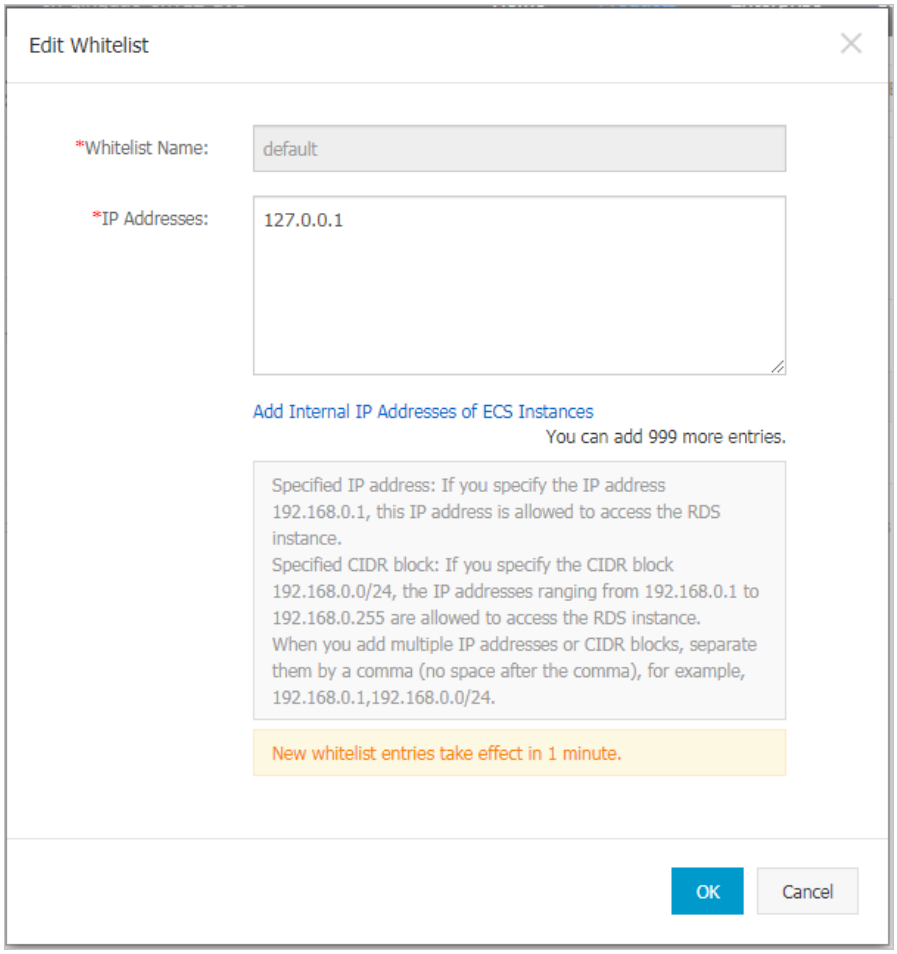
1. Log on to the ApsaraDB RDS console.

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** that corresponds to the **default** whitelist.



> **Note**
>
> - If you want to connect an ECS instance to an ApsaraDB 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 whitelist.

5. In the **Edit Whitelist** dialog box, 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 ApsaraDB 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

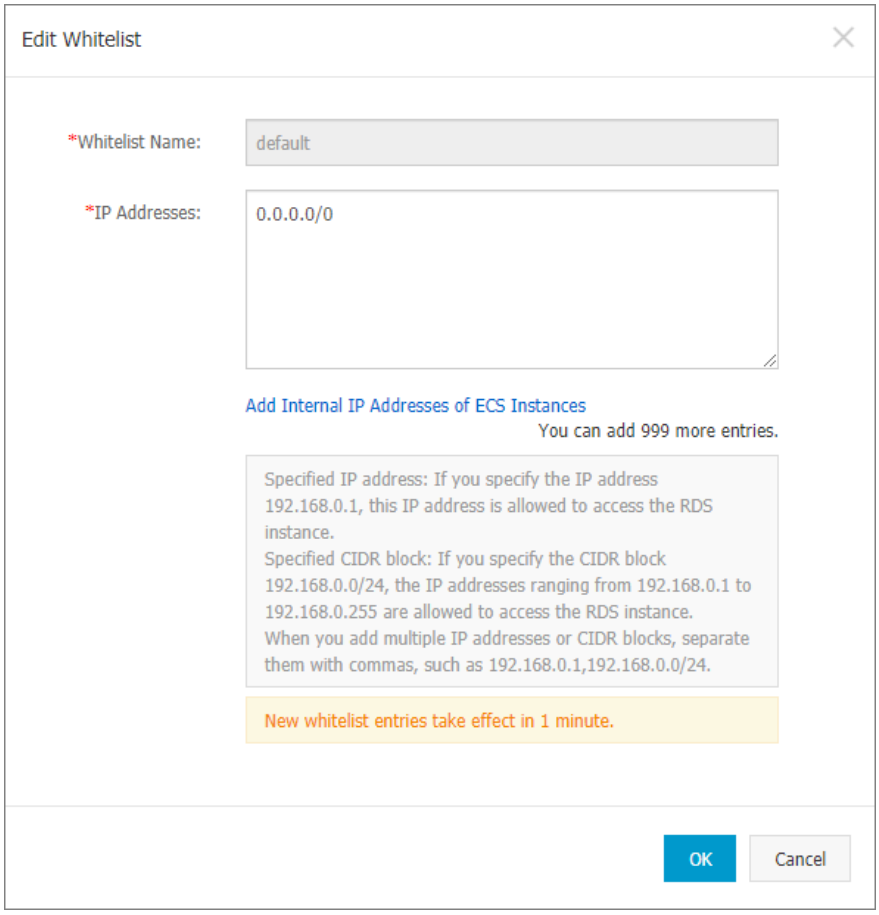
whitelist.

> **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.

FAQ

- 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.

3.1.5. Data development

3.1.5.1. 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**.

3.1.5.2. Development

3.1.5.2.1. 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.

3.1.5.2.2. 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.

3.1.5.2.3. 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.

3.1.5.3. 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 SQL code that implements the computing logic, perform the following steps to debug the code:

1. Log on to the Realtime Compute console to go to the homepage of Alibaba Cloud Realtime Compute.
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 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.

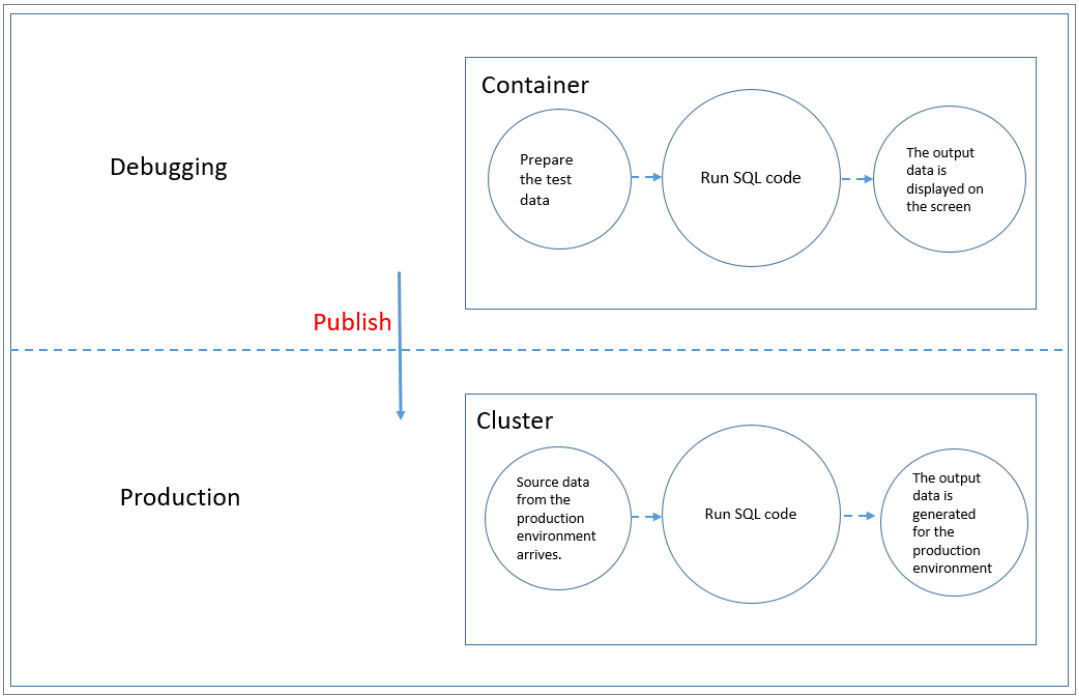
The debugging feature of Realtime Compute provides the following functions:

- 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 RDS and the length of character strings reaches the upper limit for an ApsaraDB 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 ApsaraDB 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. The following example shows a file for debugging:

```
id,name,age
1,alicloud,13
2,stream,1
```

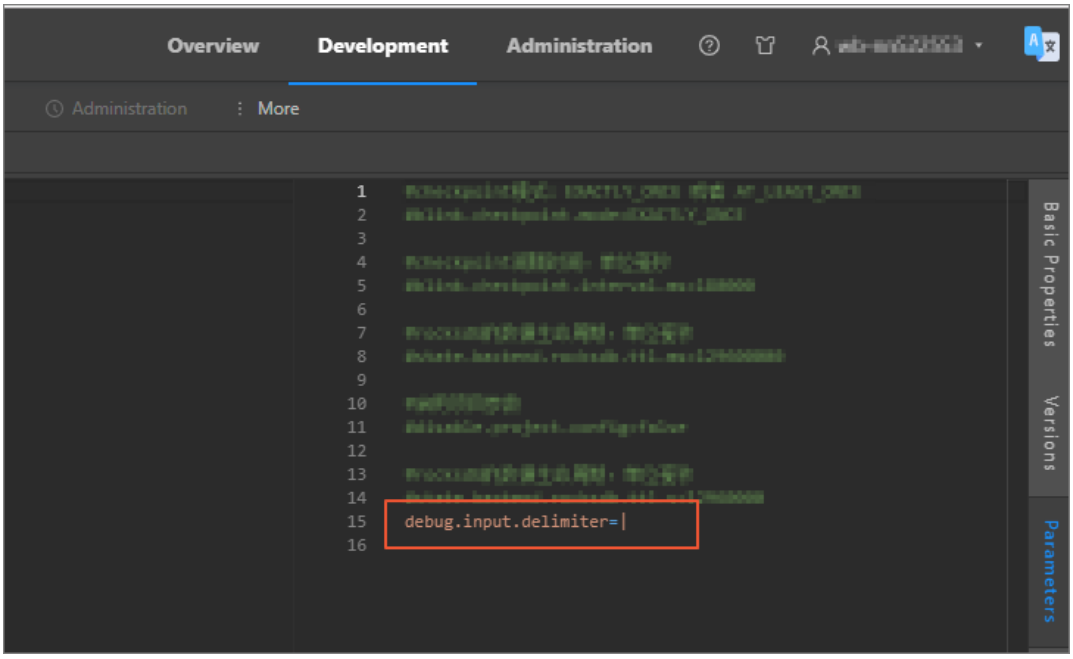
If the separator is not specified, a comma (,) is used to separate fields. If you need to use a JSON string as the field data and the string contains commas (,), you must specify another character as the separator.

> **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, set `debug.input.delimiter=|`.

Specify a separator



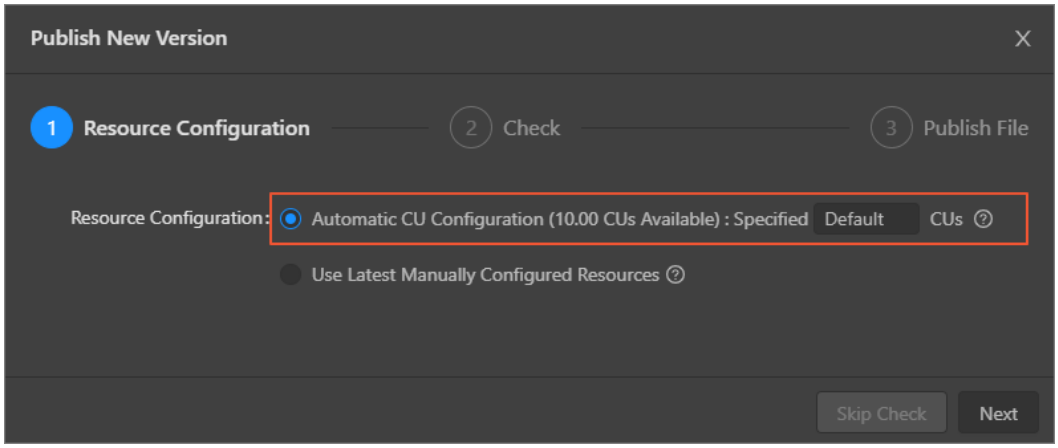
3.1.5.4. Publish a job SQL file

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



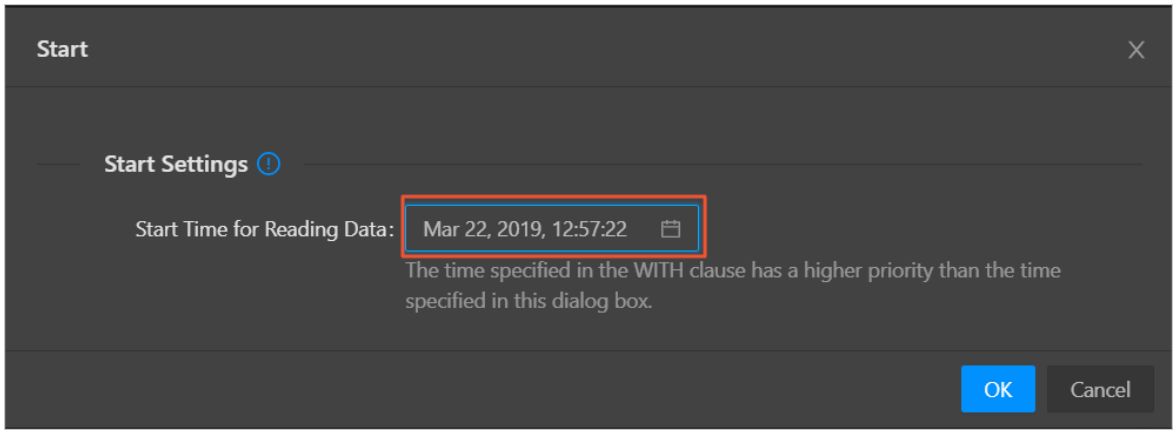
5. Check the data. After the check is completed, click **Next**.
6. Click **Publish**.
7. Go to the **Administration** page to start the job.
   i. In the top navigation bar, click **Administration**.
   ii. On the **Administration** page, find the target job, and click **Start** in the **Actions** column.

3.1.5.5. Start a job

After you develop and publish a job, you can start the job on the Administration page.

Procedure

1. Log on to the Realtime Compute console.
2. In the top navigation bar, click **Administration**.
3. Find the job that you want to start, and click **Start** in the **Actions** column.
4. In the **Start** dialog box, set the **Start Time for Reading Data** parameter.



5. Click **OK**. The job is started.

   Start Time for Reading Data indicates the time at which the system starts to read data from the source table.

   - If you select the current time, Realtime Compute reads the data generated after the current time.
   - If you select a previous time, Realtime Compute reads the data generated after the specified time. This is used to track historical data.

3.1.5.6. Suspend a job

After you modify the resource configuration of a job, you must suspend and resume the job to make the changes take effect. This topic describes how to suspend a job.

Context

Notice

- You can only **suspend** a job that is in the **Running** state.
- If you **suspend** a job, its task status is not cleared. For example, if the job you **suspend** is running a COUNT operation, the COUNT operation continues from the last successful checkpoint after you **resume** the job.
- The Suspend (checkpoint) operation is supported in Realtime Compute V3.5.0 and later. If your Realtime Compute is earlier than V3.5.0, the following error message appears when you try to perform this operation: **An error occurred. System error: The Blink version is abnormal. Error reason: blink version >= blink-3.5 is required, instance blink-3.4.4.**

Procedure

1. Log on to the Realtime Compute console.
2. In the top navigation bar, click **Administration**.
3. On the **Administration** page, find the job that you want to suspend, and click **Suspend** in the **Actions** column.

Note The **Suspend (checkpoint)** operation in **More** suspends the job and triggers a checkpoint event. Therefore, the time consumed to suspend a job by performing the **Suspend (checkpoint)** operation is longer than that by performing the **Suspend** operation.

3.1.5.7. Terminate a job

After you modify the SQL logic, change the job version, add parameters to the WITH clause, or add job parameters for a job, you must terminate and then start the job to make the changes take effect. This topic describes how to terminate a job.

Notice

- You can only **terminate** a job that is in the **Running** or **Starting** state.
- If you **terminate** a job, its task status is cleared. For example, if the job you **terminate** is running a COUNT operation, the COUNT operation starts from 0 after you **start** the job.
- The **Terminate (checkpoint)** operation is supported in Realtime Compute V3.5.0 and later. If your Realtime Compute version is earlier than V3.5.0, the following error message appears when you try to perform this operation: **An error occurred. System error: The Blink version is abnormal. Error reason: blink version >= blink-3.5 is required, instance blink-3.4.4.**

To terminate a job, perform the following steps:

1. Log on to the Realtime Compute console.
2. In the top navigation bar, click **Administration**.
3. On the **Administration** page, find the job that you want to terminate, and click **Terminate** in the **Actions** column.

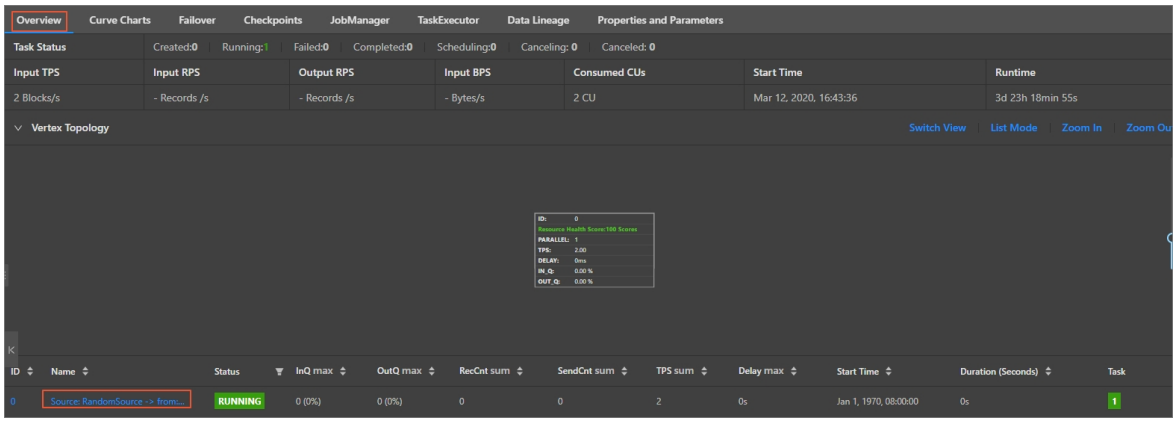
> **Note** The **Terminate (checkpoint)** operation under **More** is different from the **Terminate** operation. The system triggers a checkpoint when you perform the **Terminate (checkpoint)** operation to terminate a job. Therefore, the time consumed to terminate a job by performing the **Terminate (checkpoint)** operation is longer than that by performing the **Terminate** operation. The job status is cleared after the job is terminated. The Terminate (checkpoint) operation has other functions in some scenarios. For example, if the upstream storage system is Message Queue for Apache Kafka, the system submits an offset each time it triggers a checkpoint. This ensures that the number of offsets submitted to the Kafka server is consistent with the amount of data consumed.

3.1.5.8. View logs

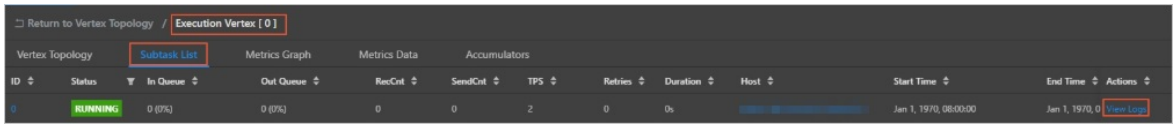
You can view the operational logs of a job to learn the job operation information. This topic describes how to view job logs.

Procedure

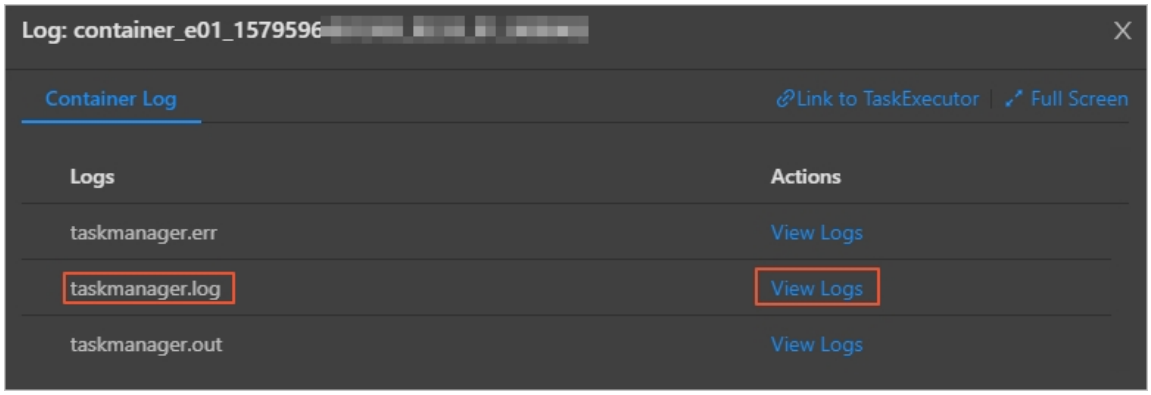
1. Go to the **Administration** page in Realtime Compute.
   i. Log on to the Realtime Compute console.
   ii. In the top navigation bar, click **Administration**.
   iii. On the **Jobs** page, click the name of the job whose logs you want to view in the **Job Name** column.
2. At the bottom of the **Overview** tab, click the name of the desired vertex.



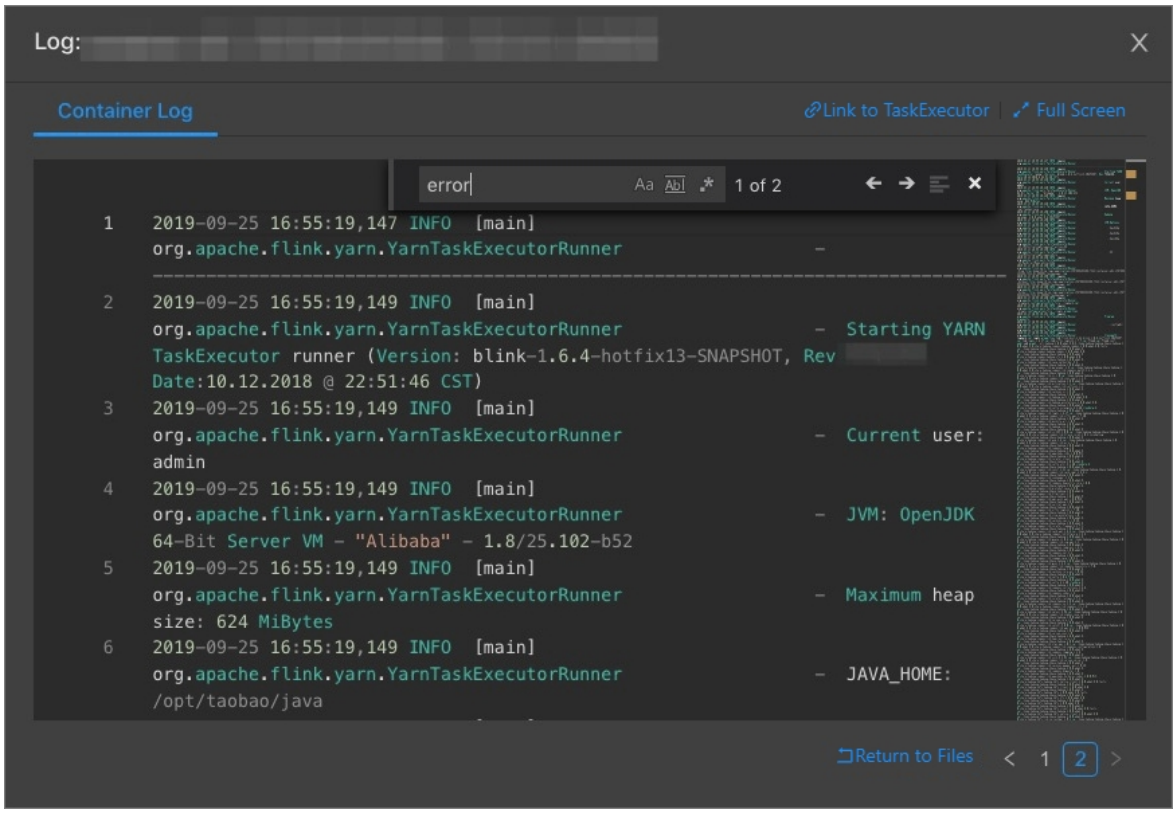
3. On the **Execution Vertex** page, click the **Subtask List** tab. Then, find the desired subtask and click **View Logs** in the **Actions** column.



4. In the **Log** dialog box, 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 log entries from the last page. The first error recorded in the log describes the Root cause of the job error.

4.Machine Learning Platform for AI

4.1. User Guide

4.1.1. 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.

4.1.2. Features supported by Hygon servers and Intel servers

This topic describes the features that are supported by Hygon servers and Intel servers.

Feature	Hygon server	Intel server
Data labeling	Not supported	Supported

Feature		Hygon server	Intel server
Machine Learning Studio	General algorithm components	Supported	Supported
	Deep learning components	Not supported	Supported
	Video intelligent platforms	Not supported	Supported
	AutoML	Not supported	Supported
	Algorithm market	Not supported	Supported
Data Science Workshop (DSW)		Supported	Supported
Elastic Algorithm Service (EAS)		Supported	Supported

4.1.3. Log on to the PAI console

This topic describes how to log on to the PAI console.

Prerequisites

- The IP address or domain name of the Apsara Uni-manager Management Console is obtained from deployment engineers before you log on to the Apsara Uni-manager Management Console.
- A browser is available. We recommend that you use Google Chrome.

Procedure

1. Enter the URL of the Apsara Uni-manager Management Console in the address bar and press the Enter key.
2. Enter your username and password.

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

   > **Note** When you log on to the Apsara Uni-manager Management Console for the first time, you must change the password of your username. For higher security, the 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 such as exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Log On**.
4. If your account has multi-factor authentication (MFA) enabled, perform corresponding operations in the following scenarios:
   - It is the first time that you log on to the console after MFA is forcibly enabled by the administrator.
     a. On the Bind Virtual MFA Device page, bind an MFA device.
     b. Enter the account and password again as in Step 2 and click **Log On**.
     c. Enter a six-digit MFA verification code and click **Authenticate**.
   - You have enabled MFA and bound an MFA device.

Enter a six-digit MFA authentication code and click **Authenticate**.

> **Note** For more information, see the *Bind a virtual MFA device to enable MFA* topic in *Apsara Uni-manager Management Console User Guide*.

5. In the top navigation bar of the Apsara Uni-manager Management Console, choose **Products > Big Data > Machine Learning Platform for AI** to go to the PAI console.
6. On the page that appears, set the Organization and Region parameters and click **Access as Administrator** to go to the PAI console.

> **Note** If this is the first time that you log on to the PAI console, you must perform the following steps:
>
> i. Create an organization
>
> Create an organization to store resource sets and their resources.
>
> ii. Create users
>
> Create users and assign the users different roles to meet different requirements for system access control.
>
> iii. Create a resource set
>
> Create a resource set before you apply for resources.
>
> iv. Add members to the resource set
>
> Add the created users to the resource set.
>
> v. Go to the homepage of the Apsara Uni-manager Management Console. In the top navigation bar, choose **Product > Big Data > MaxCompute**. Then, create a task account and a project in MaxCompute.
>
> a. When you create a task account, set the **Organization** parameter to the organization that you created in Step 1.
>
> b. When you create a MaxCompute project, set the **Organization** parameter to the organization that you created in Step 1. Set the **Resource Set** parameter to the resource set you created in Step 3. Set the **Task Account** parameter to the task account that you created.
>
> vi. Go to the Project Management page of the DataWorks console to create a DataWorks workspace. In the **Advanced Settings** section of the Create a workspace dialog box, set the **MaxCompute Project Name** parameter to the name of the MaxCompute project that you created in the preceding step.

4.1.4. Data labeling

> **Note** The data labeling feature is applicable only to Intel servers, but not to Hygon servers.

4.1.4.1. Register a dataset

PAI allows you to register a dataset by creating a dataset or importing a dataset file. It also allows you to use **manifest** files to manage all registered datasets.

Register a dataset by creating a dataset

If your source data, such as image, text, video, and audio files, is stored in OSS buckets, you can **create a dataset** in the PAI console. The system scans all files of the specified type in the specified OSS folder and generates a manifest file in the specified OSS path.

1. Go to the **Register Dataset** page.
   i. In the left-side navigation pane of the PAI console, choose **Data Preprocessing > Dataset Manager**.
   ii. On the **Dataset Manager** page, click **Register Dataset**.
2. On the **Register Dataset** page, set the parameters.

Parameter	Description
Dataset Name	The name must be 1 to 24 characters in length, and can contain underscores (_) and hyphens (-). It must start with a letter or a digit.
Method	Set the **Method** parameter to **New Dataset**.
Storage Type	Only **OSS** is supported. You cannot change the value.
Path	Set the **Path** parameter to the OSS folder where your source data is stored.
Data Type	Only **Image** is supported. You cannot change the value.
Tags	You can add a maximum of 10 tags to each dataset. Each tag can contain underscores (_) and hyphens (-). It must start with a letter or a digit.

3. Click **Submit**. Then, a manifest file is generated. The following code provides an example of the manifest file.

```
null
null
null
...
```

After you register a dataset, you can view the dataset on the **Dataset Manager** page.

Register a dataset by importing a dataset file

If you have an on-premises CSV file or manifest file, you can register a dataset by **importing the dataset file**. If you import a CSV file, the system automatically converts it to a **manifest** file.

1. Go to the **Register Dataset** page.
   i. In the left-side navigation pane of the PAI console, choose **Data Preprocessing > Dataset Manager**.
   ii. On the **Dataset Manager** page, click **Register Dataset**.
2. On the **Register Dataset** page, set the parameters.

Parameter	Description
Dataset Name	The name must be 1 to 24 characters in length, and can contain underscores (_) and hyphens (-). It must start with a letter or a digit.
Method	Set the **Method** parameter to **Import Dataset**.
Storage Type	Only **OSS** is supported. You cannot change the value.
Path	Select an OSS path.

Parameter	Description
Data Type	Drag an on-premises CSV file or **manifest** file to the upload area on the right side of the **Data Type** parameter. **Note** If the imported file is used in a labeling job, the names of the fields in the file must comply with the data structure of the template that is used to create the labeling job. For more information, see Data labeling templates.
Tags	You can add a maximum of 10 tags to each dataset. Each tag must be 1 to 10 characters in length, and can contain underscores (_) and hyphens (-). It must start with a letter or a digit.

3. Click **Submit**.

   After you register a dataset, you can view the dataset on the **Dataset Manager** page.

4.1.4.2. Data labeling templates

Machine Learning Platform for AI provides the following templates: object detection, semantic segmentation, comprehensive image labeling, Optical Character Recognition (OCR), single-label image classification, and multi-label image classification. When you create a labeling job, you can select a template based on your requirements.

Object detection

Object detection is used to locate a specific object in an image. The rectangle selection tool is commonly used.

- Scenarios

  Object detection applies to scenarios such as vehicle detection, passenger detection, and image search.

- Data structure
  - Input data

    Each row in the **manifest** file contains a topic. Each topic must contain the picUrl field.

```
{"data":{"picUrl":"oss://****/pics/fruit/apple-1.jpg"}}
...
```

- Output data

  Each row in the **manifest** file contains a topic and the corresponding labeling result. The following code provides an example on the JSON string in each row:

```
{
    "data": {
        "picUrl": "oss://****/pics/fruit/apple-1.jpg"
    },
    "label-****(Labeling job ID)": {
        "results": [{
            "data": [{
                "id":"Znyumd-*****",
                "type":"image/rectangleLabel",
                "value":{
                    "rotation":0,
                    "x":40.68320610687023,
                    "width":327.52035623409665,
                    "y":5.762467474590647,
                    "height":296.68117192104745
                },
                "labelColor":"#72bf7d",
                "labels":["apple"]
            }],
            "id":"44****",
            "type":"image"
        }]
    }
}
```

Semantic segmentation

Semantic segmentation is used to recognize an object in an image and retrieve the coordinates of the object by scanning all pixels of the object. The commonly used tools are the polygon selection tool, brush tool, and superpixel tool.

- Scenarios

  Semantic segmentation applies to scenarios such as autonomous driving, facial expression recognition, and apparel classification.

- Data structure
  - Input data

    Each row in the **manifest** file contains a topic. Each topic must contain the picUrl field.

```
{"data":{"picUrl":"oss://****/pics/fruit/apple-1.jpg"}}
...
```

- Output data

  Each row in the **manifest** file contains a topic and the corresponding labeling result. The following code provides an example on the JSON string in each row:

```
{
    "data": {
        "picUrl": "oss://****/pics/fruit/apple-1.jpg"
    },
    "label-****(Labeling job ID)": {
        "results": [{
            "data": [{
                "id":"Znyumd-*****",
                "type":"image/polygonLabel",
                "value":{
                    "points": [
                        [110, 46],
                        [52, 196],
                        [48, 168],
                        [48, 145],
                        [54, 120],
                        [63, 93],
                        [76, 74]
                    ]
                },
                "labelColor":"#72bf7d",
                "labels":["apple"]
            }],
            "id":"44****",
            "type":"image"
        }]
    }
}
```

Comprehensive image labeling

Comprehensive image labeling is used to match the content of the input images against a set of labels. This template allows you to use all image labeling tools as needed.

- Scenarios

  Comprehensive image labeling applies to scenarios such as autonomous driving, content moderation, and content recognition.

- Data structure
  - Input data

    Each row in the **manifest** file contains a topic. Each topic must contain the picUrl field.

```
{"data":{"picUrl":"oss://****/pics/fruit/apple-10.jpg"}}
```

- Output data

  Each row in the **manifest** file contains a topic and the corresponding labeling result. The following code provides an example on the JSON string in each row:

```
{
    "data": {
        "picUrl": "oss://****/pics/fruit/apple-10.jpg"
    },
    "label-****(Labeling job ID)": {
        "results": [{
            "data": [{
                "id":"Znyumd-****",
                "type":"image/rectangleLabel",
                "value":{
                    "rotation":0,
                    "x":40.68320610687023,
                    "width":327.52035623409665,
                    "y":5.762467474590647,
                    "height":296.68117192104745
                },
                "labelColor":"#72bf7d",
                "labels":["Red apple"]
            }],
            "id":"44****",
            "type":"image"
        }]
    }
}
```

OCR

OCR is used to extract text from input images, and then classify the images based on the text.

- Scenarios

  OCR applies to scenarios such as the recognition of identity cards, tickets, license plates, and bank cards.

- Data structure
  - Input data

    Each row in the **manifest** file contains a topic. Each topic must contain the picUrl field.

```
{"data":{"picUrl":"oss://****/img/ocr_card/img0.jpeg"}}
```

- Output data

  Each row in the **manifest** file contains a topic and the corresponding labeling result. The following code provides an example on the JSON string in each row:

```
{
    "data": {
        "picUrl": "oss://****/img/ocr_card/img0.jpeg"
    },
    "label-****(Labeling job ID)": {
        "results": [{
            "data": [{
                "direction_of_picture":"downward",
                "type":"ocr/meta"
            },
            {
                "id": "Y4ZFoC-****",
                "direction_of_text": "downward",
                "text": "Alibaba Cloud Intelligence",
                "type": "ocr/polygonLabel",
                "value": {
                    "points": [[325.08789110183716,397.47582054138184]]
                },
                "labelColor": "#67bd3a",
                "labels": "Enterprise"
            }],
            "id":"24****",
            "type":"ocr"
        }]
    }
}
```

Single-label image classification

Single-label image classification is used to find a label from a set of labels to match the content of an input image, and then attach the label to the image.

- Scenarios

  Single-label image classification applies to scenarios such as photo classification, image recognition, and image search.

- Data structure
  - Input data

    Each row in the **manifest** file contains a topic. Each topic must contain the picUrl field.

```
{"data":{"picUrl":"oss://****/img/ocr_card/img0.jpeg"}}
```

- Output data

  Each row in the **manifest** file contains a topic and the corresponding labeling result. The following code provides an example of the JSON string in each row:

```
{
    "data": {
        "picUrl": "oss://****/img/ocr_card/img0.jpeg"
    },
    "label-****(Labeling job ID)": {
        "results": [{
            "data": [{
                "data":"red",
                "id":"33****",
                "type":"survey/value"
            }],
            "id":"33****",
            "type":"survey"
        }]
    }
}
```

Multi-label image classification

Multi-label image classification is used to find multiple labels from a set of labels to match the content of an input image, and then attach the labels to the image.

- Scenarios

  Multi-label image classification applies to scenarios such as content recommendation, advertising, and image search.

- Data structure
  - Input data

    Each row in the **manifest** file contains a topic. Each topic must contain the picUrl field.

```
{"data":{"picUrl":"oss://****/img/ocr_card/img0.jpeg"}}
```

  - Each row in the **manifest** file contains a topic and the corresponding labeling result. The following code provides an example on the JSON string in each row:

```
{
    "data": {
        "picUrl": "oss://****/img/ocr_card/img0.jpeg"
    },
    "label-****(Labeling job ID)": {
        "results": [{
            "data": [{
                "data":["red","more","green"],
                "id":"33****",
                "type":"survey/multivalue"
            }],
            "id":"33****",
            "type":"survey"
        }]
    }
}
```

4.1.4.3. Create a labeling job

This topic describes how to create a labeling job.

Prerequisites

A dataset is registered. For more information about how to register a dataset, see Register a dataset.

Procedure

1. Log on to the Machine Learning Platform for AI console
2. In the left-side navigation pane of the PAI console, choose **Data Preprocessing > Smart Labeling**.
3. On the **Smart Labeling** page, click **Create Labeling Job**.
4. In the **Basic Information** step, set the parameters and click **Next**.

Parameter	Description
Task Name	The task name must be 1 to 24 characters in length, and can contain underscores (_) and hyphens (-). It must start with a letter or digit.
Description	The description must be 1 to 64 characters in length, and can contain underscores (_) and hyphens (-). It must start with a letter or digit.
Input Dataset	You can select one or more datasets to create a labeling job. The datasets must correspond to the topic of the labeling job. If no dataset is available, click **Register Dataset** next to the **Input Dataset** parameter to register a dataset.
Output Dataset Path	Select an Object Storage Service (OSS) path to store the labeling results. When you handle a labeling job, every time you click **Generate Result Dataset**, a result dataset is generated in the specified OSS path. The dataset contains the results of all topics that you have completed.

5. In the **Template** step, set the parameters and click **Next**.

Parameter		Description
Template		The template to be used in the labeling job. The system supports the following templates: ○ **Target detection** ○ **Semantic segmentation** ○ **Image comprehensive labeling** ○ **OCR template** ○ **Single-label image classification** ○ **Multi-label image classification**
Categories		The labels that are used to classify images. This parameter takes effect only if **Single-label image classification** or **Multi-label classification** is selected for the **Template** parameter.
Label Image Orientation	**Label Image Orientation**	Specifies whether to annotate image orientation. This parameter takes effect only if you select **OCR template** for the **Template** parameter.
	Flip Text	Specifies whether to annotate text orientation. If the text in an image is placed in the same direction as the image, you can turn off this switch. This parameter takes effect only if you select **OCR template** for the **Template** parameter.

Parameter	Description
Text Type	The labels that are used to classify the text. This parameter takes effect only if you select **OCR template** for the **Template** parameter.
Add Custom Label	The custom labels that are used to classify the text. This parameter takes effect only if you select **OCR template** for the **Template** parameter.
Labels	The labels that are used to classify images. Labels are displayed in different colors. This parameter takes effect only if you select **Target detection**, **Semantic segmentation**, or **Image comprehensive labeling** for the **Template** parameter.

6. In the **Labeling Policy** step, set the parameters and click **Submit**.

Parameter	Description
Dispatch Policy	The default dispatch policy is **Number of topics collected by a worker each time** and cannot be changed.
Topics per Collection	The number of topics that are collected by each worker each time. **Note** The value of the **Topics per Collection** parameter can be smaller than the total number of topics divided by the total number of workers. This allows workers with high efficiency to collect more topics and improves the overall efficiency of data labeling.
Add Worker	You can specify one or more workers. You can select both Apsara Stack tenant accounts and RAM users.

4.1.4.4. Label images

This topic describes how to label images.

Prerequisites

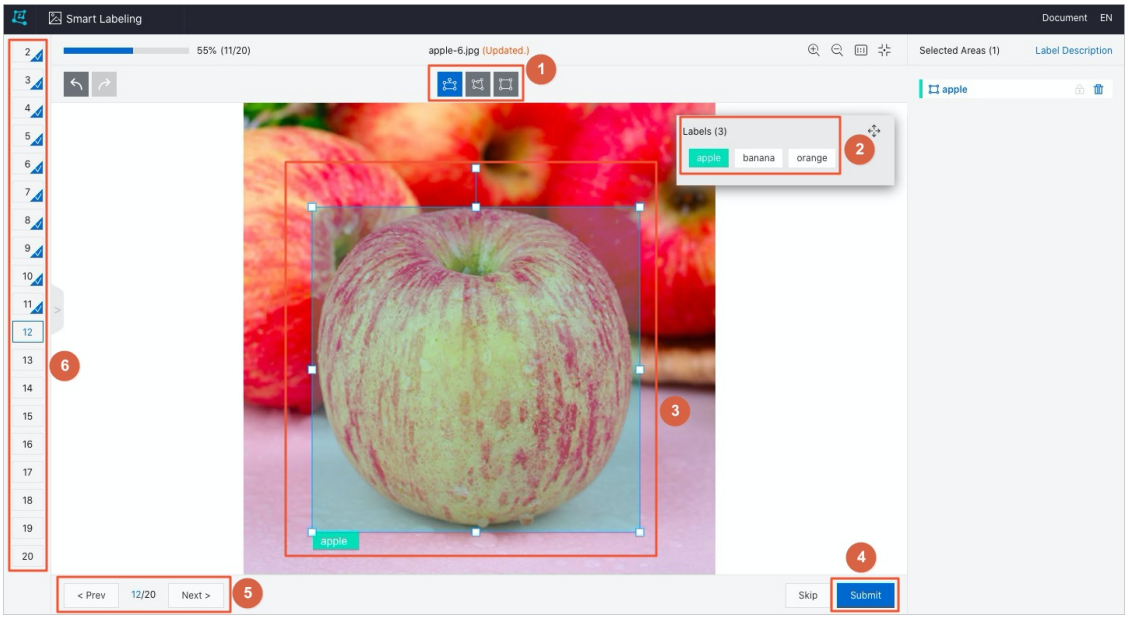
A labeling job is created or assigned to you by the administrator. For more information, see Create a labeling job.

Procedure

1. Go to the Smart Labeling page.
   i. In the left-side navigation pane of the PAI console, choose **Data Preprocessing > Smart Labeling**.
   ii. On the **Smart Labeling** page, click **My Labeling Jobs**.
   iii. In the job list, find your labeling job and click **Start** in the **Actions** column.
2. Label images.
   i. On the labeling page, click the icon.
   ii. In the **Labels** section, select a label.

   Note The system adds the selected label to each image unless you select another label.

iii. Use the selection tool to select an area in the image.



iv. (Optional)If you do not need to label the image, click **Skip**.

v. Click **Submit**.

vi. You can use one of the following methods to browse and complete topics:

- Click **Prev** or **Next** in the lower part of the Smart Labeling page.
- Click the thumbnails of the topics in the left-side navigation pane.

4.1.5. Machine Learning Studio

4.1.5.1. Quick start

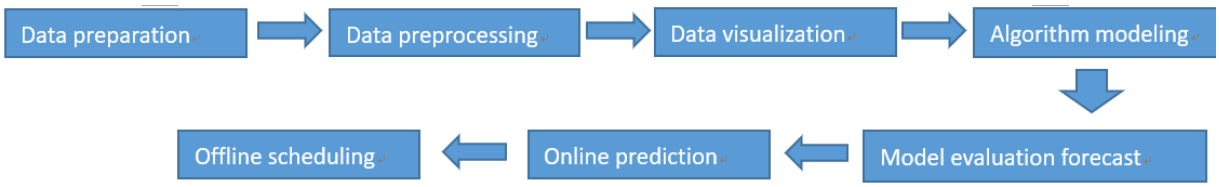
4.1.5.1.1. Overview

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 prediction evaluation

   Make predictions from and evaluate the model, and use the prediction results to create business development strategies.

6. Deploy an online model service

   Use online prediction to deploy the generated model and adjust your business strategy based on the prediction results.

7. DataWorks task scheduling

   Deploy experiments in DataStudio and run them on a regular basis.

4.1.5.1.2. Prepare data

This topic describes how to import data into the PAI console for modelling.

Prerequisites

A MaxCompute project is created and your data is imported to the MaxCompute project. You can download datasets from Machine Learning Repository.

Procedure

1. Log on to Apsara Stack Machine Learning Platform for AI.
2. In the left-side navigation pane, choose **Model Training > Studio-Modeling Visualization**.
3. On the **PAI Visualization Modeling** page, find your project and click **Machine Learning** in the **Operation** column.
4. In the left-side navigation pane, click **Experiments**.
5. On the **Experiments** page, right-click **My Experiments** and select **New Experiment** from the shortcut menu. In the New Experiment dialog box, set the **Name** and **Description** parameters and click **OK**. The **Components** pane appears.
6. In the **Components** pane, click **Data Source/Target** and drag the **Read MaxCompute Table** component to the canvas.
7. Click the **Read MaxCompute Table** component and set the parameters for the component. Enter the name of the MaxCompute table in **Table Name** in the right-side pane.
8. In the right-side pane, click the **Fields Information** tab to view the column name, data type, and the first 100 rows of data in the input table. The following figure shows the Fields Information tab.

4.1.5.1.3. Preprocess data

This topic describes how to preprocess data by using methods such as normalization, SQL scripts, and data splitting.

Prerequisites

Data is prepared before data preprocessing. For more information, see Data preparation.

Procedure

1. Log on to Apsara Stack Machine Learning Platform for AI.
2. In the left-side navigation pane, choose **Model Training > Studio-Modeling Visualization**.
3. On the **PAI Visualization Modeling** page, find your project and click **Machine Learning** in the **Operation** column.
4. In the left-side navigation pane, click **Components**.
5. In the **Components** pane, click **Tools**. Drag the **SQL Script** component to the canvas. Click **Data Preprocessing**, drag the **Normalization** component to the canvas, and then connect the components.
6. Click the **SQL Script** component. In the **SQL Script** section of the **Parameters Settings** pane, enter the following SQL scripts to convert the features from strings to numeric values.

```
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};
```

7. Click the **Normalization** component and select all columns to normalize the numeric features to values ranging from 0 to 1.
8. Click **Data Preprocessing**. Drag the **Split** component to the canvas and set the **Split Fraction** parameter 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.

4.1.5.1.4. Visualize data

This topic describes how to view the features and value distribution by using statistical analysis components.

Prerequisites

Data is preprocessed before data visualization. For more information, see Preprocess data.

Procedure

1. Log on to Apsara Stack Machine Learning Platform for AI.
2. In the left-side navigation pane, choose **Model Training > Studio-Modeling Visualization**.
3. On the **PAI Visualization Modeling** page, find your project and click **Machine Learning** in the **Operation** column.
4. In the left-side navigation pane, click **Components**.
5. In the **Components** pane, click **Statistics**. Drag the **Whole Table Statistics** component to the canvas.

Connect the components and click **Run** in the upper-left corner of the canvas.

6. After the experiment stops running, right-click the **Whole Table Statistics** component and select **View Data** from the shortcut menu. The analysis report is displayed.

4.1.5.1.5. Generate a model

This topic describes how to perform feature training and generate models by using the machine learning components.

Prerequisites

The following operations are performed before algorithm modeling: Preprocess data and learn the data characteristics and value distribution by Visualize data.

Procedure

1. Log on to Apsara Stack Machine Learning Platform for AI.
2. In the left-side navigation pane, choose **Model Training > Studio-Modeling Visualization**.
3. On the **PAI Visualization Modeling** page, find your project and click **Machine Learning** in the **Operation** column.
4. In the left-side navigation pane, click **Components**.
5. In the Components pane, choose **Modeling > Binary Classification**. Drag the **Logistic Regression for Binary Classification** component to the canvas and connect the components on the canvas.
6. Click the component, and select 13 feature columns for the **Training Feature Columns** parameter on the Fields Setting tab of the right-side pane. All parameters use the default settings.
7. Click **Run**.
8. Click **Models** in the left-side navigation pane to view the generated model.

4.1.5.1.6. Use a model for prediction and evaluation

This topic describes how to use a model to make predictions and evaluate its results by using the prediction and evaluation components.

Prerequisites

A machine learning model is generated from an experiment before predication and evaluation. For more information, see Generate a model.

Procedure

1. Log on to Apsara Stack Machine Learning Platform for AI.
2. In the left-side navigation pane, choose **Model Training > Studio-Modeling Visualization**.
3. On the **PAI Visualization Modeling** page, find your project and click **Machine Learning** in the **Operation** column.
4. In the left-side navigation pane, click **Components**.
5. In the **Components** pane, click **Modeling**. Drag the **Prediction** component to the canvas and connect the components on the canvas.
6. In the Components pane, choose **Modeling > Evaluation**. Drag the **Binary Classification Evaluation** component to the canvas and connect the components on the canvas.
7. Click **Run** in the upper-left corner of the canvas.

   During experiment execution, select a component and click the **Developer Tool** icon in the lower-right corner of the canvas to view the status of the component.

8. Right-click the **Binary Classification Evaluation** component and select **View Evaluation Report** from the shortcut menu to generate the receiver operating characteristic (ROC) curve of the LR model trained with different parameters.

4.1.5.1.7. Schedule an experiment

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 you schedule an experiment, all nodes are run in the experiment and the experiment is deployed to DataWorks.

Procedure

1. Open the **Experiments** pane.
   i. Log on to Apsara Stack Machine Learning Platform for AI.
   ii. In the left-side navigation pane, choose **Model Training > Studio-Modeling Visualization**.
   iii. On the **PAI Visualization Modeling** page, find your project and click **Machine Learning** in the **Operation** column.
   iv. In the left-side navigation pane, click **Experiments**.
2. Click **My Experiments** and click the experiment that you want to schedule.

   > **Notice** Make sure that all components are run as expected in the experiment. A green tick means that the component is successful.

3. In the upper-left corner of the canvas, choose **Deploy > DataWorks Offline Schedule** to go to DataStudio.
4. On the DataStudio page, move the pointer over the **Create** icon, choose **Machine Learning > Machine Learning Platform for AI**, and then create a machine learning node.
5. In the **Create Node** dialog box, enter the node name, select the folder where the node resides, and click **Commit**.

   > **Notice** You must select the Algorithm folder of a workflow.

   After the machine learning node is created, perform the steps on the canvas, as shown in the following figure.
6. Select the experiment from the drop-down list.
7. Configure scheduling properties for the machine learning node, including the recurrence, input, and output parameters.
8. Click the **Submit** icon. The node will be run the next day.
9. Click **Operation Center** in the upper-right corner to go to Operation Center. You can view the status of the machine learning node and the system log. You can also perform other operations, such as generating retroactive data and testing the experiment.

4.1.5.2. Components

4.1.5.2.1. 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.

4.1.5.2.2. 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.

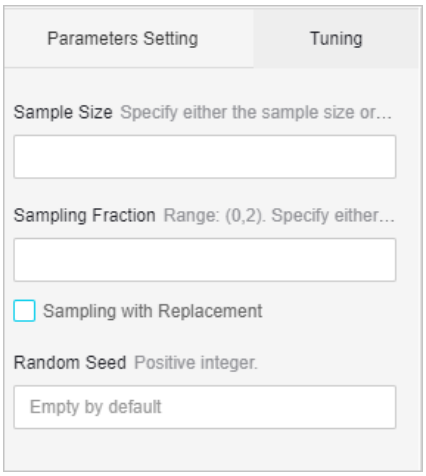
4.1.5.2.3. Data preprocessing

4.1.5.2.3.1. Sampling and filtering

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

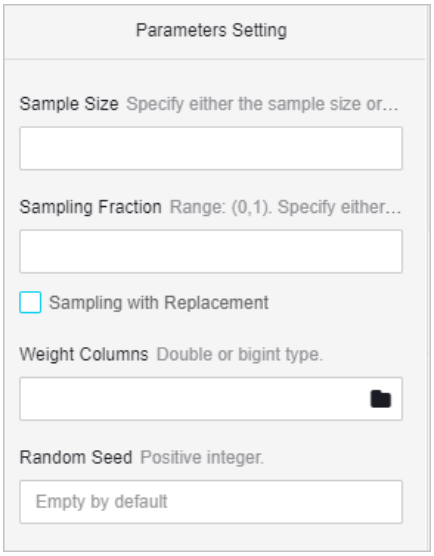
Parameters

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.
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.

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.
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.
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.

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.

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.
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.
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.

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
inputTableName	Required. The name of the input 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.
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.	-	
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.

4.1.5.2.3.2. Data merge

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.

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.

Parameter	Description
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 (,).
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 (;).

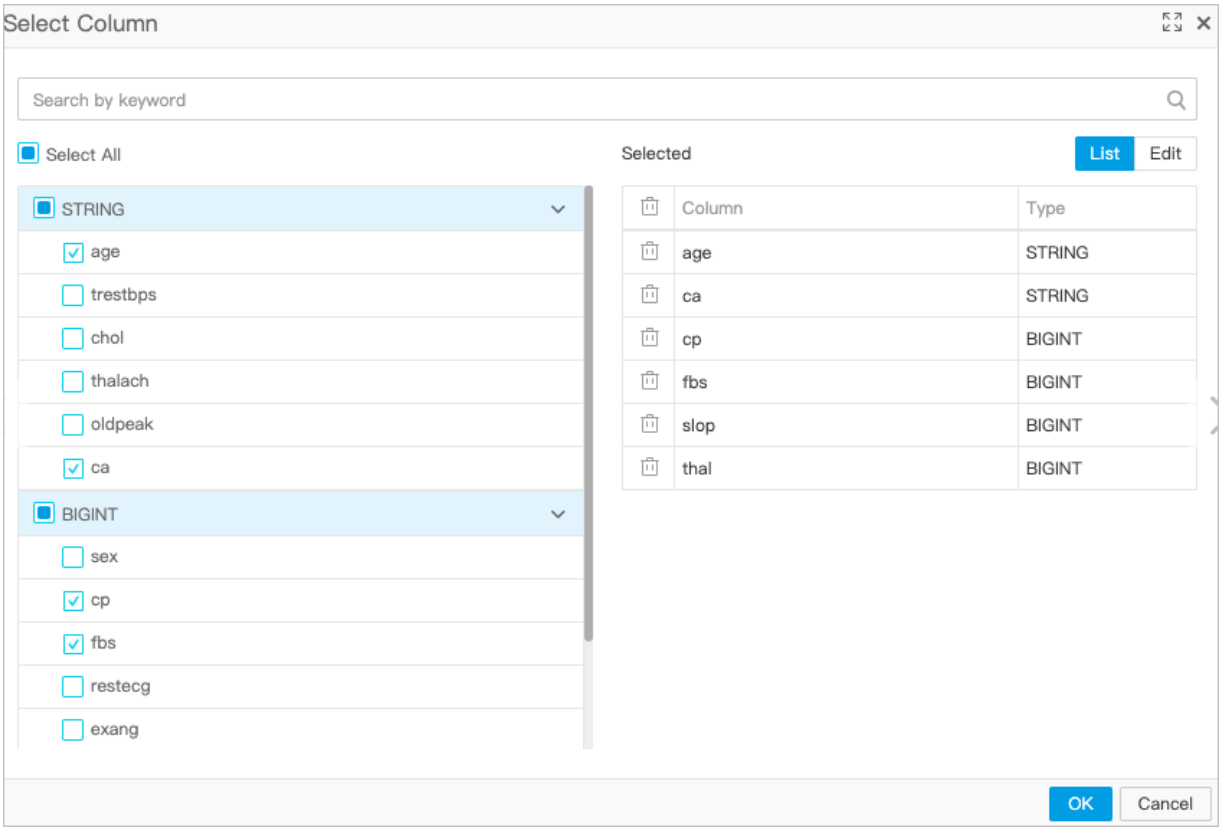
Merge rows (UNION)

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.

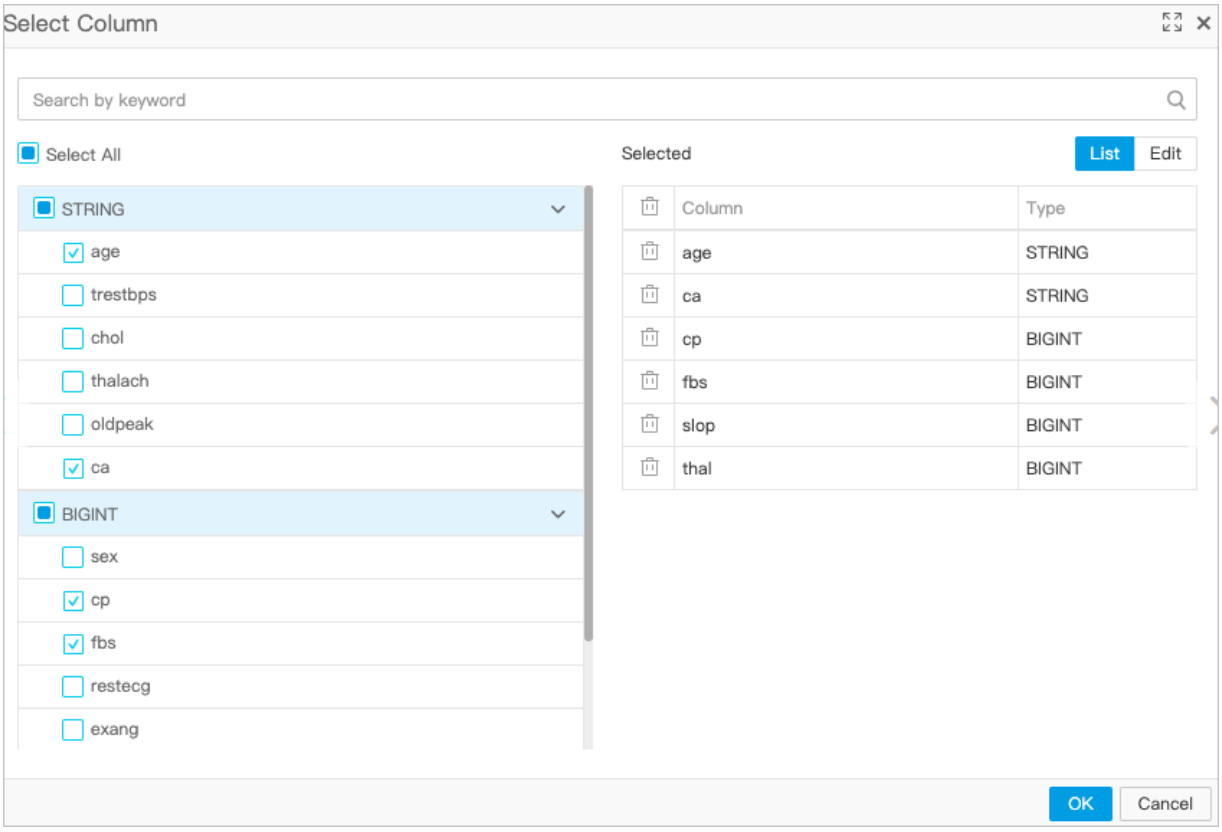
Parameter settings

- 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

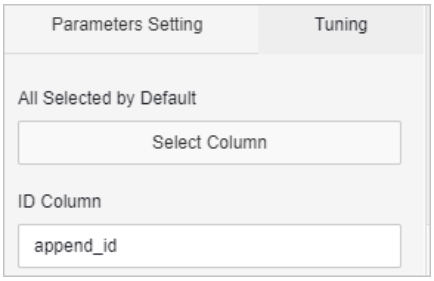
No PAI command is available.

4.1.5.2.3.3. Others

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_employed,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.
selectedColNames	The names of the columns to be retained. Separate multiple columns with commas (,).

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.	-	-
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.

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-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}} |
+------------+------------+
```

- 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}} |
+------------+------------+
```

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
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.

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.	-	-

Parameter	Description	Valid values	Default value
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

col_string	col_bigint	col_double	col_boolean	col_datetime	stdized_col_bigint	stdized_col_double
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	-1.264911064068383	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
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}}

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.

Parameter	Description	Valid values	Default value
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.
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.

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	-
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

Parameter	Description	Valid values	Default value
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.

4.1.5.2.4. Feature engineering

4.1.5.2.4.1. Feature transformation

Principal Component Analysis

Principal Component Analysis (PCA) is used to reduce dimensions.

- For more information, see Wikipedia.
- PCA supports dense data.

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_employed
-DtransType=Simple
-DcalcuType=CORR
-DcontriRate=0.9;
```

Algorithm parameters

Parameters

Parameter	Description	Default value
inputTableName	Required. The input table for PCA.	-
eigOutputTableName	Required. The output table that contains feature vectors and feature values.	-
princompOutputTableName	Required. The output table that contains the results of PCA dimension reduction and noise reduction.	-
selectedColNames	Required. The feature columns that are involved in PCA.	-
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 dimension reduction.	0.9
remainColumns	Optional. The columns retained from the original table after dimension reduction.	-

4.1.5.2.4.2. Feature importance evaluation

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

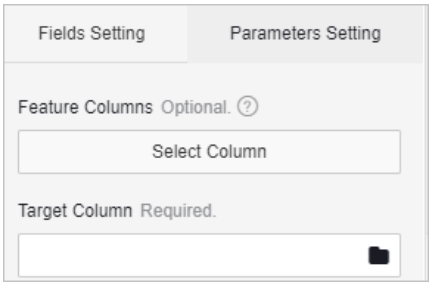
Field	Description
SST	Total sum of squares.
SSE	Sum of squared errors.
SSR	Sum of squares due to regression.
R2	Coefficient of determination.
R	Coefficient of multiple correlation.

Field	Description
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.

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.	-

Parameter	Description	Default value
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.

4.1.5.2.5. Statistical analysis

4.1.5.2.5.1. Data Pivoting

The Data Pivoting component allows you to view the distributions of feature values, feature columns, and label columns. This facilitates future data analysis. Both the sparse and dense data formats are supported.

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 that are selected from the input table.	No	N/A
outputTable	The name of the output table.	Yes	N/A

Parameter	Description	Required	Default value
mapTable	The output mapping table. The Data Pivoting component maps data of the STRING type to the INT type to facilitate machine learning and training.	Yes	N/A
selectedCols	The columns that are selected from the input table.	Yes	N/A
categoryCols	The columns of the INT or DOUBLE type that you want to use as enumeration features.	No	Empty
maxBins	The maximum number of intervals for equal-distance division of continuous features.	No	100
isSparse	Specifies whether the features are in the sparse format.	No	false
itemSpliter	The delimiter that is used to separate sparse feature items.	No	","
kvSpliter	The delimiter that is used to separate keys and values of a feature item.	No	":"
lifecycle	The lifecycle of the output table. Unit: day.	No	28

4.1.5.2.5.2. 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";
```

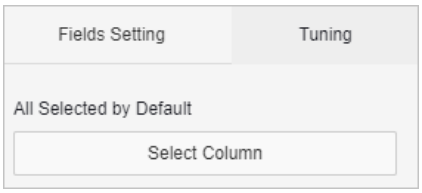
Parameters

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.

4.1.5.2.5.3. 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	-

Parameter	Description	Valid values	Default value
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

col0:double	col1:bigint	col2:double	col3:bigint	col4:double	col5:bigint	col6:double	col7:bigint	col8:double	col9:double
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
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

4.1.5.2.5.4. 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	-
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.

4.1.5.2.5.5. 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.
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.

Parameter	Description	Valid values	Default value
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

Column name	Data type
pdf	double

Output table:

```
+------------+------------+------------+------------+
| colname    | label      | x          | pdf        |
+------------+------------+------------+------------+
| col1       | NULL       | 1.0        | 0.12775155176809325 |
| col1       | NULL       | 1.0404050505050506 | 0.13042569338296 22 |
| 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.

4.1.5.2.5.6. 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

4.1.5.2.5.7. 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	-
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)

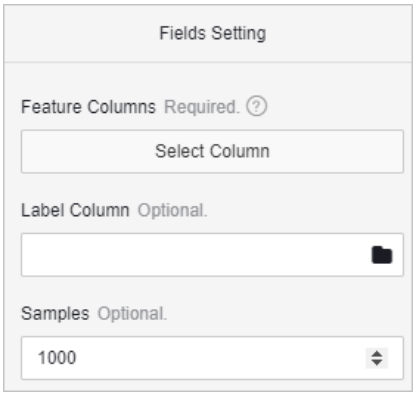
  Data:

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

4.1.5.2.5.8. 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.
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
-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

emp_var_rate	cons_price_rate	cons_conf_idx	euribor3m	y
-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
-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

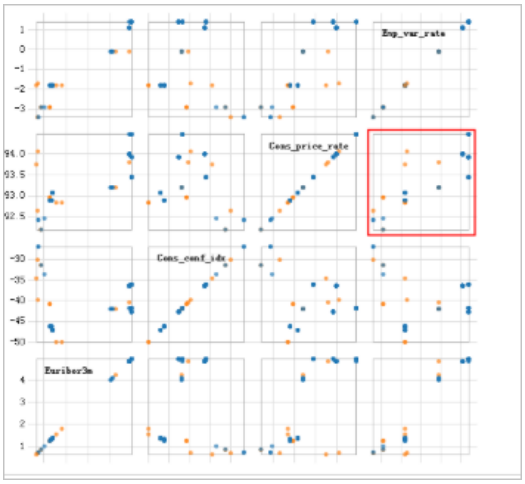
emp_var_rate	cons_price_rate	cons_conf_idx	euribor3m	y
1.1	93.994	-36.4	4.857	0
-1.8	92.893	-46.2	1.27	0
1.1	93.994	-36.4	4.857	0
1.1	93.994	-36.4	4.859	0
1.4	94.465	-41.8	4.959	0
1.1	93.994	-36.4	4.856	0
-1.8	93.075	-47.1	1.405	0
-1.8	92.843	-50.0	1.811	1
-0.1	93.2	-42.0	4.021	0
-2.9	92.469	-33.6	1.029	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
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

emp_var_rate	cons_price_rate	cons_conf_idx	euribor3m	y
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
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.

4.1.5.2.5.9. 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.
yColName	Required. The name of the column selected from table y for testing.	Column name. The type must be double or bigint.	-

Parameter	Description	Valid values	Default value
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 valu
e must 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.

4.1.5.2.5.10. 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
}
```

4.1.5.2.5.11. 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
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

Parameter	Description	Valid value	Default value
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
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
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

Quantile	col0
92	1
93	1
94	1
95	1
96	1
97	1
98	1
99	1
100	1

4.1.5.2.5.12. 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	-

Parameter	Description	Valid values	Default value
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
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

4.1.5.2.5.13. 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.

Parameter	Description
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.

4.1.5.2.5.14. 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	-

4.1.5.2.5.15. 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.

4.1.5.2.6. Machine learning

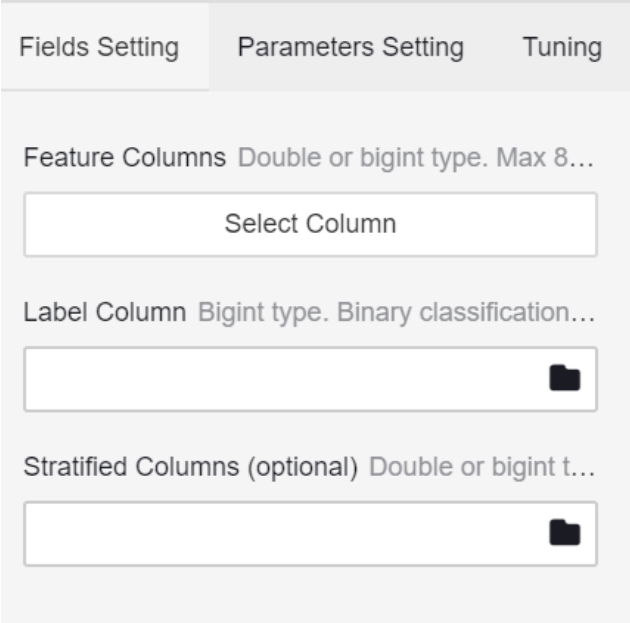
4.1.5.2.6.1. Binary classification

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.

Parameter	Description
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.
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.

Parameter	Description
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.

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.
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.

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.

Parameter	Description
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";
```

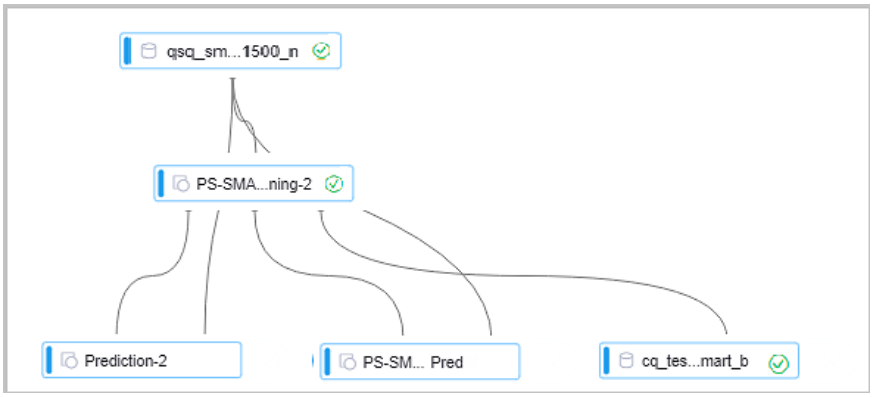
Parameters

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.

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

Command option	Parameter	Description	Valid values	Remarks
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
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.
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.
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 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.
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.

4.1.5.2.6.2. Multiclass classification

KNN

The KNN algorithm is used to resolve classification issues. For a row in a prediction table, this component selects K-nearest records from the training table. Then, it adds the row to the class that is most common among the K records.

PAI command

You can run the following PAI command to use the component:

```
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

The following table describes the parameters in the PAI command.

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 that are selected from the training table.	Column name	N/A
trainLabelColName	Required. The name of the label column that is selected from the training table.	Column name	N/A
trainTablePartitions	Optional. The partitions that are selected from the training table.	Partition name	All partitions
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 that are selected from the prediction table.	Column name	N/A
predictTablePartitions	Optional. The partitions that are selected from the prediction table.	Partition name	All partitions
appendColNames	Optional. The names of columns that are appended to the output table from the prediction table.	Column name	N/A
outputTablePartition	Optional. The partitions in the output table.	Partition name	By default, the output table is non-partitioned.
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 the sparse format.	true and false	false
itemDelimiter	Optional. The delimiter that is used to separate key-value pairs when the data in the input table is in the sparse format.	Character	The default delimiter is a space.

Parameter	Description	Valid value	Default value
kvDelimiter	Optional. The delimiter that is used to separate keys and values when the data in the input table is in the sparse format.	Character	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].	The default value is determined by the system.
memSizePerCore	Optional. The memory size of each core. Unit: MB.	A positive integer in the range of [1024, 65536]	The default value is determined by the system.
lifecycle	Optional. The lifecycle of the specified output table.	Positive integer	No lifecycle is set by default.

Example

- **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**

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.

Logistic Regression for Multiclass Classification

The common logistic regression algorithm is used for binary classification. PAI allows you to use the logistic regression algorithm for multiclass classification. The Logistic Regression for Multiclass Classification component supports both the sparse and dense data formats.

Parameters

The following figure how to set the parameters on the Parameter Setting tab for the Logistic Regression for Multiclass Classification component. The following table describes the 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 the Regularization Type parameter 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 for Multiclass Classification 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)

You can run the following PAI command to use the Logistic Regression for Multiclass Classification component.

```
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";
```

The following table describes the parameters in the PAI command.

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
modelName	The name of the output model.
regularizedLevel	Optional. The regularization coefficient. Default value: 1.0. If the regularizedType parameter 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 and is the loglikelihood deviation between two iterations. Default value: *1.0e-06*.
labelColName	The name of the label column that is selected from the input table.
featureColNames	The names of feature columns that are 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.

Random Forest

A random forest is a classifier that contains multiple decision trees. The classification result is determined by the mode of output classes of individual trees.

Procedure

1. Go to a Machine Learning Studio project. For more information, see Data preparation.
2. Drag the Random Forest component from the Components pane to the canvas and select columns. The following table describes the parameters.

Parameter	Description
Feature Columns	Optional. By default, all columns except the label and weight columns are selected.
Excluded Columns	Optional. The columns that are excluded from training. This parameter is mutually exclusive with the featureColNames parameter.
Columns Forced to Convert	Optional. Comply with the following rules to parse columns: ◦ Parse the columns of the STRING, BOOLEAN, or DATETIME type to the columns of a discrete type. ◦ Parse the columns of the DOUBLE or BIGINT type to the columns of a continuous type. ◦ To parse the columns of the BIGINT type to the columns of a categorical type, you must use the *forceCategorical* parameter to specify the type.
Weight Columns	Optional. You can select all columns except the input and label columns. Columns of the DOUBLE and BIGINT types are supported.
Label Column	You can select a column rather than the input column. Columns of the STRING, DOUBLE, and BIGINT types are supported.

3. On the **Parameter Settings** tab, set the parameters of the Random Forest component. The following table

describes the 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. The ID3, CART, and C4.5 algorithms are supported. If a forest has N trees and the condition is `algorithmTypes=[a,b]`: `[0, a)` indicates the `ID3` algorithm. `[a,b)` indicates the `CART` algorithm. `[b,n)` indicates the `C4.5` algorithm. For example, if a forest has five trees and [2,4] indicates 0, 1 indicates the ID3 algorithm, 2 and 3 indicate 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 that are randomly selected. Valid values: 1 to N. N indicates the number of features.
Minimum Number of Leaf Nodes	Optional. The minimum number of samples per leaf node. The value must be a positive integer no less than 2.
Minimum Fraction of Leaf Nodes to 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, ∞).
Number of Random Data Entries Input per Tree	The number of input random samples for a tree. Valid values: (1000, 1000000].

Note

- After the bagging method is optimized, the Random Forest component builds a forest without correlated trees based on a large dataset. Random forests are similar to the boosting method in many aspects, particularly their training processes.
- The ID3, CART, C4.5 algorithms are supported for the growth of a single tree. 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)

You can run the following PAI command to use the component:

```
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_employed"
-DisFeatureContinuous="1,1,1,1,1,1,1,1"
-DminNumPer="-1"
-DminNumObj="2"
-DinputTableName="bank_data"
-DweightColName="y"
-DtreeNum="10";
```

The following table describes the parameters in the PAI command.

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.
randomColNum	Optional. The number of features that are 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 that is 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 100,000 samples.
featureColNames	The names of feature columns that are selected from the input table for training.
isFeatureContinuous	Specifies whether the values of the feature columns are continuous or discrete. *1* indicates that the values of the feature columns are continuous. *0* indicates that the values of the feature columns are 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 that is selected from the input table. If the input table does not contain a weight column, set this parameter to *None*. The values of the weight column must be greater than 0.

Parameter	Description
treeNum	The number of trees. Valid values: (0, 1000).

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.

Overview

The following figure shows how to set the parameters for the Naive Bayes component. For more information, see Random forest.

PAI command

You can run the following PAI command to use the component:

```
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";
```

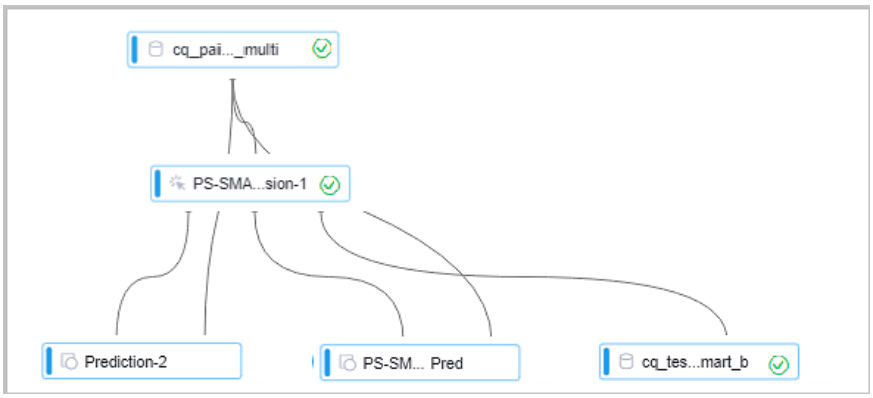
The following table describes the parameters in the PAI command.

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 that are selected from the input prediction table. If you want to select the entire table, set the parameter to None.
labelColName	The name of the label column that is selected from the input table.
featureColNames	The names of feature columns that are selected from the input table for training.
isFeatureContinuous	Specifies whether the feature for subsequent columns is continuous or discrete. *1* indicates continuous and *0* indicates 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.

PS-SMART Multiclass Classification

A parameter server (PS) is used to process a large number of offline and online training tasks. SMART is short for scalable multiple additive regression tree. PS-SMART is an algorithm that is implemented by using a PS-based gradient boosting decision tree (GBDT). PS-SMART supports training tasks for tens of billions of samples and hundreds of thousands of features. It can run training tasks 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-based approximation for training acceleration.

Quick start



In the preceding figure, a PS-SMART multiclass classification model is learned based on training data. The PS-SMART Multiclass Classification component has three output ports:

- Output model: an 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 commands

- 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

Parameter in PAI commands	Corresponding parameter in the console	Description	Valid value	Required/Default value

Parameter in PAI commands	Corresponding parameter in the console	Description	Valid value	Required/Default value
featureColNames	Feature Columns	The names of feature columns that are selected from the input table for training.	If data in the input table is in the dense format, only the columns of the BIGINT and DOUBLE types are supported. If data in the input table is key-value pairs in the sparse format, only columns of the STRING type are supported and the keys and values must be numeric.	Required.
labelColName	Label Column	The name of the label column that is selected from the input table.	The column name can be a string or a numeric value, but only numeric data can be stored in the columns. For multiclass classification, column values range from 0 to n-1. n is the number of classes.	Required.
weightCol	Weight Column	The column that contains the weight of each row of samples.	The columns of numeric data types are supported.	Optional. By default, this parameter is empty.
enableSparse	Use Sparse Format	Specifies whether data in the input table is in the sparse format. If data in the input table is in the sparse format, key-value pairs are separated by spaces whereas keys and values are separated by colons (:). Example: **1:0.3 3:0.9**.	true and false	Optional. Default value: false.
inputTableName	Input Table Name	N/A	N/A	Required.
modelName	Output Model Name	N/A	N/A	Required.

Parameter in PAI commands	Corresponding parameter in the console	Description	Valid value	Required/Default value
outputImportanceTableName	Output Feature Importance Table	N/A	N/A	Optional. By default, this parameter is empty.
inputTablePartitions	Partition	N/A	N/A	Optional. The parameter value must be in the 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 PS-SMART prediction component can be used to generate leaf node numbers.	String	Optional.
lifecycle	Lifecycle	N/A	Positive integer	Optional. Default value: 3.

- Algorithm parameters

Parameter in PAI commands	Parameter	Description	Valid value	Required/Default value
classNum	Classes	The number of classes for multiclass classification. If the number of classes is n, the values of the label column range from 0 to n-1.	A non-negative integer, greater than or equal to 3.	Required.
objective	Objective Function Type	The objective function type affects learning. You must select an appropriate objective function type. Set the parameter to **multi:softprob** for multiclass classification.	N/A	Required.

Parameter in PAI commands	Parameter	Description	Valid value	Required/Default value
metric	Evaluation Indicator Type	The evaluation metric type in the training set, which is contained in stdout of the coordinator in a logview.	**mlogloss** and **merror**	Optional. By default, this parameter is empty.
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 the parameter to 5, indicating that 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. Default value: 1.0. The default value indicates that 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. Default value: 1.0. The default value indicates that 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.

Parameter in PAI commands	Parameter	Description	Valid value	Required/Default value
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 bins is O(1.0/sketchEps). The smaller the parameter value, the more bins are generated. Typically, you do not need to change 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.
featureNum	Features	The number of features or the maximum feature ID. Specify this parameter for resource usage estimation.	Positive integer	Optional.
baseScore	Global Offset	The initial predicted values of all samples.	Real number	Optional. Default value: 0.5.

Parameter in PAI commands	Parameter	Description	Valid value	Required/Default value
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**.

- Usage notes
  - Specify different values for the objective parameter in different learning models. In the PAI console, 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**.
- Tuning parameters

Parameter in PAI commands	Corresponding parameter in the console	Description	Valid value	Required/Default value
coreNum	Cores	The number of cores. The greater the value, the faster the computing algorithm runs.	Positive integer	Optional. By default, the system determines the value.
memSizePerCore	Memory Size per Core (MB)	The memory size of each core. A value of 1024 represents 1 GB of memory.	Positive integer	Optional. By default, the system determines the value.

Example

- Generate input data

  The following sample code is used to generate data in the sparse format that consists of key-value pairs:

```
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;
```

- Configure the component for training

  Configure the training data and training component, as shown in Quick start. Select the label column as the label 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 Tuning tab. The greater the number, the faster the algorithm runs. Typically, you do not need to enter the memory size of each 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 a large number of resources are requested.

- Configure the components for 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 need only to select feature columns. All columns are selected by default, and extra columns do not affect the prediction. If data consists of key-value pairs, set the data format to sparse format and select the correct delimiter. In the SMART model, key-value pairs are separated by spaces. Therefore, the delimiter must be set to space or **\u0020** (escape expression of a space).

    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 the 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, objective column, and number of classes. The ID column can only be of a STRING-type column other than a feature column or the label 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 column of the STRING type must be selected as the label column. You can enter strings in the column but cannot be empty 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 > View Output Port 3** from the shortcut menu. The following figure shows the output feature importance table.

  In the table, the ID column lists the numbers of input features. In this example, the data is in key-value 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 I use large-scale 0-1 features?
- A: Although PS-SMART supports tasks that contain hundreds of thousands of features, such tasks consume a large number 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 directly used for training with the GBDT algorithm. Discretization is not recommended for numerical features.
- Q: Why is the result different every time even though 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 parameters 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. Different merging orders result in different tree structures, but this does not introduce too much variation to the output model. Therefore, it is a normal situation to obtain different results after the algorithm runs multiple times with the same data and same parameter settings.

> **Note**
>
> - The label column in a PS-SMART multiclass classification model supports only positive integer IDs . Class numbers range from 0 to n-1 and 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 from 0 to 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.

4.1.5.2.6.3. 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.
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.

Parameter	Description
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*.

4.1.5.2.6.4. Regression

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.
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
treeCount	Optional. The number of trees.	[1, 10000]	500

Parameter	Description	Valid values	Default value
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}

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.	-

Parameter	Description	Valid values	Default value
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
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	99.26757440771128	NULL	NULL	NULL	goodness

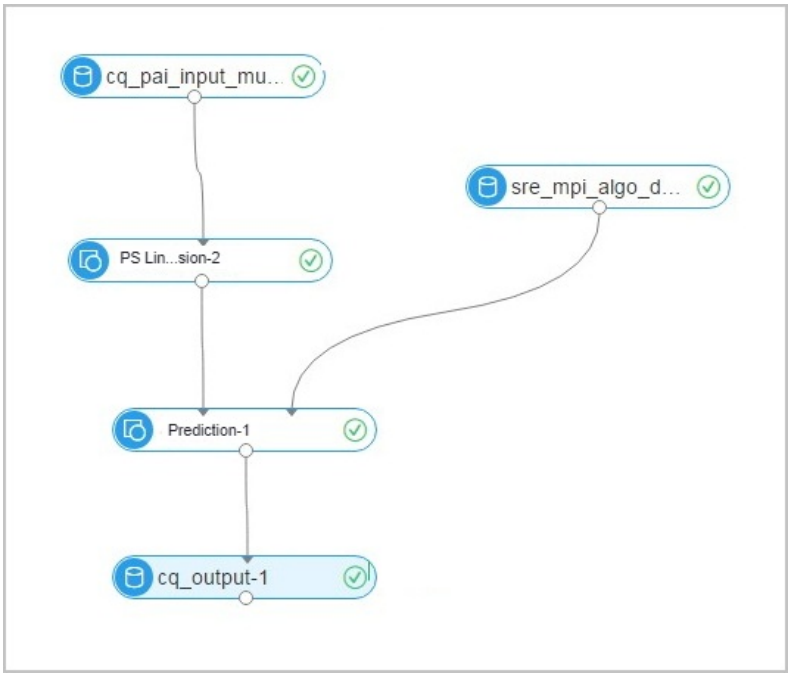
- Output prediction table: lm_test_input_predict_out.

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}

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
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

Command option	Parameter	Description	Valid values	Default value
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
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.

Command option	Parameter	Description	Valid values	Default value
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

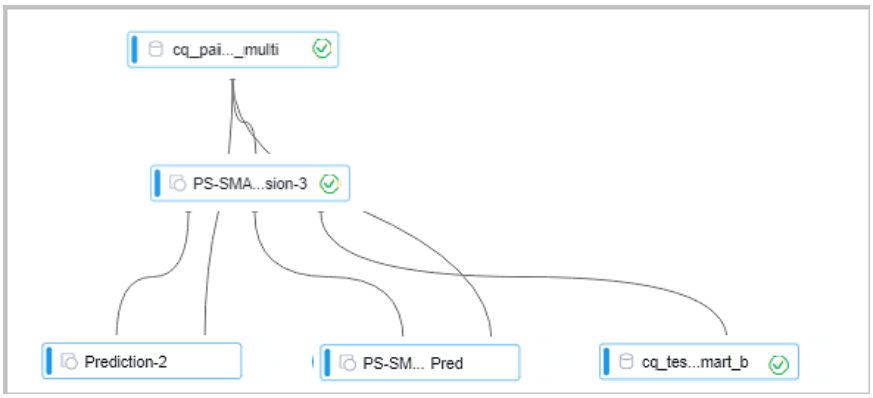
In the key-value format, feature IDs must be positive integers, and feature values must be real numbers. If feature IDs are strings, use the serialization component to serialize them. If the feature values are classification type strings, perform feature engineering, such as discretization.

PS-SMART Regression

A parameter server (PS) is used to train a large number of models online and offline. SMART is short for scalable multiple additive regression tree. PS-SMART is an algorithm that is implemented by using a PS-based gradient boosting decision tree (GBDT). PS-SMART supports training tasks for tens of billions of samples and hundreds of thousands of features. It can run training tasks 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-based approximation for training acceleration.

Quick start

The following figure shows the output details of the PS-SMART Regression component.



The PS-SMART Regression component has three output ports:

- Output model: an 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 commands

You can run the following PAI commands to use the PS-SMART Regression component and the prediction component.

- 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

- The following table describes the data parameters in PAI commands.

Parameter in PAI commands	Corresponding parameter in the console	Description	Valid value	Required/Default value

Parameter in PAI commands	Corresponding parameter in the console	Description	Valid value	Required/Default value
featureColNames	Feature Columns	The names of feature columns that are selected from the input table for training.	If data in the input table is in the dense format, only the columns of the BIGINT and DOUBLE types are supported. If data in the input table is key-value pairs in the sparse format, only columns of the STRING type are supported and the keys and values must be numeric.	Required.
labelColName	Label Column	The name of the label column that is selected from the input table.	The column name can be a string or a numeric value, 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	The column that contains the weight of each row of samples.	The columns of numeric data types are supported.	Optional. By default, this parameter is empty.
enableSparse	Use Sparse Format	Specifies whether data in the input table is in the sparse format. If data in the input table is in the sparse format, key-value pairs are separated by spaces whereas keys and values are separated by colons (:). Example: **1:0.3 3:0.9**.	true and false	Optional. Default value: false.
inputTableName	The name of the input table.	N/A	N/A	Required.
modelName	Output Model Name	N/A	N/A	Required.
outputImportanceTableName	Output Feature Importance Table	N/A	N/A	Optional. By default, this parameter is empty.

Parameter in PAI commands	Corresponding parameter in the console	Description	Valid value	Required/Default value
inputTablePartitions	Partition	N/A	N/A	Optional. The parameter value must be in the 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 PS-SMART prediction component can be used to generate leaf node numbers.	String	Optional.
lifecycle	Lifecycle	N/A	Positive integer	Optional. Default value: 3.

- The following table describes the algorithm parameters in PAI commands.

Parameter in PAI commands	Corresponding parameter in the console	Description	Valid value	Required/Default value
objective	Objective Function Type	The objective function type affects learning. You must select an appropriate objective function type. Multiple loss functions are available for regression. For more information, see Usage notes.	For more information, see Usage notes.	Required. The default type is linear regression.
metric	Evaluation Indicator Type	The evaluation metric type 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 Usage notes.	For more information, see Usage notes.	Optional. By default, this parameter is empty.
treeCount	Trees	The number of trees. The training time is proportional to this number.	Positive integer	Optional. Default value: 1.

Parameter in PAI commands	Corresponding parameter in the console	Description	Valid value	Required/Default value
maxDepth	Maximum Tree Depth	The maximum depth of a tree. We recommend that you set the parameter to 5, indicating that 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. Default value: 1.0. The default value indicates that 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. Default value: 1.0. The default value indicates that 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.
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.

Parameter in PAI commands	Corresponding parameter in the console	Description	Valid value	Required/Default value
sketchEps	Sketch-based Approximate Precision	The threshold for selecting quantiles when you build a sketch. The number of bins is O(1.0/sketchEps). The smaller the parameter value, the more bins are generated. Typically, you do not need to change 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.
featureNum	Features	The number of features or the maximum feature ID. Specify this parameter for resource usage estimation.	Positive integer	Optional.
baseScore	Global Offset	The initial 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**.

Parameter in PAI commands	Corresponding parameter in the console	Description	Valid value	Required/Default value
tweedieVarPower	N/A	The relationship between the variance and mean value of Tweedie distribution. Example: `$Var(y) ~ E(y)^tweedie_variance_power$`.	(1,2)	Optional. Default value: 1.5.

- The following table describes the tuning parameters in PAI commands.

Parameter in PAI commands	Corresponding parameter in the console	Description	Valid value	Required/Default value
coreNum	Cores	The number of cores. The greater the value, the faster the computing algorithm runs.	Positive integer	Optional. By default, the system determines the value.
memSizePerCore	Memory Size per Core (MB)	The memory size of each core. A value of 1024 represents 1 GB of memory.	Positive integer	Optional. By default, the system determines the value.

Usage notes

When you set algorithm parameters, take note of the following items:

- Specify different values for the objective parameter in different learning models. On the web UI, the PS-SMART Regression component provides multiple types of objective functions.

```
reg:linear (Linear regression) // The value range of the label column is (-∞, +∞).
reg:logistic (Logistic regression) // The value range of the label column is [0, 1].
count:poisson (Poisson regression for count data, output mean of poisson distribution) // The values of the label column must be greater than or equal to 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)) // The values of the label column must be greater than or equal to 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).)  // The values of the label column must be greater than or equal to 0.
```

- Metrics for these objective function types are:
  - rmse: Rooted Mean Square Error for the `reg:linear` objective function type.
  - mae: Mean Absolute Error for the `reg:linear` objective function type.
  - poisson-nloglik: Negative Loglikelihood for Poisson Regression for the `count:poisson` objective function type.
  - gamma-deviance: Residual Deviance for Gamma Regression for the `reg:gamma` objective function type.
  - gamma-nloglik: Negative Log-Likelihood for Gamma Regression for the `reg:gamma` objective function type.

- tweedie-nloglik: Negative Log-Likelihood for Tweedie Regression for the `reg:tweedie` objective function type.

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 I use large-scale 0-1 features?
- A: Although PS-SMART supports tasks that contain hundreds of thousands of features, such tasks consume a large number 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 directly used for training with the GBDT algorithm. Discretization is not recommended for numerical features.
- Q: Why is the result different every time even though 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 parameters 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. Different merging orders result in different tree structures, but this does not introduce too much variation to the output model. Therefore, it is a normal situation to obtain different results after the algorithm runs multiple times with the same data and same parameter settings.

> **Note**
>
> - The label 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 the feature values are classification type strings, perform feature engineering, such as discretization.

4.1.5.2.6.5. 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
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

Parameter	Description	Valid value	Default value
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
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

4.1.5.2.6.6. Evaluation

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.	-

Parameter	Description	Default value
inputTablePartitions	Optional. The partitions selected from the input table for training.	All partitions of the input table are selected by default.
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.

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
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

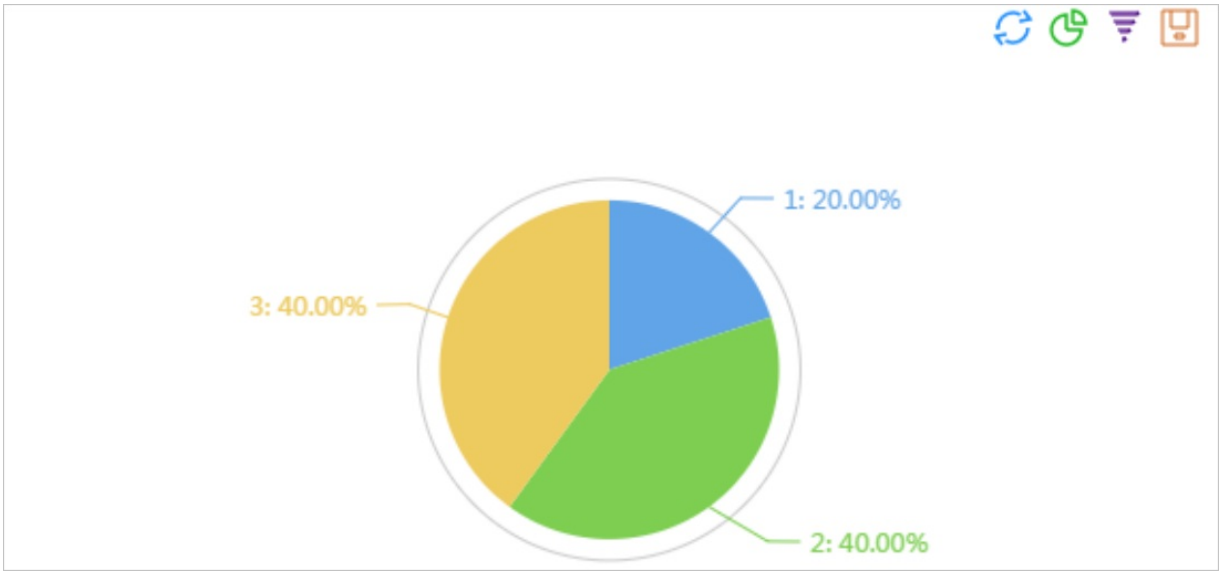
```
PAI -name cluster_evaluation
-project algo_public
-DinputTableName=pai_cluster_evaluation_test_input
-DselectedColNames=f0,f3
-DmodelName=pai_kmeans_test_model
-DoutputTableName=pai_ft_cluster_evaluation_out;
```

- 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.

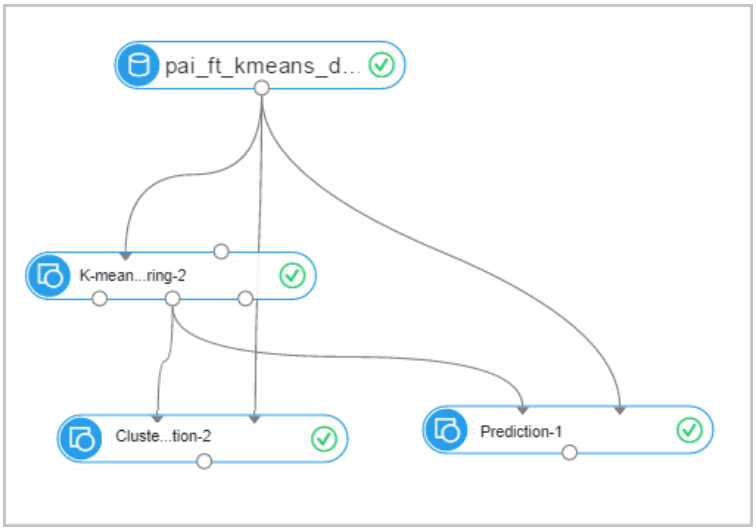
PaiWeb demonstration

Clustering model evaluation



PaiWeb-Pipeline

PaiWeb-Pipeline



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.
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%.

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.
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.

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.	-	-
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.

4.1.5.2.6.7. 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.

Parameter	Description
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.

4.1.5.2.7. Deep learning (must be separately activated)

Note The deep learning components are applicable only to Intel servers, but not to Hygon servers.

4.1.5.2.7.1. 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**.

4.1.5.2.7.2. 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.

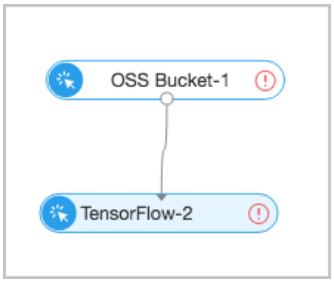
RAM authorization

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.



4.1.5.2.7.3. 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.
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
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

Parameter	Description	Valid values	Default value
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
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.

4.1.5.2.8. Time series

4.1.5.2.8.1. 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
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 value	Default value
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.
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
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

Cycle	p1
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

4.1.5.2.8.2. 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	-
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	-

Parameter	Description	Valid values	Default value
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.
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.

Parameter	Description	Valid values	Default value
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
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.
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
1950	2
1951	3
1951	4
1952	5
1953	6
1954	7
1955	8
1956	9
1957	10

Period	p1
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.

4.1.5.2.9. Text analysis

4.1.5.2.9.1. 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
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

Parameter	Description	Valid values	Default value
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

4.1.5.2.9.2. 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.	-	-

Parameter	Description	Valid values	Default value
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.

4.1.5.2.9.3. 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.	-	-

Parameter	Description	Valid values	Default value
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.
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.

Parameter	Description	Valid values	Default value
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

4.1.5.2.9.4. 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.	-
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.

Parameter	Description	Default value
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

id	word	count
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

4.1.5.2.9.5. 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.
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

Parameter	Description	Valid values	Default value
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.
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

4.1.5.2.9.6. 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

Parameter	Description	Valid values	Default value
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
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

4.1.5.2.9.7. 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
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

Parameter	Description	Valid values	Default value
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.

4.1.5.2.9.8. 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	-
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

Parameter	Description	Valid values	Default value
maxIter	Optional. The maximum number of iterations of the TextRank algorithm.	-	100
epsilon	Optional. The convergence residual threshold of the TextRank algorithm.	-	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.	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

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 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
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

4.1.5.2.9.9. 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.
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.

4.1.5.2.9.10. 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.
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

Parameter	Description	Valid values	Default value
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
..	...	...	...

4.1.5.2.9.11. 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.	-	-

Parameter	Description	Valid values	Default value
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.
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

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.	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

4.1.5.2.9.12. 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.
minCount	The minimum word truncation frequency. Words that appear for a number of times less than this value are filtered out.	[0, 2e63]	5

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.
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
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

word1	word2	word1_count	word2_count	co_occurrences_count	pmi
w6	w9	7	12	1	-0.9650808960435872
w7	w8	4	7	2	0.8266785731844679
w8	w8	7	7	1	-0.4260843953109014
w8	w9	7	12	2	-0.2719337154836418
w9	w9	12	12	2	-0.8109302162163288

4.1.5.2.9.13. 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.	-
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.

4.1.5.2.9.14. TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) is a common weighting technique for information retrieval and text mining. TF-IDF is used by search engines as a tool in scoring and ranking the relevance of a document for a given search query.

TF-IDF is also 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. 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.

Example

The output table in the example of the Word Frequency Statistics component is used as the input table for the TF-IDF component. Set the following parameters for the TF-IDF component.

- Document ID Column: id
- Word Column: word
- Word Counting Column: count

The output table contains the following columns: docid, word, word_count, total_word_count, doc_count, total_doc_count, tf, idf, and tfidf. word_count indicates the number of times a word appears in a document. total_word_count indicates the total number of words in a document. doc_count indicates the number of documents that contain a word. total_doc_count indicates the total number of documents.

PAI command

You can run the following PAI command to use the component:

```
pai -name tfidf
-project algo_public
-DinputTableName=rgdoc_split_triple_out
-DdocIdCol=id
-DwordCol=word
-DcountCol=count
-DoutputTableName=rg_tfidf_out;
```

Algorithm parameters

The following table describes the parameters in the PAI command.

Parameter	Description	Valid value	Required/Default value
inputTableName	The name of the input table.	Table name	Required.
inputTablePartitions	The partitions that are selected from the input table for word splitting.	This value must be in the `partition_name=value` format. If you want to specify multiple levels of partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	Optional. By default, all partitions in the input table are selected.
docIdCol	The name of the document ID column.	Only one column can be specified.	Required.
wordCol	The name of the word column.	Only one column can be specified.	Required.

Parameter	Description	Valid value	Required/Default value
countCol	The name of the count column.	Only one column can be specified.	Required.
outputTableName	The name of the output table.	Table name	Required.
outputTablePartition	The partitions in the output table.	Partition name	Optional. By default, the output table is non-partitioned.

4.1.5.2.9.15. PLDA

Latent Dirichlet allocation (LDA) is a topic model that provides topics of each document based on probability distribution. LDA is an unsupervised learning algorithm. You need only to specify the number of topics in a document set by using K. You do not need to manually annotate training sets. K is the Topics parameter of the PLDA component. LDA is used to recognize texts, classify texts, and calculate the similarity between texts in the text mining field.

Input parameters

The following figure shows the parameters of the PLDA component on the Parameters Setting tab. The following table describes the parameters.

Parameter	Description
Topics	The number of topics that are generated by LDA.
Alpha	The prior Dirichlet distribution parameter of `P(z/d)`.
Beta	The prior 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 value must be a positive integer. Default value: 150.

Note z represents topics, w represents words, and d represents documents.

Input and output settings

- **Input**:

  The data must be sparse arrays and can be converted by using the Convert Row, Column, and Value to KV Pair component.

  The following figure shows the input format.

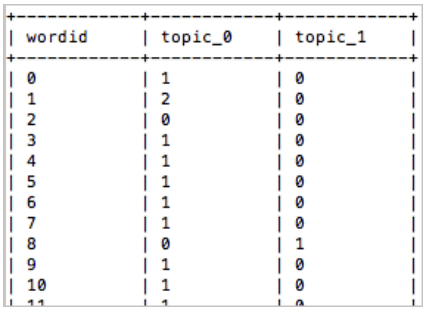
```
| id          | features |
+-------------+----------+
| 2           | 38:3.0,39:1.0,40:3.0,41:1.0,42:1.0,43:2.0,44:1.0,45:1.0,46:1.0,47:1.0,48:1.0,49:2.0,50:1.0,51:1.0,52:1.0,53:1.0,54:1.0,55:1.0,5
6:1.0,57:1.0,58:1.0,59:1.0,60:1.0,61:1.0,62:1.0,63:1.0,64:1.0,65:1.0,66:1.0,67:1.0,68:1.0,69:1.0,70:1.0,71:1.0,72:1.0,73:1.0,74:1.0,75:1.0,76:
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: key-value data of words and word frequencies.

- **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 figure 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

PAI command

You can run the following PAI command to use the component:

```
pai -name PLDA
-project algo_public
-DinputTableName=lda_input
-DtopicNum=10
-topicWordTableName=lda_output;
```

Algorithm parameters

The following table describes the parameters in the PAI command.

Parameter	Description	Valid value	Required/Default value
inputTableName	The name of the input table.	Table name	Required.
inputTablePartitions	The partitions that are selected from the input table for word splitting.	This value must be in the `partition_name=value` format. If you want to specify multiple levels of partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	Optional. By default, all partitions in the input table are selected.
selectedColNames	The names of the columns that are selected from the input table for LDA.	Separate column names with commas (,).	Optional. By default, all columns in the input table are selected.
topicNum	The number of topics.	[2, 500]	Required.
kvDelimiter	The delimiter that is used to separate keys and values.	Space, comma (,), colon (:)	Optional. Default value: colon (:).
itemDelimiter	The delimiter that is used to separate keys.	Space, comma (,), colon (:)	Optional. By default, keys are separated by spaces.

Parameter	Description	Valid value	Required/Default value
alpha	The prior Dirichlet distribution parameter of P(z/d).	(0, ∞)	Optional. Default value: 0.1.
beta	The prior Dirichlet distribution parameter of P(w/z).	(0, ∞)	Optional. Default value: 0.01.
topicWordTableName	The name of the topic-word frequency contribution table.	Table name	Required.
pwzTableName	The name of the P(w/z) table.	Table name	Optional. By default, no P(w/z) table is generated.
pzwTableName	The name of the P(z/w) table.	Table name	Optional. By default, no P(z/w) table is generated.
pdzTableName	The name of the P(d/z) table.	Table name	Optional. By default, no P(d/z) table is generated.
pzdTableName	The name of the P(z/d) table.	Table name	Optional. By default, no P(z/d) table is generated.
pzTableName	The name of the P(z) table.	Table name	Optional. By default, no P(z) table is generated.
burnInIterations	The number of burn-in iterations.	Positive integer	Optional. This value must be smaller than the total number of iterations. Default value is 100.
totalIterations	The number of iterations.	Positive integer	Optional. Default value: 150.

4.1.5.2.9.16. Word2Vec

Word2Vec is an open-source algorithm used to convert words into vectors. The Word2Vec component uses a neural network to map words to vectors in the K-dimensional space based on extensive training. The component supports operations on the vectors to show the semantics of the vectors.

For more information about the tool kit provided by Google Word2Vec, see word2vec.

Input parameters

The Word2Vec component has the following core settings:

- Word Feature Dimension: We recommend that you specify a value from 0 to 1000.
- Downsampling Threshold: We recommend that you specify a value from 1e-3 to 1e-5.
- Input: uses a word column and a word list as the input data.
- Output: generates a word vector table and a word list.

PAI command

You can run the following PAI command to use the component:

```
pai -name Word2Vec
-project algo_public
-DinputTableName=w2v_input
-DwordColName=word
-DoutputTableName=w2v_output;
```

Algorithm parameters

The following table describes the parameters in the PAI command.

Parameters

Parameter	Description	Valid value	Required/Default value
inputTableName	The name of the input table.	Table name	Required.
inputTablePartitions	The partitions that are selected from the input table for word splitting.	This value must be in the `partition_name=value` format. If you want to specify multiple levels of partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	Optional. By default, all partitions in the input table are selected.
wordColName	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	Required.
inVocabularyTableName	The name of the input word list, which contains the wordcount output of inputTableName.	Table name	Optional. By default, word count is performed for the input table.
inVocabularyPartitions	The partitions in the input word list.	Partition name	Optional. By default, all partitions in the table specified by inVocabularyTableName are selected.
layerSize	The dimension of word features.	0-1000	Optional. Default value: 100.
cbow	The language model.	1: cbow 0: skip-gram	Optional. Default value: 0.
window	The window size of words.	Positive integer	Optional. Default value: 5.
minCount	The minimum frequency of words.	Positive integer	Optional. Default value: 5.

Parameter	Description	Valid value	Required/Default value
hs	Specifies whether to use hierarchical softmax.	1: Hierarchical softmax is used. 0: Hierarchical softmax is not used.	Optional. Default value: 1.
negative	NEGATIVE SAMPLING	0: Negative sampling is disabled. Recommended value range: 5 to 10.	Optional. Default value: 0.
sample	The downsampling threshold.	0 or smaller values: Downward sampling is disabled. Recommended value range: 1e-3 to 1e-5.	Optional. Default value: 0.
alpha	The initial learning rate.	A value greater than 0	Optional. Default value: 0.025.
iterTrain	The number of training iterations.	A value greater than or equal to 1	Optional. Default value: 1.
randomWindow	Specifies whether to randomly set the size of the window.	1: The window size is randomly generated in a range from 1 to 5. 0: The window size is determined by the window parameter.	Optional. Default value: 1.
outVocabularyTableName	Optional. The name of the output word list.	Table name	Optional. By default, the **output word list** is generated.
outVocabularyPartition	The partition in the output word list.	Partition name	Optional. By default, the output word list is non-partitioned.
outputTableName	Output table	Table name	Required.
outputPartition	The information about partitions in the output table.	Partition name	Optional. By default, the output table is non-partitioned.

4.1.5.2.10. Network analysis

4.1.5.2.10.1. K-Core

The k-core of a graph is the subgraph that remains after all vertices with a degree less than or equal to K are removed. If a vertex belongs to the k-core but is not included in the (k+1)-core, the coreness of the vertex is k. Therefore, the coreness of a vertex whose degree is 1 must be 0. The largest coreness among the corenesses of all vertices is considered to be the coreness of the graph.

Input parameters

The core parameter k of the K-Core component specifies the coreness. This parameter is required. The default value is 3.

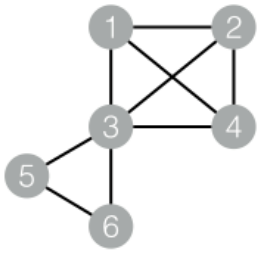
Sample test data

Execute the following SQL statements to generate test 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;
```

The Graph structure figure shows the graph structure of the test data.

Graph structure



Set the k parameter to 2. The Output figure 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

You can run the following PAI command to use the component:

```
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

The following table describes the parameters in the PAI command.

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	N/A
inputEdgeTablePartitions	The partitions that are selected from the input edge table.	No	All partitions
fromVertexCol	The start vertex column in the edge table.	Yes	N/A
toVertexCol	The end vertex column in the edge table.	Yes	N/A
outputTableName	The name of the output table.	Yes	N/A
outputTablePartitions	The partitions in the output table.	No	N/A
lifecycle	The lifecycle of the output table.	No	N/A
workerNum	The number of processes.	No	N/A
workerMem	The memory size of each process. Unit: MB.	No	4096
splitSize	The size of each part after data is split. Unit: MB.	No	64
k	The coreness.	Yes	3

4.1.5.2.10.2. Single-source Shortest Path

The single-source shortest path (SSSP) refers to the shortest path between a vertex and all other vertices. The shortest path is calculated by using the Dijkstra algorithm.

Parameter settings

The core parameter of the Single-source Shortest Path component is Start Vertex ID. You must set this parameter to specify the start vertex that is used to calculate the shortest path.

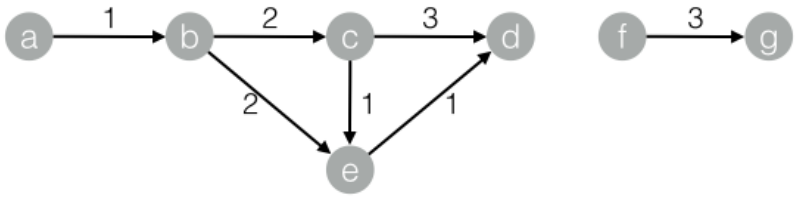
Sample test data

Execute the following SQL statements to generate test 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 ;
```

The Graph structure figure shows the graph structure of the test data.

Graph structure



Output

The following figure shows the output of the Single-source Shortest Path component that is run based on the preceding test data.

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

You can run the following PAI command to use the component:

```
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

The following table describes the parameters in the PAI command.

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	N/A
inputEdgeTablePartitions	The partitions that are selected from the input edge table.	No	All partitions
fromVertexCol	The start vertex column in the input edge table.	Yes	N/A
toVertexCol	The end vertex column in the input edge table.	Yes	N/A
outputTableName	The name of the output table.	Yes	N/A
outputTablePartitions	The partitions in the output table.	No	N/A
lifecycle	The lifecycle of the output table.	No	N/A
workerNum	The number of processes.	No	N/A
workerMem	The memory size of each process. Unit: MB.	No	4096
splitSize	The size of each part after data is split. Unit: MB.	No	64
startVertex	The ID of the start vertex.	Yes	N/A
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	N/A

4.1.5.2.10.3. Page Rank

PageRank is an algorithm that is used to sort and calculate the rankings of web pages based on their links.

Introduction

The PageRank algorithm implements ranking based on the following principle: The more links 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 an 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 following picture shows the PageRank formula with the link weight.

$$W(A) = (1 - d) + d * (\sum_{i} W(i) * C(Ai))$$

In the formula, W(i) indicates the weight of Node i, C(Ai) indicates the link weight, and d indicates the damping coefficient. W(A) indicates the influence index of each user and represents the node weight after the algorithm iteration becomes stable.

Input parameters

The core parameter of the Page Rank component is **Maximum Iterations**. You can use this parameter to specify the number of iterations before the algorithm automatically converges. The default value is 30 and this parameter is optional.

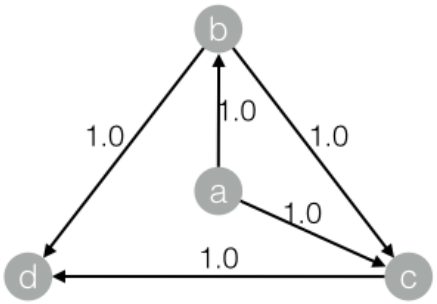
Sample test data

You can execute the following SQL statements to generate test 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 ;
```

The Graph structure figure shows the graph structure of the test data.

Graph structure



Output

The following figure shows the output of the Page Rank component that is run based on the preceding test data.

node	weight
a	0.0375
b	0.06938
c	0.12834
d	0.20556

PAI command

You can run the following PAI command to use the component:

```
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

The following table describes the parameters in the PAI command.

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	N/A
inputEdgeTablePartitions	The partitions that are selected from the input edge table.	No	All partitions
fromVertexCol	The start vertex column in the input edge table.	Yes	N/A
toVertexCol	The end vertex column in the input edge table.	Yes	N/A

Parameter	Description	Required	Default value
outputTableName	The name of the output table.	Yes	N/A
outputTablePartitions	The partitions in the output table.	No	N/A
lifecycle	The lifecycle of the output table.	No	N/A
workerNum	The number of processes.	No	N/A
workerMem	The memory size of each process. Unit: MB.	No	4096
splitSize	The size of each part after data is split. Unit: MB.	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	N/A
maxIter	The maximum number of iterations.	No	30

4.1.5.2.10.4. 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 updates.

Parameter settings

The core parameter of this component is maxIter, which indicates the maximum number of iterations. This parameter is optional. Default value: 30.

Examples - Testing data

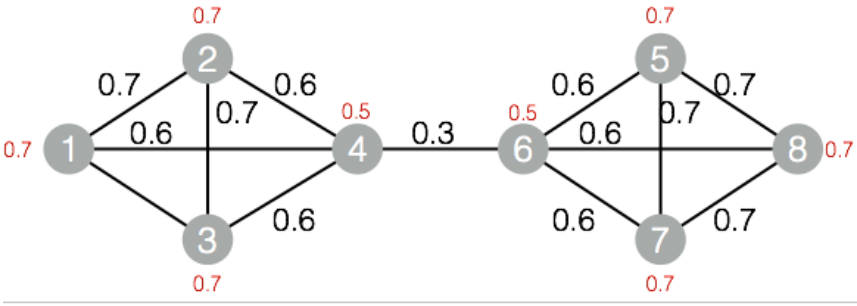
Execute the following SQL statements to generate testing data:

```
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

The following figure shows the output based on the testing data.

node	group_id
1	1
2	1
3	1
4	1
5	5
6	5
7	5
8	5

PAI command

You can run the following Machine Learning Platform for AI (PAI) command to use this component:

```
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

The following table describes the parameters that are used in the PAI command.

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	N/A

Parameter	Description	Required	Default value
inputEdgeTablePartitions	The partitions that are 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	N/A
toVertexCol	The end vertex column in the input edge table.	Yes	N/A
inputVertexTableName	The name of the input vertex table.	Yes	N/A
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	N/A
outputTableName	The name of the output table.	Yes	N/A
outputTablePartitions	The partitions in the output table.	No	N/A
lifecycle	The lifecycle of the output table.	No	N/A
workerNum	The number of workers.	No	N/A
workerMem	The memory size per worker, in MB.	No	4096
splitSize	The data split size, in MB.	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	N/A
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	N/A
randSelect	Specifies whether the maximum label value is to be randomly selected.	No	false
maxIter	The maximum number of iterations.	No	30

4.1.5.2.10.5. 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

This algorithm component uses the following core parameters:

- alpha: the damping coefficient. Default value: 0.8.
- epsilon: the convergence coefficient. Default value: 0.000001.

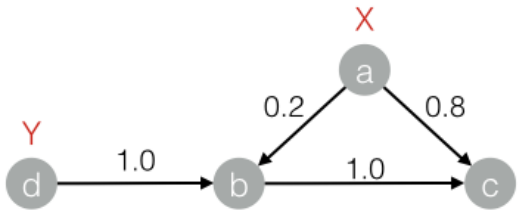
Examples - Testing data

Execute the following SQL statements to generate testing 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

The following figure shows the output based on the testing data.

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

You can run the following Machine Learning Platform for AI (PAI) command to use this algorithm component:

```
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

The following table describes the parameters that are used in the PAI command.

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	N/A
inputEdgeTablePartitions	The partitions that are 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	N/A

Parameter	Description	Required	Default value
toVertexCol	The end vertex column in the input edge table.	Yes	N/A
inputVertexTableName	The name of the input vertex table.	Yes	N/A
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	N/A
vertexLabelCol	The vertex label column in the input vertex table.	Yes	N/A
outputTableName	The name of the output table.	Yes	N/A
outputTablePartitions	The partitions in the output table.	No	N/A
lifecycle	The lifecycle of the output table.	No	N/A
workerNum	The number of workers.	No	N/A
workerMem	The memory size per worker, in MB.	No	4096
splitSize	The data split size, in MB.	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	N/A
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	N/A
alpha	The damping coefficient.	No	0.8
epsilon	The convergence coefficient.	No	0.000001
maxIter	The maximum number of iterations.	No	30

4.1.5.2.10.6. 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.

PAI command

You can run the following Machine Learning Platform for AI (PAI) command to use this component:

```
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

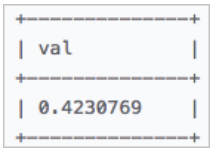
Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	N/A
inputEdgeTablePartitions	The partitions that are selected from the input edge table.	No	The whole table is selected by default.
fromVertexCol	The start vertex column in the input edge table.	Yes	N/A
fromGroupCol	The start vertex group in the input edge table.	Yes	N/A
toVertexCol	The end vertex column in the input edge table.	Yes	N/A
toGroupCol	The end vertex group in the input edge table.	Yes	N/A
outputTableName	The name of the output table.	Yes	N/A
outputTablePartitions	The partitions in the output table.	No	N/A
lifecycle	The lifecycle of the output table.	No	N/A
workerNum	The number of workers.	No	N/A
workerMem	The memory size per worker, in MB.	No	4096
splitSize	The data split size, in MB.	No	64

Examples - Testing data

The testing data is the same as that in Label propagation clustering.

Output

The following figure shows the output.



val
0.4230769

4.1.5.2.10.7. Maximum connected subgraph

In 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.

PAI command

You can run the following Machine Learning Platform for AI (PAI) command to use this component:

```
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

The following table describes the parameters that are used in the PAI command.

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	N/A
inputEdgeTablePartitions	The partitions that are 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	N/A
toVertexCol	The end vertex column in the input edge table.	Yes	N/A
outputTableName	The name of the output table.	Yes	N/A
outputTablePartitions	The partitions in the output table.	No	N/A
lifecycle	The lifecycle of the output table.	No	N/A
workerNum	The number of workers.	No	N/A

Parameter	Description	Required	Default value
workerMem	The memory size per worker, in MB.	No	4096
splitSize	The data split size, in MB.	No	64

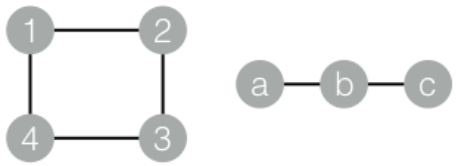
Examples - Testing data

Execute the following SQL statements to generate input 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 );
```

shows the graph structure.

Graph structure



Output

The following figure shows the output based on the testing data.

```
+-------+-------+
| node  | grp_id|
+-------+-------+
| 1     | 4     |
| 2     | 4     |
| 3     | 4     |
| 4     | 4     |
| a     | c     |
| b     | c     |
| c     | c     |
+-------+-------+
```

4.1.5.2.10.8. Vertex clustering coefficient

A vertex clustering coefficient is used to calculate the peripheral density of a vertex in Undirected Graph G. The density of a star network is 0, and that of a fully meshed network is 1.

Parameter settings

The core parameter of this component is maxEdgeCnt. If the node degree is larger than the value of this parameter, sampling is performed. This parameter is optional. Default value: 500.

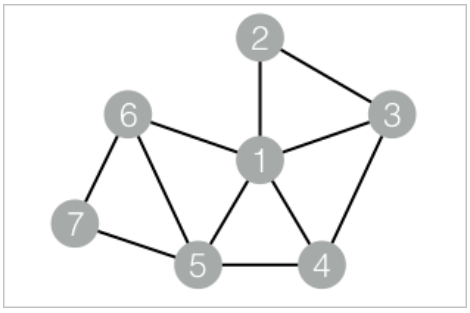
Examples - Testing data

Execute the following SQL statements to generate testing 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,   d
ensity 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

You can run the following Machine Learning Platform for AI (PAI) command to use the algorithm component:

```
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

The following table describes the parameters that are used in the PAI command.

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	N/A
inputEdgeTablePartitions	The partitions that are 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	N/A
toVertexCol	The end vertex column in the input edge table.	Yes	N/A
outputTableName	The name of the output table.	Yes	N/A
outputTablePartitions	The partitions in the output table.	No	N/A
lifecycle	The lifecycle of the output table.	No	N/A
maxEdgeCnt	If the node degree is larger than the value of this parameter, sampling is required.	No	500
workerNum	The number of workers.	No	N/A
workerMem	The memory size per worker, in MB.	No	4096

Parameter	Description	Required	Default value
splitSize	The data split size, in MB.	No	64

4.1.5.2.10.9. Edge clustering coefficient

An edge clustering coefficient is used to calculate the edge density in Undirected Graph G.

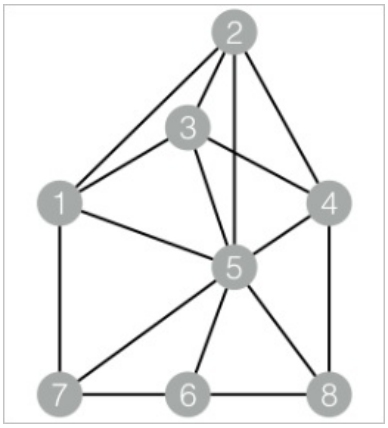
Examples - Testing data

Execute the following SQL statements to generate testing 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,
node2_edge_cnt bigint,   triangle_cnt bigint,   density double );
```

Graph structure shows the graph structure.

Graph structure



Output

The following figure shows the 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

You can run the following Machine Learning Platform for AI (PAI) command to use this component:

```
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

The following table describes the parameters that are used in the PAI command. Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	N/A
inputEdgeTablePartitions	The partitions that are 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	N/A

Parameter	Description	Required	Default value
toVertexCol	The end vertex column in the input edge table.	Yes	N/A
outputTableName	The name of the output table.	Yes	N/A
outputTablePartitions	The partitions in the output table.	No	N/A
lifecycle	The lifecycle of the output table.	No	N/A
workerNum	The number of workers.	No	N/A
workerMem	The memory size per worker, in MB.	No	4096
splitSize	The data split size, in MB.	No	64

4.1.5.2.10.10. Counting triangle

This component generates all triangles in Undirected Graph G.

Parameter settings

The core parameter of this component is maxEdgeCnt. If the node degree is larger than the value of the maxEdgeCnt parameter, sampling is performed. This parameter is optional. Default value: 500.

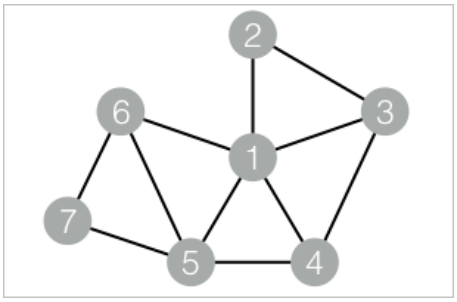
Examples - Testing data

Execute the following SQL statements to generate testing 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

You can run the following Machine Learning Platform for AI (PAI) command to use this component:

```
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

The following table describes the parameters that are used in the PAI command. Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	N/A
inputEdgeTablePartitions	The partitions that are 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	N/A
toVertexCol	The end vertex column in the input edge table.	Yes	N/A
outputTableName	The name of the output table.	Yes	N/A
outputTablePartitions	The partitions in the output table.	No	N/A
lifecycle	The lifecycle of the output table.	No	N/A
maxEdgeCnt	If the node degree is larger than the value of this parameter, sampling is required.	No	500
workerNum	The number of workers.	No	N/A
workerMem	The memory size per worker, in MB.	No	4096
splitSize	The data split size, in MB.	No	64

4.1.5.2.10.11. Tree depth

In a tree network, this component generates the depth of each node in a tree and the tree ID.

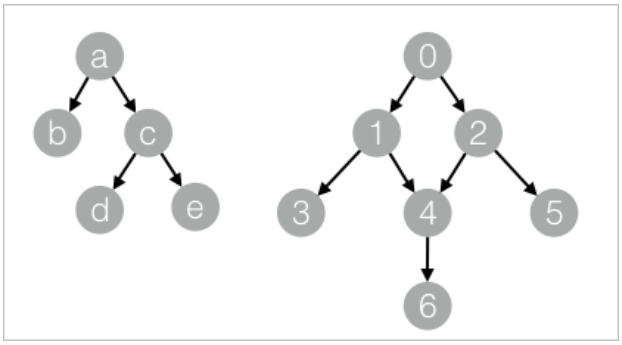
Examples - Testing data

Execute the following SQL statements to generate testing 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

You can run the following Machine Learning Platform for AI (PAI) command to use this component:

```
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

The following table describes the parameters that are used in the PAI command. Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	N/A
inputEdgeTablePartitions	The partitions that are 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	N/A
toVertexCol	The end vertex column in the input edge table.	Yes	N/A
outputTableName	The name of the output table.	Yes	N/A
outputTablePartitions	The partitions in the output table.	No	N/A
lifecycle	The lifecycle of the output table.	No	N/A
workerNum	The number of workers.	No	N/A
workerMem	The memory size per worker, in MB.	No	4096
splitSize	The data split size, in MB.	No	64

4.1.5.2.11. Tools

4.1.5.2.11.1. SQL Script

The SQL script component allows you to write SQL statements in the SQL script editor.

1. Go to a Machine Learning Studio project. For more information, see Data preparation.
2. In the left-side navigation pane, click **Components**.
3. Drag the **SQL Script** component in the **Tools** folder to the canvas.
4. After you connect the component that generates the input table and the **SQL Script** component, click the **SQL Script** component to open the **Parameters Setting** panel.

5. In the SQL Script section, enter the SQL script that is used to implement the required feature. Take note of the following items:
   - The component supports one to four inputs and one output.
   - You can write only one SQL statement.
   - Input tables are automatically mapped to tables t1 to t4. You can directly use ${t1}, ${t2}, ${t3}, and ${t4} without the need to specify the table names.
   - The sample SQL script calculates the number of rows in the input table.

4.1.5.2.12. Financials

4.1.5.2.12.1. Binning

The Binning component supports data binning in equal frequency or equal width.

PAI command

You can run the following PAI command to use the Binning component.

```
PAI -name binning
    -project algo_public
    -DinputTableName=input
    -DoutputTableName=output
```

Parameters

The following table describes the parameters in the PAI command.

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	N/A
selectedColNames	Optional. The columns that are selected from the input table for binning.	Column name	All columns except the label column (If no label column exists, all columns are selected.)
labelColumn	Required. The name of the label column.	Column name	Empty
validTableName	Optional. The name of the validation table. This parameter is required if the binningMethod parameter is set to auto.	Table name	Empty
validTablePartitions	Optional. The partitions that are selected from the validation table.	Partition name	All partitions
inputTablePartitions	Optional. The partitions that are selected from the input table.	Partition name	All partitions

Parameter	Description	Valid value	Default value
inputBinTableName	Optional. The input binning table.	Table name	Empty
selectedBinColNames	Optional. The columns that are selected from the input binning table.	Column name	Empty
positiveLabel	Optional. The label of the output positive samples.	N/A	1
nDivide	Optional. The number of bins.	Positive integer	10
colsNDivide	Optional. The numbers of bins for specific columns. Specify this parameter in the format of Column name 1:Number of bins,Column name 2:Number of bins. Example: col0:3,col2:5. If the columns that are specified for the colsNDivide parameter are not included in those specified for the selectedColNames parameter, the columns are also used in binning. For example, the selectedColNames parameter is set to col0,col1, the colsNDivide parameter is set to col0:3,col2:5, and the nDivide parameter is set to 10. In this case, binning is performed based on col0:3,col1:10,col2:5.	N/A	Empty
isLeftOpen	Optional. Specifies whether the interval is left-open and right-closed. A value of false indicates a left-closed, right-closed interval.	true and false	true
stringThreshold	Optional. The threshold for discrete values in the else bin.	N/A	Empty
colsStringThreshold	Optional. The threshold for specific columns. Specify this parameter in the same format as the colsNDivide parameter.	N/A	Empty

Parameter	Description	Valid value	Default value
binningMethod	Optional. The binning mode. A value of quantile indicates data binning in equal frequency. A value of bucket indicates data binning in equal width. A value of auto indicates the system automatically selects a binning mode.	quantile, bucket, and auto	quantile
lifecycle	Optional. The lifecycle of the output table.	Positive integer	Empty
coreNum	Optional. The number of cores.	Positive integer	Determined by the system
memSizePerCore	Optional. The memory size of each core.	Positive integer	Determined by the system

Constraints

If you want to specify constrains, the Binning component must be used with the Scorecard Training component. During scorecard training, the Binning component converts continuous features into multiple discrete dummy variables to achieve feature engineering. You can specify constraints for the weights of the dummy variables. The following information describes the constraints:

- Ascending order: Weights must be added to the dummy variables of a feature based on index values in ascending order. This indicates that a dummy variable with a large index value has a high weight.
- Descending order: Weights must be added to the dummy variables of a feature based on index values in descending order. This indicates that a dummy variable with a large index value has a low weight.
- Same weight: The weights of two dummy variables of a feature must be the same.
- Zero weight: The weight of a dummy variable must be 0.
- Specific weight: The weight of a dummy variable must be a specific floating-point value.
- WOE order: Weights must be added to the dummy variables of a feature based on the weight of evidence (WOE) values in ascending order. This indicates that a dummy variable with a large WOE value has a high weight.

4.1.5.2.12.2. Data Conversion Module

The Data Conversion Module component uses binning results to convert the input features that you provide. You can use this component to perform normalization, Weight of Evidence (WoE) conversion, and discretization.

Parameters

The following table describes the command parameters of the Data Conversion Module component.

Parameter	Description
inputFeatureTableName	Required. The name of the input feature table.
inputBinTableName	Required. The name of the binning result table.
inputFeatureTablePartitions	Optional. The partitions that are selected from the input feature table. By default, all partitions are selected.
outputTableName	Required. The name of the output table.

Parameter	Description
featureColNames	Optional. The feature columns that are selected from the input table. By default, all feature columns are selected.
metaColNames	The columns that do not need to be converted. These columns in the output are the same as those in the input. By default, the metaColNames parameter is left empty. You can specify the label and sample_id columns.
transformType	Optional. The type of data conversion. Valid values: normalize, dummy, and woe. A value of normalize indicates normalization. A value of dummy indicates discretization. A value of woe indicates WOE conversion. Default value: dummy.
itemDelimiter	Optional. The delimiter that is used to separate features. The default delimiter is the comma (,). This parameter is valid only if the transformType parameter is set to dummy.
kvDelimiter	Optional. The delimiter that is used to separate keys and values. The default delimiter is the comma (,). This parameter is valid only if the transformType parameter is set to dummy.
lifecycle	Optional. The lifecycle of the output table. By default, this parameter is left empty.
coreNum	Optional. The number of CPU cores that are required. By default, the system determines the value.
memSizePerCore	Optional. The memory size of each CPU core. By default, the system determines the value.

Example

PAI command

You can run the following PAI command to call the Data Conversion Module component.

```
PAI -name data_transform
-project algo_public
-DinputFeatureTableName=feature_table
-DinputBinTableName=bin_table
-DoutputTableName=output_table
-DmetaColNames=label
-DfeatureColNames=feaname1,feaname2
```

Normalization algorithm

To implement normalization, the Data Conversion Module component converts variable values into values between 0 and 1 based on input binning information, and imputes missing values with 0. The following algorithm is used:

```
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

The Data Conversion Module component generates regular output tables for normalization and WOE conversion.

During discretization in which data is converted into dummy variables, the component generates a table in the key-value format. Each variable in the table is in the `[${feaname}]\_bin\_${bin_id}` format. In this example, the variable is sns.

- If sns falls into the second bin, the generated variable is `[sns]_bin_2`.
- If sns does not have a value, it falls into the empty bin, and the generated variable is `[sns]_bin_null`.
- If sns has a value but does not fall into a defined bin, it falls into the else bin, and the generated variable is `[sns]_bin_else`.

4.1.5.2.12.3. Scorecard Training

The scorecard is a modeling tool that is commonly used in credit risk evaluation. Scorecard modeling performs discretization on original variables by using binning and uses linear models such as logistic regression and linear regression, to conduct model training. The scorecard supports feature selection and score conversion. In addition, it allows you to add constraints to variables during model training.

> **Note** If you use the scorecard without binning, scorecard training is equivalent to logistic 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 Training component supports two methods for feature engineering. Both methods use the Binning component for feature discretization. Then, one method performs one-hot encoding for each variable based on binning results to generate N dummy variables. N represents the number of bins. The other method performs weight of evidence (WOE) conversion to replace the original value of a variable with the WOE value of the bin into which the variable falls.

> **Note** When you convert original variables into dummy variables, you can specify constraints for these dummy variables.

Score transformation

In scenarios such as credit scoring, you must perform a linear transformation to convert the predicted sample odds into a score. The following formula is used for linear transformation:

```
log(odds) = \sum(wx) = a scaled_score + b
```

You can use the following parameters to specify the linear transformation relationship:

- scaledValue: a scaled score.
- odds: the odds of the scaled score.
- pdo: the points at which the odds are doubled.

For example, the value of scaledValue is 800, that of odds is 50, and that of pdo is 25. In this case, the following two dots are determined for a line:

```
log(40) = a * 800 + b
log(80) = a * 825 + b
```

Calculate the values of a and b. Then, perform a linear transformation to obtain the scores of a and b.

The scaling information is specified in the JSON format by using the -Dscale parameter.

```
null
```

If you specify a value for the -Dscale parameter, you must also specify a value for each of the scaledValue, odds, and pdo parameters.

Constraint addition during training

During scorecard training, you can add constraints to variables. For example, you can set the score of a specific bin to a fixed value, specify the scores of two bins to a specific proportion, or limit the scores between bins, such as sorting bin scores by WOE value. The implementation of constraints depends on the underlying optimization algorithm that contains constraints. You can specify the constraints in the Binning component in the PAI console. After the constraints are specified, the Binning component generates JSON-formatted constraints and automatically transfers them to its connected training component.

- <: The weights of variables must be sorted in ascending order.
- >: The weights of variables must be sorted in descending order.
- =: The weight of a specific variable must be a fixed value.
- %: The weights of two variables must meet a proportional relationship.
- UP: the upper limit for the weights of variables.
- LO: the lower limit for the weights of variables.

Each JSON-formatted constraint is stored in a table as a JSON string. The table contains only one row and one column. The following code shows a sample JSON string:

```
{
    "name": "feature0",
    "<": [
        [0,1,2,3]
    ],
    ">": [
        [4,5,6]
    ],
    "=": [
        "3:0","4:0.25"
    ],
    "%": [
        ["6:1.0","7:1.0"]
    ]
}
```

Built-in constraints

Each original variable has a built-in constraint. For each variable, the average score of a population must be 0. Due to the constraint, the value of scaled_weight in the intercept options of the scorecard model equals the average score of the population in terms of all variables.

Optimization algorithms

On the Parameters Setting tab, select Advanced Options. Then, you can configure the optimization algorithm that is used during scorecard training. The system supports the following optimization algorithms:

- L-BFGS
- Newton's method
- Barrier method
- SQP

The L-BFGS algorithm is a first-order optimization algorithm that is used to process large amounts of feature data. The Newton's method algorithm is a classic second-order optimization algorithm. It is fast in convergence and accurate. However, the Newton's method algorithm is not suitable for processing large amounts of feature data because it needs to calculate a second-order Hessian matrix. Both algorithms do not contain constraints. If you select one of the two algorithms, the system automatically ignores the constraints that you specify.

If you require constraints in training, you can select the barrier method and SQP algorithms. Both algorithms are second-order optimization algorithms. If the algorithm does not contain constraints, it is completely equivalent to the Newton's method algorithm. The barrier method algorithm provides almost the same computing performance and accuracy as the SQP algorithm. In most cases, we recommend that you select SQP. If you are not familiar with optimization algorithms, we recommend that you set the Optimization Algorithm parameter to Auto-selected by default. In this case, the system selects the most appropriate algorithm based on the data amount and constraints.

Feature selection

The Scorecard Training component supports stepwise feature selection. Stepwise feature selection is a fusion of forward and backward selection. Each time the system performs a forward selection to select a new variable and adds it to the model, the system also performs a backward selection. The backward selection is used to remove the variables whose significance does not meet the requirements. Stepwise feature selection supports various functions and feature transformation methods. Therefore, stepwise feature selection also supports multiple selection standards. The following standards are supported:

- Marginal contribution: This standard can be applied to all functions and feature engineering methods.
- Score test: This standard supports only WOE conversion and logistic regression without feature engineering.
- F test: This standard supports only WOE conversion and linear regression without feature engineering.

Marginal contribution

This standard needs two models to be trained: Model A and Model B. Model A does not contain Variable X, and Model B contains Variable X in addition to all the variables of Model A. The difference between the functions of the two models in final convergence is the marginal contribution of Variable X to all the other variables in Model B. In scenarios where variables are converted into dummy variables, the marginal contribution of Variable X is the difference between the functions of all dummy variables in Model A and the functions of all dummy variables in Model B. Therefore, marginal contribution is supported by all feature engineering methods.

Marginal contribution is flexible and is not limited to a specific type of model. Only variables that contribute to functions are passed to the model. Marginal contribution has disadvantages when compared with statistical significance. Typically, 0.05 is used as the threshold for statistical significance. Marginal contribution does not provide a recommended threshold for beginners. We recommend that you set the threshold to 10E-5.

Score test

This standard is applicable only to feature selection with logistic regression. During a forward selection, a model that has only intercept options is trained first. In each subsequent iteration, the score chi-squares of the variables that are not passed to the model are measured. The variable with the largest score chi-square is passed to the model. In addition, the p-value of the variable with the largest score chi-square is calculated based on chi-square distribution. If the p-value of the variable is greater than the given SLENTRY value, the variable is not passed to the model, and feature selection is terminated.

After the forward selection is complete, a backward selection is performed for the variable that is passed to the model. The Wald chi-square of the variable and the related p-value are calculated. If the p-value is greater than the given SLSTAY value, the variable is removed from the model. Then, the system starts a new iteration.

F test

This standard is applicable to feature selection with linear regression. During a forward selection, a variable that has only intercept options is trained first. In each subsequent iteration, the F-values of the variables that are not passed to the model are calculated. F-value calculation is similar to marginal contribution calculation. Two models must be trained to calculate the F-value of a variable. The F-value follows F distribution. The related p-value can be calculated based on the probability density function of 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 terminated.

During the backward selection, the F-value is used to calculate the significance of a variable in a way similar to a score test.

Forced selection of the variables that you want to pass to a model

Before a feature selection is performed, you can specify the variables that you want to forcibly pass to the model. These variables are directly passed to the model regardless of their significance. No forward or backward selection is performed for the specified variables.

You can specify the number of iterations and significance thresholds by using the -Dselected parameter. Specify this parameter in the JSON format. Example:

```
{"max_step":2, "slentry": 0.0001, "slstay": 0.0001}
```

If the -Dselected parameter is left empty or the max_step parameter is set to 0, no feature selection is performed.

Model report

The Scorecard Training component generates data to a model report. The model report contains basic model evaluation statistics, such as binning information of variables, binning constraints, WOE values, and marginal contribution information. The following table describes the columns in a model report that is generated in the PAI console.

Column name	Data type	Description
feaname	string	The name of the feature.
binid	bigint	The ID of the bin.
bin	string	The description of the bin, which indicates the interval of the bin.
constraint	string	The constraints that are added to the bin during 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.
scaled_weight	double	The score that is linearly transformed from the weight of a binning variable in scorecard training.
woe	double	A statistical metric. It indicates the WOE value of a bin in the training set.

Column name	Data type	Description
contribution	double	A statistical metric. It indicates the marginal contribution value of a bin in the training set.
total	bigint	A statistical metric. It indicates the total number of samples in a bin in the training set.
positive	bigint	A statistical metric. It indicates the number of positive samples in a bin in the training set.
negative	bigint	A statistical metric. It indicates the number of negative samples in a bin in the training set.
percentage_pos	double	A statistical metric. It indicates the proportion of positive samples in a bin to total positive samples in the training set.
percentage_neg	double	A statistical metric. It indicates the proportion of negative samples in a bin to total negative samples in the training set.
test_woe	double	A statistical metric. It indicates the WOE value of a bin in the testing set.
test_contribution	double	A statistical metric. It indicates the marginal contribution value of a bin in the testing set.
test_total	bigint	A statistical metric. It indicates the total number of samples in a bin in the testing set.
test_positive	bigint	A statistical metric. It indicates the number of positive samples in a bin in the testing set.
test_negative	bigint	A statistical metric. It indicates the number of negative samples in a bin in the testing set.
test_percentage_pos	double	A statistical metric. It indicates the proportion of positive samples in a bin to total positive samples in the testing set.
test_percentage_neg	double	A statistical metric. It indicates the proportion of negative samples in a bin to total negative samples in the testing set.

The following table describes the algorithm parameters.

Algorithm parameters

Parameter	Description	Valid value	Default value
inputTableName	Required. The name of the input feature table.	N/A	N/A
inputTablePartitions	Optional. The partitions selected from the input table.	N/A	All partitions
inputBinTableName	Optional. The name of the binning result table. If you specify a value for this parameter, the system automatically performs discretization for features based on the binning rules in the binning result table.	N/A	N/A
featureColNames	Optional. The feature columns that are selected from the input feature table.	N/A	All columns except the label column
labelColName	Required. The label column.	N/A	N/A
outputTableName	Required. The output model table.	N/A	N/A
inputConstraintTableName	Optional. The name of the table that stores constraints. The constraints are a JSON string that is stored in a cell of the table.	N/A	N/A
optimization	Optional. The optimization algorithm.	lbfgs, newton, barrier_method, sqp, and auto. Only sqp and barrier_method support constraints. If you set the optimization parameter to auto, the system automatically selects an appropriate optimization algorithm based on user data and related parameter settings. If you are not familiar with optimization algorithms, we recommend that you set the optimization parameter to auto.	auto
loss	Optional. The loss type.	logistic_regression and least_square	logistic_regression
iterations	Optional. The maximum number of iterations of optimizations.	N/A	100

Parameter	Description	Valid value	Default value
l1Weight	Optional. The parameter weight of L1 regularization. Only the L-BFGS algorithm supports this parameter.	N/A	0
l2Weight	Optional. The parameter weight of L2 regularization.	N/A	0
m	Optional. The historical step size for optimization that is performed by using the L-BFGS algorithm. Only the L-BFGS algorithm supports this parameter.	N/A	10
scale	Optional. The weight scaling information of the scorecard.	N/A	Empty
selected	Optional. Specifies whether to enable feature selection during scorecard training.	N/A	Empty
convergenceTolerance	Optional. The convergence tolerance.	N/A	1e-6
positiveLabel	Optional. The label of the positive samples.	N/A	"1"
lifecycle	Optional. The lifecycle of the output table.	N/A	N/A
coreNum	Optional. The number of cores.	N/A	Determined by the system
memSizePerCore	Optional. The memory size of each core.	N/A	Determined by the system

4.1.5.2.12.4. Scorecard Prediction

The Scorecard Prediction component uses the model that is generated by a training component to predict scores. The following training components are supported: the Scorecard Training, Logistic Regression for Binary Classification, and Linear Regression components.

Input parameters

The Scorecard Prediction component has the following parameters:

- **Feature Columns**: the feature columns to be used for prediction. By default, all columns are selected.
- **Reserved Columns**: the columns that are appended to the prediction result table without processing, such as the ID and label columns.
- **Output Variable Score**: specifies whether to generate a score for each feature variable. The total predicted score is the score of intercept options plus the score of each variable.

Sample score table

The churn column is a reserved column. The data in this column does not affect the prediction results.

Column name	Data type	Description
prediction_score	double	The column that contains predicted scores. In a linear model, the feature values and model weight values are multiplied and summed up to obtain the predicted scores. In a scorecard model, if score transformation is performed, the transformed scores are generated in this column.
prediction_prob	double	The column that contains the probability values of positive samples in binary classification. The probability values are transformed from the original scores (without score transformation) by using the sigmoid function.
prediction_detail	string	The column that contains the probability values of positive and negative samples. The probability values are described in JSON strings. The value 0 represents negative and the value 1 represents positive. Example: { “0” :0.1813110520,” 1” :0.8186889480}.

PAI command

You can run the following PAI command to use the component:

```
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

The following table describes the parameters in the PAI command.

Parameter	Description	Default value
inputFeatureTableName	Required. The name of the input feature table.	N/A
inputFeatureTablePartitions	The partitions that are selected from the input feature table.	All partitions
inputModelTableName	Required. The name of the input model table.	N/A

Parameter	Description	Default value
featureColNames	Optional. The feature columns that are selected from the input feature table.	All columns
metaColNames	The columns that do not need to be converted. These columns in the output are the same as those in the input.	By default, this parameter is left empty. You can specify the label and sample_id columns.
outputFeatureScore	Optional. Specifies whether to generate the scores of variables in the prediction results. Valid values: true and false.	false
outputTableName	Required. The name of the output table.	N/A
lifecycle	Optional. The lifecycle of the output table.	Empty
coreNum	Optional. The number of cores.	Determined by the system
memSizePerCore	Optional. The memory size per core.	Determined by the system

4.1.5.2.12.5. Population Stability Index

Population stability index (PSI) is an important metric to identify a shift in two samples of a population.

For example, PSI can measure the stability of samples over the past two months. A PSI value less than 0.1 indicates insignificant changes. A PSI value between 0.1 and 0.25 indicates changes that are relatively significant. A PSI value greater than 0.25 indicates sharp changes and requires special attention.

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 column chart.

The preceding 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. Therefore, the population stability cannot be automatically monitored. To resolve this issue, you can use the Binning and Population Stability Index components.

PAI command

You can run the following PAI command to use the component:

```
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

The following table describes the parameters in the PAI command.

Parameter	Description	Default value
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
inputBaseTablePartitions	Optional. The partitions that are selected from the base table.	All partitions
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
inputTestTablePartitions	Optional. The partitions that are selected from the test table.	All partitions
inputBinTableName	Required. The name of the binning result table.	N/A
featureColNames	Optional. The feature columns that are required for PSI value calculation.	All feature columns
outputTableName	Required. The name of the output table.	N/A
lifecycle	Optional. The lifecycle of the output table.	Empty
coreNum	Optional. The number of cores.	Determined by the system
memSizePerCore	Optional. The memory size of each core.	Determined by the system

4.1.5.2.13. Video intelligence platform (must be separately activated)

Note Video intelligence platforms are applicable only to Intel servers, but not to Hygon servers.

4.1.5.2.13.1. Video preprocessing

Convert datasets to TFRecord files

EasyVision of Machine Learning Platform for AI (PAI) allows you to convert datasets that are labeled by the data labeling module to TFRecord files. TFRecord files can be used to train models. You can use the TFRecord conversion component of EasyVision to convert labeled datasets that are stored in Object Storage Service (OSS) to TFRecord files and store the TFRecord files in OSS. If the datasets are labeled by using a method other than the data labeling module, you must first convert the labeled datasets to a format that is required by PAI.

PAI command

Convert a labeled dataset for classification

```
pai -name easy_vision
      -Dbuckets='oss://pai-vision-data-hz/?role_arn=acs:ram::xxx:role/pai-vision-hz&host=oss-cn-zhangjiakou.aliyuncs.com'
      -Dcmd  convert
      -Dlabel_file 'oss://pai-vision-data-hz/data/pai_test/watchdog.csv'
      -Dconvert_param_config '
        --class_list_file oss://pai-vision-data-hz/data/pai_test/watchdog_class_list.txt
        --max_image_size 600
        --write_parallel_num 8
        --num_samples_per_tfrecord 128
        --test_ratio 0.1
        --model_type CLASSIFICATION
      '
      -Doutput_tfrecord 'oss://pai-vision-data-hz/test/convert/watchdog_tfrecord/watchdog'
      -Dcluster='{
              \"worker\" : {
                \"count\" : 3,
                \"cpu\" : 800
              }
            }'
```

Convert a labeled dataset for text detection or recognition

```
pai -name easy_vision
      -Dbuckets='oss://pai-vision-data-hz/?role_arn=acs:ram::xxx:role/pai-vision-hz&host=oss-cn-zhangjiakou.aliyuncs.com'
      -Dcmd  convert
      -Dlabel_file 'oss://pai-vision-data-hz/data/recipt_text/qince_data/qince_receipt_test.csv'
      -Dconvert_param_config '
        --model_type TEXT_END2END
        --default_class text
        --max_image_size 2000
        --char_replace_map_path oss://pai-vision-data-hz/data/recipt_text/qince_data/char_replace_map.csv
        --default_char_dict_path oss://pai-vision-data-hz/data/recipt_text/qince_data/char_dict
        --test_ratio 0.1
        --write_parallel_num 8
        --num_samples_per_tfrecord 64
      '
      -Doutput_tfrecord 'oss://pai-vision-data-hz/test/convert/recipt_text_end2end/'
```

Parameters

Parameter	Required	Description	Value type or sample value	Default value
cmd	Yes	Set the value to convert.	String	convert

Parameter	Required	Description	Value type or sample value	Default value
buckets	No	The OSS bucket in which the input data is stored. You can specify multiple OSS buckets by separating them with commas (,). Each bucket name must end with a forward slash (/).	`"oss://bucket_name/?role_arn=xxx&host=yyy"` `"oss://bucket_1/?role_arn=xxx&host=yyy,oss://bucket_2/"`	""
label_file	Yes	The OSS path of the labeled CSV file.	`oss://your_bucket/xxx.csv`	Required
convert_param_config	No	The information about the conversion task. For more information, see the following table. You can also replace the convert_param_config parameter with the convert_config parameter.	--parama valuea --paramb valueb	""
convert_config	No	The OSS path of the conversion task profile.	String	""
output_tfrecord	No	The OSS path of the TFRecord file.	`oss://your_dir/prefix`	""
cluster	No	The information about the workers that are used to perform conversion in a distributed manner.	JSON string	`"{\"worker\":{\"count\":3,\"cpu\": 800,\"gpu\":0,\"memory\":20000}}"`

convert_param_config

Parameter	Required	Description	Value type or sample value	Default value

Parameter	Required	Description	Value type or sample value	Default value
model_type	Yes	The type of model to which the converted data applies. Valid values: • CLASSIFICATION: single-label or multi-label image classification • DETECTION: object detection • POLYGON_SEGMENTATION: semantic segmentation based on polygon labeling • SEGMENTATION: semantic segmentation • INSTANCE_SEGMENTATION: instance segmentation • TEXT_END2END: end-to-end optical character recognition (OCR) • TEXT_RECOGNITION: single-line text recognition • TEXT_DETECTION: text detection • VIDEO_CLASSIFICATION: video classification • SELF_DEFINED: custom conversion If the value of the model_type parameter is set to TEXT_END2END or TEXT_RECOGNITION, the char_replace_map_path and default_char_dict_path parameters take effect. If the value of the model_type parameter is set to VIDEO_CLASSIFICATION, the decode_type, sample_fps, reshape_size, decode_batch_size, and decode_keep_size parameters take effect.	String	N/A

Parameter	Required	Description	Value type or sample value	Default value
class_list_file	No	The OSS path of the category file. The file contains a list of category names. The category name in each line may be presented in one of the following formats: • Category name • Category name:Name of the mapping category If the default_class parameter is set, you must set the class_list_file or class_list parameter.	oss://path/to/your/classlit	""
class_list	No	The list of category names. Separate category names with commas (,). If the default_class parameter is set, you must set the class_list_file or class_list parameter.	String list	""
test_ratio	No	The ratio that is used to divide the set of test data into different subsets. Valid values: 0 to 1. If the value is set to 0, the total set of test data is used for training. If the value is set to 0.1, 10% of the test data is used for verification.	0.1	0.1
max_image_size	No	The maximum pixel value for the longer side of the images. If you have set this parameter and the size of an image exceeds the upper limit, the image is resized and saved to the TFRecord file. This reduces storage space and accelerates data reading.	INT	N/A
max_test_image_size	No	The maximum pixel value for the longer side of the image that is used for testing. The value of this parameter is the same as that of the max_image_size parameter. This parameter is used to configure test data.	INT	${max_image_size}

Parameter	Required	Description	Value type or sample value	Default value
default_class	No	The name of the default category. Category names that are not included in the class_list parameter will be specified as the default category.	STRING	""
error_class	No	The name of the invalid category. Objects or bounding boxes that belong to this category are not used for training.	STRING	""
ignore_class	No	The name of the category that is used only to detect models. Bounding boxes that belong to this category are not used for training.	STRING	""
converter_class	No	The name of the conversion class. Valid values: • PaiConverter: converts the datasets to the PAI format • QinceConverter: converts the datasets to the traditional labeling format • A custom class name	STRING	PaiConverter
seperator	No	The separator that is used in the split() method to break the label file into substrings.	STRING	""
image_format	No	The encoding format of the images in the TFRecord file.	STRING	jpg
read_parallel_num	No	The number of concurrent reads.	INT	10
write_parallel_num	No	The number of concurrent writes to the TFRecord file.	INT	1
num_samples_per_tfrecord	No	The number of images that are saved in each TFRecord file.	INT	256
user_defined_converter_path	No	The HTTP or OSS path of the user-defined converter code. Example: http://path/to/your/converter.py.	STRING	""
user_defined_generator_path	No	The HTTP or OSS path of the user-defined generator code. Example: http://path/to/your/generator.py.	STRING	""

Parameter	Required	Description	Value type or sample value	Default value
generator_class	No	The name of the user-defined generator class.	STRING	""
char_replace_map_path	No	The OSS path of the CSV file for character map replacement. The CSV file contains two columns named original and replaced. The strings in the original column are replaced by the strings in the replaced column.	STRING	""
default_char_dict_path	No	The OSS path of the file that is used to map characters to IDs. In the file, each character occupies a line. The ID of a character equals the line number minus 1.	STRING	""
decode_type	No	The method that is used to decode the videos. Valid values: • 1: intra only • 2: keyframe only • 3: without bidir • 4: decode all	INT	4
sample_fps	No	The number of frames that are extracted for sampling per second.	FLOAT	5
reshape_size	No	The size of the output frames, in pixels.	INT	224
decode_batch_size	No	The number of frames that are decoded at a time.	INT	0
task_id	No	The ID of the PAI labeling job, in the format of label-xxxx. If the manifest file contains a labeling job, this parameter is not required.	STRING	""

Custom conversion

PAI command

```
pai -name easy_vision
      -Dbuckets='oss://pai-vision-data-hz/?role_arn=acs:ram::xxx:role/pai-vision-hz&host=oss-cn-zhangjiakou.aliyuncs.com'
      -Dcmd  convert
      -Dlabel_file 'oss://pai-vision-data-hz/data/image_mat/qince/imagemat.txt'
      -Dconvert_param_config '
        --model_type SELF_DEFINED
        --converter_class ImageMatConverter
        --user_defined_converter_path  oss://pai-vision-data-hz/samples/code/imagemat_converter.py
        --user_defined_generator_path oss://pai-vision-data-hz/samples/code/custom_generator_example.py
        --generator_class CustomGenerator
        --max_image_size 1200
        --write_parallel_num 2
        --num_samples_per_tfrecord 6
        --test_ratio 0.0
```

To customize dataset conversion to TFRecord files, you must implement the following classes:

- A row class that inherits the DataRow class. The row class is used to parse data in each line of the label file and obtain image URLs and label information.
- A generator class that inherits the TFRecordGenerator class. The generator class is used to convert the data that is parsed by the row class to the tf.train.Example protocol buffers and specify the data format in TFRecord.
- A converter class that inherits the MultiProcConvert class. The converter class is used to load label files and configure train-test split policies to complete the conversion.

Example on custom_converter

```
from __future__ import unicode_literals
from tensorflow.python.platform import gfile
import tensorflow as tf
import numpy as np
import cv2
import json
import os
from easy_vision.python.protos.data_config_pb2 import DataConfig
from easy_vision.python.dataset_tools.tf_record_converter import MultiProcConverter
from easy_vision.python.dataset_tools.tf_record_converter import DataRow
from easy_vision.python.dataset_tools.converter_register import register_converter
import logging
import sys
try:
  reload(sys)
  sys.setdefaultencoding('utf8')
except:
  pass
class ImageMatRow(DataRow):
  '''
  parse an image_path mask_path line to get image url and anwser
  The following methods of DataRow are reimplemented:
     get_mask
  '''
  def __init__(self, row):
    '''
    Args:
      row: generate by csv.reader, array of columns in csv file
    '''
    super(ImageMatRow, self).__init__(row)
```

```
    super(ImageMatRow, self).__init__(row)
  def get_img_url(self):
    ''' get img url from raw_data
    Return:
      url, string, path of the image
    '''
    return self._row[0]
  def get_anws(self):
    ''' get anws string from anws field
    Return:
      anws, json object, depends on task
    '''
    return self._row[1]
  def get_mask_url(self):
    ''' must be implemented for segmentation task
    Return:
      string, mask url
    '''
    return self._row[1]
@register_converter()
class ImageMatConverter(MultiProcConverter):
  ''' implement methods for converting a text line file into tfrecord
      required by image segmentation models.
  '''
  def __init__(self, data_config):
    '''
    Args:
      data_config: DataConfig instance
    '''
    super(ImageMatConverter, self).__init__(data_config)
  def _check_lines(self, file_path):
    ''' only check the first 1k lines, to save time.
    Args:
      file_path: string, input file path to be checked
    Raises:
      AssertionError, if separator['\t'] is not appear in one of the text lines
    '''
    line_num = 1000
    with gfile.GFile(file_path, mode='r') as fin:
      for line_str in fin:
        line_str = line_str.strip()
        assert '\t' in line_str, 'separator[\t] is not in line: %s' % line_str
  def split_train_test(self, input_path, output_dir, test_ratio=0.1):
    data_arr = []
    with gfile.GFile(input_path, mode='r')  as fin:
      for line in fin:
        data_arr.append(line)
      np.random.shuffle(data_arr)
    train_num = int(len(data_arr) * (1-test_ratio))
    # save train input
    _, input_file = os.path.split(input_path)
    input_name, _ = os.path.splitext(input_file)
    train_input_path = output_dir.rstrip('/') + '/' + input_name + '_train.txt'
    with gfile.GFile(train_input_path, mode='w') as train_out:
      for line_str in data_arr[:train_num]:
        train_out.write(line_str)
    # save test input
    test_input_path = output_dir.rstrip('/') + '/' + input_name + '_test.txt'
    with gfile.GFile(test_input_path, mode='w') as test_out:
      for line_str in data_arr[train_num:]:
```

```
      for line_str in data_arr[train_num:]:
        test_out.write(line_str)
    return train_input_path, test_input_path
  def _load_input(self, input_path):
    ''' load input from easy_vision.python.input_path(a text format file)
        each line is image_path + '\t' + label_path
        label_path store an image with alpha channel,
        the alpha channel store the class of the image
    Args:
      input_path: string, input file path
    '''
    with gfile.GFile(input_path, mode='r') as f:
      for line in f:
        line = line.strip()
        line_tok = [ x for x in line.split('\t') if x != '']
        yield ImageMatRow(line_tok)
  def _save_for_fix(self, file_name):
    # currently no errors are checked and saved
    pass
```

Example on custom_generator

```
from __future__ import print_function
from __future__ import absolute_import
from __future__ import division
from __future__ import unicode_literals
import cv2
import tensorflow as tf
import numpy as np
import logging
import os
from easy_vision.python.utils import dataset_util
from easy_vision.python.dataset_tools import dataset_info as dataset_info_lib
from easy_vision.python.dataset_tools.tf_record_generator import TFRecordGenerator
class CustomGenerator(TFRecordGenerator):
  def _to_tf_example(self, img, row, url):
    mask_url = row.get_mask_url()
    mask_data = row.get_url_data(mask_url)
    if mask_data is None:
      logging.error('prefetch mask %s failed' % mask_url)
      return []
    mask = cv2.imdecode(np.fromstring(mask_data, dtype=np.uint8), cv2.IMREAD_UNCHANGED)
    if mask is None:
      logging.error('decode mask %s failed' % mask_url)
      return []
    mask = np.array(mask > 0, dtype=np.uint8)
    height, width = img.shape[:2]
    succeed, encoded = cv2.imencode('.' + self.image_format, img)
    succeed, mask_data = cv2.imencode('.png', mask)
    _, img_name = os.path.split(url)
    example_info = dataset_info_lib.DatasetInfo()
    example_info.add_single(
        {
            'image/height': height,
            'image/width': width,
        }
    )
    example = tf.train.Example(
        features=tf.train.Features(
            feature={
                'image/encoded': dataset_util.bytes_feature(encoded.tobytes()),
                'image/filename':
                  dataset_util.bytes_feature(img_name),
                'image/dataset_name':
                  dataset_util.bytes_feature(self._dataset_name.encode('utf-8')),
                'image/height': dataset_util.int64_feature(height),
                'image/width': dataset_util.int64_feature(width),
                'image/source_id':
                  dataset_util.bytes_feature(url),
                'image/format':
                  dataset_util.bytes_feature(self.image_format),
                'image/segmentation/class/encoded': (
                    dataset_util.bytes_feature(mask_data.tobytes())),
                'image/segmentation/class/format': dataset_util.bytes_feature('png')
            }))
    return [(example, example_info)]
```

General video preprocessing

This topic provides an example on how to run a Machine Learning Platform for AI (PAI) command to train a model to preprocess videos. This topic also describes the command parameters.

You can run the `ev_video_process` command to train a model to preprocess videos. When you run this command, you need only to set the common parameters without the need to know the rules or logic of the EasyVision profile.

PAI command

```
pai -name ev_video_process
             -Dinput_oss_file='oss://pai-vision-data-hz/data/pai_test/video_decode_test.list'
             -Doutput_oss_file='oss://pai-vision-data-hz/test/tmp/ev_video_decode_result.txt'
             -Doutput_dir='oss://pai-vision-data-hz/data/ucf101_train/frames/'
             -Ddecode_type=4
             -Dsample_fps=5
             -Dreshape_size=224
             -Ddecode_batch_size=10
             -Ddecode_keep_size=0
             -Dqueue_size=256
             -Dnum_worker=4
             -DcpuRequired=400
             -DgpuRequired=100
             -Dbuckets='YOUR_OSS_BUCKET_CONFIG'
```

Parameters

Parameter	Required	Description	Value type or sample value	Default value
buckets	Yes	The endpoint of the Object Storage Service (OSS) bucket in which your model is stored. This parameter is required when you use your own model to perform prediction.	`"oss://bucket_name/?role_arn=xxx&host=yyy"`	N/A
input_oss_file	Yes	The path of the input OSS file. Each line of the OSS object includes a video path.	`oss://your_bucket/filelist.txt`	N/A
output_oss_file	Yes	The path of the output OSS file in which the preprocessing result is stored. The system may generate multiple result files and merge these result files into an output OSS file. The result files are prefixed with the name of the output OSS file. The number of result files is the same as the number of workers, which is specified by the num_worker parameter.	`oss://your_bucket/result.txt`	N/A
output_dir	Yes	The directory that is used to store the frames extracted from the videos.	`oss://your_bucket/your_dir`	N/A

Parameter	Required	Description	Value type or sample value	Default value
decode_type	No	The method that is used to decode the videos. Valid values: • 1: intra only • 2: keyframe only • 3: without bidir • 4: decode all	INT	4
sample_fps	No	The number of frames that are extracted for sampling per second.	FLOAT	5
reshape_size	No	The size of the output frames, in pixels. If you set this parameter to -1, the frames are not resized.	INT	224
decode_batch_size	No	The number of frames that are decoded at a time.	INT	10
decode_keep_size	No	The number of overlapped frames in different batches.	INT	0
queue_size	No	The length of the preprocessing queue.	INT	256
thread_num	No	The number of threads on each worker.	INT	4
num_worker	No	The number of prediction workers.	INT	2
cpuRequired	No	The number of CPUs for a worker.	A value of 100 indicates one CPU.	800
gpuRequired	No	The number of GPUs for a worker.	A value of 100 indicates one GPU.	100

4.1.5.2.13.2. Offline training

Train a model for semantic image segmentation

This topic provides an example on how to run a Machine Learning Platform for AI (PAI) command to train a model for semantic image segmentation in EasyVision. This topic also describes the command parameters.

A semantic segmentation model based on DeepLabv3 is implemented. For more information, see Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation.

PAI command

The following code provides an example on the training of semantic image segmentation with a single GPU:

```
pai -name easy_vision
           -Dbuckets='oss://pai-vision-data-hz/?role_arn=xxx&host=oss-cn-zhangjiakou.aliyuncs.com'
           -DgpuRequired=100
           -Dcmd train
           -Dparam_config '
             --model_type DeeplabV3
             --backbone  resnet_v1_50
             --backbone_feature_stride 16
             --bn_trainable true
             --num_classes 21
             --num_epochs 1
             --model_dir oss://pai-vision-data-hz/test/deeplab_stage1
             --train_data oss://pai-vision-data-hz/data/test/pascal_voc_seg_aug/voc_ev_train.tfrecor
d
             --test_data oss://pai-vision-data-hz/data/test/pascal_voc_seg_aug/voc_ev_train.tfrecord
             --num_test_example 2
             --train_batch_size 6
             --test_batch_size 1
             --image_crop_size 513
             --lr_type polynomial_decay
             --initial_learning_rate 0.007
             --power 0.9
           '
```

Parameters

Parameter	Required	Description	Value type or sample value	Default value
buckets	Yes	The endpoint of the Object Storage Service (OSS) bucket.	oss://pai-vision-data-hz/?role_arn=acs:ram::1217060697xxxx:role/pai-vision-hz&host=cn-hangzhou.oss.aliyun-inc.com	N/A
cluster	No	The configuration of parameters that are used for distributed training.	JSON string	""
gpuRequired	No	Specifies whether to use GPUs. By default, each worker uses one GPU. If you set this parameter to 200, each worker uses two GPUs.	100	100
cmd	Yes	The type of the EasyVision task. To train a model, you must set this parameter to train.	train	N/A
param_config	Yes	The configuration of parameters that are used for model training. The format of the param_config parameter is the same as that of the ArgumentParser() object in Python. For more information, see param_config.	STRING	N/A

param_config

The param_config parameter contains the parameters that are used for model training. The format of the param_config parameter is the same as that of the ArgumentParser() object in Python. The following code provides an example of the param_config parameter:

```
-Dparam_config = '
--backbone resnet_v1_50
--num_classes 200
--model_dir oss://your/bucket/exp_dir
'
```

Note Do not enclose a parameter value of the STRING type in double quotation marks (") or single quotation marks (').

The following table describes all parameters that are contained in the param_config parameter.

Parameter	Required	Description	Value type or sample value	Default value
model_type	Yes	The type of the model to train. Set this parameter to DeeplabV3.	STRING	N/A
backbone	Yes	The name of the backbone network that is used by the model. Valid values: • resnet_v1_50 • resnet_v1_101 • resnet_v1a_18 • resnet_v1a_34 • resnet_v1d_50 • resnet_v1d_101 • xception_41 • xception_65 • xception_71	STRING	N/A
weight_decay	No	The value of L2 regularization.	FLOAT	1e-4
num_classes	Yes	The number of categories, including background categories.	21	N/A
backbone_feature_stride	No	The feature downsampling step size of the backbone network.	INT	16
bn_trainable	No	Specify whether the batch normalization (BN) layer is trainable. If the value of the train_batch_size parameter is greater than 8, set the bn_trainable parameter to true.	BOOL	true
image_crop_size	No	The size of the image after cropping, in pixels.	INT	513

Parameter	Required	Description	Value type or sample value	Default value
optimizer	No	The type of the optimizer. Valid values: • momentum: stochastic gradient descent (SGD) with momentum • adam	STRING	momentum
lr_type	No	The policy that is used to adjust the learning rate. Valid values: • exponential_decay: The learning rate is subject to exponential decay. • polynomial_decay: The learning rate is subject to polynomial decay. If you set the lr_type parameter to polynomial_decay, the num_steps parameter is automatically set to the total number of training iterations. The value of the end_learning_rate parameter is automatically set to one thousandth of the value of the initial_learning_rate parameter. • manual_step: The learning rate of each epoch is manually adjusted. If you set the lr_type parameter to manual_step, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rate. You must also set the learning_rates parameter to specify the learning rate as needed. • cosine_decay: The learning rate of each epoch is adjusted based on the cosine curve until the learning rate decreases to 0. For more information, see SGDR: Stochastic Gradient Descent with Warm Restarts.	STRING	exponential_decay
initial_learning_rate	No	The initial learning rate.	FLOAT	0.01

Parameter	Required	Description	Value type or sample value	Default value
decay_epochs	No	If you set the lr_type parameter to exponential_decay, the decay_epochs parameter is equivalent to the decay_steps parameter of tf.train.exponential.decay. In this case, the decay_epochs parameter specifies the epoch interval at which you want to adjust the learning rate. The system automatically converts the value of the decay_epochs parameter to the value of the decay_steps parameter based on the total number of training data entries. Typically, you can set the decay_epochs parameter to half of the total number of epochs. For example, if the total number of epochs is 20, you can set this parameter to 10. If you set the lr_type parameter to manual_step, the decay_epochs parameter specifies the epochs for which you want to adjust the learning rate. For example, a value of 16 18 indicates that you want to adjust the learning rate for the 16th and 18th epochs. Typically, if the total number of epochs is N, you can set the two values of the decay_epochs parameter to 8/10 × N and 9/10 × N.	INTEGER list. Sample value: 20 20 40 60.	20
decay_factor	No	The decay rate. This parameter is equivalent to the decay_factor parameter of tf.train.exponential.decay.	FLOAT	0.95
staircase	No	Specifies whether the learning rate changes based on the decay_epochs parameter. This parameter is equivalent to the staircase parameter of tf.train.exponential.decay.	BOOL	true
power	No	The power of the polynomial. This parameter is equivalent to the power parameter of tf.train.polynomial.decay.	FLOAT	0.9

Parameter	Required	Description	Value type or sample value	Default value
learning_rates	No	The learning rate that you want to set for each specified epoch. This parameter is required if you set the lr_type parameter to manual_step. If you want to adjust the learning rate for two epochs, specify two learning rates in the value. For example, if the decay_epoches parameter is set to 20 40, you must specify two learning rates in the learning_rates parameter, such as 0.001 0.0001. This indicates that the learning rate of the 20th epoch is adjusted to 0.001 and the learning rate of the 40th epoch is adjusted to 0.0001. We recommend that you adjust the learning rate to one tenth, one hundredth, and one thousandth of the initial learning rate in sequence.	FLOAT list	N/A
lr_warmup	No	Specifies whether to warm up the learning rate.	BOOL	false
lr_warm_up_epochs	No	The number of epochs for which you want to warm up the learning rate.	FLOAT	1
train_data	Yes	The OSS path of the data that is used to train the model.	`oss://path/to/train_*.tfrecord`	N/A
test_data	Yes	The OSS path of the data that is evaluated during the training.	`oss://path/to/test_*.tfrecord`	N/A
train_batch_size	Yes	The size of the data that is used to train the model in the current batch.	INT. Sample value: 32.	N/A
test_batch_size	Yes	The size of the data that is evaluated in the current batch.	INT. Sample value: 32.	N/A
train_num_readers	No	The number of concurrent threads that are used to read the training data.	INT	4
model_dir	Yes	The OSS path of the model.	`oss://path/to/model`	N/A
pretrained_model	No	The OSS path of the pretrained model. If this parameter is specified, the actual model is finetuned based on the pretrained model.	`oss://pai-vision-data-hz/pretrained_models/inception_v4.ckpt`	`""`
use_pretrained_model	No	Specifies whether to use a pretrained model.	BOOL	true

Parameter	Required	Description	Value type or sample value	Default value
num_epochs	Yes	The number of times the data is iterated for the training. A value of 1 indicates that all data is iterated once for the training.	INT. Sample value: 40.	N/A
num_test_example	No	The number of data entries that are evaluated during the training. A value of -1 indicates that all training data is evaluated.	INT. Sample value: 2000.	-1
num_visualizations	No	The number of data entries that can be visualized during the evaluation.	INT	10
save_checkpoint_epochs	No	The epoch interval at which a checkpoint is saved. A value of 1 indicates that a checkpoint is saved each time an epoch is complete.	INT	1
save_summary_epochs	No	The epoch interval at which a summary is saved. A value of 0.01 indicates that a summary is saved each time 1% of the training data is iterated.	FLOAT	0.01
num_train_images	No	The total number of data entries that are used for the training. If you use custom TFRecord files to train the model, this parameter is required.	INT	0
label_map_path	No	The category mapping file. If you use custom TFRecord files to train the model, this parameter is required.	STRING	`""`

Train an image detection model

This topic provides an example on how to run a Machine Learning Platform for AI (PAI) command to train an image detection model. This topic also describes the command parameters.

EasyVision allows you to run a command to train an image detection model. You can use the `-Dparam_config` parameter to set common parameters without the need to know the rules or logic of the EasyVision profile. If you need advanced parameters to train an image detection model, you can use the `-Dconfig` parameter to pass the profile to EasyVision. The following table describes the models that are supported for image detection training.

Model	Backbone	Whether FPN is supported
Faster R-CNN	• resnet_v1_50 • resnet_v1_101 • resnet_v1a_18 • resnet_v1a_34 • resnet_v1d_50 • resnet_v1d_101 • resnet_v1d_152	Yes

Model	Backbone	Whether FPN is supported
R-FCN	• resnet_v1_50 • resnet_v1_101 • resnet_v1a_18 • resnet_v1a_34 • resnet_v1d_50 • resnet_v1d_101 • resnet_v1d_152	No
SSD	• resnet_v1_50 • resnet_v1d_50	Yes
SSD	vgg16_reduce_fc	No
SSD	mobilenet_v1	No

PAI command

- The following code provides an example on how to train a single-shot detection (SSD) model by using a single GPU:

```
pai -name easy_vision
           -Dbuckets='oss://YOUR_BUCKET_NAME/?role_arn=xxx&host=oss-cn-zhangjiakou.aliyuncs.com'
           -DgpuRequired=100
           -Dcmd train
           -Dparam_config '
             --model_type SSD
             --backbone  resnet_v1_50
             --num_classes 20
             --model_dir oss://YOUR_BUCKET_NAME/test/ssd_resnet50
             --train_data oss://pai-vision-data-hz/data/voc0712_tfrecord/voc0712_part_*.tfrecord
             --test_data oss://pai-vision-data-hz/data/voc0712_tfrecord/VOC2007_test.tfrecord
             --num_test_example 2
             --train_batch_size 32
             --test_batch_size 1
             --image_sizes 300
             --lr_type exponential_decay
             --initial_learning_rate 0.001
             --decay_epochs 20
             --staircase true
           '
```

- The following code provides an example on how to train a Faster R-CNN model or an R-FCN model by using a single GPU:

```
pai -name easy_vision
           -Dbuckets='oss://YOUR_BUCKET_NAME/?role_arn=xxx&host=oss-cn-zhangjiakou.aliyuncs.com'
           -DgpuRequired=100
           -Dcmd train
           -Dparam_config '
             --model_type FasterRCNN
             --backbone  resnet_v1_50
             --num_classes 20
             --model_dir oss://YOUR_BUCKET_NAME/test/ssd_fpn_resnet50
             --train_data oss://pai-vision-data-hz/data/voc0712_tfrecord/voc0712_part_*.tfrecord
             --test_data oss://pai-vision-data-hz/data/voc0712_tfrecord/VOC2007_test.tfrecord
             --num_test_example 2
             --train_batch_size 32
             --test_batch_size 1
             --image_min_sizes 600
             --image_max_sizes 1024
             --lr_type exponential_decay
             --initial_learning_rate 0.001
             --decay_epochs 20
             --staircase true
           '
```

- The following code provides an example on how to train an SSD model by using multiple GPUs:

```
pai -name easy_vision
           -Dbuckets='oss://YOUR_BUCKET_NAME/?role_arn=xxx&host=oss-cn-zhangjiakou.aliyuncs.com'
           -Dcmd train
           -Dcluster='{
             \"ps\": {
                 \"count\" : 1,
                 \"cpu\" : 600
             },
             \"worker\" : {
                 \"count\" : 3,
                 \"cpu\" : 800,
                 \"gpu\" : 100
             }
           }'
           -Dparam_config '
             --model_type SSD
             --backbone  resnet_v1_50
             --num_classes 20
             --model_dir oss://YOUR_BUCKET_NAME/test/ssd_fpn_resnet50
             --train_data oss://pai-vision-data-hz/data/voc0712_tfrecord/voc0712_part_*.tfrecord
             --test_data oss://pai-vision-data-hz/data/voc0712_tfrecord/VOC2007_test.tfrecord
             --num_test_example 2
             --train_batch_size 32
             --test_batch_size 1
             --image_sizes 300
             --lr_type exponential_decay
             --initial_learning_rate 0.001
             --decay_epochs 20
             --staircase true
           '
```

Parameters

Parameter	Required	Description	Value type or sample value	Default value
buckets	Yes	The endpoint of the Object Storage Service (OSS) bucket.	`oss://pai-vision-data-hz/?role_arn=acs:ram::1217060697 1xxxx:role/pai-vision-hz&host=cn-hangzhou.oss.aliyun-inc.com`	N/A
cluster	No	The configuration of parameters that are used for distributed training.	JSON string	`""`
gpuRequired	No	Specifies whether to use GPUs. By default, each worker uses one GPU. If you set this parameter to 200, each worker uses two GPUs.	INT	100
cmd	Yes	The type of the EasyVision task. To train a model, you must set this parameter to train.	train	N/A
param_config	Yes	The configuration of parameters that are used for model training. The format of the param_config parameter is the same as that of the ArgumentParser() object in Python. For more information, see param_config.	STRING	N/A

param_config

The param_config parameter contains the parameters that are used for model training. The format of the param_config parameter is the same as that of the ArgumentParser() object in Python. The following code provides an example of the param_config parameter:

```
-Dparam_config = '
--backbone resnet_v1_50
--num_classes 200
--model_dir oss://your/bucket/exp_dir
'
```

Note Do not enclose a parameter value of the STRING type in double quotation marks (") or single quotation marks (').

Parameter	Required	Description	Value type or sample value	Default value

Parameter	Required	Description	Value type or sample value	Default value
model_type	Yes	The type of the model to train. Valid values: • SSD • FasterRCNN • RFCN	STRING	N/A
backbone	Yes	The name of the backbone network that is used by the model. Valid values: • mobilenet_v1 • resnet_v1_50 • resnet_v1_101 • resnet_v1a_18 • resnet_v1a_34 • resnet_v1d_50 • resnet_v1d_101 • vgg16 _reduce_fc	STRING	N/A
weight_decay	No	The value of L2 regularization.	FLOAT	1e-4
use_fpn	No	Specifies whether to use Feature Pyramid Network (FPN).	BOOL	false
num_classes	No	The number of categories, excluding background categories.	INT. Sample value: 21.	N/A
anchor_scales	No	The size of the anchor box. The size of the anchor box is the same as that of the input image in which the anchor box resides after the image is resized. If you use an SSD model, this parameter is not required. If you use FPN, set this parameter to the size of the anchor box in the layer that has the highest resolution. The total number of layers is five. The size of the anchor box in a layer is twice as that in the previous layer. For example, if the size of the anchor box in the first layer is 32, the sizes of the anchor boxes in the next four layers are 64, 128, 256, and 512. If you use the FasterRCNN model or the RFCN model without FPN support, you can specify multiple anchor sizes as needed. For example, you can set the anchor_scales parameter to 128 256 512.	FLOAT list. Sample value: 32 (single scale) or 128 256 512 (multiple scales).	• SSD: The default value is 0.1, 0.2, 0.37, 0.54, 0.71, 0.88, 0.96, or 1.0 times of the size of the input image. • FNP: 32 • Faster R-CNN with FNP support: 32 • Faster R-CNN or R-FCN: [128 256 512]
anchor_ratios	No	The ratios of the width to the height of the anchor boxes.	FLOAT list	0.5 1 2

Parameter	Required	Description	Value type or sample value	Default value
image_sizes	No	The size of the images after they are resized. This parameter takes effect only when an SSD model is used. The value of this parameter is a list that contains two numbers, which indicate the height and width.	FLOAT list	300 300
image_min_sizes	No	The length of the shorter side of images after they are resized. This parameter is used for Faster R-CNN and R-FCN models. If you specify multiple lengths for the shorter sides of the images in the value of this parameter, the last one is used to evaluate the model, whereas one of the others is randomly selected to train the model. This way, the multi-scale training is supported. If you specify only one length for the longer sides of images, this length is used for both training and evaluation.	FLOAT list	600
image_max_sizes	No	The length of the longer side of images after they are resized. This parameter is used for Faster R-CNN and R-FCN models. If you specify multiple lengths for the longer sides of the images in the value of this parameter, the last one is used to evaluate the model, whereas one of the others is randomly selected to train the model. This way, the multi-scale training is supported. If you specify only one length for the longer sides of images, this length is used for both training and evaluation.	FLOAT list	1024
optimizer	No	The type of the optimizer. Valid values: • momentum: stochastic gradient descent (SGD) with momentum • adam	STRING	momentum

Parameter	Required	Description	Value type or sample value	Default value
lr_type	No	The policy that is used to adjust the learning rate. Valid values: • exponential_decay: The learning rate is subject to exponential decay. • polynomial_decay: The learning rate is subject to polynomial decay. If you set the lr_type parameter to polynomial_decay, the num_steps parameter is automatically set to the total number of training iterations. The value of the end_learning_rate parameter is automatically set to one thousandth of the value of the initial_learning_rate parameter. • manual_step: The learning rate of each epoch is manually adjusted. If you set the lr_type parameter to manual_step, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rate. You must also set the learning_rates parameter to specify the learning rate of each epoch as needed. • cosine_decay The learning rate of each epoch is adjusted based on the cosine curve until the learning rate decreases to 0. For more information, see SGDR: Stochastic Gradient Descent with Warm Restarts. If you set the lr_type parameter to cosine_decay, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rate.	STRING	exponential_decay
initial_learning_rate	No	The initial learning rate.	Floating point	0.01

Parameter	Required	Description	Value type or sample value	Default value
decay_epochs	No	If you set the lr_type parameter to exponential_decay, the decay_epochs parameter is equivalent to the decay_steps parameter of tf.train.exponential.decay. In this case, the decay_epochs parameter specifies the epoch interval at which you want to adjust the learning rate. The system automatically converts the value of the decay_epochs parameter to the value of the decay_steps parameter based on the total number of training data entries. Typically, you can set the decay_epochs parameter to half of the total number of epochs. For example, if the total number of epochs is 20, you can set this parameter to 10. If you set the lr_type parameter to manual_step, the decay_epochs parameter specifies the epochs for which you want to adjust the learning rate. For example, a value of 16 18 indicates that you want to adjust the learning rate for the 16th and 18th epochs. Typically, if the total number of epochs is N, you can set the two values of the decay_epochs parameter to 8/10 × N and 9/10 × N.	INTEGER list. Sample value: 20 20 40 60.	20
decay_factor	No	The decay rate. This parameter is equivalent to the decay_factor parameter of tf.train.exponential.decay.	FLOAT	0.95
staircase	No	Specifies whether the learning rate changes based on the decay_epochs parameter. This parameter is equivalent to the staircase parameter of tf.train.exponential.decay.	BOOL	true
power	No	The power of the polynomial. This parameter is equivalent to the power parameter of tf.train.polynomial.decay.	FLOAT	0.9

Parameter	Required	Description	Value type or sample value	Default value
learning_rates	No	The learning rate that you want to set for each specified epoch. This parameter is required if you set the lr_type parameter to manual_step. If you want to adjust the learning rate for two epochs, specify two learning rates in the value. For example, if the decay_epoches parameter is set to 20 40, you must specify two learning rates in the learning_rates parameter, such as 0.001 0.0001. This indicates that the learning rate of the 20th epoch is adjusted to 0.001 and the learning rate of the 40th epoch is adjusted to 0.0001. We recommend that you adjust the learning rate to one tenth, one hundredth, and one thousandth of the initial learning rate in sequence.	FLOAT list	N/A
lr_warmup	No	Specifies whether to warm up the learning rate.	BOOL	false
lr_warm_up_epochs	No	The number of epochs for which you want to warm up the learning rate.	FLOAT	1
train_data	Yes	The OSS path of the data that is used to train the model.	`oss://path/to/train_*.tfrecord`	N/A
test_data	Yes	The OSS path of the data that is evaluated during the training.	`oss://path/to/test_*.tfrecord`	N/A
train_batch_size	Yes	The size of the data that is used to train the model in the current batch.	INT. Sample value: 32.	N/A
test_batch_size	Yes	The size of the data that is evaluated in the current batch.	INT. Sample value: 32.	N/A
train_num_readers	No	The number of concurrent threads that are used to read the training data.	INT	4
model_dir	Yes	The OSS path of the model.	`oss://path/to/model`	N/A
pretrained_model	No	The OSS path of the pretrained model. If this parameter is specified, the actual model is finetuned based on the pretrained model.	`oss://pai-vision-data-sh/pretrained_models/inception_v4.ckpt`	`""`
use_pretrained_model	No	Specifies whether to use a pretrained model.	BOOL	true

Parameter	Required	Description	Value type or sample value	Default value
num_epochs	Yes	The number of training iterations. A value of 1 indicates that all data is iterated once for the training.	INT. Sample value: 40.	N/A
num_test_example	No	The number of data entries that are evaluated during the training. A value of -1 indicates that all training data is evaluated.	INT. Sample value: 2000.	-1
num_visualizations	No	The number of data entries that can be visualized during the evaluation.	INT	10
save_checkpoint_epochs	No	The epoch interval at which a checkpoint is saved. A value of 1 indicates that a checkpoint is saved each time an epoch is complete.	INT	1
save_summary_epochs	No	The epoch interval at which a summary is saved. A value of 0.01 indicates that a summary is saved each time 1% of the training data is iterated.	FLOAT	0.01
num_train_images	No	The total number of data entries that are used for the training. If you use custom TFRecord files to train the model, this parameter is required.	INT	0
label_map_path	No	The category mapping file. If you use custom TFRecord files to train the model, this parameter is required.	STRING	`""`

Train an image classification model

This topic provides examples on how to run a Machine Learning Platform for AI (PAI) command to train an image classification model in EasyVision. This topic also describes the command parameters.

EasyVision allows you to run a command to train an image classification model. When you run the command, you need only to set the common classification parameters without the need to know the rules or logic of the EasyVision profile.

PAI command

- Training on a single server

```
pai -name easy_vision
           -Dbuckets='oss://pai-vision-data-hz/?role_arn=xxx&host=oss-cn-zhangjiakou.aliyuncs.com'
           -DgpuRequired=100
           -Dcmd train
           -Dparam_config '
             --model_type Classification
             --backbone  inception_v4
             --num_classes 10
             --num_epochs 1
             --model_dir oss://pai-vision-data-hz/test/cifar_inception_v4
             --use_pretrained_model true
             --train_data oss://pai-vision-data-hz/data/test/cifar10/*.tfrecord
             --test_data oss://pai-vision-data-hz/data/test/cifar10/*.tfrecord
             --num_test_example 20
             --train_batch_size 32
             --test_batch_size=32
             --image_size 299
             --initial_learning_rate 0.01
             --staircase true   '
```

- Training on multiple servers

```
pai -name easy_vision
           -Dbuckets='oss://pai-vision-data-hz/?role_arn=xxx&host=oss-cn-zhangjiakou.aliyuncs.com'
           -Dcmd train
           -Dcluster='{
             \"ps\": {
                 \"count\" : 1,
                 \"cpu\" : 600
             },
             \"worker\" : {
                 \"count\" : 3,
                 \"cpu\" : 800,
                 \"gpu\" : 100
             }
           }'
           -Dparam_config='
             --model_type Classification
             --backbone  inception_v4
             --num_classes 10
             --num_epochs 1
             --model_dir oss://pai-vision-data-hz/test/cifar_inception_v4_dis
             --use_pretrained_model true
             --train_data oss://pai-vision-data-hz/data/test/cifar10/*.tfrecord
             --test_data oss://pai-vision-data-hz/data/test/cifar10/*.tfrecord
             --num_test_example 20
             --train_batch_size 32
             --test_batch_size=32
             --image_size 299
             --initial_learning_rate 0.01
             --staircase true
           '
```

Parameters

Parameter	Required	Description	Value type or sample value	Default value
buckets	Yes	The endpoint of the Object Storage Service (OSS) bucket.	`oss://pai-vision-data-hz/?role_arn=acs:ram::1217060697 1xxxx:role/pai-vision-hz&host=cn-hangzhou.oss.aliyun-inc.com`	N/A
cluster	No	The configuration of parameters that are used for distributed training.	JSON string	""
gpuRequired	No	Specifies whether to use GPUs. By default, each worker uses one GPU. If you set this parameter to 200, each worker uses two GPUs.	100	100
cmd	Yes	The type of the EasyVision task. To train a model, you must set this parameter to train.	train	N/A
param_config	Yes	The configuration of parameters that are used for model training. The format of the param_config parameter is the same as that of the ArgumentParser() object in Python. For more information, see the following table.	STRING	N/A

param_config

The param_config parameter contains the parameters that are used for model training. The format of the param_config parameter is the same as that of the ArgumentParser() object in Python. The following code provides an example of the param_config parameter:

```
-Dparam_config = '
--model_type Classification
--backbone inception_v4
--num_classes 200
--model_dir oss://your/bucket/exp_dir
'
```

Note Do not enclose a parameter value of the STRING type in double quotation marks (") or single quotation marks (').

Parameter	Required	Description	Value type or sample value	Default value

Parameter	Required	Description	Value type or sample value	Default value
model_type	Yes	The type of the model to train. To train a model for multi-label image classification, set this parameter to Classification.	STRING	N/A
backbone	No	The name of the backbone network that is used by the model. Valid values: • vgg_16 • vgg_19 • inception_v1 • inception_v2 • inception_v3 • inception_v4 • mobilenet_v1 • mobilenet_v2 • mobilenet_v3 • resnet_v1_50 • resnet_v1_101 • resnet_v1_152 • resnet_v2_50 • resnet_v2_101 • resnet_v2_152 • resnet_v1a_18 • resnet_v1a_34 • resnet_v1a_50 • resnet_v1a_101 • resnet_v1a_152 • resnet_v1b_50 • resnet_v1b_101 • resnet_v1b_152 • resnet_v1c_50 • resnet_v1c_101 • resnet_v1c_152 • resnet_v1d_50 • resnet_v1d_101 • resnet_v1d_152 • efficientnet-b0 • efficientnet-b1 • efficientnet-b2 • efficientnet-b3 • efficientnet-b4 • efficientnet-b5 • efficientnet-b6 • efficientnet-b7 • efficientnet-b8	STRING	inception_v4

Parameter	Required	Description	Value type or sample value	Default value
weight_decay	No	The value of L2 regularization.	FLOAT	1e-4
num_classes	Yes	The number of categories.	INT	N/A
hidden_size	No	The size of the custom Fiber Channel (FC) layer. A value of -1 indicates that the size of the original FC layer is used.	INT	-1
image_size	No	The size of the images after they are resized, in pixels.	INT	224
use_crop	No	Specifies whether to crop images for data enhancement.	BOOL	true
crop_min_area	No	The minimum proportion of the original image area that is occupied by the crop box when you set the use_crop parameter to true.	FLOAT	0.7
eval_each_category	No	Specifies whether to separately evaluate each category.	BOOL	false
optimizer	No	The type of the optimizer. Valid values: • momentum: stochastic gradient descent (SGD) with momentum • adam	STRING	momentum

Parameter	Required	Description	Value type or sample value	Default value
lr_type	No	The policy that is used to adjust the learning rate. Valid values: • exponential_decay: The learning rate is subject to exponential decay. • polynomial_decay: The learning rate is subject to polynomial decay. If you set the lr_type parameter to polynomial_decay, the num_steps parameter is automatically set to the total number of training iterations. The value of the end_learning_rate parameter is automatically set to one thousandth of the value of the initial_learning_rate parameter. • manual_step: The learning rate of each epoch is manually adjusted. If you set the lr_type parameter to manual_step, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rate. You must also set the learning_rates parameter to specify the learning rate as needed. • cosine_decay: The learning rate of each epoch is adjusted based on the cosine curve. For more information, see SGDR: Stochastic Gradient Descent with Warm Restarts.	STRING	exponential_decay
initial_learning_rate	No	The initial learning rate.	FLOAT	0.01

Parameter	Required	Description	Value type or sample value	Default value
decay_epochs	No	If you set the lr_type parameter to exponential_decay, the decay_epochs parameter is equivalent to the decay_steps parameter of tf.train.exponential.decay. In this case, the decay_epochs parameter specifies the epoch interval at which you want to adjust the learning rate. The system automatically converts the value of the decay_epochs parameter to the value of the decay_steps parameter based on the total number of training data entries. Typically, you can set the decay_epochs parameter to half of the total number of epochs. For example, if the total number of epochs is 20, you can set this parameter to 10. If you set the lr_type parameter to manual_step, the decay_epochs parameter specifies the epochs for which you want to adjust the learning rate. For example, a value of 16 18 indicates that you want to adjust the learning rate for the 16th and 18th epochs. Typically, if the total number of epochs is N, you can set the two values of the decay_epochs parameter to 8/10 × N and 9/10 × N.	INTEGER list. Sample value: 20 20 40 60.	20
decay_factor	No	The decay rate. This parameter is equivalent to the decay_factor parameter of tf.train.exponential.decay.	FLOAT	0.95
staircase	No	Specifies whether the learning rate changes based on the decay_epochs parameter. This parameter is equivalent to the staircase parameter of tf.train.exponential.decay.	BOOL	true
power	No	The power of the polynomial. This parameter is equivalent to the power parameter of tf.train.polynomial.decay.	FLOAT	0.9

Parameter	Required	Description	Value type or sample value	Default value
learning_rates	No	The learning rate that you want to set for each specified epoch. This parameter is required if you set the lr_type parameter to manual_step. If you want to adjust the learning rate for two epochs, specify two learning rates in the value. For example, if the decay_epoches parameter is set to 20 40, you must specify two learning rates in the learning_rates parameter, such as 0.001 0.0001. This indicates that the learning rate of the 20th epoch is adjusted to 0.001 and the learning rate of the 40th epoch is adjusted to 0.0001. We recommend that you adjust the learning rate to one tenth, one hundredth, and one thousandth of the initial learning rate in sequence.	FLOAT list	N/A
lr_warmup	No	Specifies whether to warm up the learning rate.	BOOL	false
lr_warm_up_epochs	No	The number of epochs for which you want to warm up the learning rate.	FLOAT	1
train_data	Yes	The OSS path of the data that is used to train the model.	`oss://path/to/train_*.tfrecord`	N/A
test_data	Yes	The OSS path of the data that is evaluated during the training.	`oss://path/to/test_*.tfrecord`	N/A
train_batch_size	Yes	The size of the data that is used to train the model in the current batch.	INT. Sample value: 32.	N/A
test_batch_size	Yes	The size of the data that is evaluated in the current batch.	INT. Sample value: 32.	N/A
train_num_readers	No	The number of concurrent threads that are used to read the training data.	INT	4
model_dir	Yes	The OSS path of the model.	`oss://path/to/model`	N/A
pretrained_model	No	The OSS path of the pretrained model. If this parameter is specified, the actual model is finetuned based on the pretrained model.	`oss://pai-vision-data-sh/pretrained_models/inception_v4.ckpt`	`""`
use_pretrained_model	No	Specifies whether to use a pretrained model.	BOOL	true

Parameter	Required	Description	Value type or sample value	Default value
num_epochs	Yes	The number of times the data is iterated for the training. A value of 1 indicates that all data is iterated once for the training.	INT. Sample value: 40.	N/A
num_test_example	No	The number of data entries that are evaluated during the training. A value of -1 indicates that all training data is evaluated.	INT. Sample value: 2000.	-1
num_visualizations	No	The number of data entries that can be visualized during the evaluation.	INT	10
save_checkpoint_epochs	No	The epoch interval at which a checkpoint is saved. A value of 1 indicates that a checkpoint is saved each time an epoch is complete.	INT	1
save_summary_epochs	No	The epoch interval at which a summary is saved. A value of 0.01 indicates that a summary is saved each time 1% of the training data is iterated.	FLOAT	0.01
num_train_images	No	The total number of data entries that are used for the training. If you use custom TFRecord files to train the model, this parameter is required.	INT	0
label_map_path	No	The category mapping file. If you use custom TFRecord files to train the model, this parameter is required.	STRING	`""`

Train a video classification model

This topic provides an example on how to run a Machine Learning Platform for AI (PAI) command to train a video classification model. This topic also describes the command parameters.

You can run the `ev_video_classification` command to train a model to classify videos. When you run this command, you need only to set the common classification parameters without the need to know the rules or logic of the EasyVision profile.

PAI command

- Training on a single server

```
pai -name ev_video_classification
             -Dbackbone='resnet_3d_50'
             -Dnum_classes=101
             -Dnum_epochs=2
             -Ddecay_epochs=1
             -Dsave_checkpoints_epochs=1
             -Dmodel_dir=oss://pai-vision-data-hz/test/UCF101_resnet_3d_50
             -Dpretrained_model='oss://pai-vision-data-hz/pretrained_models/resnet_3d_50/resent_3d
_50_model.ckpt'
             -Dtrain_data='oss://pai-vision-data-hz/data/ucf101/train/ucf101_train_*.tfrecord'
             -Dtest_data='oss://pai-vision-data-hz/data/ucf101/test/ucf101_test_0.tfrecord'
             -Dlabel_map_path='oss://pai-vision-data-hz/data/ucf101/test/ucf101_label_map.pbtxt'
             -Dnum_test_example=3783
             -Dtrain_batch_size=32
             -Dtest_batch_size=128
             -Dinitial_learning_rate=0.0001
             -Dstaircase=true
             -Dbuckets='YOUR_OSS_BUCKET_CONFIG'
             -DgpuRequired=100
```

- Training on multiple servers

```
pai -name ev_video_classification
             -Dbackbone='resnet_3d_50'
             -Dnum_classes=101
             -Dnum_epochs=2
             -Ddecay_epochs=1
             -Dsave_checkpoints_epochs=1
             -Dmodel_dir=oss://pai-vision-data-hz/test/UCF101_resnet_3d_50
             -Dtrain_data='oss://pai-vision-data-hz/data/ucf101/train/ucf101_train_*.tfrecord'
             -Dtest_data='oss://pai-vision-data-hz/data/ucf101/test/ucf101_test_0.tfrecord'
             -Dlabel_map_path='oss://pai-vision-data-hz/data/ucf101/test/ucf101_label_map.pbtxt'
             -Dpretrained_model='oss://pai-vision-data-hz/pretrained_models/resnet_3d_50/resent_3d
_50_model.ckpt'             -Dnum_test_example=3783
             -Dtrain_batch_size=32
             -Dtest_batch_size=128
             -Dinitial_learning_rate=0.0001
             -Dstaircase=true
             -Dbuckets='YOUR_OSS_BUCKET_CONFIG'
             -Dcluster='{
               \"ps\": {
                   \"count\" : 1,
                   \"cpu\" : 600
               },
               \"worker\" : {
                   \"count\" : 3,
                   \"cpu\" : 800,
                   \"gpu\" : 100
               }
             }'
```

Parameters

Parameter	Required	Description	Value type or sample value	Default value

Parameter	Required	Description	Value type or sample value	Default value
backbone	Yes	The name of the backbone network that is used by the model. Valid values: • c3d • resnet_3d_18 • resnet_3d_34 • resnet_3d_50 • resnet_3d_101 • resnet_3d_152 • resnet_3d_200 • resnext_3d_50 • resnext_3d_101 • resnext_3d_152 • resnext_3d_200	STRING	resnet_3d_50
num_classes	Yes	The number of categories.	100	N/A
feature_name	No	The names of features. Example: ["resnet_3d_50/poo2","resnet_3d_50/scale5/conv5","resnet_3d_50/scale4/conv4","resnet_3d_50/scale3/conv3","resnet_3d_50/scale2/conv2","resnet_3d_50/pool1","resnet_3d_50/scale1/conv1",].	Character List	null
optimizer	No	The type of the optimizer. Valid values: • momentum: stochastic gradient descent (SGD) with momentum • adam	STRING	momentum
decay_epochs	No	The epoch interval at which you want to adjust the learning rate. This parameter is equivalent to the decay_steps parameter of tf.train.exponential.decay.	INT	10
decay_factor	No	The decay rate. This parameter is equivalent to the decay_factor parameter of tf.train.exponential.decay.	FLOAT	0.95
staircase	No	Specifies whether the learning rate changes based on the decay_epochs parameter. This parameter is equivalent to the staircase parameter of tf.train.exponential.decay.	BOOL	true
cluster	No	The configuration of parameters that are used for distributed training.	JSON string	`""`
gpuRequired	No	Specifies whether to use GPUs. By default, each worker uses one GPU. If you set this parameter to 200, each worker uses two GPUs.	INT	100

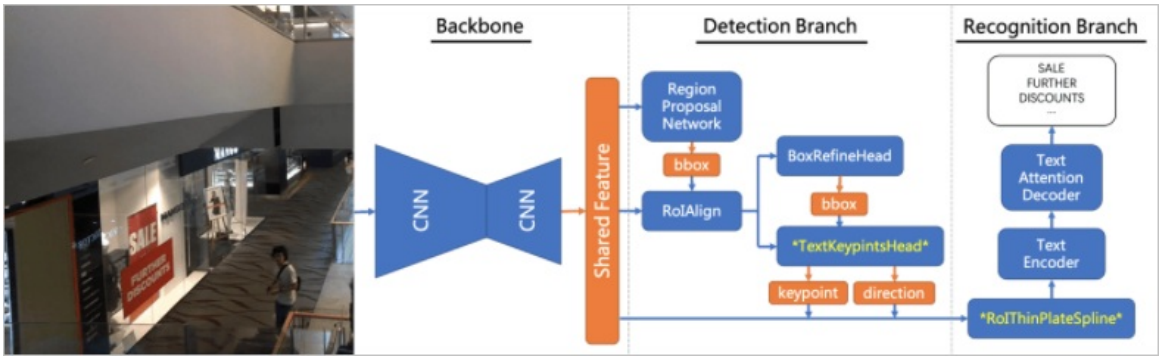
Parameter	Required	Description	Value type or sample value	Default value
train_data	Yes	The Object Storage Service (OSS) path of the data that is used to train the model.	`oss://path/to/train_*.tfrecord`	N/A
test_data	Yes	The OSS path of the data that is evaluated during the training.	`oss://path/to/test_*.tfrecord`	N/A
train_batch_size	Yes	The size of the data that is used to train the model in the current batch.	INT. Sample value: 32.	N/A
test_batch_size	Yes	The size of the data that is evaluated in the current batch.	INT. Sample value: 32.	N/A
model_dir	Yes	The OSS path of the model.	oss://path/to/model	N/A
buckets	Yes	The endpoint of the OSS bucket.	`oss://pai-vision-data-hz/?role_arn=acs:ram::1217060697 1xxxx:role/pai-vision-hz&host=cn-hangzhou.oss.aliyun-inc.com`	N/A
train_num_readers	No	The number of concurrent threads that are used to read the training data.	INT	4
pretrained_model	No	The OSS path of the pretrained model. If this parameter is specified, the actual model is finetuned based on the pretrained model.	`oss://pai-vision-data-sh/pretrained_models/inception_v4.ckpt`	`""`
initial_learning_rate	No	The initial learning rate.	FLOAT	0.0001
num_epochs	Yes	The number of training iterations. All data is iterated once for the training.	INT. Sample value: 20.	N/A
num_test_example	Yes	The number of data entries that are evaluated during the training.	INT. Sample value: 2000.	N/A
save_checkpoint_epochs	No	The epoch interval at which a checkpoint is saved.	INT	1

Parameter	Required	Description	Value type or sample value	Default value
label_map_path	Yes	The category mapping file.	STRING. Sample value: `oss://path/to/*_labelmap.pbtxt`.	N/A

Training of end-to-end text recognition

EasyVision of Machine Learning Platform for AI (PAI) allows you to train models for end-to-end text recognition and use the trained models to make predictions. This topic shows you how to use PAI commands to train a model for end-to-end text recognition. This topic also provides sample PAI commands and describes related parameters.

EasyVision simplifies the training configuration. You can use the `-Dparam_config` parameter to set common parameters. This way, you do not need to know the rules or logic of the configuration files of EasyVision. If you need advanced parameters to train a model for end-to-end text recognition, you can use the `-Dconfig` parameter to pass the configuration file to EasyVision. The following figure shows the algorithm framework of the models for end-to-end text recognition.



Training of end-to-end text recognition

```
pai -name easy_vision
           -Dbuckets='oss://pai-vision-data-hz/?role_arn=xxx/pai-vision-hz&host=oss-cn-zhangjiakou.aliyuncs.com'
           -DgpuRequired=100
           -Dcmd train
           -Dparam_config '
             --model_type TextEnd2End
             --backbone  resnet_v1_50
             --num_classes 1
             --use_pretrained_model true
             --train_batch_size 1
             --test_batch_size 1
             --image_min_sizes 960
             --image_max_sizes 1440
             --initial_learning_rate 0.0001
             --optimizer adam
             --lr_type exponential_decay
             --decay_epochs 40
             --decay_factor 0.5
             --num_epochs 10
             --staircase true
             --predict_text_direction true
             --text_direction_trainable true
             --text_direction_type smart_unified
             --feature_gather_type fixed_height_pyramid
             --train_data oss://pai-vision-data-hz/data/recipt_text/end2end_tfrecords/train_*.tfrecord
             --test_data oss://pai-vision-data-hz/data/recipt_text/end2end_tfrecords/test.tfrecord
             --model_dir oss://pai-vision-data-hz/test/recipt_text/text_end2end_krcnn_resnet50_attn
           '
```

Parameters

Parameter	Required	Description	Value type or value format	Default Value
		The information about the Object Storage Service (OSS) buckets. This parameter supports three value formats. Set the following variables in the value formats to their actual values and keep the remaining part unchanged. • <your_bucket>: Set this variable to the name of the OSS bucket that you created. • <your_arn>: You can obtain the value of this variable by	The following three value formats are	

Parameter	Required	Description	Value type or value format	Default Value
buckets	Yes	performing the following operations: i. In the top navigation bar of the Apsara Uni-manager Management Console, click **Configurations**. ii. On the **Service-linked Roles** page, view the role identifier in the **Role Identifier** column. The role identifier is the Apsara Stack resource name that globally and uniquely identifies a RAM role. If no role exists, you can create a role on the page. • <your_host>: Set this variable to the endpoint of your OSS bucket. You can obtain the endpoint in the OSS console.	supported: • "oss://<your_bucket>/?role_arn=<your_arn>&host=<your_host>" • "oss://<your_bucket1>/?role_arn=<your_arn>,oss://<your_bucket2>/?role_arn=<your_arn>&host=<your_host>" • "oss://<your_bucket1>/?role_arn=<your_arn>&host=<your_host>,oss://<your_bucket2>/"	N/A
cluster	No	The configuration of parameters that are used for distributed training.	JSON string	""
gpuRequired	No	Specifies whether to use GPUs. By default, each worker uses one GPU. If you set this parameter to 200, each worker uses two GPUs.	INT	100

Parameter	Required	Description	Value type or value format	Default Value
cmd	Yes	The type of the EasyVision task. To train a model, you must set this parameter to train.	STRING	N/A
param_config	Yes	The configuration of parameters that are used for model training. The format of the param_config parameter is the same as that of the ArgumentParser() object in Python. For more information, see param_config.	STRING	N/A

param_config

The param_config parameter contains the parameters that are used for model training. The format of the param_config parameter is the same as that of the ArgumentParser() object in Python. The following code provides an example of the param_config parameter:

```
-Dparam_config = '
--backbone resnet_v1_50
--model_dir oss://your/bucket/exp_dir
'
```

Note The values of all string parameters in the param_config parameter are not enclosed in double quotation marks (") or single quotation marks (').

Parameter	Required	Description	Value type or value format	Default Value
model_type	Yes	The type of the model to train. Set this parameter to TextEnd2End when you train a model for end-to-end text recognition.	STRING	N/A
backbone	Yes	The name of the backbone network that is used by the model. Valid values: • resnet_v1_50 • resnet_v1_101	STRING	N/A
weight_decay	No	The value of L2 regularization.	FLOAT	1e-4
num_classes	No	The number of categories. By default, the value is obtained by analyzing the dataset that is used for model training.	INT	-1

Parameter	Required	Description	Value type or value format	Default Value
anchor_scales	No	The size of the anchor box. The size of the anchor box is the same as that of the input image where the anchor box resides after the image is resized. Set this parameter to the size of the anchor box in the layer that has the highest resolution. The total number of layers is five. The size of the anchor box in a layer is twice as that in the previous layer. For example, if the size of the anchor box in the first layer is 32, the sizes of the anchor boxes in the next four layers are 64, 128, 256, and 512.	FLOAT list. Example value: 32 (single scale).	24
anchor_ratios	No	The ratios of the width to the height of the anchor boxes.	FLOAT list	0.2 0.5 1 2 5
predict_text_direction	No	Specifies whether to predict the text orientation.	BOOL	false
text_direction_trainable	No	Specifies whether to train the model to predict the text orientation.	BOOL	false
text_direction_type	No	The type of the prediction of text orientation. Valid values: • normal: greedy prediction. • unified: The orientation of the most text lines is determined as the text orientation. • smart_unified: The orientation of the most text lines excluding the lines of which the height is twice the width is determined as the text orientation.	STRING	normal
feature_gather_type	No	The type of the extractor that is used to extract features of the text lines. Valid values: • fixed_size: extracts features based on the specified width and height. • fixed_height: extracts features based on the specified height and the specified ratio of the width to the height. • fixed_height_pyramid: extracts features from multi-scale features based on the specified height and the specified ratio of the width to the height.	STRING	fixed_height

Parameter	Required	Description	Value type or value format	Default Value
feature_gather_aspect_ratio	No	The ratio of the width to the height of the text lines. If you set the feature_gather_type parameter to fixed_size, this parameter specifies the ratio of the specified height to width after the features are resized. If you set the feature_gather_type parameter to fixed_height, this parameter specifies the maximum ratio of the specified height to the custom width after the features are resized.	FLOAT	40
feature_gather_batch_size	No	The size of the current batch of the text lines that are used to train the model.	INT	160
recognition_norm_type	No	The type of the normalization that is used by the encoder and feature extractor. Valid values: • batch_norm • group_norm	STRING	group_norm
recognition_bn_trainable	No	Specifies whether the batch normalization value that is obtained by the encoder and feature extractor can be used for the training. This parameter takes effect only when the norm_type parameter is set to batch_norm.	BOOL	false
encoder_type	No	The type of the encoder. Valid values: • crnn: hybrid Convolutional Neural Network (CNN)-Recurrent Neural Network (RNN) encoder. • cnn_line: CNN encoder. • cnn_spatial: the encoder that uses spatial attention CNN.	STRING	crnn
encoder_cnn_name	No	The type of CNN that is used by the encoder. Valid values: • conv5_encoder • senet5_encoder	STRING	senet5_encoder
encoder_num_layers	No	The number of RNN layers in the encoder. CNN layers are not counted.	INT	2
encoder_rnn_type	No	The type of RNN that is used by the encoder. Valid values: • bi: bidirectional RNN. • uni: unidirectional RNN.	STRING	uni
encoder_hidden_size	No	The number of neurons in the hidden layer of the encoder.	INT	512

Parameter	Required	Description	Value type or value format	Default Value
encoder_cell_type	No	The type of RNN cells in the encoder. Valid values: • basic_lstm • gru • layer_norm_basic_lstm • nas	STRING	basic_lstm
decoder_type	No	The type of the decoder. Valid values: • attention • ctc	STRING	attention
decoder_num_layers	No	The number of layers in the decoder.	INT	2
decoder_hidden_size	No	The number of neurons in the hidden layer of the decoder.	INT	512
decoder_cell_type	No	The type of RNN cells in the decoder. Valid values: • basic_lstm • gru • layer_norm_basic_lstm • nas	STRING	basic_lstm
embedding_size	No	The embedding size of the dictionary.	INT	64
beam_width	No	The beam width of the beam search.	INT	0
length_penalty_weight	No	The length penalty score of the beam search. This prevents shorter sentences from receiving higher scores.	FLOAT	0.0
attention_mechanism	No	The type of the attention mechanism of the decoder. Valid values: • luong • scaled_luong • bahdanau • normed_bahdanau	STRING	normed_bahdanau
aspect_ratio_min_jitter_coef	No	The minimum ratio of the width to the height at which images can be resized during the training. The value 0 indicates that the ratios of the width to the height of images remain unchanged during the training.	FLOAT	0.8

Parameter	Required	Description	Value type or value format	Default Value
aspect_ratio_max_jitter_coef	No	The maximum ratio of the width to the height at which images can be resized during the training. The value 0 indicates that the ratios of the width to the height of images remain unchanged during the training.	FLOAT	1.2
random_rotation_angle	No	The maximum angle to which images can be randomly rotated during the training, in the clockwise or anticlockwise direction. The value 0 indicates that images are not randomly rotated during the training.	FLOAT	10
random_crop_min_area	No	The minimum ratio of the size of an image after it is randomly cropped to the size of the original image. The value 0 indicates that images are not randomly cropped during the training.	FLOAT	0.1
random_crop_max_area	No	The maximum ratio of the size of an image after it is randomly cropped to the size of the original image. The value 0 indicates that images are not randomly cropped during the training.	FLOAT	1.0
random_crop_min_aspect_ratio	No	The minimum ratio of the width to the height of images after they are randomly cropped during the training. The value 0 indicates that images are not randomly cropped during the training.	FLOAT	0.2
random_crop_max_aspect_ratio	No	The maximum ratio of the width to the height of images after they are randomly cropped during the training. The value 0 indicates that images are not randomly cropped during the training.	FLOAT	5
image_min_sizes	No	The length of the shorter side of images after they are resized. If you specify multiple lengths for the shorter sides of the images in the value of this parameter, the last one is used to evaluate the model, whereas one of the others is randomly selected to train the model. This way, the multi-scale training is supported. If you set only one length for the longer sides of images, this length is used for both training and evaluation.	FLOAT list	800

Parameter	Required	Description	Value type or value format	Default Value
image_max_sizes	No	The length of the longer side of images after they are resized. If you specify multiple lengths for the longer sides of the images in the value of this parameter, the last one is used to evaluate the model, whereas one of the others is randomly selected to train the model. This way, the multi-scale training is supported. If you set only one length for the longer sides of images, this length is used for both training and evaluation.	FLOAT list	1200
random_distort_color	No	Specifies whether to randomly change the brightness, contrast, and saturation of images during the training.	BOOL	true
optimizer	No	The type of the optimizer. Valid values: • momentum: stochastic gradient descent (SGD) with momentum • adam	STRING	momentum

Parameter	Required	Description	Value type or value format	Default Value
lr_type	No	The policy that is used to adjust the learning rate. Valid values: • exponential_decay: the exponential decay. • polynomial_decay: the polynomial decay. If you set the lr_type parameter to polynomial_decay, the num_steps parameter is automatically set to the total number of training iterations. The value of the end_learning_rate parameter is automatically set to one thousandth of the value of the initial_learning_rate parameter. • manual_step: specifies to manually adjust the learning rate for each epoch. If you set the lr_type parameter to manual_step, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rate. You must also set the learning_rates parameter to specify the learning rate as needed. • cosine_decay adjusts the learning rate by following the cosine curve. For more information, see SGDR: Stochastic Gradient Descent with Warm Restarts. If you set the lr_type parameter to cosine_decay, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rates.	STRING	exponential_decay
initial_learning_rate	No	The initial learning rate.	FLOAT	0.01

Parameter	Required	Description	Value type or value format	Default Value
decay_epochs	No	If you set the lr_type parameter to exponential_decay, the decay_epochs parameter is equivalent to the decay_steps parameter of tf.train.exponential.decay. In this case, the decay_epochs parameter specifies the epoch interval at which you want to adjust the learning rate. The system automatically converts the value of the decay_epochs parameter to the value of the decay_steps parameter based on the total number of training data entries. Typically, you can set the decay_epochs parameter to half of the total number of epochs. For example, if the total number of epochs is 20, you can set this parameter to 10. If you set the lr_type parameter to manual_step, the decay_epochs parameter specifies the epochs for which you want to adjust the learning rate. For example, a value of 16 18 indicates that you want to adjust the learning rate for the 16th and 18th epochs. Typically, if the total number of epochs is N, you can set the two values of the decay_epochs parameter to 8/10 × N and 9/10 × N.	INTEGER list. Sample value: 20 20 40 60.	20
decay_factor	No	The decay rate. This parameter is equivalent to the decay_factor parameter of tf.train.exponential.decay.	FLOAT	0.95
staircase	No	Specifies whether the learning rate changes based on the decay_epochs parameter. This parameter is equivalent to the staircase parameter of tf.train.exponential.decay.	BOOL	true
power	No	The power of the polynomial. This parameter is equivalent to the power parameter of tf.train.polynomial.decay.	FLOAT	0.9

Parameter	Required	Description	Value type or value format	Default Value
learning_rates	No	The learning rate that you want to set for the specified epochs. This parameter is required when you set the lr_type parameter to manual_step. If you want to adjust the learning rate for two epochs, specify two learning rates in the value. For example, if the decay_epoches parameter is set to 20 40, you must specify two learning rates in the learning_rates parameter, such as 0.001 0.0001. This indicates that the learning rate of the 20th epoch is adjusted to 0.001 and that of the 40th epoch is adjusted to 0.0001. We recommend that you adjust the learning rate to one tenth, one hundredth, and one thousandth of the initial learning rate in sequence.	FLOAT list	N/A
lr_warmup	No	Specifies whether to warm up the learning rate.	BOOL	false
lr_warm_up_epochs	No	The number of epochs for which you want to warm up the learning rate.	FLOAT	1
train_data	Yes	The OSS path of the data that is used to train the model.	oss://path/to/train_*.tfrecord	N/A
test_data	Yes	The OSS path of the data that is evaluated during the training.	oss://path/to/test_*.tfrecord	N/A
train_batch_size	Yes	The size of the data that is used to train the model in the current batch.	INT. Sample value: 32.	N/A
test_batch_size	Yes	The size of the data that is evaluated in the current batch.	INT. Sample value: 32.	N/A
train_num_readers	No	The number of concurrent threads that are used to read the training data.	INT	4
model_dir	Yes	The OSS path of the model.	`oss://path/to/model`	N/A
pretrained_model	No	The OSS path of the pretrained model. If this parameter is specified, the actual model is finetuned based on the pretrained model.	`oss://pai-vision-data-sh/pretrained_models/inception_v4.ckpt`	""
use_pretrained_model	No	Specifies whether to use a pretrained model.	BOOL	true
num_epochs	Yes	The number of training iterations. A value of 1 indicates that all data is iterated once for the training.	INT. Sample value: 40.	N/A

Parameter	Required	Description	Value type or value format	Default Value
num_test_example	No	The number of data entries that are evaluated during the training. A value of -1 indicates that all training data is evaluated.	INT. Sample value: 2000.	-1
num_visualizations	No	The number of data entries that can be visualized during the evaluation.	INT	10
save_checkpoint_epochs	No	The epoch interval at which a checkpoint is saved. A value of 1 indicates that a checkpoint is saved each time an epoch is complete.	INT	1
save_summary_epochs	No	The epoch interval at which a summary is saved. A value of 0.01 indicates that a summary is saved each time 1% of the training data is iterated.	FLOAT	0.01
num_train_images	No	The total number of data entries that are used for the training. If you use custom TFRecord files to train the model, this parameter is required.	INT	0
label_map_path	No	The category mapping file. If you use custom TFRecord files to train the model, this parameter is required.	STRING	""

4.1.5.2.13.3. Offline prediction

General image prediction

The general image prediction component is used to make predictions. It reads data from MaxCompute tables or Object Storage Service (OSS) files for prediction, and writes the prediction results to the MaxCompute tables or OSS files. This topic provides examples on how to run a Machine Learning Platform for AI (PAI) command to use the general image prediction component. This topic also describes how to define the input and output files and customize the process and prediction classes for different types of models.

PAI command

- Detect objects based on a region of interest

```
pai -name ev_predict
   -Dmodel_path='oss://pai-vision-data-hz/pretrained_models/saved_models/faster_rcnn_with_roi_features_coco/'
   -Dmodel_type='detector'
   -Dinput_oss_file='oss://pai-vision-data-hz/data/pai_test/predict_oss_io_test.list'
   -Doutput_oss_file='oss://pai-vision-data-hz/test/tmp/ev_predict_result.txt'
   -Dimage_type='url'
   -Dbuckets='oss://pai-vision-data-hz/?role_arn=xxx:role/pai-vision-hz&host=cn-hangzhou.oss.aliyun-inc.com'
```

- Customize the process and predictor classes

```
pai -name ev_predict
   -Dmodel_path='oss://pai-vision-data-hz/pretrained_models/pipeline_text_models/'
   -Dmodel_type='self_define'
   -Dinput_oss_file='oss://pai-vision-data-hz/data/pai_test/predict_oss_io_test.list'
   -Doutput_oss_file='oss://pai-vision-data-hz/test/tmp/ev_predict_result.txt'
   -Dimage_type='url'
   -Duser_resource='oss://pai-vision-data-hz/data/pai_test/ev_predict/test_data/self_define/user_res.tar.gz'
   -Duser_predictor_cls='user_predictor.MyPredictor'
   -Duser_process_config='
      [
        {
         \"job_name\":\"myprocess\",
         \"num_threads\":2,
         \"batch_size\":1,
         \"class\":\"user_process.MyProcess\"
       }
      ]'
   -Dbuckets='oss://pai-vision-data-hz/?role_arn=xxx:role/pai-vision-hz&host=oss-cn-zhangjiakou.aliyuncs.com'
```

The code file that is used to customize the process class is *user_process.py*. The class name is MyProcess. The code file that is used to customize the predictor class is *user_predictor.py*. The class name is MyPredictor. The *user_process.py* and *user_predictor.py* code files are compressed in the *user_res.tar.gz* package.

- Specify the input and output OSS files

```
pai -name ev_predict
   -Dmodel_path='oss://pai-vision-data-hz/pretrained_models/pipeline_text_models/'
   -Dmodel_type='self_define'
   -Duser_resource='oss://your_predictor_res.tar.gz'
   -Duser_predictor_cls='your_predictor.DummyModalPredictor'
   -Dinput_oss_file='oss://pai-vision-data-hz/data/pai_test/predict_oss_io_test.list'
   -Doutput_oss_file='oss://pai-vision-data-hz/test/tmp/ev_predict_multi_modal_result.json'
   -Doutput_dir='oss://pai-vision-data-hz/test/tmp/multi_modal_output'
   -Dimage_type='url'
   -Dbuckets='oss://pai-vision-data-hz/?role_arn=xxx/pai-vision-hz&host=oss-cn-zhangjiakou.aliyuncs.com'
```

The code file that is used to customize the predictor class is *user_predictor.py*. The class name is DummyModalPredictor.

- Customize the process class

  You can specify the input tables to read and process, and the output table to which the processed data is written. The following content provides an example on how to extract video features. In this example, two input tables are used. One of them contains the information about video frames. The other contains the IDs of the videos from which you want to extract features. You can use the following PAI command to read data from the input tables and write the prediction results to the output table.

- Input table 1

```
+-------------------------------------------------------------------------------------+
| Field           | Type       | Label | Comment                                      |
+-------------------------------------------------------------------------------------+
| video_id        | bigint     |       |                                              |
| frame_id        | bigint     |       |                                              |
| image           | string     |       |                                              |
| shot_id         | bigint     |       |                                              |
| extra_info      | string     |       |                                              |
| video_id_str    | string     |       | string type video id                         |
+-------------------------------------------------------------------------------------+
```

- Input table 2

```
+-------------------------------------------------------------------------------------+
| Field           | Type       | Label | Comment                                      |
+-------------------------------------------------------------------------------------+
| video_id        | bigint     |       |                                              |
+-------------------------------------------------------------------------------------+
```

- Output table

```
+-------------------------------------------------------------------------------------+
| Field           | Type       | Label | Comment                                      |
+-------------------------------------------------------------------------------------+
| video_id        | bigint     |       |                                              |
| frame_num       | bigint     |       |                                              |
| feature         | string     |       |                                              |

+-------------------------------------------------------------------------------------+
```

- PAI command

```
pai -name ev_predict
            -Dmodel_path='oss://pai-vision-data-hz/pretrained_models/saved_models/inception_v4_central_crop_1.0/'
            -Dmodel_type='feature_extractor'
            -Dfeature_name='PreLogitsFlatten'
            -Dinput_table='odps://sre_mpi_algo_dev/tables/zwm_dummy_table_data,odps://sre_mpi_algo_dev/tables/zwm_dummy_video_id'
            -Doutput_table='odps://sre_mpi_algo_dev/tables/zwm_dummy_result_video_feature'
            -Dimage_col='image'
            -Dimage_type='url'
            -Dreserved_columns='video_id'
            -Dresult_column='frame_num,feature'
            -Dbuckets='oss://pai-vision-data-hz/?role_arn=acs:ram::xxx:role/pai-vision-hz&host=oss-cn-zhangjiakou.aliyuncs.com'
            -Dauto_create_table=false
            -DuseSparseClusterSchema=true
            -DenableDynamicCluster=false
            -DautoEnablePsTaskFailover=false
            -DgpuRequired=50
            -Dnum_worker=1
            -Dqueue_size=12
            -Dbatch_size=1
            -Dpredict_thread_num=1
            -Dpreprocess_thread_num=3
            -Duser_resource http://pai-vision-data-hz.oss-cn-zhangjiakou.aliyuncs.com/data/pai_test/ev_predict/test_data/self_define/video_feature_extract_pad.py
            -Duser_build_process_fn 'video_feature_extract_pad.build_and_run_video_decode_process'
```

Parameters

Parameter	Required	Description	Value type	Default value
model_path	Yes	The OSS path of the model. Example: "oss://your_bucket/your_model_dir".	STRING	N/A

Parameter	Required	Description	Value type	Default value
model_type	Yes	The type of the model. Valid values: • feature_extractor: feature extraction • classifier: image classification • detector: object detection • text_detector: text detection • text_recognizer: text line recognition • text_spotter: end-to-end text recognition • segmentor: semantic image segmentation • self_define: custom prediction If the model_type parameter is set to self_define, the predictor class that is specified by the user_predictor_cls parameter is loaded.	STRING	N/A
buckets	No	The information about the OSS buckets. If you use a custom model, you must specify the OSS bucket in which your model is stored. Example: "oss://bucket_name/?role_arn=xxx&host=yyy".	STRING	""
feature_name	No	The name of the feature to extract. This parameter is required if the model_type parameter is set to feature_extractor. Example: resnet_v1_50/logits.	STRING	""
input_table	No	The name of the input table. For example, you can use a non-partitioned table "odps://prj_name/tables/table_name" or a partitioned table "odps://prj_name/tables/table_name/pt=xxx".	STRING	""
image_col	No	The name of the column that contains the image data.	STRING	"image"
image_type	No	The format of the image data. Valid values: • base64: indicates that the table stores the image data in the Base64 format. • url: indicates that the table stores the URLs or OSS paths of the images.	STRING	"base64"

Parameter	Required	Description	Value type	Default value
reserved_columns	No	The names of reserved data columns. Separate multiple names with commas (,). Example: "col1,col2,col3".	STRING	""
result_column	No	The name of the result column.	STRING	"prediction_result"
output_table	No	The output table. The value format is the same as that of the input_table parameter. If the specified output table does not exist, an output table is automatically created and partitioned. You can also create a partitioned table as the output table before you make the prediction.	STRING	""
lifecycle	No	The lifecycle of the output table.	INT	10
num_worker	No	The number of prediction workers. More workers can accelerate offline prediction.	INT	2
cpuRequired	No	The number of CPUs for a worker. A value of 100 indicates one CPU.	INT	1600
gpuRequired	No	The number of GPUs for each worker. A value of 100 indicates one GPU. You can use up to 100 GPUs for a worker. A value of 0 indicates that a CPU cluster is used.	INT	100
memory	No	The memory size of each worker, in MB.	INT	30000
input_oss_file	No	The path of the input OSS file. Each line in the input file can be in one of the following formats: • The OSS path or URL of an image to predict. Example: oss://your_bucket/filelist.txt. • A JSON string.	STRING	""
output_oss_file	No	The path of the output OSS file that is used to store the prediction results. The system may generate multiple result files and merge these result files into an output OSS file. The result files are prefixed with the name of the output OSS file. The number of result files is the same as the number of workers, which is specified by the num_worker parameter.	STRING	""

Parameter	Required	Description	Value type	Default value
output_dir	No	The directory of the output file. If you customize the output format, all the result images are stored in this directory.	STRING	""
user_resource	No	The path to which you want to upload your resources. You can upload tar.gz, .zip, and .py files. OSS paths and HTTP URLs are supported. Example: oss://xxx/a.tar.gz or http://a.com/c.zip.	STRING	""
user_predictor_cls	No	The module path of the custom predictor class. For example, if you implement Predictor A in *module.py*, the module path of Predictor A is module.A.	STRING	""
user_process_config	No	The configurations of the custom process class. You can use the following fields to configure the process class. You can also configure other custom fields. • job_name: the name of the custom process class. • num_threads: the number of concurrent threads that are used by the custom process class. • batch_size: the batch size of the data to be processed. • user_process_cls: the module path of the custom process class. For example, if you implement Process A in *module.py*, the module path of Process A is module.A. Example: '{["job_name":"myprocess","user_process_cls": module.ClassA"num_threads":2, "batch_size":1]}'	JSON string	""
queue_size	No	The length of the cache queue.	INT	1024
batch_size	No	The size of data that is used for prediction in the current batch.	INT	1
preprocess_thread_num	No	The number of concurrent threads that are used for preprocessing, such as image decoding and download.	INT	4
predict_thread_num	No	The number of concurrent threads that are used for prediction.	INT	2

Parameter	Required	Description	Value type	Default value
is_input_video	No	Specifies whether the input files are video files.	BOOL	false
use_image_predictor	No	Specifies whether the predictor supports only image input.	BOOL	true
decode_type	No	The method that is used to decode the videos. Valid values: • 1: intra only • 2: keyframe only • 3: without bidir • 4: decode all	INT	4
sample_fps	No	The number of frames that are extracted for sampling per second.	FLOAT	5
reshape_size	No	The size of the output frames, in pixels. If you set this parameter to -1, the frames are not resized.	INT	-1
decode_batch_size	No	The number of frames that are decoded at a time.	INT	10
decode_keep_size	No	The number of overlapped frames in different batches.	INT	0
float_digits	No	The number of FLOAT digits for JSON serialization. For example, if you set this parameter to 3, the generated string contains three decimal places. A value of -1 indicates that the generated string uses the original precision.	INT	-1
mediaflow_class_name	No	The path of the Mediaflow class. Example: demo.mediaflow_caller.	STRING	""
self_defined_result_formater	No	The class that is used to format the custom results. Example: a.table_result_format.	STRING	""
user_build_process_fn	No	The custom process function. Example: a.build_process_fn.	STRING	""
enableDynamicCluster	No	Specifies whether to enable the dynamic cluster feature. If this feature is enabled, the failover of a single worker is allowed. If task exceptions frequently occur, we recommend that you enable this feature.	BOOL	false

Parameter	Required	Description	Value type	Default value
useSparseClusterSchema	No	If the enableDynamicCluster parameter is set to true, you must also set the useSparseClusterSchema parameter to true.	BOOL	false
enable_elastic_inference	No	Specifies whether to enable the elastic inference feature.	BOOL	false
restore_works_dir	No	The cache directory that is used to store the data being processed by each worker. When a worker fails, the data can be used to recover the worker. You can set this parameter to an OSS path or an Apsara File Storage for HDFS directory. If the enable_elastic_inference parameter is set to true, this parameter is required. Example: egoss://path/to/dir/hdfs://path/to/dir/. **Note** The value must end with a forward slash (/).	STRING	""
slice_size	No	The size of a data shard that a worker processes at a time. A large value may lead to out of memory (OOM) errors, whereas a small value may lead to low reading efficiency. We recommend that you set this parameter to 1024 for image processing and 16 for video processing.	INT	N/A

Input data formats

- Input table

  An input table can contain one or more columns. One of the columns must contain the image URLs or Base64-encoded binary image data. The data type of the column must be STRING. The following example shows a table schema:

```
+------------------------------------------------------------------------------------+
| Field           | Type       | Label | Comment                                     |
+------------------------------------------------------------------------------------+
| id              | STRING     |       |                                             |
| url             | STRING     |       |                                             |
+------------------------------------------------------------------------------------+
```

- Input OSS file

  Each line of the input OSS file is a URL or an OSS path, for example:

```
oss://your/path/to/image.jpg
http://your.path/to/image.jpg
```

- Custom input format

  By default, only image URLs or Base64-encoded image data entries are read from the input tables. Only image URLs are read from the input OSS files to download and decode images. Both of the two methods generate only the numPy arrays of images in the {"image": np.ndarry} format. The numPy arrays are used by the process and predictor classes to make predictions. More and more users customize the predictor and process classes. A single input data format cannot satisfy user needs. To support custom formats, the method for reading data from OSS files is improved.

  The custom format can be the original OSS file format or JSON string format. Each line in the input files is a JSON string. You can enter multiple key-value pairs. All key-value pairs are saved in a dictionary and passed to the custom predictor and process classes. You can obtain the corresponding values based on custom keys.

  If the values are OSS paths or URLs, the system automatically uses multiple threads to download the file content, and saves the values in a Python file-like object. You can call methods such as `read()` or `readlines()` to obtain the content of the Python file-like object. If a value points to a file with an image extension, the system automatically decodes the image. You can obtain the value from the input_data dictionary based on the corresponding key. The value is of the numpy.ndarray type.

  The following code provides an example of the input data:

  ```
  {"image":"http://your/path/to/image.jpg", "prior":"oss://your/path/to/prior.txt", "config": {"key1":1, "key2":"value2"}}
  {"image":"http://your/path/to/image.jpg", "prior":"oss://your/path/to/prior.txt", "config": {"key2":1, "key2":"value2"}}
  ```

  The preceding input data is converted into data in the input_data dictionary with the following fields:

  - image: the decoded data of an image.
  - prior: the file-like object.
  - config: a dictionary of JSON strings.

  The input_data dictionary is in the following format. For all custom process and predictor classes, you can query image data based on the key.

  ```
  input_dict = {
    "image": np.ndarray,
    "prior" : file_like_object,
    "config": null
  }
  ```

  Note All built-in predictor classes of EasyVision use the image key to obtain input images. If you want to use a custom input format to call the built-in predictor classes of EasyVision, you must set the image key for image data.

Custom third-party library

The offline prediction framework is implemented based on PAI-TensorFlow. Therefore, the framework does not allow you to use PyTorch models to make custom predictions. In addition, if you make a custom prediction by using a third-party library that does not exist in the preset Python environment, the prediction fails. To resolve this issue, PAI allows you to use the *requirements.txt* file to install a custom Python library.

You must add the *requirements.txt* file to the top-level code directory:

```
user_resource
├── pytorch_sr_script.py
├── requirements.txt
├── user_predictor.py
├── user_process.py
```

Then, you must package the entire code directory and upload it to OSS. The following code provides an example:

```
tar -zcf user_resource.tar.gz user_resource
osscmd put user_resource.tar.gz oss://your/path/user_resource.tar.gz
```

You can specify the address of the code package in the PAI command parameter `-Duser_resource`. When the prediction task runs, PAI automatically installs the dependencies that are specified in the *requirements.txt* file.

Custom predictor class

The custom predictor class must inherit `easy_vision.python.inference.predictor.PredictorInterface` or `easy_vision.python.inference.predictor.PredictorInterfaceV2`. The names and default values of all functions and parameters of the custom predictor class must be the same as those in the definition of the UserProcess operation.

The general image prediction component is compatible with V1 and V2 operation definitions of the custom predictor class.

- V1 (only serializes all prediction results to JSON strings)

  The following code shows the operation definition:

```
class PredictorInterface(six.with_metaclass(abc.ABCMeta, object)):
  def __init__(self, model_path, model_config=None):
    """
    init tensorflow session and load tf model
    Args:
      model_path  init model from this directory
      model_config config string for model to init, in json format
    """
    pass
  @abc.abstractmethod
  def predict(self, input_data, batch_size):
    """
    using session run predict a number of samples using batch_size
    Args:
      input_data:  a list of numpy array, each array is a sample to be predicted
      batch_size: batch_size passed by the caller, you can also ignore this param and
        use a fixed number if you do not want to adjust batch_size in runtime
    Return:
      result: a list of dict, each dict is the prediction result of one sample
        eg, {"output1": value1, "output2": value2}, the value type can be
        python int str float, and numpy array
    """
    pass
```

  The following code provides an example of the custom predictor class:

```
#!/usr/bin/env python
# -*- encoding:utf-8 -*-
# Filename: user_predictor.py
# Author: Qianyan (wenmeng.zwm@alibaba-inc.com)
# Date: 2019-06-10
# Description:
from easy_vision.python.inference import predictor
import tensorflow as tf
class MyPredictor(predictor.PredictorInterface):
  def __init__(self, model_path, model_config=None):
    """
    init tensorflow session and load tf model
    Args:
      model_path  init model from this directory
      model_config config string for model to init, in json format
    """
    tf.logging.info('my predictor init done')
  def predict(self, input_data, batch_size):
    """
    using session run predit a number of samples using batch_size
    Args:
      input_data:  a list of numpy array, each array is a sample to be predicted
      batch_size: batch_size passed by the caller, you can also ignore this param and
        use a fixed number if you do not want to adjust batch_size in runtime
    Return:
      result: a list of dict, each dict is the prediction result of one sample
        eg, {"output1": value1, "output2": value2}, the value type can be
        python int str float, and numpy array
    """
    results = [{'result1':'my_predictor_result1', 'result2':'my_predictor_result2'}
               for i in range(len(input_data))]
    return results
```

- V2

  The custom predictor class has the following limits:

  - You can use the custom predictor class to pass only the list of images. No other information can be passed.
  - The output dictionary is converted into only JSON strings and cannot save image files.

  To remove the limits, PAI provides V2 operation definition to enable you to specify the output format. V2 is developed based on V1 by adding the get_output_type function. This function returns a dictionary that indicates how to save the values in the key-value pairs in the dictionary returned by the prediction. The values can be saved in the following ways:

  - json: All the output JSON data of keys is aggregated into a dictionary and serialized to a JSON string. Then, the JSON string is written to the output OSS file.
  - image: The output data of the keys that correspond to values of the image type is saved in an OSS file *output_dir/key/filename.jpg*. The value of the filename variable is obtained from the input URL. If the output data of a key is a list of images, each image in the list is saved as *output_dir/key/filename_idx.jpg*, where, idx is the index number of each image.
  - video: not supported.

  To pass more information, you can also pass a list of dictionaries to the predictor class by using custom input data types.

  The following code shows the operation definition:

```
class PredictorInterfaceV2(six.with_metaclass(abc.ABCMeta, object)):
  def __init__(self, model_path, model_config=None):
    """
    init tensorflow session and load tf model
    Args:
      model_path  init model from this directory
      model_config config string for model to init, in json format
    """
    pass
  def get_output_type(self):
    """
    in this function user should return a type dict, which indicates
    which type of data should the output of predictor be converted to
    * type json, data will be serialized to json str
    * type image, data will be converted to encode image binary and write to oss file,
      whose name is output_dir/${key}/${input_filename}_${idx}.jpg, where input_filename
      is extracted from url, key corresponds to the key in the dict of output_type,
      if the type of data indexed by key is a list, idx is the index of element in list, otherwhil
e ${idx} will be empty
    * type video, data will be converted to encode video binary and write to oss file,
    eg:  return  {
      'ret_image': 'image',
      'feature': 'json'
    }
    indicating that the image data in the output dict will be save to image
    file and feature in output dict will be converted to json
    """
    return {}
  @abc.abstractmethod
    def predict(self, input_data_dict, batch_size):
    """
    using session run predict a number of samples using batch_size
    Args:
      input_data_dict:  a list of dict, each dict is a sample data to be predicted
      batch_size: batch_size passed by the caller, you can also ignore this param and
        use a fixed number if you do not want to adjust batch_size in runtime
    Return:
      result: a list of dict, each dict is the prediction result of one sample
        eg, {"output1": value1, "output2": value2}, the value type can be
        python int str float, and numpy array
    """
    pass
```

The following code provides an example of the custom predictor class:

```
  class DummyMultiModalPredictor(ev_predictor.PredictorInterfaceV2):
    def __init__(self, model_path, model_config=None):
      pass
    def get_output_type(self):
      return {'result1': 'json',
              'result2':'json',
              'result_image': 'image',
              'result_image_list':'image',
              }
    def predict(self, images, batch_size=1):
      results = [{
                  'result1':'dummy',
                  'result2':'dummy',
                  'result_image': np.zeros([224,224,3],dtype=np.uint8),
                  'result_image_list': [255* np.ones([224,224,3],dtype=np.uint8) for i in range(2
))]
                } for i in range(len(images))]
      return results
```

Custom process class

The custom process class must inherit `easy_vision.python.pai_utils.ev_process.UserProcess`. The names and default values of all functions and parameters of the custom process class must be the same as those in the definition of the UserProcess operation.

The following code shows the operation definition:

```
class UserProcess(Process):
  def __init__(self,
               job_name,
               num_threads=1,
               input_queue=None,
               output_queue=None,
               process_config=None,
               batch_size=1):
    """
    Args:
      job_name:  job name for this process
      num_threads: number of threads to run your process
      input_queue:  a queue containing input data, you should not read input_queue by
        yourself
      output_queue:  you can push data to output_queue or just return data in process
        which will be pushed into output queue automatically
      process_config:  config for your process, dict type
      batch_size:  if batch_size is 1, input_data passed to in_data is a dict of one sample data
        if batch_size greater than 1, input_data passed to process is a list of dict, each dict
                contains one sample data
    """
    super(UserProcess, self).__init__(
        job_name,
        num_threads,
        input_queue=input_queue,
        output_queue=output_queue,
        batch_size=batch_size)
  def process(self, in_data):
    """
    method need to be reimplemented, one can add result to output_queue in this func
    or just return the result which will be pushed to output queue automatically.
    you should add the process result to input_data dict, and push it to the output
    queue or just return it. You can refer to easy_vision.python.pai_utils.ev_process.DataFields
    attribute to acquire what information you can get from input_dict, you can only
    write data with key like DataFields.image or DataFields.clip
    Args:
      in_data   if self.batch_size is 1, input_data is a dict of one sample data
                if self.batch_size greater than 1, input_data is a list of dict, each dict
                  contains one sample data
    Return
      if None, result should be added to output_queue by yourself in this func
      if not None, the returned result will be add to output_queue automatically
    """
    return input_data
  def destroy(self):
    """
    destroy resources that has been used for this process
    """
    pass
```

The following code provides an example of the custom process class:

```
#!/usr/bin/env python
# -*- encoding:utf-8 -*-
# Filename: user_process.py
# Author: Qianyan (wenmeng.zwm@alibaba-inc.com)
# Date: 2019-06-10
# Description:
from easy_vision.python.pai_utils import ev_process
import logging
class MyProcess(ev_process.UserProcess):
  def __init__(self,
               job_name,
               num_threads=1,
               input_queue=None,
               output_queue=None,
               process_config=None,
               batch_size=1):
    """
    Args:
      job_name:  job name for this process
      num_threads: number of threads to run your process
      input_queue:  a queue containing input data, you should not read input_queue by
        yourself
      output_queue:  you can push data to output_queue or just return data in process
        which will be pushed into output queue automatically
      process_config:  config for your process, dict type
      batch_size:  if batch_size is 1, input_data passed to in_data is a dict of one sample data
        if batch_size greater than 1, input_data passed to process is a list of dict, each dict
                  contains one sample data
    """
    super(MyProcess, self).__init__(job_name, num_threads,
        input_queue = input_queue,
        output_queue = output_queue,
        batch_size = batch_size)
  def process(self, in_data):
    """
    method need to be reimplemented, one can add result to output_queue in this func
    or just return the result which will be pushed to output queue automatically.
    you should add the process result to input_data dict, and push it to the output
    queue or just return it. You can refer to easy_vision.python.pai_utils.ev_process.DataFields
    attribute to acquire what information you can get from input_dict, you can only
    write data with key like DataFields.image or DataFields.clip
    Args:
      in_data   if self.batch_size is 1, input_data is a dict of one sample data
                if self.batch_size greater than 1, input_data is a list of dict, each dict
                  contains one sample data
    Return
      if None, result should be added to output_queue by yourself in this func
      if not None, the returned result will be add to output_queue automatically
    """
    logging.info('my process process data')
    return in_data
  def destroy(self):
    """
    destroy resources that has been used for this process
    """
    logging.info('my process destroyed')
```

Result format

This section describes the result format of various models.

- feature_extractor

  Sample result:

  ```
  {"feature": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.4583122730255127, 0.0]}
  ```

  The following table describes the fields in the sample result.

Field	Description	Shape	Type
feature	The extracted feature.	[feature_dim]	FLOAT

- classifier

  Sample result:

  ```
  {
  "class": 3,
  "class_name": "coho4",
  "class_probs": {"coho1": 4.028851974258174e-10,
            "coho2": 0.48115724325180054,
            "coho3": 5.116515922054532e-07,
            "coho4": 0.5188422446937221}
  }
  ```

  The following table describes the fields in the sample result.

Field	Description	Shape	Type
class	The ID of the category.	[]	INT 32
class_name	The name of the category.	[]	STRING
class_probs	The matching probabilities of all categories.	[num_classes]	DICT{key: STRING, value: FLOAT}

- detector

  Sample result:

  ```
  {
    "detection_boxes": [[243.5308074951172, 197.69570922851562, 385.59625244140625, 247.724777221679
  7], [292.1929931640625, 114.28043365478516, 571.2748413085938, 165.09771728515625]],
    "detection_scores": [0.9942291975021362, 0.9940272569656372],
    "detection_classes": [1, 1],
    "detection_classe_names": ["text", "text"]
   }
  ```

  The following table describes the fields in the sample result.

Field	Description	Shape	Type

Field	Description	Shape	Type
detection_boxes	The bounding boxes that mark the recognized objects. The coordinates of each bounding box are specified in the following order: [top, left, bottom, right].	[num_detections, 4]	FLOAT
detection_scores	The probabilities that the commodities are recognized.	num_detections	FLOAT
detection_classes	The IDs of the categories to which the objects belong.	num_detections	INT
detection_class_names	The names of the categories to which the commodities belong.	num_detections	ISTRING
detection_masks	The segmentation masks of the object. Optional.	[num_detection, image_height, image_width]	FLOAT
detection_keypoints	The landmarks of the object. Optional.	[num_detection, num_keypoints, 2]	FLOAT
detection_roi_features	The local features of the object. Optional.	[num_detection, roi_height, roi_width, channels]	FLOAT

- segmentor

  Sample result:

```
{
  "probs" : [[[0.8, 0.8], [0.6, 0.7]],[[0.8, 0.5], [0.4, 0.3]]],
  "preds" : [[[1,1], [0, 0]], [[0, 0], [1,1]]]
}
```

  The following table describes the fields in the sample result.

Field	Description	Shape	Type
probs	The probability that each pixel obtained through segmentation belongs to a specific category.	[output_height, output_width, num_classes]	FLOAT
preds	The IDs of the categories to which the pixels obtained after segmentation belong.	[output_height, output_widths]	INT

- text_detector

  Sample result:

```
{
  "detection_keypoints": [[[243.57516479492188, 198.84210205078125], [243.91038513183594, 247.62425231933594], [385.5513916015625, 246.61660766601562], [385.2197570800781, 197.79345703125]], [[292.2718200683594, 114.44700622558594], [292.2237243652344, 164.684814453125], [571.1962890625, 164.931640625], [571.2444458007812, 114.67433166503906]]],
  "detection_boxes": [[243.5308074951172, 197.69570922851562, 385.59625244140625, 247.7247772216797], [292.1929931640625, 114.28043365478516, 571.2748413085938, 165.09771728515625]],
  "detection_scores": [0.9942291975021362, 0.9940272569656372],
  "detection_classes": [1, 1],
  "detection_classe_names": ["text", "text"],
   "image_shape": [1024, 968, 3]
 }
```


The following table describes the fields in the sample result.

Field	Description	Shape	Type
detection_boxes	The detected text area with coordinates in the following order: [top, left, bottom, right].	[num_detections, 4]	FLOAT
detection_scores	The probabilities that the text areas are detected.	num_detections	FLOAT
detection_classes	The IDs of the categories to which the text areas belong.	num_detections	INT
detection_class_names	The names of the categories to which the text areas belong.	num_detections	STRING
detection_keypoints	The (y, x) coordinates of the four vertices of the detected text area.	[num_detections, 4, 2]	FLOAT
image_shape	The information about the input image. The value is a list that contains the following information in order: height, width, and channel.	[3]	LIST

- text_recognizer

  Sample result:


```
{
  "sequence_predict_ids": [1,2,2008,12],
  "sequence_predict_texts": "This is an example",
  "sequence_probability": 0.88
}
```


The following table describes the fields in the sample result.

Field	Description	Shape	Type

Field	Description	Shape	Type
sequence_predict_ids	The ID of the category to which a single line of the recognized text belongs.	[text_length]	INT
sequence_predict_texts	The recognition result of each single-line text.	[]	STRING
sequence_probability	The probability that each single-line text is recognized.	[]	FLOAT

- text_spotter

  Sample result:

```
{
  "detection_keypoints": [[[243.57516479492188, 198.84210205078125], [243.91038513183594, 247.6242
5231933594], [385.5513916015625, 246.61660766601562], [385.2197570800781, 197.79345703125]], [[292
.2718200683594, 114.44700622558594], [292.2237243652344, 164.684814453125], [571.1962890625, 164.9
31640625], [571.2444458007812, 114.674331665039061]]],
  "detection_boxes": [[243.5308074951172, 197.69570922851562, 385.59625244140625, 247.724777221679
7], [292.1929931640625, 114.28043365478516, 571.2748413085938, 165.09771728515625]],
  "detection_scores": [0.9942291975021362, 0.9940272569656372],
  "detection_classes": [1, 1],
  "detection_classe_names": ["text", "text"],
  "detection_texts_ids" : [[1,2,2008,12], [1,2,2008,12]],
  "detection_texts": ["This is an example", "This is an example"],
  "detection_texts_scores" : [0.88, 0.88],
  "image_shape": [1024, 968, 3]
 }
```

  The following table describes the fields in the sample result.

Field	Description	Shape	Type
detection_boxes	The detected text area with coordinates in the following order: [top, left, bottom, right].	[num_detections, 4]	FLOAT
detection_scores	The probabilities that the text areas are detected.	num_detections	FLOAT
detection_classes	The IDs of the categories to which the text areas belong.	num_detections	INT
detection_class_names	The names of the categories to which the text areas belong.	num_detections	STRING
detection_keypoints	The (y, x) coordinates of the four vertices of the detected text area.	[num_detections, 4, 2]	FLOAT

Field	Description	Shape	Type
detection_texts_ids	The ID of the category to which a single line of the recognized text belongs.	[num_detections, max_text_length]	INT
detection_texts	The recognition result of each single-line text.	[num_detections]	STRING
detection_texts_scores	The probability that each single-line text is recognized.	[num_detections]	FLOAT
image_shape	The information about the input image. The value is a list that contains the following information in order: height, width, and channel.	[3]	LIST

- Custom result format

  By default, `ev_predict` extracts the dictionary from the cache queue, obtains the value of `prediction_result`, serializes the value to a JSON string, and then writes the JSON string to the result column. To write results to different columns by field, you can set the `self_defined_result_formater` parameter to the path of the custom result formatting class and the user_resource parameter to the path of your code.

  The process class is a derived class. For more information about the operation definition of this class, see the "Custom process class" section of this topic. You need only to save the formatted results in the input_data dictionary with the key as result_to_save. Then, TableWriter automatically writes the prediction results to the specified table.

  The following code provides an example on how to write the decoded data of a video to a table by frame. Each line is the encoded data of a frame.

```
class SelfDefineTableFormatProcess(ev_process.Process):
  def __init__(self,
               input_queue,
               output_queue,
               reserved_col_names=[],
               output_col_names=[]):
    """
    extract data from dict to preprare result dict which will be written to table or oss file
    Args:
      reserved_col_names list of str column names for reserved col
      output_col_names  list of column name for output table
      input_queue  the python queue for input
      output  the python queue for output
      float_digits: The number of digits of precision when writing floats out
    Return:
      push a k-v pair {'result_to_save': result_dict} into data_dict and return
      a result dict or a tuple, or a list of tuple or a list of dict
        if dict is returned, each key should be the same as column name of output table
        if tuple is returned, this tuple corespond to one table record
        if list of tuple is returned, they correspond to a list of table record
        if list of dict is returned, each element in list will be automatically converted to table
record in table writer
    """
    super(SelfDefineTableFormatProcess, self).__init__('SelfDefineTableFormatProcess', 1,
                                                       input_queue=input_queue,
                                                      output_queue=output_queue)
  def process(self, input_data):
    output_record = []
    frames = input_data['frames']
    frames = [base64.urlsafe_b64encode(np.array(frame)) for frame in frames]
    frame_ids = input_data['frame_id']
    url = input_data['url']
    video_records = []
    for url, fid, frame in zip(url, frame_ids, frames):
      video_records.append((url, fid, frame))
    # result should be returned with key result_to_save
    input_data['result_to_save'] = video_records
    return input_data
```

Add the preceding code to *oss://path/to/you/code.py*. Then, use the following PAI command parameters to customize the logic for writing data to the table.

```
pai -name ev_predict
   ...
   -Duser_resource=oss://path/to/you/code.py
   -Dself_defined_result_formater=code.SelfDefineTableFormatProcess
```

General video prediction

This topic provides examples on how to run a Machine Learning Platform for AI (PAI) command to use the video prediction component. This topic also describes the command parameters and provides the operation definitions of and examples on custom process and predictor classes.

You can run the `ev_video_predict` command to make video predictions. When you run this command, you need only to set the common parameters without the need to know the rules or logic of the EasyVision profile.

PAI command

- Input OSS file

```
pai -name ev_video_predict
             -Dinput_oss_file='oss://pai-vision-data-hz/data/pai_test/video_decode_test.list'
             -Doutput_oss_file='oss://pai-vision-data-hz/test/tmp/ev_video_predict_result.txt'
             -Dmodel_path='oss://pai-vision-data-hz/data/pai_test/ev_predict/test_data/models/ucf1
01_resnet3d_50'
             -Dtask_type='video_classifier'
             -Dfeature_name=''
             -Ddecode_type=4
             -Dsample_fps=-1
             -Dreshape_size=-1
             -Dsample_duration=16
             -Dbatch_size=16
             -Dpredict_thread_num=1
             -Dqueue_size=256
             -Dnum_worker=4
             -DcpuRequired=400
             -Dbuckets='YOUR_OSS_BUCKET_CONFIG'
```

- Input Swift file

```
pai -name ev_video_predict
             -Dswift_end_point='SWIFT_END_POINT'
             -Dinput_topic='INPUT_TOPIC'
             -Doutput_topic='OUTPUT_TOPIC'
             -Dmodel_path='oss://pai-vision-data-hz/data/pai_test/ev_predict/test_data/models/ucf1
01_resnet3d_50'
             -Dtask_type='video_classifier'
             -Dfeature_name=''
             -Ddecode_type=4
             -Dsample_fps=-1
             -Dreshape_size=-1
             -Dsample_duration=16
             -Dbatch_size=16
             -Dpredict_thread_num=1
             -Dqueue_size=256
             -Dnum_worker=4
             -DcpuRequired=400
             -Dbuckets='YOUR_OSS_BUCKET_CONFIG'
```

- Custom process and predictor classes

```
pai -name ev_video_predict
             -Dswift_end_point='SWIFT_END_POINT'
             -Dinput_topic='INPUT_TOPIC'
             -Doutput_topic='OUTPUT_TOPIC'
             -Dmodel_path='oss://pai-vision-data-hz/data/pai_test/ev_predict/test_data/models/ucf101_resnet3d_50'
             -Dtask_type='self_define'
             -Dfeature_name=''
             -Ddecode_type=4
             -Dsample_fps=-1
             -Dreshape_size=-1
             -Dsample_duration=16
             -Dbatch_size=1
             -Dpredict_thread_num=1
             -Dqueue_size=256
             -Dnum_worker=4
             -DcpuRequired=400
             -Duser_resource='oss://pai-vision-data-hz/data/pai_test/ev_predict/test_data/self_define/video_res.tar.gz'
             -Duser_predictor_cls='user_predictor.MyPredictor'
             -Duser_process_config='
                [
                  {
                   \"job_name\":\"myprocess\",
                   \"num_threads\":4,
                   \"batch_size\":1,
                   \"class\":\"user_process.MyProcess\"
                  }
                ]'
             -Dbuckets='oss://pai-vision-data-hz/?role_arn=acs:ram::1217060697l8XXXX:role/pai-vision-hz&host=oss-cn-zhangjiakou.aliyuncs.com'
```

The code file that is used to customize the predictor class is *user_predictor.py*. The class name is `MyPredictor`. The code file that is used to customize the process class is *user_process.py*. The class name is `MyProcess`. The *user_predictor.py* and *user_process.py* code files are compressed in the `video_res.tar.gz` package.

Parameters

Parameter	Required	Description	Value type or sample value	Default value
buckets	Yes	The information about the Object Storage Service (OSS) buckets.	`"oss://bucket_name/?role_arn=xxx&host=yyy"`	N/A

Parameter	Required	Description	Value type or sample value	Default value
model_path	Yes	The OSS path of the model.	`"oss://your_bucket/your_model_dir"` The OSS path of the model that EasyVision provides is `oss://pai-vision-data-hz/pretrained_models/saved_models/ucf101_resnet3d_50/`.	N/A
task_type	Yes	The type of the model. Only the video classification model is supported. Set this parameter to video_classifier.	STRING	N/A
input_oss_file	Yes	The path of the input OSS file. Each line in the input file can be in one of the following formats: • The path of a video. • A JSON string. Example: {"video_binary_data": VIDEO_URL, "xxx":xxx}	`oss://your_bucket/filelist.txt`	N/A
output_oss_file	Yes	The path of the output OSS file that is used to store the prediction results. The system may generate multiple result files and merge these result files into an output OSS file. The result files are prefixed with the name of the output OSS file. The number of result files is the same as the number of workers, which is specified by the num_worker parameter.	`oss://your_bucket/result.txt`	N/A
output_dir	No	The directory of the output file. If you customize the output format, all the result files are stored in this directory.	`oss://your_bucket/dir`	`""`
swift_end_point	No	The URL of the online ZooKeeper server.	STRING	`""`

Parameter	Required	Description	Value type or sample value	Default value
input_topic	No	The input topic. Each element in the topic is the OSS path of a video or a JSON string, for example, {"url":xxx, "user_define": xxx}.	STRING	""
output_topic	No	The output topic.	STRING	""
feature_name	No	The name of the output feature. Example: "resnet_3d_50/poo2","resnet_3d_50/scale5/conv5","resnet_3d_50/scale4/conv4","resnet_3d_50/scale3/conv3","resnet_3d_50/scale2/conv2","resnet_3d_50/pool1","resnet_3d_50/scale1/conv1".	STRING	""
decode_type	No	The method that is used to decode the videos. Valid values: • 1: intra only • 2: keyframe only • 3: without bidir • 4: decode all	INT	4
sample_fps	No	The number of frames that are extracted for sampling per second.	FLOAT	-1, which indicates that 25 frames are extracted per second.
reshape_size	No	The size of the output frames, in pixels.	INT	If you set this parameter to -1, the frames are not resized.
sample_duration	No	The length of consecutive frames that are imported to the model.	INT	16
batch_size	No	The number of clips that are imported to the model.	INT	16
predict_thread_num	No	The number of threads that are used for prediction.	INT	1
queue_size	No	The length of the processing queue.	INT	256
num_worker	No	The number of prediction workers.	INT	2

Parameter	Required	Description	Value type or sample value	Default value
cpuRequired	No	The number of CPUs for a worker. A value of 100 indicates one CPU.	INT	400
gpuRequired	No	The number of CPUs for a worker. A value of 100 indicates one GPU.	INT	100
user_resource	No	The path to which you want to upload your resources. You can upload tar.gz, .zip, and .py files. OSS paths and HTTP URLs are supported.	`oss://xxx/a.tar.gzhttp://a.com/c.zip`	`""`
user_predictor_cls	No	The module path of the custom predictor class. For example, if you implement Predictor A in module.py, the module path of Predictor A is module.A.	`module.PredictorA`	`""`
user_process_config	No	The configurations of the custom process class, in the JSON string format. You can use the following fields to configure the process class. You can also configure other custom fields. • job_name: the name of the custom process class. • num_threads: the number of concurrent threads that are used by the custom process class. • batch_size: the batch size of the data to be processed. • user_process_cls: the module path of the custom process class. For example, if you implement Process A in module.py, the module path of Predictor A is module.A.	`'{["job_name":"myprocess","user_process_cls":module.ClassA"num_threads":2,"batch_size":1]}'`	`""`

Result format

The result of each video is a JSON string, which includes the URL or OSS path and the prediction result of the video. The following code provides an example:

```
oss://pai-vision-data-hz/data/ucf101_train/ApplyEyeMakeup/v_ApplyEyeMakeup_g08_c01,{"class_name": "ApplyEyeMakeup", "class": 0, "probs": {"MilitaryParade": 6.785883670090698e-06, "TrampolineJumping": 1.9762439933401765e-06, "PlayingDaf": 2.7828968086396344e-05, "SalsaSpin": 7.392855422949651e-06, "CuttingInKitchen": 2.6925881684292108e-05, "ApplyEyeMakeup": 0.9972793459892273, "PlayingViolin": 1.84710111571818963e-05, "YoYo": 6.056292932044016e-06, "PlayingDhol": 4.430206445249496e-06, "PlayingCello": 2.7006941309082322e-05, "Bowling": 7.969072612468153e-06, "UnevenBars": 3.238687213524827e-06, "BalanceBeam": 5.145544491824694e-06, "SkyDiving": 3.6508849916572217e-06, "SumoWrestling": 9.847853107203264e-06, "PushUps": 9.45046849665232e-06, "FloorGymnastics": 3.89273827750003e-06, "ApplyLipstick": 0.0008350351126864552, "BreastStroke": 2.5186193397530587e-06, "GolfSwing": 2.3900222458905773e-06, "HorseRiding": 3.5062098504567984e-06, "PlayingFlute": 1.801957296265755e-05, "PizzaTossing": 7.031375389487948e-06, "CleanAndJerk": 4.537632776191458e-06, "WritingOnBoard": 2.856705032172613e-05, "CricketShot": 4.059567345393589e-06, "FieldHockeyPenalty": 4.859361069975421e-06, "HammerThrow": 4.741576958622318e-06, "BodyWeightSquats": 5.07304139318876e-06, "CliffDiving": 7.602381629112642e-06, "Typing": 6.037013008608483e-05, "MoppingFloor": 7.247148005262716e-06, "TaiChi": 7.273819846886909e-06, "PlayingPiano": 9.108782251132652e-06, "Punch": 1.3467762983054854e-05, "Nunchucks": 7.87033604865428e-06, "RopeClimbing": 4.0648433241585735e-06, "Swing": 2.3137952211982338e-06, "Knitting": 2.0819035853492096e-05, "Rafting": 2.7667874746839516e-06, "PlayingGuitar": 2.7430540285422467e-05, "VolleyballSpiking": 4.066964265803108e-06, "ShavingBeard": 0.0002367567940382287, "JugglingBalls": 1.6967631381703354e-05, "Diving": 5.02031662108493e-06, "JumpingJack": 1.1237583748879842e-05, "PoleVault": 3.3857772905321326e-06, "SkateBoarding": 7.812836884113494e-06, "BoxingPunchingBag": 6.725965249643195e-06, "IceDancing": 1.026979043672327e-05, "WallPushups": 2.3817137844162062e-05, "FrisbeeCatch": 7.734954124316573e-06, "Drumming": 4.163131961831823e-06, "JumpRope": 9.629309715819545e-06, "HeadMassage": 6.168564868858084e-05, "PlayingTabla": 2.9489216103684157e-05, "TableTennisShot": 6.383983418345451e-06, "PommelHorse": 5.876625891687581e-06, "HighJump": 3.0327462354762247e-06, "BasketballDunk": 5.7146194194501736e-06, "BoxingSpeedBag": 2.9891823942307383e-05, "PullUps": 7.257571269292384e-06, "SoccerPenalty": 5.835005140397698e-06, "RockClimbingIndoor": 9.057837814907543e-06, "BlowingCandles": 5.086950750410324e-06, "Skiing": 6.789191957068397e-06, "WalkingWithDog": 3.242800858060945 8e-06, "Basketball": 4.267759322829079e-06, "SoccerJuggling": 4.374186573841143e-06, "Fencing": 6.132470389275113e-06, "Billiards": 5.670899554388598e-06, "BaseballPitch": 4.975987394573167e-06, "BlowDryHair": 0.0001697591069387272, "CricketBowling": 2.135262002411764e-05, "BandMarching": 4.6766035666223615e-06, "PlayingSitar": 3.0760954814468278e-06, "ThrowDiscus": 1.1850976079585962e-05, "StillRings": 5.105009677208727e-06, "Lunges": 1.5529138863712433e-06, "Skijet": 3.188117261743173e-06, "BabyCrawling": 5.77771561438567e-06, "Mixing": 4.743058525491506e-05, "Hammering": 7.2947509579535 1e-06, "Shotput": 3.6980163713451475e-06, "Archery": 4.3028112486354075e-06, "Surfing": 5.7839174587 5706e-06, "FrontCrawl": 8.995367352326866e-06, "HulaHoop": 9.055997907125857e-06, "JavelinThrow": 8.379415703529958e-06, "Rowing": 6.835780368419364e-06, "Kayaking": 3.7107411117176525e-06, "ParallelBars": 6.430962002923479e-06, "HorseRace": 9.8691098173731e-06, "HandstandWalking": 4.089476078661391 5e-06, "BrushingTeeth": 0.0001521828380646184, "LongJump": 1.0337393177906051e-05, "Biking": 5.36460 6295188423e-06, "HandstandPushups": 6.265170213737292e-06, "BenchPress": 1.0012458915298339e-05, "Haircut": 0.0003677888307720423, "TennisSwing": 3.352387921040645e-06}}
```


If you want to obtain video features, the result contains the feature field with the corresponding feature value. The following code provides an example:


```
{"feature": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.4583122730255127, 0.0]}
```


Parameter	Description	Shape	Value type
class	The ID of the category.	[]	INT32
class_name	The name of the category.	[]	STRING
probs	The matching probabilities of all categories.	[num_classes]	DICT{Key: STRING, Value: FLOAT}
feature	The video feature.	[]	FLOAT

Custom process class

The custom process class must inherit `easy_vision.python.pai_utils.ev_process.UserProcess`. The names and default values of all functions and parameters of the custom process class must be the same as those in the definition of the UserProcess operation.

The following code shows the operation definition:

```
class UserProcess(Process):
  def __init__(self,
               job_name,
               num_threads=1,
               input_queue=None,
               output_queue=None,
               process_config=None,
               batch_size=1):
    """
    Args:
      job_name:  job name for this process
      num_threads: number of threads to run your process
      input_queue:  a queue containing input data, you should not read input_queue by
        yourself
      output_queue:  you can push data to output_queue or just return data in process
        which will be pushed into output queue automatically
      process_config:  config for your process, dict type
      batch_size:  if batch_size is 1, input_data passed to in_data is a dict of one sample data
        if batch_size greater than 1, input_data passed to process is a list of dict, each dict
                contains one sample data
    """
    super(UserProcess, self).__init__(
        job_name,
        num_threads,
        input_queue=input_queue,
        output_queue=output_queue,
        batch_size=batch_size)
  def process(self, in_data):
    """
    method need to be reimplemented, one can add result to output_queue in this func
    or just return the result which will be pushed to output queue automatically.
    you should add the process result to input_data dict, and push it to the output
    queue or just return it. You can refer to easy_vision.python.pai_utils.ev_process.DataFields
    attribute to acquire what information you can get from input_dict, you can only
    write data with key like DataFields.image or DataFields.clip
    Args:
      in_data   if self.batch_size is 1, input_data is a dict of one sample data
                if self.batch_size greater than 1, input_data is a list of dict, each dict
                  contains one sample data
    Return
      if None, result should be added to output_queue by yourself in this func
      if not None, the returned result will be add to output_queue automatically
    """
    return in_data
  def destroy(self):
    """
    destroy resources that has been used for this process
    """
    pass
```

To use the custom process class for video prediction, you must pass a dictionary that contains the following fields:

- url: the URL of the video.
- video_binary_data: the original video stream, in the STRING format.
- frames: the frames that are decoded from the video, in the LIST format. In the list, each element indicates a frame in the STRING format.
- time_stamp: the timestamps, in the LIST format. In the list, each element indicates a timestamp in the FLOAT format.
- xxx: the custom information.

The following code provides an example of the custom process class:

```
#!/usr/bin/env python
# -*- encoding:utf-8 -*-
# Filename: user_process.py
from easy_vision.python.pai_utils import ev_process
import logging
class MyProcess(ev_process.UserProcess):
  def __init__(self,
               job_name,
               num_threads=1,
               input_queue=None,
               output_queue=None,
               process_config=None,
               batch_size=1):
    """
    Args:
      job_name:  job name for this process
      num_threads: number of threads to run your process
      input_queue:  a queue containing input data, you should not read input_queue by
        yourself
      output_queue:  you can push data to output_queue or just return data in process
        which will be pushed into output queue automatically
      process_config:  config for your process, dict type
      batch_size:  if batch_size is 1, input_data passed to in_data is a dict of one sample data
        if batch_size greater than 1, input_data passed to process is a list of dict, each dict
                contains one sample data
    """
    super(MyProcess, self).__init__(job_name, num_threads,
        input_queue = input_queue,
        output_queue = output_queue,
        batch_size = batch_size)
  def process(self, in_data):
    """
    method need to be reimplemented, one can add result to output_queue in this func
    or just return the result which will be pushed to output queue automatically.
    you should add the process result to input_data dict, and push it to the output
    queue or just return it. You can refer to easy_vision.python.pai_utils.ev_process.DataFields
    attribute to acquire what information you can get from input_dict, you can only
    write data with key like DataFields.image or DataFields.clip
    Args:
      in_data   if self.batch_size is 1, input_data is a dict of one sample data
                if self.batch_size greater than 1, input_data is a list of dict, each dict
                  contains one sample data
    Return
      if None, result should be added to output_queue by yourself in this func
      if not None, the returned result will be add to output_queue automatically
    """
    logging.info('my process process data')
    return in_data
  def destroy(self):
    """
    destroy resources that has been used for this process
    """
    logging.info('my process destroyed')
```

Custom predictor class

The custom predictor class must inherit `easy_vision.python.inference.predictor.PredictorInterface`. The names and default values of all functions and parameters of the custom predictor class must be the same as those in the definition of the UserProcess operation.

The general video prediction component is compatible with V1 and V2 operation definitions of the custom predictor class.

- V1

  The following code shows the operation definition:

```
class PredictorInterface(six.with_metaclass(abc.ABCMeta, object)):
  def __init__(self, model_path, model_config=None):
    """
    init tensorflow session and load tf model
    Args:
      model_path  init model from this directory
      model_config config string for model to init, in json format
    """
    pass
  @abc.abstractmethod
  def predict(self, input_data, batch_size):
    """
    using session run predict a number of samples using batch_size
    Args:
      input_data:  a list of numpy array, each array is a sample to be predicted
      batch_size: batch_size passed by the caller, you can also ignore this param and
        use a fixed number if you do not want to adjust batch_size in runtime
    Return:
      result: a list of dict, each dict is the prediction result of one sample
        eg, {"output1": value1, "output2": value2}, the value type can be
        python int str float, and numpy array
    """
    pass
```

  The following code provides an example of the custom predictor class:

```
#!/usr/bin/env python
# -*- encoding:utf-8 -*-
# Filename: user_predictor.py
# Description:
from easy_vision.python.inference import predictor
import tensorflow as tf
class MyPredictor(predictor.PredictorInterface):
  def __init__(self, model_path, model_config=None):
    """
    init tensorflow session and load tf model
    Args:
      model_path  init model from this directory
      model_config config string for model to init, in json format
    """
    tf.logging.info('my predictor init done')
  def predict(self, input_data, batch_size):
    """
    using session run predit a number of samples using batch_size
    Args:
      input_data:  a list of numpy array[l*h*w*c], each array is a sample to be predicted
      batch_size: batch_size passed by the caller, you can also ignore this param and
        use a fixed number if you do not want to adjust batch_size in runtime
    Return:
      result: a list of dict, each dict is the prediction result of one sample
        eg, {"output1": value1, "output2": value2}, the value type can be
        python int str float, and numpy array
    """
    results = []
    for i in range(len(input_data)):
      output = {'result1':'my_predictor_result1'}
      if 'url' in input_data:
        output['url'] = input_data['url']
      if 'user_define' in input_data:
        output['user_define'] = input_data['user_define']
      results.append(output)
    return results
```

- V2

  The custom predictor class has the following limits:

  - You can use the custom predictor class to pass only the list of images. No other information can be passed.
  - The output dictionary is converted into only JSON strings and cannot save image files.

  To remove the limits, PAI provides V2 operation definition to enable you to specify the output format. V2 is developed based on V1 by adding the get_output_type function. This function returns a dictionary that indicates how to save the values in the key-value pairs in the dictionary returned by the prediction. The values can be saved in the following ways:

  - json: All the output JSON data of keys is aggregated into a dictionary and serialized to a JSON string. Then, the JSON string is written to the output OSS file.
  - image: The output data of the keys that correspond to values of the image type is saved in an OSS file *output_dir/key/filename.jpg*. The value of the filename variable is obtained from the input URL. If the output data of a key is a list of images, each image in the list is saved as *output_dir/key/filename_idx.jpg*, where, idx is the index number of each image.
  - video: not supported.

  To pass more information, you can also pass a list of dictionaries to the predictor class by using custom input data types.

The following code shows the operation definition:

```
class PredictorInterfaceV2(six.with_metaclass(abc.ABCMeta, object)):
  def __init__(self, model_path, model_config=None):
    """
    init tensorflow session and load tf model
    Args:
      model_path  init model from this directory
      model_config config string for model to init, in json format
    """
    pass
  def get_output_type(self):
    """
    in this function user should return a type dict, which indicates
    which type of data should the output of predictor be converted to
    * type json, data will be serialized to json str
    * type image, data will be converted to encode image binary and write to oss file,
      whose name is output_dir/${key}/${input_filename}_${idx}.jpg, where input_filename
      is extracted from url, key corresponds to the key in the dict of output_type,
      if the type of data indexed by key is a list, idx is the index of element in list, otherwhil
e ${idx} will be empty
    * type video, data will be converted to encode video binary and write to oss file,
    eg:  return  {
      'image': 'jpg',
      'feature': 'json'
    }
    indicating that the image data in the output dict will be save to image
    file and feature in output dict will be converted to json
    """
    return {}
  @abc.abstractmethod
    def predict(self, input_data_dict, batch_size):
    """
    using session run predict a number of samples using batch_size
    Args:
      input_data_dict:  a list of dict, each dict is a sample data to be predicted
      batch_size: batch_size passed by the caller, you can also ignore this param and
        use a fixed number if you do not want to adjust batch_size in runtime
    Return:
      result: a list of dict, each dict is the prediction result of one sample
        eg, {"output1": value1, "output2": value2}, the value type can be
        python int str float, and numpy array
    """
    pass
```

The following code provides an example of the custom predictor class:

```
  class DummyMultiModalPredictor(ev_predictor.PredictorInterfaceV2):
    def __init__(self, model_path, model_config=None):
      pass
    def get_output_type(self):
      return {'result1': 'json',
              'result2':'json',
              'result_image': 'image',
              'result_image_list':'image',
              }
    def predict(self, images, batch_size=1):
      results = [{
                  'result1':'dummy',
                  'result2':'dummy',
                  'result_image': np.zeros([224,224,3],dtype=np.uint8),
                  'result_image_list': [255* np.ones([224,224,3],dtype=np.uint8) for i in range(2
))]
                } for i in range(len(images))]
      return results
```

ASR prediction

This topic provides an example on how to run a Machine Learning Platform for AI (PAI) command to use the automatic speech recognition (ASR) prediction component. This topic also describes the command parameters.

You must specify a MaxCompute table that contains the URL of the raw data as the input. Then, the ASR prediction component returns all the input data with an extra row at the end that contains the ASR result.

> **Note** You must create an empty table in advance as the output table. The schema of the output table must be the same as that of the input table. The last row of the output table contains the prediction result in the STRING format.

PAI command

```
PAI -name asr_predict_180 -project sre_mpi_algo_dev
-Dbuckets='oss://pai-audio-zjk/?role_arn=xxx&host=yyy'
-Dinput_tables='odps://sre_mpi_algo_dev/tables/video_asr_dev'
-Doutput_tables='odps://sre_mpi_algo_dev/tables/video_asr_dev_output'
-Durl_col_index=0
-Dmerge=1
-Dnum_workers=1
-Dnum_processes=6
-Dnum_download=7
-Dnum_memory=10000
-Doversubscription=false;
```

Set the buckets, input_tables, and output_tables parameters as needed.

Parameters

Parameter	Required	Description	Value type	GraphKey

Parameter	Required	Description	Value type	GraphKey
buckets	Yes	The information about the Object Storage Service (OSS) buckets. Set this parameter to the path of a directory in an OSS bucket. The value format is the same as that of the buckets parameter of PAI-TensorFlow. Example: oss://pai-audio-zjk/?role_arn=xxx&host=yyy.	STRING	N/A
input_tables	Yes	The input tables. Example: odps://sre_mpi_algo_dev/tables/video_asr_dev.	STRING	N/A
output_tables	Yes	The output tables. Example: odps://sre_mpi_algo_dev/tables/video_asr_dev_output.	STRING	N/A
url_col_index	No	The serial number of the column that contains the data URL. The serial number starts from 0.	INT	0
merge	No	Specifies whether to return the merged recognition result. If you set this parameter to 0, the recognition result contains the speech segments and their start and stop timestamps. Example: {"detection_texts": ["你看人家想妈了吧小妈了", "小妈了", "太好了", "你呀赶紧给你妈打电话话"], "time": [["0", "2660"], ["4040", "4660"], ["5800", "6560"], ["6840", "10069"]]}.	INT	1
num_workers	No	The number of workers that are used for prediction.	INT	1
num_processes	No	The number of threads that are used by each worker for prediction. We recommend that you set this parameter to 12 or less.	INT	6
num_download	No	The number of threads that are used by each worker to download videos. We recommend that you set this parameter to 10 or less.	INT	7
num_memory	No	The size of the memory that each worker consumes, in MB.	INT	10000
oversubscription	No	The value format of this parameter is the same as that of the oversubscription parameter of PAI-TensorFlow. • true • false	BOOL	true

If an error occurs during ASR, the system returns the corresponding error code as a string in the output table. The

following table describes the error codes.

Error message	Description
data_convert_error	An error occurred during data download or transcoding. Check whether the data source is normal.
data_download_error	An error occurred during data download. Check whether the URL is valid.
predict_error	An error occurred during the model prediction.

4.1.5.3. Online model service (must be activated separately)

4.1.5.3.1. Deploy a model as an online service

This topic describes how to deploy the generated experiment model as an online service for prediction by using Elastic Algorithm Service (EAS). You can adjust your business strategy in real time based on predicted results.

Prerequisites

Before you deploy a model as an online service, make sure that the preceding steps are performed and the components are run as expected. A green check means that the component runs as expected.

Procedure

1. Log on to Apsara Stack Machine Learning Platform for AI.
2. In the left-side navigation pane, choose **Model Training > Studio-Modeling Visualization**.
3. On the **PAI Visualization Modeling** page, find your project and click **Machine Learning** in the **Operation** column.
4. In the left-side navigation pane, click **Experiments**.
5. Click **My Experiments** and click the experiment that you want to manage.

   > **Notice** Make sure that all components are run as expected in the experiment. A green tick means that the component is successful.

6. In the upper-left corner of the canvas, choose **Deploy > Online Model Service**.
7. In the Select Model dialog box, select the model to deploy and click **Next**.
8. Select a deployment mode. You can select one of the following modes:
   - Deploy a new service
   - Add a version to an existing service
   - Implement a blue-green deployment

4.1.5.3.2. 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.

4.1.5.3.3. 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.

4.1.5.3.4. 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,"thalach":0,"oldpeak":0,"ca":0}]
```

   iii. Click **Run** and check the result.

4.1.6. DSW user guide

4.1.6.1. Overview

This topic describes the features and user interfaces of Data Science Workshop (DSW).

Features

PAI supports visualized modeling and allows you to create models in Notebook or an online integrated development environment (IDE). This way, native code can be used to create custom models to better meet your requirements. You can select a programming environment as needed on the Notebook Modeling page to create and train models. DSW provides the following features:

- Allows you to start and stop DSW instances on demand, save images with a few clicks, restore development environments, and access resources over virtual private clouds (VPCs).
- Provides integrated AI development environments.
- Provides built-in big data development packages and algorithm packages, and grants sudo permissions to you for installing third-party libraries.
- Provides official images that support different versions of mainstream computing frameworks, such as TensorFlow and PyTorch.
- Provides built-in WebIDE that allows you to install all plug-ins.

Interfaces related to modeling in DSW

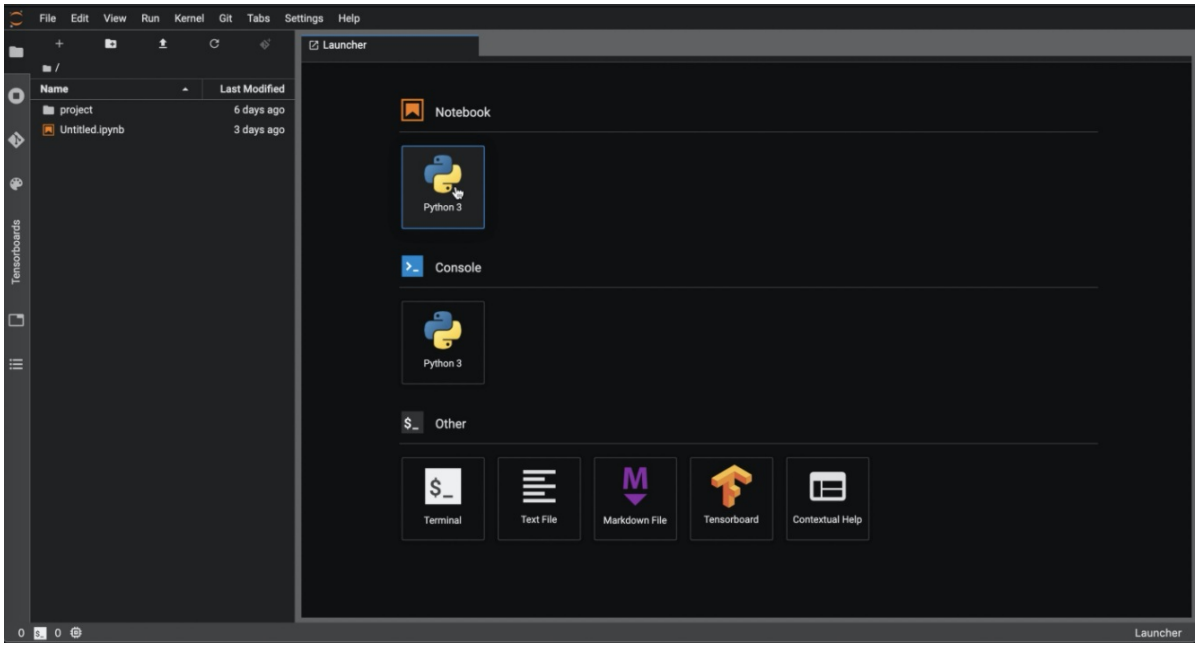
Modeling in DSW involves the following interfaces:

- DSW-Notebook Service page in the PAI console

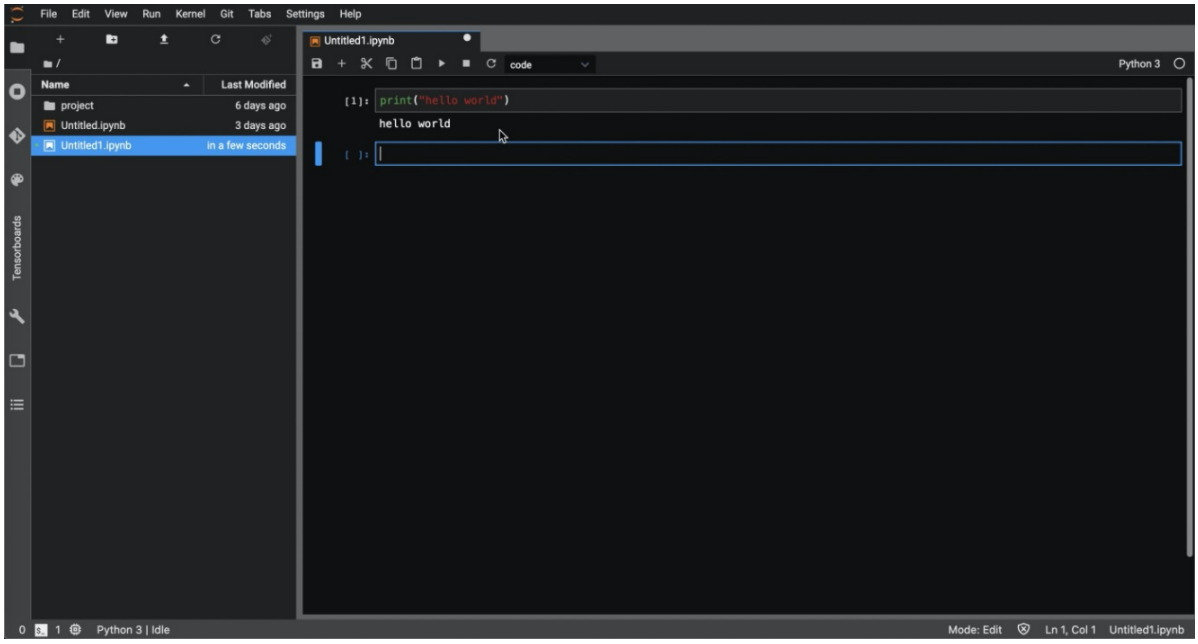
  In the PAI console, you can view the information about all created DSW instances and their statuses. You can create, stop, or delete DSW instances. In addition, you can click Lab in the Actions column to create models in Notebook. Alternatively, you can click IDE or Terminal in the Actions column to create models in WebIDE based on native code, or create models on online terminals.

- Notebook modeling interface

You can create, upload, or open an existing notebook in the Notebook modeling interface, as shown in the following figure. Notebook supports various data formats. This facilitates data exploration and modeling.

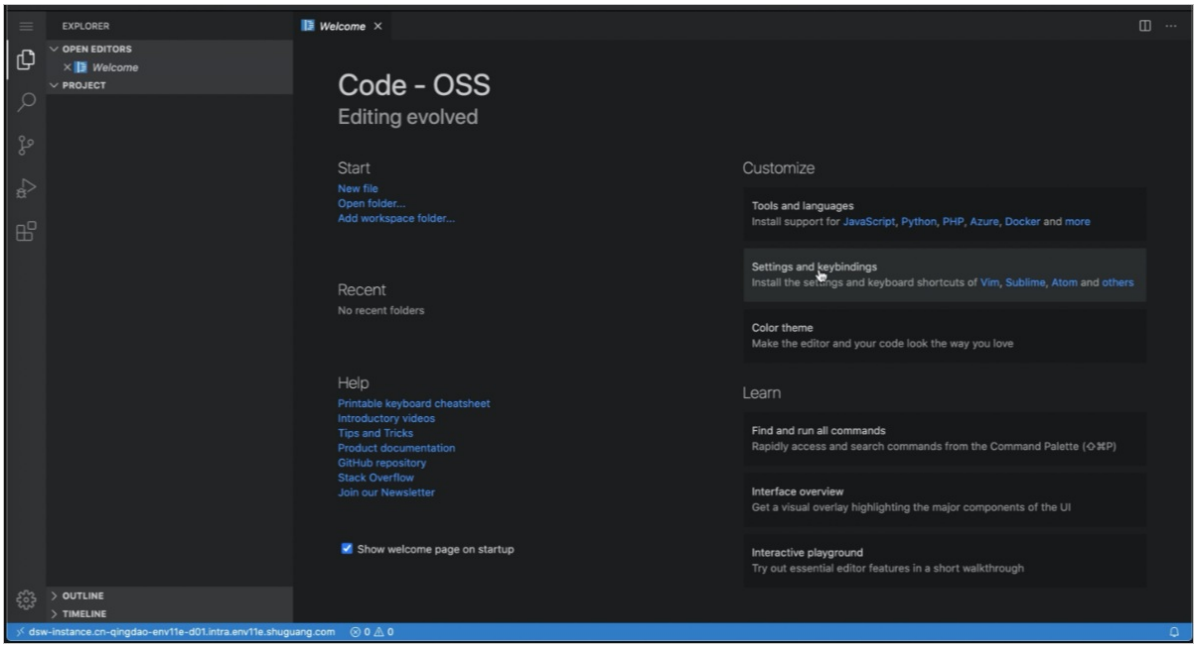


The following figure shows how to create a model in Notebook.



- WebIDE interface

You can use native code to create models in WebIDE.



- Online terminal interface

  You can use command lines to create models on online terminals.

Persistent storage

If the DSW environment contains a distributed storage system such as a bucket in Object Storage Service (OSS) or a file system in Apsara File Storage NAS (NAS), PAI automatically creates a disk for each logon user in OSS or NAS. The disk is a remote disk that is bound to each logon user and mounted to the fixed directory */home/admin/workspace*. After a DSW instance is created, DSW mounts the created disk to the fixed directory */home/admin/workspace*.

By default, all operations related to modeling in Notebook or WebIDE or on terminals are performed in this directory. The operation data that is generated when you write code, obtain data files, and export training results in this directory is permanently stored on the remote disk.

The remote disk separates computing from storage, and provides the following benefits:

- If your log off from your DSW instance for specific reason, user data is not lost.
- When you start your DSW instance for computing, the disk is automatically mounted to your instance, and you are connected to the data stored on the disk.

4.1.6.2. Manage instances

This topic describes how to manage Data Science Workshop (DSW) instances. You can create, stop, start, and delete DSW instances.

Create an instance

1. Go to the Notebook Models page.
2. On the **Notebook Modeling** page, click **Create Instance**.
3. In the **Create Instance** panel, set the parameters.

Parameter	Description
Instance	The instance name cannot exceed 27 characters in length and can contain only letters, digits, and underscores (_).
Compute Resource Type	Valid values: **CPU** and **GPU**.
Resource Type	The instance type. You can an instance type with appropriate resource specifications.
Image Source	The image based on which the instance is created. Only official images are supported. You can select one as needed.

4. Click **OK**.

Stop an instance

1. Go to the Notebook Models page.
2. On the **Notebook Models** page, find the instance that you want to stop and click **Stop** in the **Actions** column.

Start an instance

You can manually start an instance that is in the Stopped state.

1. Go to the Notebook Models page.
2. On the **Notebook Models** page, find the instance that you want to start and click **Start** in the **Actions** column.

   After the instance is started, its status changes to **Running**.

Delete an instance

You can delete instances that are not in use. After an instance is deleted, its data cannot be recovered.

1. Go to the Notebook Models page.
2. On the **Notebook Models** page, find the instance that you want to delete, and click **Delete** in the **Actions** column.
3. In the **Delete** message, click **OK**.

Go to the Notebook Models page

1. Log on to the Machine Learning Platform for AI (PAI) console. For more information, see Log on to the PAI console.
2. In the left-side navigation pane, choose **Model Training > DSW-Notebook Service**.

4.1.6.3. Work with the development environments of DSW

This article describes how to go to the development environments of Data Science Workshop (DSW), including JupyterLab, WebIDE, and Terminal.

Prerequisites

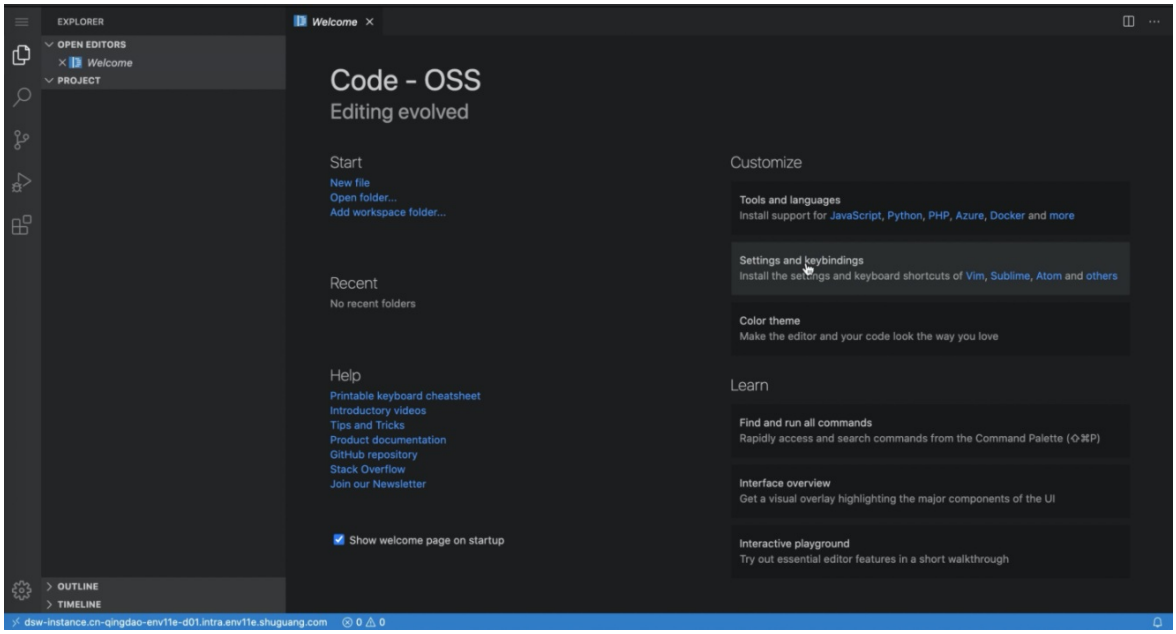
A DSW instance is created. For more information, see the "Create an instance" section of the Manage instances" topic.

Go to JupyterLab

1. Log on to the Machine Learning Platform for AI (PAI) console. For more information, see Log on to the PAI console.
2. In the left-side navigation pane, choose **Model Training > DSW-Notebook Service**.
3. On the **Notebook Models** page, find the instance that you want to manage and click **Lab** in the **Actions** column.
4. In JupyterLab, you can build Notebook models. For more information about how to build Notebook models, see JupyterLab Documentation.

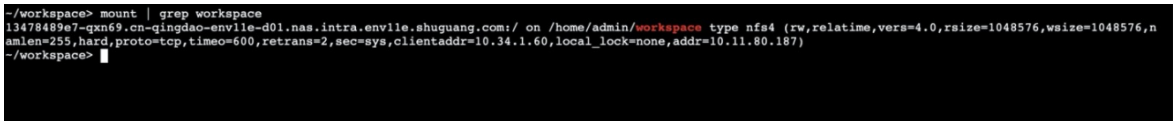
Go to WebIDE

1. Log on to the Machine Learning Platform for AI (PAI) console. For more information, see Log on to the PAI console.
2. In the left-side navigation pane, choose **Model Training > DSW-Notebook Service**.
3. On the **Notebook Models** page, find the instance that you want to manage and click **IDE** in the **Actions** column.
   The following figure shows the user interface of WebIDE.



Go to Terminal

1. Log on to the PAI console. For more information, see Log on to the PAI console.
2. In the left-side navigation pane, choose **Model Training > DSW-Notebook Service**.
3. On the **Notebook Models** page, find the instance that you want to manage and click **Terminal** in the **Actions** column.
   The following figure shows the user interface of Terminal.



4.1.6.4. Video intelligence platform (must be separately activated)

The video intelligent platform (VIP) is a one-stop platform for intelligent video analysis. VIP supports the following features for a large amount of video data: data preprocessing, feature extraction, training, and prediction. VIP integrates the storage, decoding, model training, and video algorithm analysis features for video data.

VIP supports a distributed compute engine with high performance and advanced algorithms for images, videos, audio, and natural language processing (NLP). This feature allows you to obtain and analyze video data in a flexible way. This way, you can implement an end-to-end video analysis solution to meet the needs in different scenarios.

Machine Learning Platform for AI (PAI) provides three types of upper-layer application algorithm libraries: EasyVision for images and videos, EasyTransfer for text, and EasyASR for speech recognition. You can train, evaluate and predict models in various fields by calling the API for Python in Data Science Workshop (DSW).

For more information about EasyVision, how to use EasyVision, and the API operations of EasyVision, see the topics related to EasyVison in the SDK Reference chapter.

4.1.7. Manage models

You can manage models on the Model Management page. For a registered model, you can deploy it to EAS, download the model, or delete the model.

Register a model

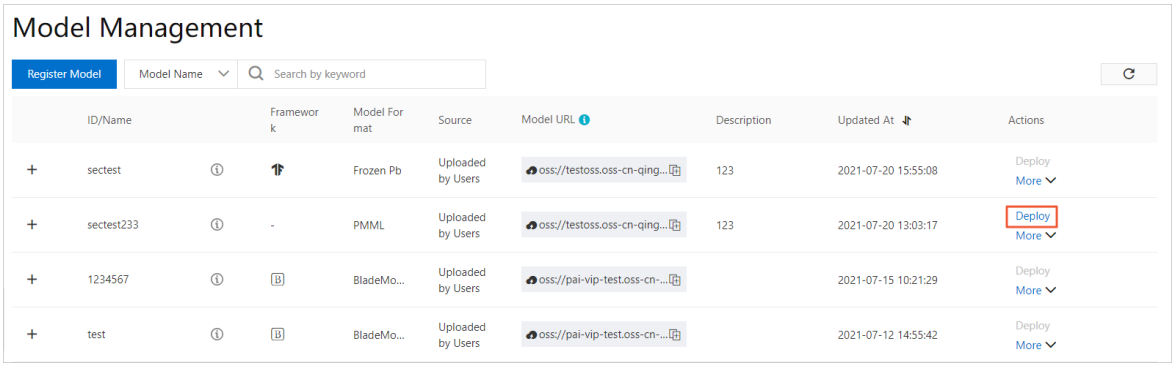
1. Go to the Model Management page.
2. On the **Model Management** page, click **Register Model**.
3. In the **Register Model** panel, set the parameters.

Parameter	Description
Model Name	The name of the model. The name must be 1 to 27 characters in length and can contain underscores (_) and hyphens (-). It must start with a letter or digit.
Description	The description of the model. Enter an informational description based on your business requirements to facilitate model management.
Model Format	Valid values: ○ SavedModel ○ Frozen Pb ○ Keras H5 ○ Caffe Prototxt ○ ONNX ○ BladeModel ○ PMML ○ Torchscript ○ TFLite
Model Framework	The system automatically select the framework based on the selected **model format**. You do not need to set this parameter.
Model URL	Model files are stored in OSS. To register a model, you must authorize PAI to access OSS. You can select **User-defined OSS Path** or **Upload to Specified OSS Path**.

4. Click **Confirm Operation**.

Deploy a model

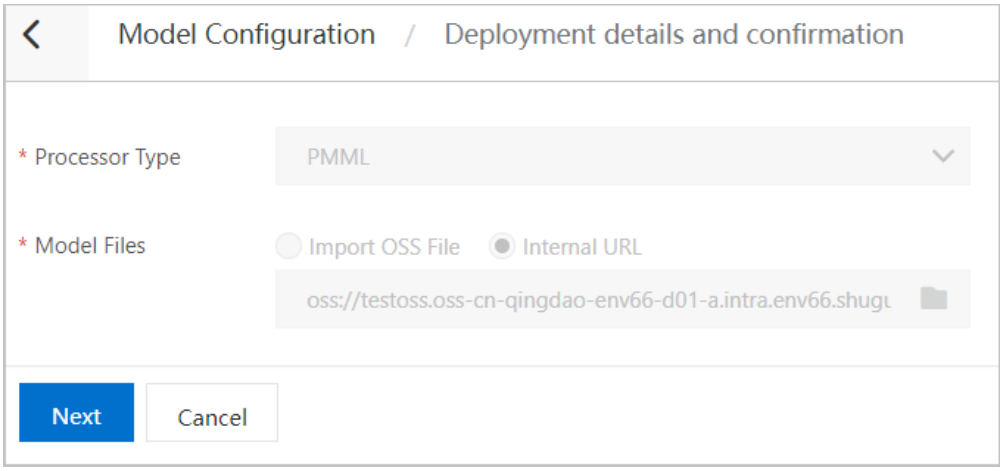
1. Go to the Model Management page.
2. On the **Model Management** page, find the model that you want to deploy and click **Deploy** in the **Actions** column.



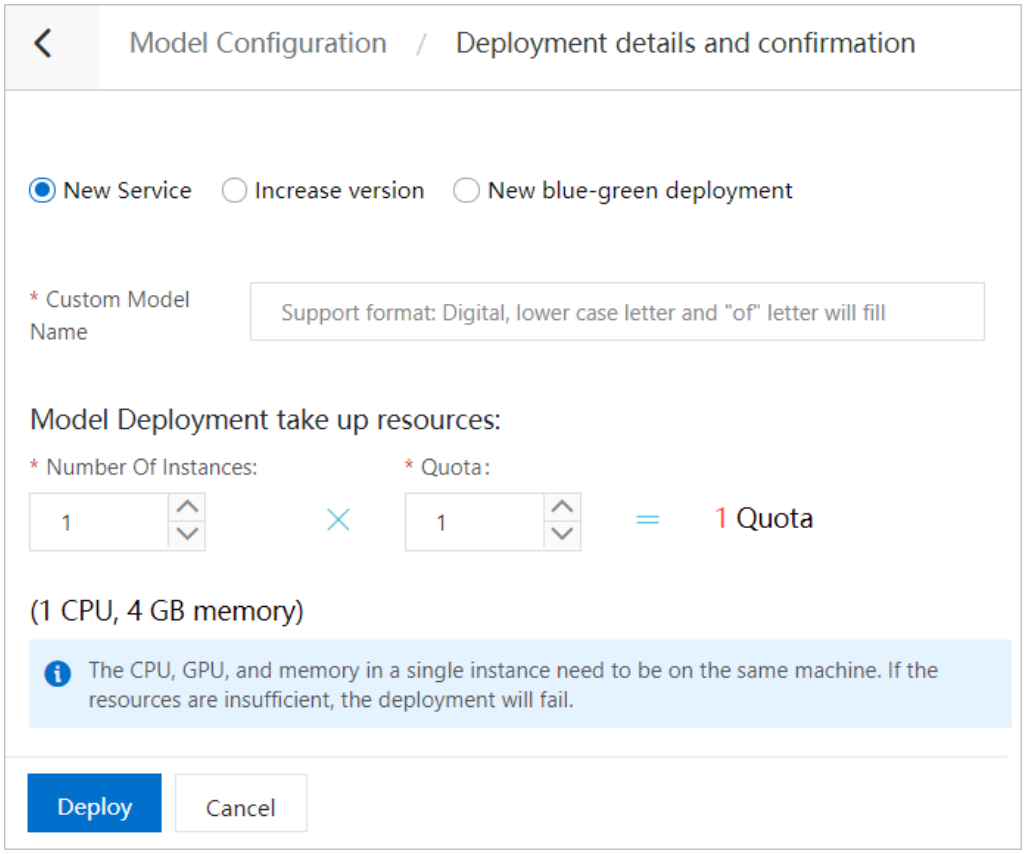
On the Model Management page, you can deploy only models in the SavedModel, PMML, and Torchscript formats to EAS. The **Deploy** button is dimmed for models in other formats. If you need to deploy models in other formats, you can use custom processors in EAS to deploy models. For more information, see Create a service by uploading a model to the PAI console.

3. Confirm the region where the model is deployed and click **OK**.

4. In the **Model Configuration** panel, the system automatically sets the **Processor Type** and **Model Files** parameters. You can directly click **Next**.



5. In the **Deployment details and confirmation** panel, set the parameters.

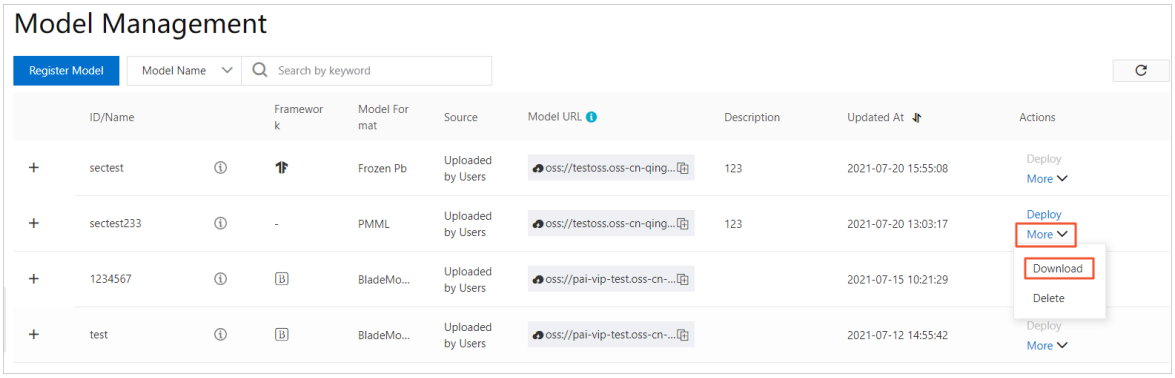


Parameter		Description
Deployment mode		EAS supports the following three deployment modes: ○ **New Service**: Deploy a new service. If you want to deploy a model that is registered on the Model Management page, select this mode to deploy the registered model as a new service. ○ **Increase version**: Add a new version to an existing service. The version number of the existing service increases by one. ○ **New blue-green deployment**: Deploy an associated service for an existing service. You can configure the traffic allocation between the two services.
Custom Model Name		The name can contain digits, lowercase letters, and underscores (_). It must start with a letter.
Model Deployment take up resources	**Number Of Instances**	We recommend that you use multiple service instances to prevent risks caused by single-machine deployment.
	Quota	One quota contains 1 core and 4 GB of memory.

6. Click **Deploy**.

Download a model

1. Go to the Model Management page.
2. On the **Model Management** page, find the model that you want to download and choose **More > Download** in the **Actions** column.



Delete a model

1. Go to the Model Management page.
2. On the **Model Management** page, choose **More > Delete** in the **Actions** column.
3. In the **Delete** message, click **Delete**.

Go to the Model Management page

1. Log on to the Machine Learning Platform for AI console.
2. In the left-side navigation pane, click **Model Management and Optimization**.

4.1.8. EAS user guide

4.1.8.1. Overview

EAS allows you to deploy machine learning models as online services with simple configurations.

EAS allows you to deploy machine learning models, such as PMML, PAI-OfflineModel, TensorFlow, and Caffe models, as online services with simple configurations. 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 PAI SDK for Java to deploy it as an online service. The service can be accessed from the production environment and the Internet. EAS supports various regions and different types of hardware resources, including CPUs and GPUs.

4.1.8.2. EASCMD client

This topic describes how to download the EASCMD client.

You can use the EASCMD client to manage services, including creating, deleting, and modifying services, and view the service status. In Machine Learning Platform for AI (PAI), the EASCMD client is stored in the eas-utils bucket. You can download the EASCMD client by using the wget or curl command. Perform the following steps:

1. Obtain the endpoint of the storage service.

   MinIO is an easy-to-use service that provides storage for Elastic Algorithm Service (EAS). The Internet endpoint of MinIO is the endpoint of the storage service for EAS. Sample Internet endpoint: `eas.minio.cn-hangzhou-env50-d01.inter.env50e.shuguang.com`. You can run the following command to obtain the Internet endpoint.

   ```
   kubectl get cm -n eas-system eas-config -oyaml
   ```

2. Download the EASCMD client by running the following `wget` command:

   ```
   wget eas.minio.cn-hangzhou-env50-d01.inter.env50e.shuguang.com/eas-utils/eascmd64
   ```

Replace `eas.minio.cn-hangzhou-env50-d01.inter.env50e.shuguang.com` with the endpoint of the storage service that you actually obtain in the preceding step. The following figure shows the output of the wget command.

4.1.8.3. User authentication

You can use the EASCMD client to manage the services in Elastic Algorithm Service (EAS). This topic shows you how to download the EASCMD client and complete user authentication.

In Machine Learning Platform for AI (PAI), prediction services use JSON Web Token (JWT) for user authentication. When you use the EASCMD client, run the following command to configure an AccessKey pair for the EASCMD client:

```
./eascmd64 config -i <AccessKeyId> -k <AccessKeySecret> -e {{Domain name of EAS in PAI}}
// New users can use the admin account to pass authentication.
./eascmd64 config -i admin -k admin -e {{Domain name of EAS in PAI}}
```

You can use the following methods to obtain the domain name of EAS in PAI:

- Concatenate a domain name:
  - To call the API for service management, you can concatenate a domain name in the following format:

    ```
    eas.${global:region}.${global:internet-domain}
    ```

  - To call the API for service calls, you can concatenate a domain name in one of the following formats based on whether the prediction service is deployed across regions:
    - Secondary region: `eas.pai.${global:region}.${global:internet-domain}`
    - Primary region: `eas.pai.${global:internet-domain}`
- Use the EAS Meta Server to obtain the access information by performing the following steps:
  i. Run the following command to obtain the string with Region as the key:

  ```
  curl http://easmeta.pai.${global:internet-domain}/regions
  ```

  ii. Find the access information of the current region.

4.1.8.4. 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.

4.1.8.5. Create a service

To create a service, you can use the EASCMD client tool or upload a model in the Machine Learning Platform for AI console.

Create a service by using the EASCMD client

You can run the `create` command to create a service. When you create the service, you need to configure resources such as the model or processor by specifying HTTP URLs. You can upload resources to Object Storage Service (OSS) and obtain the HTTP URL provided by OSS.

```
eascmd create <service_desc_json>
```

In the preceding command, <service_desc_json> indicates the name of the service description file. The file includes information about the service. The metadata parameter in the file specifies the region where the cluster is deployed and the information about resources.

> **Note** The metadata parameter specifies only the deployment information about the cluster. When you run the test command to debug the on-premises model, the system ignores the metadata parameter.

The following code provides an example of the <service_desc_json> file:

```
{
  "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 following table describes the parameters in the service description file.

Key	Value
name	The name of the service. The name of the service must be unique in a region. You can specify the region where the service is deployed in the metadata parameter.
generate_token	Specifies whether a token is generated. If you set the parameter to true, you must include the token in the HTTP URL that is used to access the prediction service. If you set the parameter to false, the service is a public service and can be accessed without authentication.
token	Optional. The token string that is used for authentication. If this parameter is not specified, the system automatically generates a token when the generate_token parameter is set to true.
model_path	The path of the model package. For more information, see the note that follows this table.
model_entry	Optional. The entry file of the model package. If this parameter is not specified, the file name in the value of the model_path parameter is used. The path of the main file is passed to the Load() function in the processor.
model_config	Optional. The configurations of the model. The value is of the TEXT type. The value of this parameter is passed to the second parameter of the LoadWithConfig() function in the processor.
processor	Optional. The built-in processor that is to be used for prediction. When this parameter is specified, the processor_path, processor_entry, processor_mainclass, and processor_type parameters are ignored.

Key	Value
processor_path	Optional. The package path of the custom processor to be used. For more information, see the note that follows this table.
processor_entry	Optional. The main file of the processor package. The main file contains the implementations of the Load() and process() functions that are required for prediction. This parameter is required when the language used to implement the custom processor is C or C++.
processor_mainclass	Optional. The main class in the JAR package of the processor. This parameter is required when the language used to implement the custom processor is Java.
processor_type	Optional. The language that is used to implement the custom processor. Valid values: cpp, java, and python.
metadata	The metadata of the service. For more information, see Description of the metadata parameter.

Note The model_path parameter specifies the path in which the input model package is stored. The processor_path parameter specifies the path in which the input processor package is stored. The values of the two parameters can be HTTP URLs or OSS URLs. If you want to run the test command to perform on-premises debugging, you can use an on-premises path.

If an HTTP URL is used, the input package must be in the TAR.GZ, TAR, BZ2, or ZIP format.

Description of the metadata parameter

Category	Parameter	Description
Common parameters	workers	Optional. Default value: 5. This parameter specifies the number of threads that are used to concurrently process requests on each instance.
	instance	The number of instances that are required to run the service.
	cpu	The number of CPUs that are required by each instance.
	gpu	The number of GPUs that are required by each instance.
	resource	The name of the resource group. You can ignore this parameter if the service uses only CPUs. If the service uses only GPUs, you can specify the P4_4CORE or P4_8CORE resource group.

Category	Parameter	Description
Advanced parameters (modify them with caution)	rpc.batching	Optional. Default value: false. This parameter specifies whether batch processing is enabled on the server to improve the speed of a GPU-based model.
	rpc.keepalive	Optional. Default value: 5000. Unit: milliseconds. This parameter specifies the maximum processing time for a single request. If the request processing time exceeds this value, the server returns the timeout error code 408 and closes the connection.
	rpc.io_threads	Optional. Default value: 4. This parameter specifies the number of threads that are used to process the input and output network data on each instance.
	rpc.max_batch_size	Optional. Default value: 16. This parameter specifies the maximum size of each batch. This parameter takes effect only when the rpc.batching parameter is set to true.
	rpc.max_batch_timeout	Optional. Default value: 50. Unit: milliseconds. This parameter specifies the maximum timeout period of each batch. This parameter takes effect only when the rpc.batching parameter is set to true.
	rpc.max_queue_size	Optional. The size of the request queue. Default value: 64. When the queue is full, the server returns the error code 450 and closes the connection. To prevent the server from being overloaded, the request queue instructs the client to send requests to other instances when the queue is full. If the response time is too long, set this parameter to a smaller value to prevent a request from timing out.
	rpc.worker_threads	Optional. Default value: 5. This parameter specifies the number of threads that are used to concurrently process requests on each instance. This parameter works in the same way as the workers parameter.

The following code provides an example on how to configure the metadata parameter:

```
{
 ...
"metadata": {
  "cpu": 4,
"rpc.max_queue_size": 32,
...
 }
```

For example, if the service description file pmml.json is created, you can run the following command to create a service:

```
eascmd create pmml.json
```

The system displays information similar to the following output:

```
[RequestId]: 1651567F-8F8D-4A2B-933D-F8D3E2DD****
+------------------+-----------------------------------------------------------------------------+
| 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
```

Create a service by uploading a model to the Machine Learning Platform for AI console

On the **Elastic Algorithm Service** page, you can upload trained models and deploy them as online model services. Perform the following steps:

1. Go to the **Elastic Algorithm Service** page.
   i. Log on to the PAI console.
   ii. In the left-side navigation pane, choose **Model Deployment > EAS-Model Serving**.
2. In the upper-left corner of the page, select the region where you want to create a service.

3. On the **Elastic Algorithm Service** page, click **Model Deploy**.
4. In the **Model Configuration** step of the panel that appears, set the parameters.

Parameter	Description
Processor Type	Valid values: **PMML**, **TensorFlow1.12**, and **Self-definition processor**.

Parameter	Description
Processor Language	This parameter takes effect only when the **Processor Type** parameter is set to **Self-definition processor**. Valid values: **Cpp**, **Java**, and **python**.
Processor package	This parameter takes effect only when the **Processor Type** parameter is set to **Self-definition processor**. In the **Processor package** field, enter a public URL.
Processor Master File	This parameter takes effect only when the **Processor Type** parameter is set to **Self-definition processor**. This parameter specifies the main file of the custom processor package.
Model Files	You can set this parameter in one of the following ways: ○ **Import OSS File** Select **Import OSS File**. Then, select the OSS path where the model file resides. ○ **Internal URL** Select **Internal URL**. Then, enter a public URL.

5. Click **Next**.
6. In the **Deployment details and confirmation** step, set the parameters.
   i. Select **New Service**.

      EAS supports the following deployment modes:

      - **New Service**: Deploy a new service. In this example, this deployment mode is used.
      - **Increase version**: Add a new version to an existing model service. The version number of the existing model service increases by one.
      - **New blue-green deployment**: Deploy an associated service for an existing service. You can configure the traffic allocation between the two services.

   ii. Set the **Custom Model Name** parameter and other parameters.

Parameter		Description
Model Deployment take up resources	**Number Of Instances**	We recommend that you use multiple service instances to prevent risks caused by single-machine deployment.
	Quota	One quota contains 1 core and 4 GB of memory.

> **Note**
>
> - The CPU, GPU, and memory of a single service instance must belong to the same machine. If the resources are insufficient, the deployment fails.
> - To deploy the model as an online service with high stability requirements, we recommend that you use a resource group that includes multiple machines and deploy multiple service instances.

   iii. Click **Deploy**.

4.1.8.6. Perform on-premises debugging

Before you deploy a service to a cluster, you can use the on-premises debugging feature to start an on-premises service for debugging. This topic describes how to perform on-premises debugging.

This feature requires a Docker container and Internet access. You must install Docker on the machine where the EASCMD client runs and run the following command to debug the service:

```
sudo eascmd test [service_desc_json]
```

Specify the JSON file that is used to deploy the service. Run the following sample command:

```
bin/eascmd test tf.json
```

The system displays information similar to the following output:

```
[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 an on-premises TensorFlow model service. The endpoint of the service is `http://localhost:6942/api/predict/savedmodel_exanple`.

4.1.8.7. Modify configurations

This topic describes how to modify the configurations of a service, such as the instance, CPU, and memory 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 run the following command to 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
```

4.1.8.8. 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.

4.1.8.9. Delete a service

This topic describes how to 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.

For example, to delete the savedmodel_exanple service, run the following command:

```
eascmd delete savedmodel_exanple
```

The system returns the following confirmation message:

```
Are you sure to delete the service [savedmodel_exanple] in [cn-shanghai]? [Y/n]
```

Enter Y. The system displays information similar to the following output:

```
[RequestId]: 1651567F-8F8D-4A2B-933D-F8D3E2DD****
[OK] Service [savedmodel_exanple] in region [cn-shanghai] is terminating
[OK] Service is terminating
[OK] Service is terminating
[OK] Service was deleted successfully
```

4.1.8.10. 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]
```

4.1.8.11. Enable HTTPS

To meet the requirements for encrypted transmission, (EAS) allows you to access services over HTTPS. This topic describes how enable or disable HTTPS.

You can enable HTTPS when you create a service. You can also enable HTTPS after a service is deployed. The following content describes the methods used to enable HTTPS for a new service or a deployed service:

- Enable HTTPS when you create a service.

  When you configure the service description file, set the enable_https parameter in metadata to true. The following code provides an example of the service description file. For more information about the parameters in the service description file, see Create a service.

```
  "name": "test",
  "token": "test-token",
  "model_path": "http://xxx/echo_processor_release.tar.gz",
  "processor_path": "http://xxx/echo_processor_release.tar.gz",
  "processor_entry": "libecho.so",
  "processor_type": "cpp",
  "metadata": {
    "instance": x,
    "memory": xxx,
    "enable_https": "true"
  }
}
```

- Enable HTTPS for a deployed service by running the following command:

```
eascmd modify <servicename> -Dmetadata.enable_https=true
```

Replace <servicename> with the name of the deployed service.

Note You cannot use both HTTPS and HTTP to access the service. If HTTPS is enabled, the service can be accessed only over HTTPS rather than HTTP.

If you want to disable HTTPS for a service, run the following command:

```
eascmd modify <servicename> -Dmetadata.enable_https=false
```

Replace <servicename> with the name of the service.

4.1.8.12. View services

This topic describes how to view deployed services.

You can run the `list(ls)` command to view the services that you deploy.

```
eascmd ls
```

The system displays information similar to the following output:

```
[RequestId]: 83945D4EED3E-4D35-A989-831E6BB****
+--------------------------+-------------+----------+---------------------+---------------------+--
-------+--------+-----------------------------------------+
|        SERVICENAME       |   REGION    | INSTANCE |     CREATETIME      |     UPDATETIME      | S
TATUS  | WEIGHT |              SERVICEPATH                |
+--------------------------+-------------+----------+---------------------+---------------------+--
-------+--------+-----------------------------------------+
| mnist_saved_model_example | cn-shanghai |        1 | 2019-02-21 16:35:41 | 2019-02-21 16:35:41 | R
unning |      0 | /api/predict/mnist_saved_model_example |
```

4.1.8.13. View the details of a service

This topic describes how to view the details of a deployed service.

You can run the `desc` command to view the details of a deployed service.

```
eascmd desc [service_name]
```

For example, to view the details of the mnist_saved_model_example service, you can run the following command:

```
eascmd desc mnist_saved_model_example
```

The system displays information similar to the following output:

```
+--------------------+-----------------------------------------------------------------------------------
---------------------------------+
|              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-shang
hai:v0.0.1-20190221163541        |
|              Weight | 0
|
|       LatestVersion | 1
|
|      CurrentVersion | 1
|
|             Message | Service start successfully
|
|       APIGatewayUrl | 1c3b37ea83c047efa0dc6df0cacb70d3-cn-shanghai.alicloudapi.com/EAPI_1828488879
222746_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"
|
|                      | }
```

4.1.8.14. View service processes

This topic describes how to view the status of service processes.

You can run the `showworkers(w)` command to view the status of the service processes.

```
eascmd w <service_name>
```

Replace <service_name> with the name of the service.

For example, to view the process status of the mnist_saved_model_example service, run the following command:

```
eascmd w mnist_saved_model_example
```

The system displays information similar to the following output:

```
[RequestId]: 4E905404-E617-4BD8-85D6-EC5C6A0D****
+--------------+----------------+---------------------+----------+---------+-------+---------+
|   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] |        |
+--------------+----------------+---------------------+----------+---------+-------+---------+
```

4.1.8.15. Call a prediction service

This topic describes how to call a prediction service.

To call a prediction service, you must use the HTTP URL that is generated when you create the prediction service. The HTTP URLs for on-premises testing services are generated in the same format as those for cluster-based prediction services, except that the HTTP URLs use different hosts. When you send a request to call a service, 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 a PMML prediction service by running the curl command:

```
curl http://{{Domain name of EAS in the environment}}/api/predict/pmml_example -H 'Authorization: NG
E3MzM4YmRhODZjMTE2NmZjZjNlNTJlNDgyNzM3YzdjMmIwOD****==' -d '[{}]'
[{"prediction_score":0.0902926730924553}]
```

You can use the following methods to obtain the domain name of EAS in PAI:

- Concatenate a domain name:
  - To call the API for service management, you can concatenate a domain name in the following format:

    ```
    eas.${global:region}.${global:internet-domain}
    ```

  - To call the API for service calls, you can concatenate a domain name in one of the following formats based on whether the prediction service is deployed across regions:
    - Secondary region: `eas.pai.${global:region}.${global:internet-domain}`
    - Primary region: `eas.pai.${global:internet-domain}`
- Use the EAS Meta Server to obtain the access information by performing the following steps:
  i. Run the following command to obtain the string with Region as the key:

  ```
  curl http://easmeta.pai.${global:internet-domain}/regions
  ```

  ii. Find the access information of the current region.

The input and output formats of a prediction service are determined by the processor of the service.

4.1.8.16. Use Java or C++ to develop a model service

4.1.8.16.1. What are processors?

A processor is a package that contains the online prediction logic. This topic describes how processors work.

A processor is a package that contains online prediction logic. Your requests are processed by processors and then returned to clients. A processor contains the logic for loading models and making predictions upon requests. PAI supports general processors, such as 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 without the need to use an SDK. To develop a processor, you only need to define the relevant classes and functions. This facilitates on-premises debugging.

4.1.8.16.2. Processors in C or C++

This topic describes how to develop custom processors in C or C++.

If you want to develop a processor by using C or C++, you must define the initialize() and Process() functions. The initialize() function is used to load the model during the initialization process. The Process() function is used to process service calls and return the processing results to clients. You can declare the two functions based on the following information:

- initialize()

```
void *initialize(const char *model_entry, const char *model_config, int *state)
```

Parameter	Type	Description
model_entry	Input parameter	The entry file of the model package. This parameter corresponds to the model_entry field in the configuration file when the service is created. You can specify a file name such as randomforest.pmml, or a directory such as ./model.
model_config	Input parameter	The custom configuration information of the model. This parameter corresponds to the model_config field in the configuration file when the service is created.
state	Output parameter	The status of model loading. If the return value is 0, the model is loaded. Otherwise, the model fails to be loaded.
-	Return value	The memory address of the model, which supports all types. The model is specified in the model variable.

- Process()

```
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	The model memory address that is returned by the initialize() function.
input_data	Input parameter	The input data, which can be a string or of the BINARY type.
input_size	Output parameter	The length of the input data.
output_data	Output parameter	The data that is returned by the processor. The heap memory must be allocated for the data. The model releases the memory as configured.
output_size	Output parameter	The length of the data that is returned by the processor.
-	Return value	If 0 or 200 is returned, the request is successful. Otherwise, an HTTP error code can be returned. If an undefined HTTP status code is returned, it is automatically converted to http 400 error.

Example for developing a processor in C or C++

In the following sample code, no model data is loaded. The prediction service returns the user 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;
    }
}
```

In the following sample code, the processor does not read model information and returns the input data without changes. You can compile the input data as an SO file based on the following Makefile:

```
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 other dependent SO files exist, you must package both the dependent SO files and the SO file that is generated after compilation and then deploy the package as a processor.

4.1.8.16.3. Processors in Java

This topic describes the prototype of the processors in Java.

To develop a processor by using Java, you must define only one class. In addition to the constructor, this class requires only the Load() and Process() functions. The following information specifies the prototype of the class:

```
package com.alibaba.eas;
import java.util.*;
public class TestProcessor {
  public TestProcessor(String modelEntry, String modelConfig) {
    /* Pass the model file name for initialization.*/
  }
  public void Load() {
    /* Load model information based on the model name.*/
  }
  public String Process(String input) {
    /* /* Predict the input data and return the prediction result. The input data and output data can only be strings* /
  }
  public static void main(String[] args) {
    /* /* The main function is optional, and its class functions can be verified on an on-premises standalone machine.* /
  }
}
```

If an exception occurs, the framework captures the exception and returns the message in the exception as an error message to the client. In addition, the HTTP status code 400 is returned. You can also capture exceptions and return corresponding error messages, as shown in the following example:

```
try{
} catch (com.alibaba.fastjson.JSONException e) {
  throw new RuntimeException("bad json format, " + e.getMessage());
}
```

4.1.9. Automatic parameter tuning with AutoML (must be separately activated)

Note The AutoML feature is applicable only to Intel servers, but not to Hygon servers.

4.1.9.1. Automatic parameter tuning with AutoML

This topic describes the automatic parameter tuning feature of AutoML.

Parameter

1. Log on to the PAI 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 on to the PAI 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.

4.1.9.2. Parameter tuning methods

AutoML supports four parameter tuning methods.

Evolutionary Optimizer

Principle:

1. Randomly selects A parameter candidate sets. 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 as the standard deviation range around these parameters to explore new parameter sets. R indicates the **convergence coefficient**. 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 based on the preceding logic until the optimal parameter set is found. M indicates the **number of explorations**.

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 N/2 - 1. If the result of N/2 - 1 is a decimal number, the number is rounded up.

Parameter	Description
Data Splitting Ratio	You can set this parameter to split input data sources into training and evaluation sets. A value of 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 higher the accuracy, the more the consumed computing resources. 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 higher the exploration accuracy, the more the consumed computing resources. This parameter must be set to a positive integer in the range of 1 to 10.
Convergence Coefficient	You can set this parameter to adjust the exploration ranges (R times the standard deviation range search). The smaller the parameter value, the faster the convergence. However, optimal parameters may be missed if the convergence is fast. Valid values: 0.1 to 1. The value can contain only one decimal place.

> **Note** You must enter the tuning range for each parameter. If the current parameter range is not specified, the parameter range is specified 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 and then sorts the output models. M indicates the **number of iterations**.

Parameter	Description
Data Splitting Ratio	You can set this parameter to split input data sources into training and evaluation sets. A value of 0.7 indicates that 70% of the data is used for model training and 30% for evaluation.
Iterations	The number of searches in the specified range. Valid values: 2 to 50.

Note You must enter the tuning range for each parameter. If the current parameter range is not specified, the parameter range is specified by default.

Grid Search

Principle:

1. Splits the value range of each parameter into N segments. N indicates the **number of split grids**.
2. Randomly selects a value from the N segments. If M parameters exist, N^M parameter sets can be combined.
3. Generates N^M models by training based on the N^M parameter sets and then sorts the models.

Parameter	Description
Data Splitting Ratio	You can set this parameter to split input data sources into training and evaluation sets. A value of 0.7 indicates that 70% of the data is used for model training and 30% for evaluation.
Grids	The number of grids after splitting. Valid values: 2 to 10.

Note You must enter the tuning range for each parameter. If the current parameter range is not specified, the parameter range is specified by default.

Custom Parameters

The following figure shows the configurations if you use the Custom Parameters method.

- You can enumerate parameter candidate sets. Then, the system helps score all the combinations of the candidate sets.
- You can define enumeration ranges. Separate parameters with commas (,). If the ranges are not specified, the default ranges of parameters are tuned.

4.1.10. OpenAPI

4.1.10.1. Query PMML models

This topic how to use the ListPMMLModels operation to query Predictive Model Markup Language (PMML) models.

Operation name

ListPMMLModels

Description

You can call this operation to query the PMML models that belong to a specified owner in a specified project. You can also query the PMML models that are generated from a specified experiment.

Request parameters

Parameter	Type	Required	Example	Description
OnwerId	String	Yes	11769368777159105	The UID of a model owner.
ProjectId	Long	Yes	10009	The ID of the project to which a model belongs.
ExperimentId	Long	No	4294	The ID of the experiment from which a model is generated.
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 from which the models are generated.
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 that is generated by the corresponding algorithm.
CreateTime	Date	The time when the model was generated.

Parameter	Type	Description
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 operation that you want to perform.
AccessKeyId	String	The AccessKey ID provided to you by Apsara Stack.
Signature	String	The signature string.
SignatureMethod	String	The signature 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>0a9481861569500365643474 3d0059</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": "Random forest1-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": "0a9481861569500365643474 3d0059",
    "ErrMsg": "Successful",
```

```
    "ErrCode": "success"
  }
}
```

4.1.10.2. Query the details of a PMML model

This topic describes the API operation that is used to query the details of a PMML model.

API operation

DescribePMMLMode

Description

You can call this operation to query the details of 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. For more information about the structure of each element, see the sample code.
Model	ModelInfo	The details of a 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 that is 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.

Parameter	Type	Description
UpdateTime	Date	The time when the model was last 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 request

```
http://pop.pai.idst.inter.env8d.com/?Action=DescribePMMLMode&ModelId=6200&<Common request parameters
>
```

Sample success response

```
<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>0a9441531569499268686817 3d0065</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": "0a9441531569499268686817 3d0065",
    "ErrMsg": "Successful",
    "ErrCode": "success"
  }
}
```

4.1.10.3. Download PMML models

4.1.10.3.1. Generate a download URL for a model

This topic describes how to use the GeneratePMMLModelUrl operation to generate a download URL for a model.

Operation name

GeneratePMMLModelUrl

Description

You can call this operation to generate a Predictive Model Markup Language (PMML) download URL for a model in asynchronous mode. After you call this operation, the system starts a task in the background to generate a download URL and returns the JobId parameter. The system uses the JobId parameter to query the execution status of the task. If the task is successful, a download URL, which is an Object Storage Service (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 task that is used to generate a download URL.

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>0a948186156950130543566 97d0059</RequestId>
<ErrMsg>Successful</ErrMsg>
<ErrCode>success</ErrCode>
</GeneratePMMLModelUrlResponse>{
  "GeneratePMMLModelUrlResponse": {
    "Data": {
      "JobId": "
asynUploadModel2Oss_395fd769-1568-4015-8aa8-a56b2e0da11c
"
    },
    "RequestId": "0a948186156950130543566 97d0059",
    "ErrMsg": "Successful",
    "ErrCode": "success"
  }
}
```

4.1.10.3.2. Poll the status of a download URL generation task

This topic describes the API operation that is used to poll the status of a download URL generation task.

API operation

QueryAsynJobStatus

Description

You can call this operation to poll the status of a download URL generation task and obtain the generated OSS URL for you to download models when the task is successful.

Request parameters

Parameter	Type	Required	Example	Description
JobId	String	Yes	asynUploadModel2Oss_395fd769-1568-4015-8aa8-a56b2e0da11c	The ID of the download URL generation task. The ID is returned by 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 return data in a structure.

Parameter	Type	Description
Status	String	The status of the download URL generation task. Valid values: done, failed, and running.
Info	String	A URL is returned if the task is successful. An error message is returned if the task fails.

Sample request

```
http://pop.pai.idst.inter.env8d.com/?Action=QueryAsynJobStatus&JobId=asynUploadModel2Oss_395fd769-1568-4015-8aa8-a56b2e0da11c&<Common request parameters>
```

Sample success response

```
<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_v0_m_0-%E9%9A%8F%E6%9C%BA%E6%A3%AE%E6%9E%971-AUC-0.pmml?Expires=1569501906&OSSAccessKeyId=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_v0_m_0-%E9%9A%8F%E6%9C%BA%E6%A3%AE%E6%9E%971-AUC-0.pmml?Expires=1569501906&OSSAccessKeyId=yMPdrqaNstzXKsNY&Signature=uRM1OFDZeDNzZo%2BjG87s09uUd6****
"
    },
    "RequestId": "0a9481861569501342195757 9d0059",
    "ErrMsg": "Successful",
    "ErrCode": "success"
  }
}
```

4.1.10.4. SDK

This topic describes how to use Alibaba Cloud SDK.

Alibaba Cloud SDK

```
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 cases

Example: Call a remote procedure call (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 DefaultAcsClient instance 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 specify its 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 a 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 DefaultAcsClient instance 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 specify its 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 parameter must be in the pop.${pai_domain} format. Set the pai_domain parameter to the domain of the Machine Learning Platform for AI console.

The Version parameter must be set to 2019-09-25.

The Action parameter specifies an API operation.

The QueryParameter parameter 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", "***");
```

4.1.11. SDK Reference

4.1.11.1. Overview

Machine Learning Platform for AI (PAI) provides EasyVision, which is an enhanced algorithm framework for visual intelligence. EasyVision provides a variety of features for model training and prediction. You can use EasyVision to train and apply computer vision (CV) models for your CV applications.

Background information

The rapid development of deep learning technologies promotes large-scale commercial use of CV. As CV developers, you may encounter the following difficulties when you use deep learning technologies to build CV models:

- High costs are required to develop and debug deep learning algorithms.
- Models are frequently updated. You must spend much time understanding how the models work and other details of the models.
- To train algorithms and improve the inference performance, you must master professional and systematic knowledge.
- High costs are required for data annotation.
- To use open source algorithms in PAI, you must spend time learning and rebuilding the algorithms.

To overcome the preceding difficulties, PAI provides EasyVision, which is a simple and easy-to-use algorithm framework that allows you to train models. You can use EasyVision to build and apply CV models with ease.

Benefits

EasyVision has the following benefits:

- Ease of use

  EasyVision supports pluggable API operations that can be called to complete various tasks of various modules. In addition, EasyVision provides rich features, such as data I/O, preprocessing, model training, and offline prediction, to support the entire modeling process. You can use EasyVision in multiple modules such as Machine Learning Studio or Data Science Workshop (DSW) of PAI.

- High performance

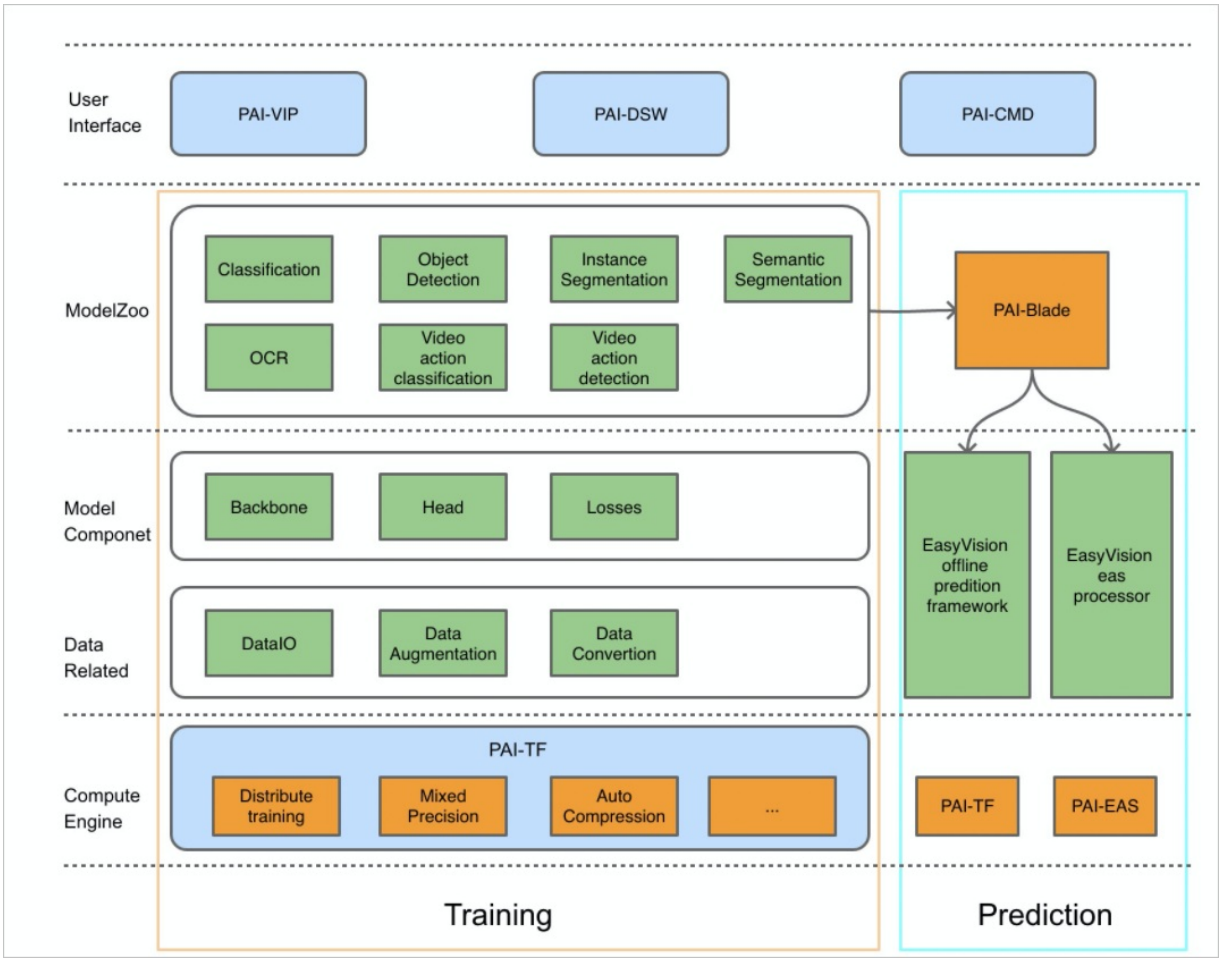
  EasyVision encapsulates a variety of optimization engines of PAI-TensorFlow, such as the optimization engines for compiling, distributed training, and mixed precision training. You can use these optimization engines to improve the system performance by configuring the related configuration files. You can also use EasyVision in open source TensorFlow systems.

- Rich models

  EasyVision provides a variety of models, such as the optical character recognition (OCR) model. The models are trained based on open source datasets. This reduces the development and training costs.

Architecture

EasyVision builds a model zoo with plenty of models. In addition, EasyVision provides a variety of model training and prediction capabilities and is compatible with the commands, Video Intelligence Platform (VIP), and DSW of PAI. This way, EasyVision can meet various modeling requirements. EasyVision uses a distributed pipeline architecture for offline prediction. This is a flexible and highly available architecture that allows EasyVision to process hundreds of millions of data records offline within a short period. Models can be used to make predictions in Elastic Algorithm Service (EAS) of PAI. The system and model optimization features of PAI allow you to make predictions with fewer parameters more efficiently. In addition, you can use EasyVision to customize operations of model training and prediction. This way, you can reuse existing features and optimize models. The following figure shows the architecture of EasyVision.



Features

- Usability

  You may have different requirements on model training and prediction. For example, you may want to train models by performing simple operations, run model training and prediction tasks as scheduled, and reuse existing models and algorithms. To meet these requirements, EasyVision is compatible with the commands, VIP, and DSW of PAI.

- Performance optimization

  EasyVision optimizes distributed training based on PAI-TensorFlow. EasyVision allows you to train a model on one or more multi-GPU servers. EasyVision also improves the inference performance, including graph optimization and model compression.

- Connection to Smart Labeling of PAI

EasyVision is connected to Smart Labeling of PAI, which is used to label data. PAI provides a conversion tool to convert files that contain labeled data to TFRecord files. You can use the TFRecord files to train EasyVision models. In addition, EasyVision provides rich data enhancement modules to dynamically inject data during training.

- Efficient offline predictions

  EasyVision allows you to use multiple servers to concurrently make predictions. Each server separately processes data. This way, you can use offline data to make predictions based on the models that are trained by EasyVision. Each processing job in a prediction task can be run in an accelerated manner by using multiple threads on multiple servers. All jobs are asynchronously run one by one. This improves the processing efficiency. You can also customize jobs.

- Connection to EAS

  A SavedModel file is generated during training. You can use the SavedModel file in your own system or EAS to make online predictions. EasyVision provides a processor that supports the powerful online prediction capabilities of EAS. You can use this processor to process data in real time after you specify the model information such as the endpoint and type in the configuration file.

4.1.11.2. Quick start

This topic describes how to install EasyVision, which is an enhanced algorithm framework provided by Machine Learning Platform for AI (PAI). This topic also describes how to use EasyVision to train computer vision (CV) models.

Prerequisites

Python and TensorFlow are installed. For more information about the limits on the versions of Python and TensorFlow, see Limits.

Limits

You can install and use EasyVision only by using Python and TensorFlow of the following versions:

- Python 2.7, or Python 3.4 or later
- TensorFlow 1.8 or later, or PAI-TensorFlow

Procedure

1. Prepare the test data.
   i. Use one of the following methods to download the Pascal dataset to the current directory:

   ```
   # Method 1: Use osscmd.
   osscmd downloadallobject oss://pai-vision-data-hz/data/voc0712_tfrecord/ data/voc0712_tfrecord --host=oss-cn-zhangjiakou.aliyuncs.com
   # Method 2: Use ossutil. To use ossutil to download the Pascal dataset, you must set the host parameter to oss-cn-zhangjiakou.aliyuncs.com in the configuration file.
   ossutil  cp -r  oss://pai-vision-data-hz/data/voc0712_tfrecord/ data/voc0712_tfrecord
   ```

   ii. Create a folder in the current directory and download the ResNet50 pre-trained model to the folder.

   ```
   mkdir -p pretrained_models/
   ossutil cp -r oss://pai-vision-data-hz/pretrained_models/resnet_v1d_50/ pretrained_models/resnet_v1d_50
   ```

2. Use one of the following methods to start a training task:
   - Use the configuration file

   ```
   import easy_vision
   easy_vision.train_and_evaluate(easy_vision.RFCN_SAMPLE_CONFIG)
   ```

- Set parameters

```
import easy_vision
param_config = """
  --model_type RFCN
  --backbone  resnet_v1d_50
  --num_classes 20
  --model_dir experiments/pascal_voc/resnet50_rfcn_model
  --train_data data/voc0712_tfrecord/voc0712_part_*.tfrecord
  --test_data data/voc0712_tfrecord/VOC2007_test.tfrecord
  --num_test_example 2
  --train_batch_size 32
  --test_batch_size 1
  --image_min_sizes 600
  --image_max_sizes 1024
  --lr_type exponential_decay
  --initial_learning_rate 0.001
  --decay_epochs 20
  --staircase true"""
easy_vision.train_and_evaluate_with_param_config(param_config)
```

- Use multiple servers

  You can use multiple servers to train the model. Make sure that each server has at least two GPUs. If you use this method, you must start the following child processes:

  - ps: the parameter server.
  - master: the master that writes summaries, saves checkpoints, and periodically evaluates the model.
  - worker: the worker that processes specific data.

  Run the following code to start a training task:

```
#-*- encoding:utf-8 -*-
import multiprocessing
import sys
import os
import easy_vision
import json
import logging
import subprocess
import time
# train config under distributed settings
config=easy_vision.RFCN_DISTRIBUTE_SAMPLE_CONFIG
print('config path: %s' % config)
# The configuration of the cluster.
TF_CONFIG={'cluster':{
             'ps': ['localhost:12921'],
             'master': ['localhost:12922'],
             'worker': ['localhost:12923']
            }
          }
def job(task, gpu):
  task_name = task['type']
  # redirect python log and tf log to log_file_name
  # [logs/master.log, logs/worker.log, logs/ps.log]
  log_file_name = "logs/%s.log" % task_name
  TF_CONFIG['task'] = task
  os.environ['TF_CONFIG'] = json.dumps(TF_CONFIG)
  os.environ['CUDA_VISIBLE_DEVICES'] = gpu
  train_cmd = 'python -m easy_vision.python.train_eval --pipeline_config_path %s' % config
```

```
  logging.info('%s > %s 2>&1 ' % (train_cmd, log_file_name))
  with open(log_file_name, 'w') as lfile:
    return subprocess.Popen(train_cmd.split(' '), stdout= lfile, stderr=subprocess.STDOUT)
if __name__ == '__main__':
  procs = {}
  # start ps job on cpu
  task = {'type':'ps', 'index':0}
  procs['ps'] = job(task, '')
  # start master job on gpu 0
  task = {'type':'master', 'index':0}
  procs['master'] = job(task, '0')
  # start worker job on gpu 1
  task = {'type':'worker', 'index':0}
  procs['worker'] = job(task, '1')
  num_worker = 2
  for k, proc in procs.items():
    logging.info('%s pid: %d' %(k, proc.pid))
  task_failed = None
  task_finish_cnt = 0
  task_has_finished = {k:False for k in procs.keys()}
  while True:
    for k, proc in procs.items():
      if proc.poll() is None:
        if task_failed is not None:
          logging.error('task %s failed, %s quit' % (task_failed, k))
          proc.terminate()
          if k != 'ps':
            task_has_finished[k] = True
            task_finish_cnt += 1
          logging.info('task_finish_cnt %d' % task_finish_cnt)
      else:
        if not task_has_finished[k]:
          #process quit by itself
          if k != 'ps':
            task_finish_cnt += 1
            task_has_finished[k] = True
          logging.info('task_finish_cnt %d' % task_finish_cnt)
          if proc.returncode != 0:
            logging.error('%s failed' %k)
            task_failed = k
          else:
            logging.info('%s run successfuly' % k)
    if task_finish_cnt >= num_worker:
      break
    time.sleep(1)
```

3. Use TensorBoard to monitor the training task.

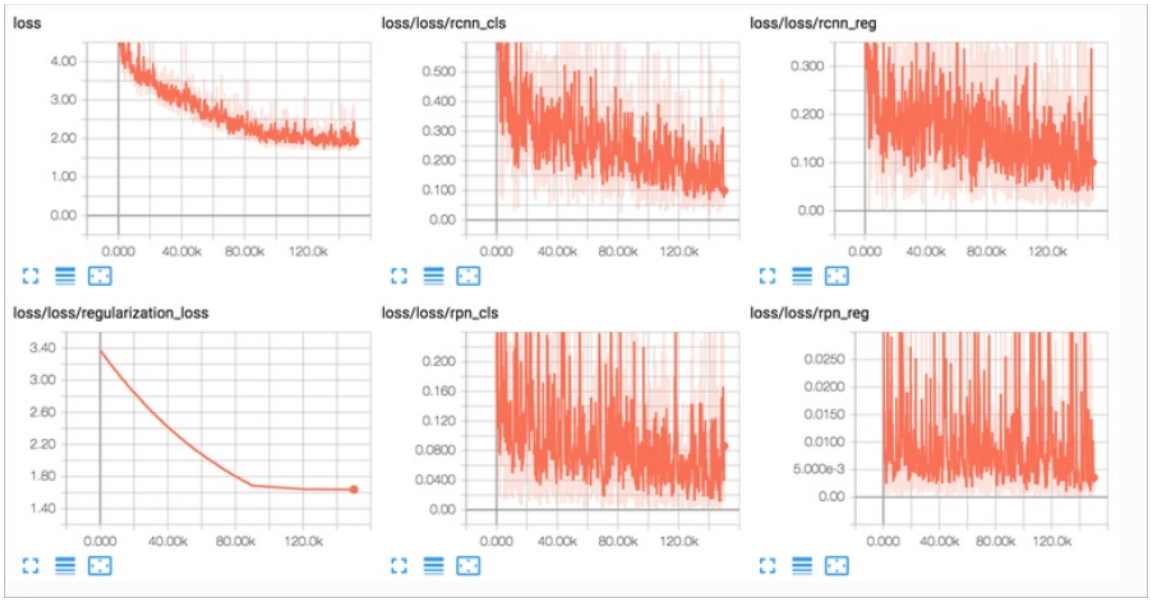
   The checkpoints and event files of the model are saved in the model directory. You can run the following command in TensorBoard to view the loss and mean average precision (mAP) of the training:

   ```
   tensorboard --port 6006 --logdir <model_dir>  [ --host 0.0.0.0 ]
   ```

   Replace <model_dir> with the directory of your model.

   In TensorBoard, you can view the following information:

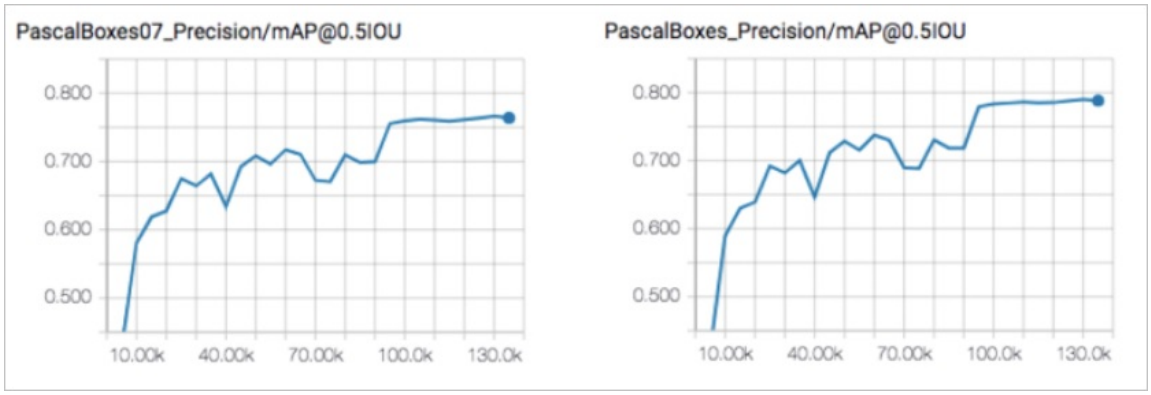
   - Training loss



TensorBoard provides the following metrics about the training loss:

- loss: the total loss of the training.
- loss/loss/rcnn_cls: the classification loss.
- loss/loss/rcnn_reg: the regression loss.
- loss/loss/regularization_loss: the regularization loss.
- loss/loss/rpn_cls: the classification loss of region proposal network (RPN).
- loss/loss/rpn_reg: the regression loss of RPN.

- Test mAP



PascalBoxes07 and PascalBoxes are used as metrics to calculate the test mAP, as shown in the preceding figure. PascalBoxes07 is commonly used in studies.

4. Test and evaluate the model.

   After the training task is complete, you can test and evaluate the trained model.

   - Use other datasets to test the model. Then, check the detection result of each image.

```
import easy_vision
test_filelist = 'path/to/filelist.txt' # each line is a image file path.
detect_results = easy_vision.predict(easy_vision.RFCN_SAMPLE_CONFIG, test_filelist=test_fileli
st)
```

   The detection result of each image is returned in the detect_results parameter in the format of [detection_boxes, box_probability, box_class]. In the format, detection_boxes and box_class indicate the

location and category of the detected object. box_probability indicates the confidence level of the detection result.

- Evaluate the trained model.

```
import easy_vision
eval_metrics = easy_vision.evaluate(easy_vision.RFCN_SAMPLE_CONFIG)
```

The eval_metrics parameter indicates evaluation metrics, including PascalBoxes07, PascalBoxes, global_step, and the following loss metrics: loss, loss/loss/rcnn_cls, loss/loss/rcnn_reg, loss/loss/rpn_cls, loss/loss/rpn_reg, and loss/loss/total_loss.

5. Export the model.

EasyVision allows you to export the model as a SavedModel file. Then, you can use the SavedModel file in Python or C++ to make predictions.

Run the following code to export the model as a SavedModel file:

```
import easy_vision
easy_vision.export(export_dir, easy_vision.RFCN_SAMPLE_CONFIG, checkpoint_path)
```

After you run the preceding code, a model directory is created in the export_dir directory. The name of the model directory contains the UNIX timestamp that indicates the time when the directory is created. All checkpoints of the model are exported to a SavedModel file in the model directory.

6. Use the model to make predictions.

EasyVision provides the predictor in Python for SavedModel files. You can run the following code to start model prediction based on a detection model. For more information about the predictors for classification, segmentation, and optical character recognition (OCR), see Offline prediction.

```
import easy_vision
detector = ev.Detector(saved_model_path)
image = np.zeros([640, 480, 3],  dtype=np.float32)
output_dict = detector.predict([image])
```

The prediction result is returned in the output_dict parameter. The parameter value is in the format of a JSON list. The number of elements in the list is equal to the number of images in the datasets for testing. The following code shows the sample response in the JSON format:

```
{
  "detection_boxes": [[243.5308074951172, 197.69570922851562, 385.59625244140625, 247.7247772216
797], [292.1929931640625, 114.28043365478516, 571.2748413085938, 165.09771728515625]],
  "detection_scores": [0.9942291975021362, 0.9940272569656372],
  "detection_classes": [1, 1],
  "detection_classe_names": ["text", "text"]
}
```

7. Evaluate the SavedModel file.

Run the following code to evaluate the exported SavedModel file. All metrics of the model are contained in the evaluation result file and logs.

```
from easy_vision.python.main import predictor_evaluate
predictor_evaluate(predictor_eval_config)
```

In the preceding code, predictor_eval_config specifies the .proto file that is used for the evaluation. For more information, see SavedModel file evaluation.

4.1.11.3. Model export and prediction

EasyVision allows you to export a trained model as a SavedModel file. Then, you can use the SavedModel file in Python or C++ to make predictions. This topic describes how to export a model as a SavedModel file and the input and output of model prediction. This topic also provides an example on how to make offline prediction.

Model export

You can run the following Python code to export a model:

```
import easy_vision
easy_vision.export(export_dir, pipeline_config_path, checkpoint_path)
```

By default, If you do not specify the checkpoint_path parameter, the checkpoints in the model directory specified by the model_dir parameter under the configuration path specified by the pipeline_config_path parameter are used.

Input and output of model prediction

To make predictions based on a SavedModel file, you must obtain the TensorFlow input and output operations.

Parameters in a placeholder in the input

Parameter	Description	Dimension	Type
image	The batched tensor of the image. The channels are in the RGB sequence.	[batch_size, None, None, 3]	tf.uint8
true_image_shape	The size of the image. The last dimension must be in the sequence of [height, width, channel], such as [[224,224,3], [448, 448, 3]].	[batch_size, 3]	tf.int32

The value of the batch_size parameter is specified in export_config in *pipeline_config* when you export the model. If the batch_size parameter is set to -1, its value is dynamically assigned. Only classification models support dynamic value assignment of the batch_size parameter.

Parameters in an output tensor

The output of a prediction model is in the format of a JSON list. The number of elements in the list is equal to the number of images in the input datasets. The following part provides the sample outputs of different types of models in the JSON format and describes the output parameters:

- **feature_extractor**

  The following code shows a sample output:

  ```
  {"feature": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.4583122730255127, 0.0]}
  ```

  Parameters

Parameter	Description	Dimension	Type
feature	The extracted features. The coordinates of each feature are specified in the following order: [top, left, bottom, right].	[feature_dim]	FLOAT

- **classifier**

  The following code shows a sample output:

```
{
"class": 3,
"class_name": "coho4",
"class_probs": {"coho1": 4.028851974258174e-10,
          "coho2": 0.48115724325180054,
          "coho3": 5.116515922054532e-07,
          "coho4": 0.5188422446937221}
}
```

Parameters

Parameter	Description	Dimension	Type
class	The ID of the category.	[]	INT32
class_name	The name of the category.	[]	STRING
class_probs	The probabilities for all categories.	[num_classes]	Dict{Key: STRING, Value: FLOAT}

- **multilabel_classifier**

  The following code shows a sample output:

```
{
"class": [3, 4],
"class_names": ["coho3", "coho4"],
"class_probs": {"coho1": 4.028851974258174e-10,
          "coho2": 0.10115724325180054,
          "coho3": 0.6188422446937221,
          "coho4": 0.5188422446937221}
}
```

  Parameters

Parameter	Description	Dimension	Type
class	The ID of the category.	[None]	INT32
class_names	The name of the category.	[None]	STRING
class_probs	The probabilities for all categories.	[num_classes]	Dict{Key: STRING, Value: FLOAT}

- **detector** (supports instance segmentation)

  The following code shows a sample output:

```
{
  "detection_boxes": [[243.5308074951172, 197.69570922851562, 385.59625244140625, 247.724777221679
7], [292.1929931640625, 114.28043365478516, 571.2748413085938, 165.09771728515625]],
  "detection_scores": [0.9942291975021362, 0.9940272569656372],
  "detection_classes": [1, 1],
  "detection_classe_names": ["text", "text"]
 }
```

  Parameters

Parameter	Description	Dimension	Type
detection_boxes	The bounding boxes that mark the recognized objects. The coordinates of each bounding box are specified in the following order: [top, left, bottom, right].	[num_detections, 4]	FLOAT
detection_scores	The probabilities that the objects are recognized.	num_detections	FLOAT
detection_classes	The IDs of the categories to which the objects belong.	num_detections	INT
detection_class_names	The names of the categories to which the objects belong.	num_detections	STRING
detection_masks	Optional. The segmentation masks of the objects.	[num_detection, image_height, image_width]	FLOAT
detection_keypoints	Optional. The landmarks of the objects.	[num_detection, num_keypoints, 2]	FLOAT
detection_roi_features	Optional. The local features of the objects.	[num_detection, roi_height, roi_width, channels]	FLOAT

- **detector_with_rpn**

  The following code shows a sample output:

```
{
  "proposal_boxes": [[243.5308074951172, 197.69570922851562, 385.59625244140625, 247.7247772216797
], 243.5308074951172, 197.69570922851562, 385.59625244140625, 247.7247772216797],
  "proposal_scores": [0.88, 0.56],
  "detection_boxes": [[243.5308074951172, 197.69570922851562, 385.59625244140625, 247.724777221679
7], [292.1929931640625, 114.28043365478516, 571.2748413085938, 165.09771728515625]],
  "detection_scores": [0.9942291975021362, 0.9940272569656372],
  "detection_classes": [1, 1],
  "detection_classe_names": ["text", "text"]
 }
```

  Parameters

Parameter	Description	Dimension	Type
proposal_boxes	The areas that cover the proposals.	[num_proposal, 4]	FLOAT
proposal_scores	The scores of the proposals.	num_proposal	FLOAT

Parameter	Description	Dimension	Type
detection_boxes	The bounding boxes that mark the recognized objects. The coordinates of each bounding box are specified in the following order: [top, left, bottom, right].	[num_detections, 4]	FLOAT
detection_scores	The probabilities that the objects are recognized.	num_detections	FLOAT
detection_classes	The IDs of the categories to which the objects belong.	num_detections	INT
detection_class_names	The names of the categories to which the objects belong.	num_detections	STRING

- segmentor

  The following code shows a sample output:

```
{
  "probs" : [[[0.8, 0.8], [0.6, 0.7]],[[0.8, 0.5], [0.4, 0.3]]],
  "preds" : [[[1,1], [0, 0]], [[0, 0], [1,1]]]
}
```

  Parameters

Parameter	Description	Dimension	Type
probs	The probabilities that the pixels obtained after segmentation belong to specific categories.	[output_height, output_width, num_classes]	FLOAT
preds	The IDs of the categories to which the pixels obtained after segmentation belong.	[output_height, output_widths]	INT

- text_detector

  The following code shows a sample output:

```
{
  "detection_keypoints": [[[243.57516479492188, 198.84210205078125], [243.91038513183594, 247.62425231933594], [385.5513916015625, 246.61660766601562], [385.2197570800781, 197.79345703125]], [[292.2718200683594, 114.44700622558594], [292.2237243652344, 164.684814453125], [571.1962890625, 164.931640625], [571.2444458007812, 114.67433166503906]]],
  "detection_boxes": [[243.5308074951172, 197.69570922851562, 385.59625244140625, 247.7247772216797], [292.1929931640625, 114.28043365478516, 571.2748413085938, 165.09771728515625]],
  "detection_scores": [0.9942291975021362, 0.9940272569656372],
  "detection_classes": [1, 1],
  "detection_classe_names": ["text", "text"],
   "image_shape": [1024, 968, 3]
 }
```

Parameters

Parameter	Description	Dimension	Type
detection_boxes	The bounding boxes that mark the recognized text areas. The coordinates of each bounding box are specified in the following order: [top, left, bottom, right].	[num_detections, 4]	FLOAT
detection_scores	The probabilities that the text areas are detected.	num_detections	FLOAT
detection_classes	The IDs of the categories to which the text areas belong.	num_detections	INT
detection_class_names	The names of the categories to which the text areas belong.	num_detections	STRING
detection_keypoints	The four vertices of each text area that is detected. The coordinates of each vertex are specified in the (y, x) format.	[num_detections, 4, 2]	FLOAT
image_shape	The size of the input image. The height and width are in pixels.	[3]. The three numbers indicate the image height, image width, and the number of channels.	List

- **text_recognizer**

  The following code shows a sample output:

```
{
  "sequence_predict_ids": [1,2,2008,12],
  "sequence_predict_texts": "This is an example",
  "sequence_probability": 0.88
}
```

  Parameters

Parameter	Description	Dimension	Type
sequence_predict_ids	The ID of the category to which a single line of the recognized text belongs.	[text_length]	INT
sequence_predict_texts	The recognition result of each single-line text.	[]	STRING
sequence_probability	The probability that each single-line text is recognized.	[]	FLOAT

- **text_spotter/text_pipeline_predictor**

The following code shows a sample output:

```
{
  "detection_keypoints": [[[243.57516479492188, 198.84210205078125], [243.91038513183594, 247.62425231933594], [385.5513916015625, 246.61660766601562], [385.2197570800781, 197.79345703125]], [[292.2718200683594, 114.44700622558594], [292.2237243652344, 164.684814453125], [571.1962890625, 164.931640625], [571.2444458007812, 114.674331665039906]]],
  "detection_boxes": [[243.5308074951172, 197.69570922851562, 385.59625244140625, 247.7247772216797], [292.1929931640625, 114.28043365478516, 571.2748413085938, 165.09771728515625]],
  "detection_scores": [0.9942291975021362, 0.9940272569656372],
  "detection_classes": [1, 1],
  "detection_classe_names": ["text", "text"],
  "detection_texts_ids" : [[1,2,2008,12], [1,2,2008,12]],
  "detection_texts": ["This is an example", "This is an example"],
  "detection_texts_scores" : [0.88, 0.88],
  "image_shape": [1024, 968, 3]
 }
```

Parameters

name	Description	Dimension	type
detection_boxes	The bounding boxes that mark the recognized text areas. The coordinates of each bounding box are specified in the following order: [top, left, bottom, right].	[num_detections, 4]	FLOAT
detection_scores	The probabilities that the text areas are detected.	num_detections	FLOAT
detection_classes	The IDs of the categories to which the text areas belong.	num_detections	INT
detection_class_names	The names of the categories to which the text areas belong.	num_detections	STRING
detection_keypoints	The four vertices of each text area that is detected. The coordinates of each vertex are specified in the (y, x) format.	[num_detections, 4, 2]	FLOAT
detection_texts_ids	The ID of the category to which a single line of the recognized text belongs.	[num_detections, max_text_length]	INT
detection_texts	The recognition result of each single-line text.	[num_detections]	STRING
detection_texts_scores	The probability that each single-line text is recognized.	[num_detections]	FLOAT

name	Description	Dimension	type
image_shape	The size of the input image. The height and width are in pixels.	[3]. The three numbers indicate the image height, image width, and the number of channels.	List

Offline prediction

EasyVision provides an operation in Python for predictions. You can call this operation to use an exported SavedModel file to make predictions. For more information about the operation, see easy_vision.python.inference. The following code shows examples of offline prediction jobs:

```
import easy_vision as ev
import numpy as np
# Recognition.
saved_model_path = 'xxx/xxx'
classifier = ev.Classifier(saved_model_path)
image = np.zeros([640, 480, 3],  dtype=np.float32)
output_dict = classifier.predict([image])
# Detection.
saved_model_path = 'xxx/xxx'
detector = ev.Detector(saved_model_path)
image = np.zeros([640, 480, 3],  dtype=np.float32)
output_dict = detector.predict([image])
# Text recognition.
saved_model_path = 'xxx/xxx'
text_recognizer = ev.TextRecognizer(saved_model_path)
image = np.zeros([640, 480, 3],  dtype=np.float32)
output_dict = text_recognizer.predict([image])
# Text detection.
saved_model_path = 'xxx/xxx'
text_detector = ev.TextDetector(saved_model_path)
image = np.zeros([640, 480, 3],  dtype=np.float32)
output_dict = text_detector.predict([image])
# End-to-End text recognition.
saved_model_path = 'xxx/xxx'
text_spotter = ev.TextSpotter(saved_model_path)
image = np.zeros([640, 480, 3],  dtype=np.float32)
output_dict = text_spotter.predict([image])
# Basic predictor.
saved_model_path = 'xxx/xxx'
predictor = ev.Predictor(saved_model_path)
image = np.zeros([640, 480, 3],  dtype=np.float32)
image_list = [image for i in range(10)]
batched_images, origin_shapes = predictor.batch(images)
input_data = {
  'image': batched_images,
  'true_image_shape', origin_shapes
}
output_data_dict = predictor.predict(input_data)
```

4.1.11.4. Model library

4.1.11.4.1. Image classification

This topic describes how to use EasyVision that is provided by Machine Learning Platform for AI (PAI) to train an image classification model. In this example, a CIFAR-10 dataset is used to train a resnet_v1d_50 model.

Context

The module of image classification provides multiple major convolutional neural networks (CNNs) for training image classification models. The following types of models are supported:

- Residual Neural Network (ResNet) models
- SE-ResNet models
- Inception models
- VGG models
- MobileNet models
- EfficientNet models
- DarkNet models

The following table describes the performance benchmarks for the preceding models. Benchmark

Model	EasyVisionTop1 Acc	EasyVisionTop5 Acc
resnet_v1a_18	70.94	89.88
resnet_v1a_34	74.26	91.97
resnet_v1_50	75.20	92.20
resnet_v1a_50	77.34	93.64
resnet_v1b_50	77.52	93.76
resnet_v1c_50	77.90	94.12
resnet_v1d_50	79.01	94.51
resnet_v1_101	76.40	92.89
resnet_v1a_101	78.26	93.97
resnet_v1b_101	79.05	94.60
resnet_v1c_101	79.49	94.74
resnet_v1d_101	80.36	95.04
resnet_v1_152	76.80	93.17
resnet_v1a_152	79.16	94.69
resnet_v1b_152	79.52	94.81
resnet_v1c_152	79.93	94.91
resnet_v1d_152	80.48	95.31
SE-Resnet-v1-50	77.51	93.6
SE-Resnet-v1-101	78.28	94.24

Model	EasyVisionTop1 Acc	EasyVisionTop5 Acc
SE-Resnet-v1-152	78.58	94.41
inception-v1	69.8	89.6
inception-v2	73.9	91.8
inception-v3	78.0	93.9
inception-v4	80.2	95.2
mobilenet-v1-1.0_224	70.7	89.5
mobilenet-v2-1.0_224	70.1	89.5
mobilenet-v3-1.0_224	75.4	92.7
efficientnet-b0	0.7750	0.9360
efficientnet-b1	0.7957	0.9437
efficientnet-b2	0.8044	0.9503
efficientnet-b3	0.8188	0.9566
efficientnet-b4	0.8330	0.9640
efficientnet-b5	0.8431	0.9694
efficientnet-b6	0.8477	0.9713
efficientnet-b7	0.8516	0.9725
efficientnet-b8	0.8545	0.8524
darknet53	0.7664	0.9342
cspdarknet53-mish	0.7794	0.9425

Step 1: Prepare data

1. Download a dataset.

   EasyVision provides the CIFAR-10 dataset that is converted to a TFRecord file. You can run the following command to download the CIFAR-10 dataset to the *data* folder:

   ```
   ossutil cp -r oss://pai-vision-data-hz/data/cifar10/ data/cifar10
   ```

2. Run the following command to download the pretrained ResNet50 model to the *pretrained_models* folder:

   ```
   ossutil cp -r oss://pai-vision-data-hz/pretrained_models/resnet_v1d_50/ pretrained_models/resnet_v1d_50
   ```

Step 2: Train the model

EasyVision allows you to train a model by referencing a configuration file or setting parameters.

- **Reference a configuration file**

You can set the parameters that are used for model training and evaluation in a configuration file and reference the file in Python code. For more information about sample configuration files, see Image classification. If you want to know the configuration details, submit a ticket to contact the customer service representatives of PAI. Run the following Python code that references a configuration file to start model training and evaluation:

```
import easy_vision
easy_vision.train_and_evaluate(easy_vision.RESNET50_FINETUNE_PRETRAINED_MODELS_SAMPLE_CONFIG)
```

If you use a custom configuration file, you must modify the preceding code by replacing `easy_vision.RESNET50_FINETUNE_PRETRAINED_MODELS_SAMPLE_CONFIG` with the path of the custom configuration file.

- **Set parameters**

  You can set the parameters that are used for model training and evaluation in Python code. Run the following Python code that contains related parameters to start model training and evaluation:

```
import easy_vision
param_config = """
    --model_type Classification
  --backbone  resnet_v1d_50
  --num_classes 10
  --num_epochs 1
  --model_dir experiments/cifar10/resnet50_imagenet_pretrain
  --use_pretrained_model true
  --train_data data/cifar10/cifar10_train_part_*.tfrecord
  --test_data data/cifar10/cifar10_test.tfrecord
  --train_batch_size 96
  --test_batch_size 96
  --image_size 224
  --lr_type manual_step
  --initial_learning_rate 0.001
  --learning_rates 0.0003 0.0001
  --decay_epochs 35 40
  --save_checkpoint_epochs 5
  --staircase true"""
easy_vision.train_and_evaluate_with_param_config(param_config)
```

In the preceding code, the param_config structure contains the parameters that are used for model training and evaluation. The format of the param_config structure is the same as that of the ArgumentParser() object in Python. The following table describes the parameters in the param_config structure.

> **Note** Do not enclose a parameter value of the STRING type in double quotation marks (") or single quotation marks (').

Parameters in the param_config structure

Parameter	Required	Description	Value type or sample value	Default value
model_type	Yes	The type of the model to train. To train an image classification model, set this parameter to Classification.	STRING	N/A

Parameter	Required	Description	Value type or sample value	Default value
backbone	No	The name of the backbone network that is used by the model. Valid values: ○ vgg_16 ○ vgg_19 ○ inception_v1 ○ inception_v2 ○ inception_v3 ○ inception_v4 ○ mobilenet_v1 ○ mobilenet_v2 ○ mobilenet_v3 ○ resnet_v1_50 ○ resnet_v1_101 ○ resnet_v1_152 ○ resnet_v2_50 ○ resnet_v2_101 ○ resnet_v2_152 ○ resnet_v1a_18 ○ resnet_v1a_34 ○ resnet_v1a_50 ○ resnet_v1a_101 ○ resnet_v1a_152 ○ resnet_v1b_50 ○ resnet_v1b_101 ○ resnet_v1b_152 ○ resnet_v1c_50 ○ resnet_v1c_101 ○ resnet_v1c_152 ○ resnet_v1d_50 ○ resnet_v1d_101 ○ resnet_v1d_152 ○ efficientnet-b0 ○ efficientnet-b1 ○ efficientnet-b2 ○ efficientnet-b3 ○ efficientnet-b4 ○ efficientnet-b5 ○ efficientnet-b6 ○ efficientnet-b7 ○ efficientnet-b8	STRING	inception_v4
weight_decay	No	The value of L2 regularization.	FLOAT	1e-4
num_classes	Yes	The number of categories.	100	N/A

Parameter	Required	Description	Value type or sample value	Default value
image_size	No	The size of the images after they are resized.	INT	224
use_crop	No	Specifies whether to crop images for data enhancement.	BOOL	true
crop_min_area	No	The minimum proportion of the original image area that is occupied by the crop box if you set the use_crop parameter to true.	FLOAT	0.7
eval_each_category	No	Specifies whether to separately evaluate each category.	BOOL	false
optimizer	No	The type of the optimizer. Valid values: ◦ momentum: stochastic gradient descent (SGD) with momentum ◦ adam	STRING	momentum

Parameter	Required	Description	Value type or sample value	Default value
lr_type	No	The policy that is used to adjust the learning rate. Valid values: ◦ exponential_decay: The learning rate is subject to exponential decay. ◦ polynomial_decay: The learning rate is subject to polynomial decay. If you set the lr_type parameter to polynomial_decay, the num_steps parameter is automatically set to the total number of training iterations. The value of the end_learning_rate parameter is automatically set to one thousandth of the value of the initial_learning_rate parameter. ◦ manual_step: The learning rate of each epoch is manually adjusted. If you set the lr_type parameter to manual_step, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rate. You must also set the learning_rates parameter to specify the learning rate of each epoch as needed. ◦ cosine_decay: The learning rate of each epoch is adjusted based on the cosine curve until the learning rate decreases to 0.	STRING	exponential_decay
initial_learning_rate	No	The initial learning rate.	FLOAT	0.01

Parameter	Required	Description	Value type or sample value	Default value
decay_epochs	No	If you set the lr_type parameter to exponential_decay, the decay_epochs parameter is equivalent to the decay_steps parameter of tf.train.exponential.decay. In this case, the decay_epochs parameter specifies the epoch interval at which you want to adjust the learning rate. The system automatically converts the value of the decay_epochs parameter to the value of the decay_steps parameter based on the total number of training data entries. Typically, you can set the decay_epochs parameter to half of the total number of epochs. For example, if the total number of epochs is 20, you can set this parameter to 10. If you set the lr_type parameter to manual_step, the decay_epochs parameter specifies the epochs for which you want to adjust the learning rate. For example, a value of 16 18 indicates that you want to adjust the learning rate for the 16th and 18th epochs. Typically, if the total number of epochs is N, you can set the two values of the decay_epochs parameter to 8/10 × N and 9/10 × N.	INTEGER list. Sample value: 20 20 40 60	20
decay_factor	No	The decay rate. This parameter is equivalent to the decay_factor parameter of tf.train.exponential.decay.	FLOAT	0.95
staircase	No	Specifies whether the learning rate changes based on the decay_epochs parameter. This parameter is equivalent to the staircase parameter of tf.train.exponential.decay.	BOOL	true
power	No	The power of the polynomial. This parameter is equivalent to the power parameter of tf.train.polynomial.decay.	FLOAT	0.9

Parameter	Required	Description	Value type or sample value	Default value
learning_rates	No	The learning rate that you want to set for each specified epoch. This parameter is required if you set the lr_type parameter to manual_step. If you want to adjust the learning rate for two epochs, specify two learning rates in the value. For example, if the decay_epoches parameter is set to 20 40, you must specify two learning rates in the learning_rates parameter, such as 0.001 0.0001. This indicates that the learning rate of the 20th epoch is adjusted to 0.001 and the learning rate of the 40th epoch is adjusted to 0.0001. We recommend that you adjust the learning rate to one tenth, one hundredth, and one thousandth of the initial learning rate in sequence.	FLOAT list	N/A
lr_warmup	No	Specifies whether to warm up the learning rate.	BOOL	false
lr_warm_up_epochs	No	The number of epochs for which you want to warm up the learning rate.	FLOAT	1
train_data	Yes	The Object Storage Service (OSS) path of the data that is used to train the model.	oss://path/to/train_*.tfrecord	N/A
test_data	Yes	The OSS path of the data that is evaluated during the training.	oss://path/to/test_*.tfrecord	N/A
train_batch_size	Yes	The size of the data that is used to train the model in the current batch.	INT. Sample value: 32.	N/A
test_batch_size	Yes	The size of the data that is evaluated in the current batch.	INT. Sample value: 32.	N/A
train_num_readers	No	The number of concurrent threads that are used to read the training data.	INT	4
model_dir	Yes	The OSS path of the model.	oss://path/to/model	N/A
pretrained_model	No	The OSS path of the pretrained model. If this parameter is specified, the actual model is finetuned based on the pretrained model.	oss://pai-vision-data-sh/pretrained_models/inception_v4.ckpt	""
use_pretrained_model	No	Specifies whether to use a pretrained model.	BOOL	true

Parameter	Required	Description	Value type or sample value	Default value
num_epochs	Yes	The number of times the data is iterated for the training. A value of 1 indicates that all data is iterated once for the training.	INT. Sample value: 40.	N/A
num_test_example	No	The number of data entries that are evaluated during the training. A value of -1 indicates that all training data is evaluated.	INT. Sample value: 2000.	-1
num_visualizations	No	The number of data entries that can be visualized during the evaluation.	INT	10
save_checkpoint_epochs	No	The epoch interval at which a checkpoint is saved. A value of 1 indicates that a checkpoint is saved each time an epoch is complete.	INT	1
save_summary_epochs	No	The epoch interval at which a summary is saved. A value of 0.01 indicates that a summary is saved each time 1% of the training data is iterated.	FLOAT	0.01
num_train_images	No	The total number of data entries that are used for the training. If you use custom TFRecord files to train the model, this parameter is required.	INT	0
label_map_path	No	The category mapping file. If you use custom TFRecord files to train the model, this parameter is required.	STRING	""

4.1.11.4.2. Object detection

This topic describes how to use EasyVision to train an object detection model. In this example, a Visual Object Classes (VOC) dataset is used and a FasterRCNN model is trained.

Context

EasyVision provides multiple types of models for common object detection tasks that return the position information of the detected objects in images. The following types of models are supported:

- FasterRCNN models
- Single Shot Detector (SSD) models
- You Only Look Once (YOLO) models

The following table describes the expected performance of the preceding models when they are trained by using a VOC dataset or the COCO 2017 dataset.

Benchmark datasets based on VOC datasets

- Train set: the trainvals of VOC 2007 and VOC 2012
- Test set: the test set of VOC 2007

Model name	Input image size	mAP@IOU0.5
faster-rcnn-resnet50	600 × 1024	0.790
faster-rcnn-resnet50-ohem	600 × 1024	0.796
faster-rcnn-resnet50-fpn	600 × 1024	0.798
rfcn-resnet50	600 × 1024	0.764
ssd-vgg16	300 × 300	0.777
ssd-vgg16	512 × 512	0.801
ssd-resnet50	300 × 300	0.749
ssd-resnet50-fpn	300 × 300	0.762

Benchmark datasets based on the COCO 2017 dataset

- Train set: train 118k
- Test set: test 5k

Model name	Input image size	mAP@IOU0.5
faster-rcnn-resnet50	800 × 1333	0.34
ssd-mobilenet-v1	512 × 512	0.21

Step 1: Prepare data

1. Download a dataset.

   EasyVision provides a VOC dataset that is converted to TFRecord files. You can download the dataset to the *data* folder by running the following command:

   ```
   ossutil  cp -r  oss://pai-vision-data-hz/data/voc0712_tfrecord/ data/voc0712_tfrecord
   ```

2. Download the ResNet50 pretrained model to the *pretrained_models* folder by running the following command:

   ```
   ossutil cp -r oss://pai-vision-data-hz/pretrained_models/resnet_v1d_50/ pretrained_models/resnet_v1d_50
   ```

Step 2: Train the model

EasyVision allows you to train a model by referencing a configuration file or setting parameters.

- **Reference a configuration file**

  You can set parameters related to training and evaluation in a configuration file and reference the file in the Python code. For more information about sample configuration files, see Object detection. If you want to know the configuration details, submit a ticket to contact the customer service representatives of Machine Learning Platform for AI (PAI). Run the following Python code that references a configuration file to start training and evaluation:

  ```
  import easy_vision
  easy_vision.train_and_evaluate(easy_vision.FASTER_RCNN_SAMPLE_CONFIG)
  ```

  If you want to use a custom configuration file, replace `easy_vision.FASTER_RCNN_SAMPLE_CONFIG` in the

preceding code with the path of the custom configuration file.

- Set parameters

  You can set the parameters related to training and evaluation in the Python code. Run the following Python code that contains related parameters to start training and evaluation:

```
import easy_vision
param_config = """
  --model_type FasterRCNN
  --backbone  resnet_v1d_50
  --num_classes 20
  --model_dir experiments/pascal_voc/resnet50_frcnn_model
  --train_data data/voc0712_tfrecord/voc0712_part_*.tfrecord
  --test_data data/voc0712_tfrecord/VOC2007_test.tfrecord
  --num_test_example 2
  --train_batch_size 32
  --test_batch_size 1
  --image_min_sizes 600
  --image_max_sizes 1024
  --lr_type exponential_decay
  --initial_learning_rate 0.001
  --decay_epochs 20
  --staircase true"""
easy_vision.train_and_evaluate_with_param_config(param_config)
```

In the preceding code, the param_config parameter contains the parameters that are used for training and evaluation. The format of the param_config parameter is the same as that of the ArgumentParser() object in Python. The following table describes the parameters.

> **Note** Do not enclose a parameter value of the STRING type in double quotation marks (") or single quotation marks (').

Parameters in the param_config structure

Parameter	Required	Description	Value type or sample value	Default value
model_type	Yes	The type of the model to train. Valid values: ○ SSD ○ FasterRCNN ○ RFCN	STRING	N/A
backbone	Yes	The name of the backbone network that is used by the model. Valid values: ○ mobilenet_v1 ○ resnet_v1_50 ○ resnet_v1_101 ○ resnet_v1a_18 ○ resnet_v1a_34 ○ resnet_v1d_50 ○ resnet_v1d_101 ○ vgg16 _reduce_fc	STRING	N/A

Parameter	Required	Description	Value type or sample value	Default value
weight_decay	No	The value of L2 regularization.	FLOAT	1e-4
use_fpn	No	Specifies whether to use Feature Pyramid Network (FPN).	BOOL	false
num_classes	No	The number of categories, excluding background categories.	INT	N/A
anchor_scales	No	The size of the anchor box. The size of the anchor box is the same as that of the input image where the anchor box resides after the image is resized. If you use an SSD model, this parameter is not required. If you use FPN, set this parameter to the size of the anchor box in the layer that has the highest resolution. The total number of layers is five. The size of the anchor box in a layer is twice as that in the previous layer. For example, if the size of the anchor box in the first layer is 32, the sizes of the anchor boxes in the next four layers are 64, 128, 256, and 512. If you use the Faster R-CNN model or the R-FCN model without FPN support, you can specify multiple anchor sizes as needed. For example, you can set the anchor_scales parameter to 128 256 512.	FLOAT list Sample value: 32 (single scale) or 128 256 512 (multiple scales)	◦ SSD: The default value is 0.1, 0.2, 0.37, 0.54, 0.71, 0.88, 0.96, or 1.0 times of the size of the input image. ◦ FPN: 32 ◦ Faster R-CNN with FNP support: 32 ◦ Faster R-CNN or R-FCN: [128 256 512]
anchor_ratios	No	The ratios of the width to the height of the anchor boxes.	FLOAT list	0.5 1 2
image_sizes	No	The size of the images after they are resized. This parameter takes effect only when an SSD model is used. The value of this parameter is a list that contains two numbers, which indicate the height and width. Unit: pixel.	FLOAT list	300 300
image_min_sizes	No	The length of the shorter side of images after they are resized. This parameter is used for Faster R-CNN and R-FCN models. Unit: pixel. If you specify multiple lengths for the shorter sides of images in the value of this parameter, the last length but one is used to train the model, whereas the last one is used to evaluate the model. If you specify only one length for the shorter sides of images, this length is used for both training and evaluation.	FLOAT list	600

Parameter	Required	Description	Value type or sample value	Default value
image_max_sizes	No	The length of the longer side of images after they are resized. This parameter is used for Faster R-CNN and R-FCN models. Unit: pixel. If you specify multiple lengths for the longer sides of images in the value of this parameter, the last length but one is used to train the model, whereas the last one is used to evaluate the model. If you specify only one length for the longer sides of images, this length is used for both training and evaluation.	FLOAT list	1024
optimizer	No	The type of the optimizer. Valid values: ◦ momentum: stochastic gradient descent (SGD) with momentum ◦ adam	STRING	momentum

Parameter	Required	Description	Value type or sample value	Default value
lr_type	No	The policy that is used to adjust the learning rate. Valid values: ◦ exponential_decay: the exponential decay. ◦ polynomial_decay: the polynomial decay. If you set the lr_type parameter to polynomial_decay, the num_steps parameter is automatically set to the total number of training iterations. The value of the end_learning_rate parameter is automatically set to one thousandth of the value of the initial_learning_rate parameter. ◦ manual_step: The learning rate of each epoch is manually adjusted. If you set the lr_type parameter to manual_step, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rate. You must also set the learning_rates parameter to specify the learning rate as needed. ◦ cosine_decay The learning rate of each epoch is adjusted based on the cosine curve. For more information, see SGDR: Stochastic Gradient Descent with Warm Restarts. If you set the lr_type parameter to cosine_decay, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rates.	STRING	exponential_decay
initial_learning_rate	No	The initial learning rate.	Floating point	0.01

Parameter	Required	Description	Value type or sample value	Default value
decay_epochs	No	If you set the lr_type parameter to exponential_decay, the decay_epochs parameter is equivalent to the decay_steps parameter of tf.train.exponential.decay. In this case, the decay_epochs parameter specifies the epoch interval at which you want to adjust the learning rate. The system automatically converts the value of the decay_epochs parameter to the value of the decay_steps parameter based on the total number of training data entries. Typically, you can set the decay_epochs parameter to half of the total number of epochs. For example, if the total number of epochs is 20, you can set this parameter to 10. If you set the lr_type parameter to manual_step, the decay_epochs parameter specifies the epochs for which you want to adjust the learning rate. For example, a value of 16 18 indicates that you want to adjust the learning rate for the 16th and 18th epochs. Typically, if the total number of epochs is N, you can set the two values of the decay_epochs parameter to 8/10 × N and 9/10 × N.	INTEGER list. Sample value: 20 20 40 60.	20
decay_factor	No	The decay rate. This parameter is equivalent to the decay_factor parameter of tf.train.exponential.decay.	FLOAT	0.95
staircase	No	Specifies whether the learning rate changes based on the decay_epochs parameter. This parameter is equivalent to the staircase parameter of tf.train.exponential.decay.	BOOL	true
power	No	The power of the polynomial. This parameter is equivalent to the power parameter of tf.train.polynomial.decay.	FLOAT	0.9

Parameter	Required	Description	Value type or sample value	Default value
learning_rates	No	The learning rate that you want to set for the specified epochs. This parameter is required when you set the lr_type parameter to manual_step. If you want to adjust the learning rate for two epochs, specify two learning rates in the value. For example, if the decay_epoches parameter is set to 20 40, you must specify two learning rates in the learning_rates parameter, such as 0.001 0.0001. This indicates that the learning rate of the 20th epoch is adjusted to 0.001 and the learning rate of the 40th epoch is adjusted to 0.0001. We recommend that you adjust the learning rate to one tenth, one hundredth, and one thousandth of the initial learning rate in sequence.	FLOAT list	N/A
lr_warmup	No	Specifies whether to warm up the learning rate.	BOOL	false
lr_warm_up_epochs	No	The number of epochs for which you want to warm up the learning rate.	FLOAT	1
train_data	Yes	The Object Storage Service (OSS) path of the data that is used to train the model.	oss://path/to/train_*.tfrecord	N/A
test_data	Yes	The OSS path of the data that is evaluated during the training.	oss://path/to/test_*.tfrecord	N/A
train_batch_size	Yes	The size of the data that is used to train the model in the current batch.	INT. Sample value: 32.	N/A
test_batch_size	Yes	The size of the data that is evaluated in the current batch.	INT. Sample value: 32.	N/A
train_num_readers	No	The number of concurrent threads that are used to read the training data.	INT	4
model_dir	Yes	The OSS path of the model.	oss://path/to/model	N/A
pretrained_model	No	The OSS path of the pretrained model. If this parameter is specified, the actual model is finetuned based on the pretrained model.	oss://pai-vision-data-sh/pretrained_models/inception_v4.ckpt	""
use_pretrained_model	No	Specifies whether to use a pretrained model.	BOOL	true
num_epochs	Yes	The number of training iterations. A value of 1 indicates that all data is iterated once for the training.	INT. Sample value: 40.	N/A

Parameter	Required	Description	Value type or sample value	Default value
num_test_example	No	The number of data entries that are evaluated during the training. A value of -1 indicates that all training data is evaluated.	INT. Sample value: 2000.	-1
num_visualizations	No	The number of data entries that can be visualized during the evaluation.	INT	10
save_checkpoint_epochs	No	The epoch interval at which a checkpoint is saved. A value of 1 indicates that a checkpoint is saved each time an epoch is complete.	INT	1
save_summary_epochs	No	The epoch interval at which a summary is saved. A value of 0.01 indicates that a summary is saved each time 1% of the training data is iterated.	FLOAT	0.01
num_train_images	No	The total number of data entries that are used for the training. If you use custom TFRecord files to train the model, this parameter is required.	INT	0
label_map_path	No	The category mapping file. If you use custom TFRecord files to train the model, this parameter is required.	STRING	""

4.1.11.4.3. Image segmentation

This topic describes how to use EasyVision that is provided by Machine Learning Platform for AI (PAI) to train an image segmentation model. In this example, a Visual Object Classes (VOC) dataset is used to train a DeepLab v3+ model.

Context

Image segmentation refers to the process of partitioning an image into multiple segments. Each segment consists of pixels that share specific characteristics. The mean Intersection-Over-Union (mIOU) is used to evaluate how well an image segmentation model performs. EasyVision implements image segmentation based on DeepLab v3+.

Step 1: Prepare data

1. Download a dataset.

   EasyVision provides the VOC dataset that is converted to a TFRecord file. You can run the following command to download the VOC dataset to the *data* folder:

   ```
   # Download the dataset to the pascal_voc_seg folder.
   ossutil cp -r  oss://pai-vision-data-hz/data/pascal_voc_seg/ data/pascal_voc_seg
   # Download the dataset to the pascal_voc_seg_aug folder.
   ossutil cp -r  oss://pai-vision-data-hz/data/pascal_voc_seg_aug/ data/pascal_voc_seg_aug
   ```

2. Run the following command to download the pretrained ResNet101 model to the *pretrained_models* folder:

   ```
   ossutil cp -r oss://pai-vision-data-hz/pretrained_models/resnet_v1d_101/ pretrained_models/resnet_v1d_101
   ```

Step 2: Train the model

EasyVision allows you to train a model by referencing a configuration file or setting parameters.

- **Reference a configuration file**

  You can set the parameters that are used for model training and evaluation in a configuration file and reference the file in Python code. For more information about sample configuration files, see Image semantic segmentation. If you want to know the configuration details, submit a ticket to contact the customer service representatives of PAI. Run the following Python code that references a configuration file to start model training and evaluation:

```
import easy_vision
# The first step of model training and evaluation.
easy_vision.train_and_evaluate(easy_vision.DEEPLAB_SAMPLE_CONFIG_STEP1)
# The second step of model training and evaluation.
easy_vision.train_and_evaluate(easy_vision.DEEPLAB_SAMPLE_CONFIG_STEP2)
```

  If you use a custom configuration file, you must modify the preceding code by replacing both `easy_vision.DEEPLAB_SAMPLE_CONFIG_STEP1` and `easy_vision.DEEPLAB_SAMPLE_CONFIG_STEP2` with the path of the custom configuration file.

- **Set parameters**

  You can set the parameters that are used for model training and evaluation in Python code. Run the following Python code that contains related parameters to start model training and evaluation:

```
import easy_vision
param_config = """
  --model_type DeeplabV3
  --backbone  resnet_v1d_101
  --backbone_feature_stride 16
  --bn_trainable true
  --num_classes 21
  --num_epochs 1
  --model_dir experiments/pascal_voc/deeplab_stage1
  --train_data data/pascal_voc_seg_aug/voc_ev_train.tfrecord
  --test_data data/pascal_voc_seg_aug/voc_ev_val.tfrecord
  --num_test_example 2
  --train_batch_size 6
  --test_batch_size 1
  --image_crop_size 513
  --lr_type polynomial_decay
  --initial_learning_rate 0.007
  --power 0.9"""
easy_vision.train_and_evaluate_with_param_config(param_config)
```

  In the preceding code, the param_config structure contains the parameters that are used for model training and evaluation. The format of the param_config structure is the same as that of the ArgumentParser() object in Python. The following table describes the parameters in the param_config structure.

  > **Note** Do not enclose a parameter value of the STRING type in double quotation marks (") or single quotation marks (').

  Parameters in the param_config structure

Parameter	Required	Description	Value type or sample value	Default value

Parameter	Required	Description	Value type or sample value	Default value
model_type	Yes	The type of the model to train. To train an image segmentation model by using EasyVision, set this parameter to DeeplabV3.	STRING	N/A
backbone	Yes	The name of the backbone network that is used by the model. Valid values: ○ resnet_v1_50 ○ resnet_v1_101 ○ resnet_v1a_18 ○ resnet_v1a_34 ○ resnet_v1d_50 ○ resnet_v1d_101 ○ xception_41 ○ xception_65 ○ xception_71	STRING	N/A
weight_decay	No	The value of L2 regularization.	FLOAT	1e-4
num_classes	No	The number of categories, including background categories.	INT	N/A
backbone_feature_stride	No	The feature downsampling step size of the backbone network.	INT	16
bn_trainable	No	Specifies whether the batch normalization (BN) layer is trainable. If the value of the train_batch_size parameter is greater than 8, set this parameter to true.	BOOL	true
image_crop_size	No	The size of the image after cropping.	INT	513
optimizer	No	The type of the optimizer. Valid values: ○ momentum: stochastic gradient descent (SGD) with momentum ○ adam	STRING	momentum

Parameter	Required	Description	Value type or sample value	Default value
lr_type	No	The policy that is used to adjust the learning rate. Valid values: ◦ exponential_decay: The learning rate is subject to exponential decay. ◦ polynomial_decay: The learning rate is subject to polynomial decay. If you set the lr_type parameter to polynomial_decay, the num_steps parameter is automatically set to the total number of training iterations. The value of the end_learning_rate parameter is automatically set to one thousandth of the value of the initial_learning_rate parameter. ◦ manual_step: The learning rate of each epoch is manually adjusted. If you set the lr_type parameter to manual_step, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rate. You must also set the learning_rates parameter to specify the learning rate of each epoch as needed. ◦ cosine_decay The learning rate of each epoch is adjusted based on the cosine curve. For more information, see SGDR: Stochastic Gradient Descent with Warm Restarts. If you set the lr_type parameter to cosine_decay, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rate.	STRING	exponential_decay
initial_learning_rate	No	The initial learning rate.	Floating point	0.01

Parameter	Required	Description	Value type or sample value	Default value
decay_epochs	No	If you set the lr_type parameter to exponential_decay, the decay_epochs parameter is equivalent to the decay_steps parameter of tf.train.exponential.decay. In this case, the decay_epochs parameter specifies the epoch interval at which you want to adjust the learning rate. The system automatically converts the value of the decay_epochs parameter to the value of the decay_steps parameter based on the total number of training data entries. Typically, you can set the decay_epochs parameter to half of the total number of epochs. For example, if the total number of epochs is 20, you can set this parameter to 10. If you set the lr_type parameter to manual_step, the decay_epochs parameter specifies the epochs for which you want to adjust the learning rate. For example, a value of 16 18 indicates that you want to adjust the learning rate for the 16th and 18th epochs. Typically, if the total number of epochs is N, you can set the two values of the decay_epochs parameter to 8/10 × N and 9/10 × N.	INTEGER list. Sample value: 20 20 40 60	20
decay_factor	No	The decay rate. This parameter is equivalent to the decay_factor parameter of tf.train.exponential.decay.	FLOAT	0.95
staircase	No	Specifies whether the learning rate changes based on the decay_epochs parameter. This parameter is equivalent to the staircase parameter of tf.train.exponential.decay.	BOOL	true
power	No	The power of the polynomial. This parameter is equivalent to the power parameter of tf.train.polynomial.decay.	FLOAT	0.9

Parameter	Required	Description	Value type or sample value	Default value
learning_rates	No	The learning rate that you want to set for each specified epoch. This parameter is required if you set the lr_type parameter to manual_step. If you want to adjust the learning rate for two epochs, specify two learning rates in the value. For example, if the decay_epoches parameter is set to 20 40, you must specify two learning rates in the learning_rates parameter, such as 0.001 0.0001. This indicates that the learning rate of the 20th epoch is adjusted to 0.001 and the learning rate of the 40th epoch is adjusted to 0.0001. We recommend that you adjust the learning rate to one tenth, one hundredth, and one thousandth of the initial learning rate in sequence.	FLOAT list	N/A
lr_warmup	No	Specifies whether to warm up the learning rate.	BOOL	false
lr_warm_up_epochs	No	The number of epochs for which you want to warm up the learning rate.	FLOAT	1
train_data	Yes	The Object Storage Service (OSS) path of the data that is used to train the model.	oss://path/to/train_*.tfrecord	N/A
test_data	Yes	The OSS path of the data that is evaluated during the training.	oss://path/to/test_*.tfrecord	N/A
train_batch_size	Yes	The size of the data that is used to train the model in the current batch.	INT. Sample value: 32.	N/A
test_batch_size	Yes	The size of the data that is evaluated in the current batch.	INT. Sample value: 32.	N/A
train_num_readers	No	The number of concurrent threads that are used to read the training data.	INT	4
model_dir	Yes	The OSS path of the model.	oss://path/to/model	N/A
pretrained_model	No	The OSS path of the pretrained model. If this parameter is specified, the actual model is finetuned based on the pretrained model.	oss://pai-vision-data-sh/pretrained_models/inception_v4.ckpt	""
use_pretrained_model	No	Specifies whether to use a pretrained model.	BOOL	true
num_epochs	Yes	The number of training iterations. A value of 1 indicates that all data is iterated once for the training.	INT. Sample value: 40.	N/A

Parameter	Required	Description	Value type or sample value	Default value
num_test_example	No	The number of data entries that are evaluated during the training. A value of -1 indicates that all training data is evaluated.	INT. Sample value: 2000.	-1
num_visualizations	No	The number of data entries that can be visualized during the evaluation.	INT	10
save_checkpoint_epochs	No	The epoch interval at which a checkpoint is saved. A value of 1 indicates that a checkpoint is saved each time an epoch is complete.	INT	1
save_summary_epochs	No	The epoch interval at which a summary is saved. A value of 0.01 indicates that a summary is saved each time 1% of the training data is iterated.	FLOAT	0.01
num_train_images	No	The total number of data entries that are used for the training. If you use custom TFRecord files to train the model, this parameter is required.	INT	0
label_map_path	No	The category mapping file. If you use custom TFRecord files to train the model, this parameter is required.	STRING	""

4.1.11.4.4. Instance segmentation

This topic describes how to use EasyVision that is provided by Machine Learning Platform for AI (PAI) to train an instance segmentation model. In this example, a Common Objects in Context (COCO) dataset is used to train a Mask R-CNN model.

Context

An instance segmentation model detects each distinct object in an image and predicts the movement of each object at the pixel level. The following figure shows an example of instance segmentation. EasyVision provides only Mask R-CNN models for you to implement instance segmentation.

Step 1: Prepare data

1. Download a dataset.

   EasyVision provides the COCO dataset that is converted to a TFRecord file. You can run the following command to download the COCO dataset to the *data* folder:

   ```
   ossutil  cp -r  oss://pai-vision-data-hz/data/coco_wmask/ data/coco_wmask
   ```

2. Run the following command to download the pretrained ResNet50 model to the *pretrained_models* folder:

   ```
   ossutil cp -r oss://pai-vision-data-hz/pretrained_models/resnet_v1d_50/ pretrained_models/resnet_v1d_50
   ```

Step 2: Train the model

You can set the parameters that are used for model training and evaluation in a configuration file to control the process of model training and evaluation. For more information about sample configuration files, see Instance segmentation. If you want to know the configuration details, submit a ticket to contact the customer service representatives of PAI. Run the following Python code that references a configuration file to start model training and evaluation:

```
import easy_vision
easy_vision.train_and_evaluate(easy_vision.MASK_RCNN_SAMPLE_CONFIG)
```

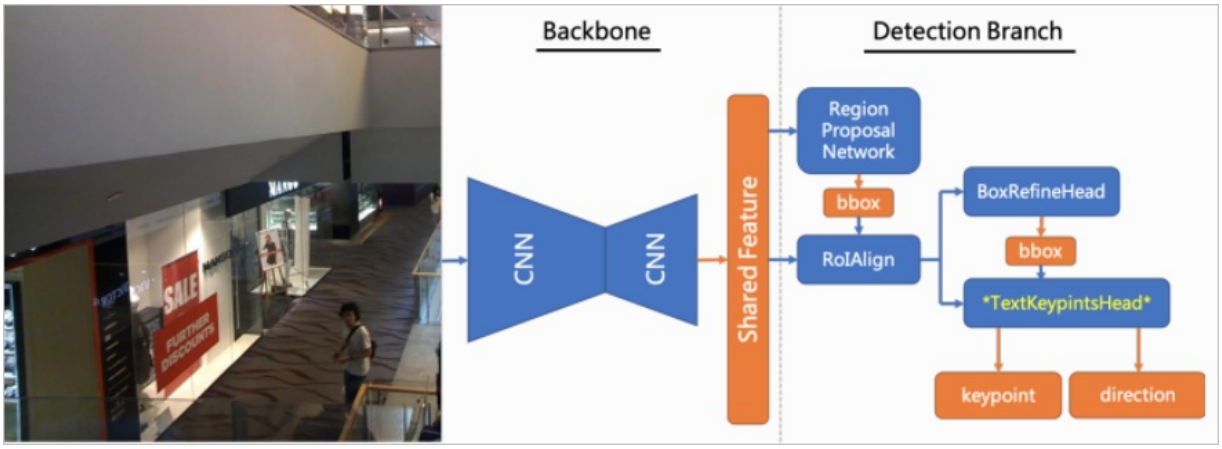
If you want to use a custom configuration file, you must modify the preceding code by replacing `easy_vision.MASK_RCNN_SAMPLE_CONFIG` with the path of the custom configuration file.

4.1.11.4.5. Text detection

This topic describes how to use EasyVision to train a text detection model. In this example, the ICDAR 2015 dataset is used and the Keypoint R-CNN model is trained.

Context

Text detection locates the text in complex text recognition scenarios. EasyVision supports the self-developed text detection model Keypoint R-CNN. Compared with the Faster R-CNN model that is used for common image detection, the Keypoint R-CNN model can predict the vertices and orientation of the text. This advantage facilitates the detection of malformed text and tilted text. The following figure shows the algorithm framework of the models for text detection.



Step 1: Prepare data

1. Download a dataset.

   EasyVision provides the ICDAR 2015 dataset that is converted to TFRecord files. You can download the dataset to the *data* folder by running the following command:

   ```
   ossutil cp -r oss://pai-vision-data-hz/data/icdar_detection_tfrecords/ data/icdar_detection_tfrecords
   ```

2. Download the ResNet50 pretrained model to the *pretrained_models* folder by running the following command:

   ```
   ossutil cp -r oss://pai-vision-data-hz/pretrained_models/resnet_v1_50/ pretrained_models/resnet_v1_50
   ```

Step 2: Train the model

EasyVision allows you to train a model by referencing a configuration file or setting parameters.

- **Reference a configuration file**

  You can set parameters related to training and evaluation in a configuration file and reference the file in the Python code. For more information about sample configuration files, see Text detection. If you want to know the configuration details, submit a ticket to contact the customer service representatives of Machine Learning Platform for AI (PAI). Run the following Python code that references a configuration file to start training and evaluation:

  ```
  import easy_vision
  easy_vision.train_and_evaluate(easy_vision.TEXT_KRCNN_SAMPLE_CONFIG_ICDARCH4)
  ```

  If you want to use a custom configuration file, replace `easy_vision.TEXT_KRCNN_SAMPLE_CONFIG_ICDARCH4` in the preceding code with the path of the custom configuration file.

- **Set parameters**

  You can set the parameters related to training and evaluation in the Python code. Run the following Python code that contains related parameters to start training and evaluation:

```
import easy_vision
param_config = """
  --model_type TextKRCNN
  --backbone  resnet_v1d_50
  --num_classes 1
  --num_steps 10
  --use_pretrained_model true
  --train_batch_size 1
  --test_batch_size 1
  --image_min_sizes 960
  --image_max_sizes 1440
  --initial_learning_rate 0.00001
  --optimizer adam
  --lr_type exponential_decay
  --decay_epochs 40
  --decay_factor 0.5
  --staircase true
  --train_data oss://pai-vision-data-hz/data/icdar_detection_tfrecords/icdar_training_*.tfrecord
  --test_data oss://pai-vision-data-hz/data/icdar_detection_tfrecords/icdar-ch4-test.tfrecord
  --model_dir oss://pai-vision-data-hz/test/icdar_ch4/text_krcnn_resnet50_fpn"""
easy_vision.train_and_evaluate_with_param_config(param_config)
```

In the preceding code, the param_config parameter contains the parameters that are used for training and evaluation. The format of the param_config parameter is the same as that of the ArgumentParser() object in Python. The following table describes the parameters.

Note Do not enclose a parameter value of the STRING type in double quotation marks (") or single quotation marks (').

Parameters in the param_config structure

Parameter	Required	Description	Value type or sample value	Default value
model_type	Yes	The type of the model to train. Set the parameter to TextKRCNN for a text detection model.	STRING	N/A
backbone	Yes	The name of the backbone network that is used by the model. Valid values: ○ resnet_v1_50 ○ resnet_v1_101	STRING	N/A
weight_decay	No	The value of L2 regularization.	FLOAT	1e-4
num_classes	No	The number of categories. By default, the value is obtained by analyzing the dataset that is used for the training.	INT	-1

Parameter	Required	Description	Value type or sample value	Default value
anchor_scales	No	The size of the anchor box. The size of the anchor box is the same as that of the input image where the anchor box resides after the image is resized. You can set this parameter based on the size of the resized input image. Set this parameter to the size of the anchor box in the layer that has the highest resolution. The total number of layers is five. The size of the anchor box in a layer is twice as that in the previous layer. For example, if the size of the anchor box in the first layer is 32, the sizes of the anchor boxes in the next four layers are 64, 128, 256, and 512.	FLOAT list	24
anchor_ratios	No	The ratios of the width to the height of the anchor boxes.	FLOAT list	0.2 0.5 1 2 5
predict_text_direction	No	Specifies whether to predict the text orientation.	BOOL	false
text_direction_trainable	No	Specifies whether to train the model to predict the text orientation.	BOOL	false
text_direction_type	No	The type of the prediction of text orientation. Valid values: ◦ normal: greedy prediction. ◦ unified: The orientation of most text lines is determined as the text orientation. ◦ smart_unified: The orientation of most text lines excluding the lines of which the height is twice as the width is determined as the text orientation.	STRING	normal
aspect_ratio_min_jitter_coef	No	The minimum ratio of the width to the height at which images can be resized during the training. A value of 0 indicates that the ratios of the width to the height of images remain unchanged during the training.	FLOAT	0.8
aspect_ratio_max_jitter_coef	No	The maximum ratio of the width to the height at which images can be resized during the training. A value of 0 indicates that the ratios of the width to the height of images remain unchanged during the training.	FLOAT	1.2

Parameter	Required	Description	Value type or sample value	Default value
random_rotation_angle	No	The maximum angle to which images can be randomly rotated during the training, in the clockwise or anticlockwise direction. A value of 0 indicates that images are not randomly rotated during the training.	FLOAT	10
random_crop_min_area	No	The minimum ratio of the size of an image after it is randomly cropped to the size of the original image. A value of 0 indicates that images are not randomly cropped during the training.	FLOAT	0.1
random_crop_max_area	No	The maximum ratio of the size of an image after it is randomly cropped to the size of the original image. A value of 0 indicates that images are not randomly cropped during the training.	FLOAT	1.0
random_crop_min_aspect_ratio	No	The minimum ratio of the width to the height of images after they are randomly cropped during the training. A value of 0 indicates that images are not randomly cropped during the training.	FLOAT	0.2
random_crop_max_aspect_ratio	No	The maximum ratio of the width to the height of images after they are randomly cropped during the training. A value of 0 indicates that images are not randomly cropped during the training.	FLOAT	5
image_min_sizes	No	The length of the shorter side of images after they are resized. If you specify multiple lengths for the shorter sides of images in the value of this parameter, the last length but one is used to train the model, whereas the last one is used to evaluate the model. If you specify only one length for the shorter sides of images, this length is used for both training and evaluation.	FLOAT list	800
image_max_sizes	No	The length of the longer side of images after they are resized. If you specify multiple lengths for the longer sides of images in the value of this parameter, the last length but one is used to train the model, whereas the last one is used to evaluate the model. If you specify only one length for the longer sides of images, this length is used for both training and evaluation.	FLOAT list	1200

Parameter	Required	Description	Value type or sample value	Default value
random_distort_color	No	Specifies whether to randomly change the brightness, contrast, and saturation of images during the training.	BOOL	true
optimizer	No	The type of the optimizer. Valid values: ◦ momentum: stochastic gradient descent (SGD) with momentum ◦ adam	STRING	momentum
lr_type	No	The policy that is used to adjust the learning rate. Valid values: ◦ exponential_decay: the exponential decay. ◦ polynomial_decay: the polynomial decay. If you set the lr_type parameter to polynomial_decay, the num_steps parameter is automatically set to the total number of training iterations. The value of the end_learning_rate parameter is automatically set to one thousandth of the value of the initial_learning_rate parameter. ◦ manual_step: The learning rate of each epoch is manually adjusted. If you set the lr_type parameter to manual_step, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rate. You must also set the learning_rates parameter to specify the learning rate as needed. ◦ cosine_decay: The learning rate of each epoch is adjusted based on the cosine curve until the learning rate decreases to 0.	STRING	exponential_decay
initial_learning_rate	No	The initial learning rate.	FLOAT	0.01

Parameter	Required	Description	Value type or sample value	Default value
decay_epochs	No	If you set the lr_type parameter to exponential_decay, the decay_epochs parameter is equivalent to the decay_steps parameter of tf.train.exponential.decay. In this case, the decay_epochs parameter specifies the epoch interval at which you want to adjust the learning rate. The system automatically converts the value of the decay_epochs parameter to the value of the decay_steps parameter based on the total number of training data entries. Typically, you can set the decay_epochs parameter to half of the total number of epochs. For example, if the total number of epochs is 20, you can set this parameter to 10. If you set the lr_type parameter to manual_step, the decay_epochs parameter specifies the epochs for which you want to adjust the learning rate. For example, a value of 16 18 indicates that you want to adjust the learning rate for the 16th and 18th epochs. Typically, if the total number of epochs is N, you can set the two values of the decay_epochs parameter to 8/10 × N and 9/10 × N.	INTEGER list. Sample value: 20 20 40 60.	20
decay_factor	No	The decay rate. This parameter is equivalent to the decay_factor parameter of tf.train.exponential.decay.	FLOAT	0.95
staircase	No	Specifies whether the learning rate changes based on the decay_epochs parameter. This parameter is equivalent to the staircase parameter of tf.train.exponential.decay.	BOOL	true
power	No	The power of the polynomial. This parameter is equivalent to the power parameter of tf.train.polynomial.decay.	FLOAT	0.9

Parameter	Required	Description	Value type or sample value	Default value
learning_rates	No	The learning rate that you want to set for the specified epochs. This parameter is required when you set the lr_type parameter to manual_step. If you want to adjust the learning rate for two epochs, specify two learning rates in the value. For example, if the decay_epoches parameter is set to 20 40, you must specify two learning rates in the learning_rates parameter, such as 0.001 0.0001. This indicates that the learning rate of the 20th epoch is adjusted to 0.001 and the learning rate of the 40th epoch is adjusted to 0.0001. We recommend that you adjust the learning rate to one tenth, one hundredth, and one thousandth of the initial learning rate in sequence.	FLOAT list	N/A
lr_warmup	No	Specifies whether to warm up the learning rate.	BOOL	false
lr_warm_up_epochs	No	The number of epochs for which you want to warm up the learning rate.	FLOAT	1
train_data	Yes	The Object Storage Service (OSS) path of the data that is used to train the model.	oss://path/to/train_*.tfrecord	N/A
test_data	Yes	The OSS path of the data that is evaluated during the training.	oss://path/to/test_*.tfrecord	N/A
train_batch_size	Yes	The size of the data that is used to train the model in the current batch.	INT. Sample value: 32.	N/A
test_batch_size	Yes	The size of the data that is evaluated in the current batch.	INT. Sample value: 32.	N/A
train_num_readers	No	The number of concurrent threads that are used to read the training data.	INT	4
model_dir	Yes	The OSS path of the model.	oss://path/to/model	N/A
pretrained_model	No	The OSS path of the pretrained model. If this parameter is specified, the actual model is finetuned based on the pretrained model.	oss://pai-vision-data-sh/pretrained_models/inception_v4.ckpt	""
use_pretrained_model	No	Specifies whether to use a pretrained model.	BOOL	true

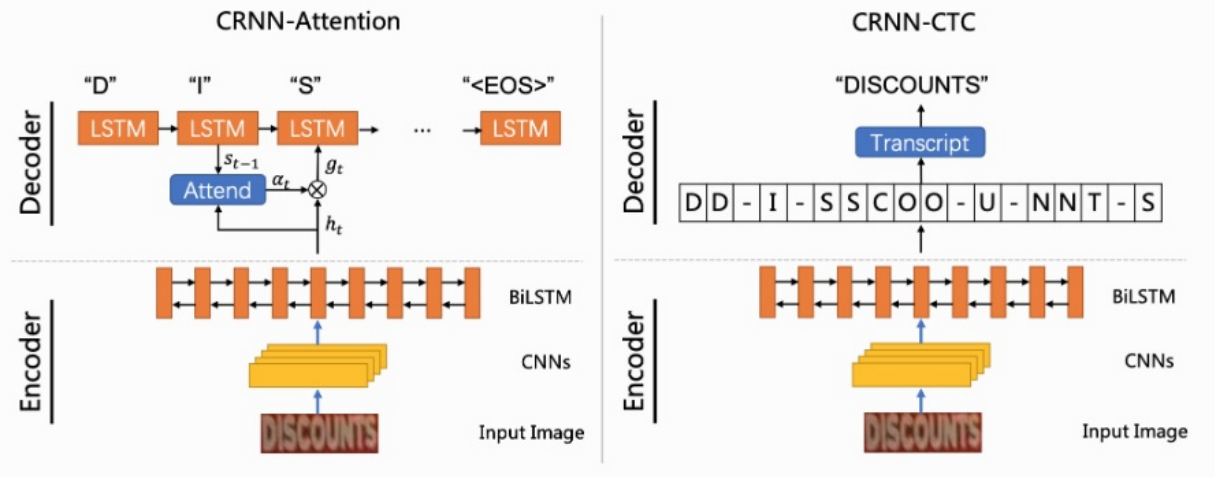
Parameter	Required	Description	Value type or sample value	Default value
num_epochs	Yes	The number of times the data is iterated for the training. A value of 1 indicates that all data is iterated once for the training.	INT. Sample value: 40.	N/A
num_test_example	No	The number of data entries that are evaluated during the training. A value of -1 indicates that all training data is evaluated.	INT. Sample value: 2000.	-1
num_visualizations	No	The number of data entries that can be visualized during the evaluation.	INT	10
save_checkpoint_epochs	No	The epoch interval at which a checkpoint is saved. A value of 1 indicates that a checkpoint is saved each time an epoch is complete.	INT	1
save_summary_epochs	No	The epoch interval at which a summary is saved. A value of 0.01 indicates that a summary is saved each time 1% of the training data is iterated.	FLOAT	0.01
num_train_images	No	The total number of data entries that are used for the training. If you use custom TFRecord files to train the model, this parameter is required.	INT	0
label_map_path	No	The category mapping file. If you use custom TFRecord files to train the model, this parameter is required.	STRING	""

4.1.11.4.6. Text recognition

This topic describes how to use EasyVision that is provided by Machine Learning Platform for AI (PAI) to train a text recognition model. In this example, a receipt dataset is used to train a CRNN-Attention model.

Context

EasyVision supports the following text recognition algorithms: CRNN-Attention, CRNN-CRCC, CNN-SpatialAttention, and TransformerOCR. The following figure shows the algorithm framework of text recognition models.



Step 1: Prepare data

1. Download a dataset.

   EasyVision provides the receipt dataset that is converted to a TFRecord file. You can run the following command to download the receipt dataset to the *data* folder:

   ```
   ssutil cp -r  oss://pai-vision-data-hz/data/recipt_text/recognition_tfrecords/ data/recipt_text/recognition_tfrecords
   ```

2. Run the following command to download the text recognition model that is provided by EasyVision to the *pretrained_models* folder. The model is pretrained by using data in Chinese and English.

   ```
   ossutil cp -r oss://pai-vision-data-hz/pretrained_models/general_crnn_attn_resnet15 pretrained_models/general_crnn_attn_resnet15
   ```

Step 2: Train the model

EasyVision allows you to train a model by referencing a configuration file or setting parameters.

- **Reference a configuration file**

  You can set the parameters that are used for model training and evaluation in a configuration file and reference the file in Python code. For more information about sample configuration files, see Text recognition. If you want to know the configuration details, submit a ticket to contact the customer service representatives of PAI. Run the following Python code that references a configuration file to start model training and evaluation:

  ```
  import easy_vision
  easy_vision.train_and_evaluate(easy_vision.TEXT_RECOG_CRNN_ATTN_CONFIG_RECIPT)
  ```

  If you want to use a custom configuration file, you must modify the preceding code by replacing `easy_vision.TEXT_RECOG_CRNN_ATTN_CONFIG_RECIPT` with the path of the custom configuration file.

- **Set parameters**

  You can set the parameters that are used for model training and evaluation in Python code. Run the following Python code that contains related parameters to start model training and evaluation:

```
import easy_vision
param_config = """
  --model_type TextRecognition
  --backbone text_resnet15
  --use_pretrained_model true
  --train_batch_size 32
  --test_batch_size 32
  --initial_learning_rate 0.0001
  --optimizer adam
  --lr_type exponential_decay
  --decay_epochs 10
  --decay_factor 0.7
  --staircase true
  --train_data data/recipt_text/recognition_tfrecords/train_*.tfrecord
  --test_data data/recipt_text/recognition_tfrecords/test.tfrecord
  --model_dir experiments/recipt_text/crnn_attn_resnet15_fixed_height"""
easy_vision.train_and_evaluate_with_param_config(param_config)
```

In the preceding code, the param_config structure contains the parameters that are used for model training and evaluation. The format of the param_config structure is the same as that of the ArgumentParser() object in Python. The following table describes the parameters in the param_config structure.

> **Note** Do not enclose a parameter value of the STRING type in double quotation marks (") or single quotation marks (').

Parameters in the param_config structure

Parameter	Required	Description	Value type or sample value	Default value
model_type	Yes	The type of the model to train. To train a text recognition model, set this parameter to TextRecognition.	STRING	N/A
backbone	Yes	The name of the backbone network that is used by the model. Valid values: ○ text_resnet15 ○ vgg_bai	STRING	N/A
weight_decay	No	The value of L2 regularization.	FLOAT	1e-4
image_resize_height	No	The height of images after they are resized.	INT	32
image_resize_width	No	The width of images after they are resized. Default value: -1, which indicates resizing based on a fixed ratio of width to height.	INT	-1
min_input_ratio	No	The minimum ratio of the width to the height of images. This parameter takes effect if the image_resize_width parameter is set to -1.	FLOAT	0.125

Parameter	Required	Description	Value type or sample value	Default value
max_input_ratio	No	The maximum ratio of the width to the height of images. This parameter takes effect if the image_resize_width parameter is set to -1.	FLOAT	38.0
num_buckets	No	The number of buckets into which images are classified. Images with similar ratios of width to height are classified into the same bucket. This parameter takes effect if the image_resize_width parameter is set to -1.	INT	10
random_distort_color	No	Specifies whether to randomly change the brightness, contrast, and saturation of images during the training.	BOOL	true
rgb_to_gray	No	Specifies whether to convert images to grayscale images.	BOOL	true
encoder_type	No	The type of the encoder. Valid values: ○ crnn: the hybrid Convolutional Neural Network (CNN)-Recurrent Neural Network (RNN) encoder. ○ cnn_line: the CNN encoder. ○ cnn_spatial: the encoder that uses spatial attention CNN.	STRING	crnn
encoder_num_layers	No	The number of RNN layers in the encoder. CNN layers are not counted.	INT	2
encoder_rnn_type	No	The type of RNN that is used by the encoder. Valid values: ○ bi: bidirectional RNN. ○ uni: unidirectional RNN.	STRING	uni
encoder_hidden_size	No	The number of neurons in the hidden layer of the encoder.	INT	512
encoder_cell_type	No	The type of RNN cells in the encoder. Valid values: ○ basic_lstm ○ gru ○ layer_norm_basic_lstm ○ nas	STRING	basic_lstm
decoder_type	No	The type of the decoder. Valid values: ○ attention ○ ctc	STRING	attention

Parameter	Required	Description	Value type or sample value	Default value
decoder_num_layers	No	The number of layers in the decoder.	INT	2
decoder_hidden_size	No	The number of neurons in the hidden layer of the decoder.	INT	512
decoder_cell_type	No	The type of RNN cells in the decoder. Valid values: ○ basic_lstm ○ gru ○ layer_norm_basic_lstm ○ nas	STRING	basic_lstm
embedding_size	No	The embedding size of the dictionary.	INT	64
beam_width	No	The beam width of the beam search.	INT	0
length_penalty_weight	No	The length penalty score of the beam search. This prevents shorter sentences from receiving higher scores.	FLOAT	0.0
attention_mechanism	No	The type of the attention mechanism of the decoder. Valid values: ○ luong ○ scaled_luong ○ bahdanau ○ normed_bahdanau	STRING	normed_bahdanau
optimizer	No	The type of the optimizer. Valid values: ○ momentum: stochastic gradient descent (SGD) with momentum ○ adam	STRING	momentum

Parameter	Required	Description	Value type or sample value	Default value
lr_type	No	The policy that is used to adjust the learning rate. Valid values: ◦ exponential_decay: The learning rate is subject to exponential decay. ◦ polynomial_decay: The learning rate is subject to polynomial decay. If you set the lr_type parameter to polynomial_decay, the num_steps parameter is automatically set to the total number of training iterations. The value of the end_learning_rate parameter is automatically set to one thousandth of the value of the initial_learning_rate parameter. ◦ manual_step: The learning rate of each epoch is manually adjusted. If you set the lr_type parameter to manual_step, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rate. You must also set the learning_rates parameter to specify the learning rate of each epoch as needed. ◦ cosine_decay The learning rate of each epoch is adjusted based on the cosine curve. For more information, see SGDR: Stochastic Gradient Descent with Warm Restarts. If you set the lr_type parameter to cosine_decay, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rate.	STRING	exponential_decay
initial_learning_rate	No	The initial learning rate.	FLOAT	0.01

Parameter	Required	Description	Value type or sample value	Default value
decay_epochs	No	If you set the lr_type parameter to exponential_decay, the decay_epochs parameter is equivalent to the decay_steps parameter of tf.train.exponential.decay. In this case, the decay_epochs parameter specifies the epoch interval at which you want to adjust the learning rate. The system automatically converts the value of the decay_epochs parameter to the value of the decay_steps parameter based on the total number of training data entries. Typically, you can set the decay_epochs parameter to half of the total number of epochs. For example, if the total number of epochs is 20, you can set this parameter to 10. If you set the lr_type parameter to manual_step, the decay_epochs parameter specifies the epochs for which you want to adjust the learning rate. For example, a value of 16 18 indicates that you want to adjust the learning rate for the 16th and 18th epochs. Typically, if the total number of epochs is N, you can set the two values of the decay_epochs parameter to 8/10 × N and 9/10 × N.	INTEGER list. Sample value: 20 20 40 60.	20
decay_factor	No	The decay rate. This parameter is equivalent to the decay_factor parameter of tf.train.exponential.decay.	FLOAT	0.95
staircase	No	Specifies whether the learning rate changes based on the decay_epochs parameter. This parameter is equivalent to the staircase parameter of tf.train.exponential.decay.	BOOL	true
power	No	The power of the polynomial. This parameter is equivalent to the power parameter of tf.train.polynomial.decay.	FLOAT	0.9

Parameter	Required	Description	Value type or sample value	Default value
learning_rates	No	The learning rate that you want to set for each specified epoch. This parameter is required if you set the lr_type parameter to manual_step. If you want to adjust the learning rate for two epochs, specify two learning rates in the value. For example, if the decay_epoches parameter is set to 20 40, you must specify two learning rates in the learning_rates parameter, such as 0.001 0.0001. This indicates that the learning rate of the 20th epoch is adjusted to 0.001 and the learning rate of the 40th epoch is adjusted to 0.0001. We recommend that you adjust the learning rate to one tenth, one hundredth, and one thousandth of the initial learning rate in sequence.	FLOAT list	N/A
lr_warmup	No	Specifies whether to warm up the learning rate.	BOOL	false
lr_warm_up_epochs	No	The number of epochs for which you want to warm up the learning rate.	FLOAT	1
train_data	Yes	The Object Storage Service (OSS) path of the data that is used to train the model.	oss://path/to/train_*.tfrecord	N/A
test_data	Yes	The OSS path of the data that is evaluated during the training.	oss://path/to/test_*.tfrecord	N/A
train_batch_size	Yes	The size of the data that is used to train the model in the current batch.	INT. Sample value: 32.	N/A
test_batch_size	Yes	The size of the data that is evaluated in the current batch.	INT. Sample value: 32.	N/A
train_num_readers	No	The number of concurrent threads that are used to read the training data.	INT	4
model_dir	Yes	The OSS path of the model.	oss://path/to/model	N/A
pretrained_model	No	The OSS path of the pretrained model. If this parameter is specified, the actual model is finetuned based on the pretrained model.	oss://pai-vision-data-sh/pretrained_models/inception_v4.ckpt	""
use_pretrained_model	No	Specifies whether to use a pretrained model.	BOOL	true
num_epochs	Yes	The number of training iterations. A value of 1 indicates that all data is iterated once for the training.	INT. Sample value: 40.	N/A

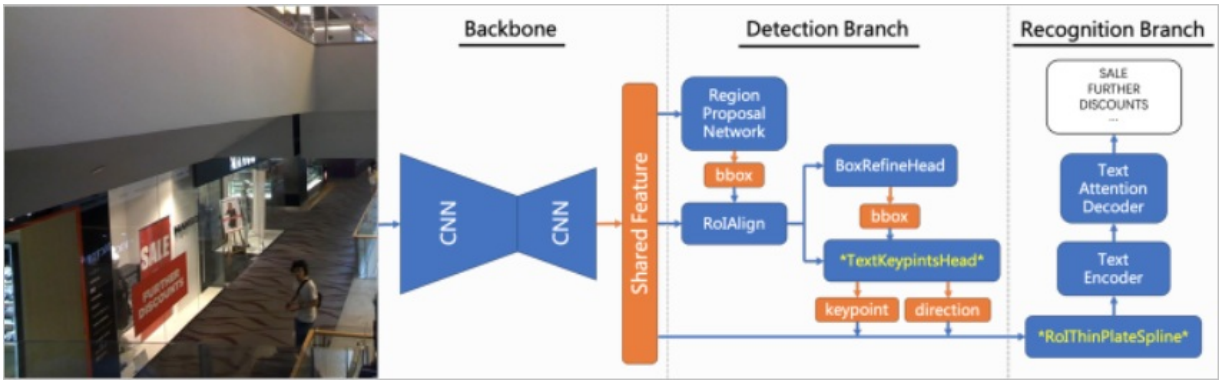
Parameter	Required	Description	Value type or sample value	Default value
num_test_example	No	The number of data entries that are evaluated during the training. A value of -1 indicates that all training data is evaluated.	INT. Sample value: 2000.	-1
num_visualizations	No	The number of data entries that can be visualized during the evaluation.	INT	10
save_checkpoint_epochs	No	The epoch interval at which a checkpoint is saved. A value of 1 indicates that a checkpoint is saved each time an epoch is complete.	INT	1
save_summary_epochs	No	The epoch interval at which a summary is saved. The value of 0.01 indicates that a summary is saved each time 1% of the training data is iterated.	FLOAT	0.01
num_train_images	No	The total number of data entries that are used for the training. If you use custom TFRecord files to train the model, this parameter is required.	INT	0
label_map_path	No	The category mapping file. If you use custom TFRecord files to train the model, this parameter is required.	STRING	""

4.1.11.4.7. End-to-end text recognition

This topic describes how to use EasyVision to train an end-to-end text recognition model. In this example, the receipt dataset is used and the TextEnd2End model is trained.

Context

The traditional optical character recognition (OCR) algorithm separates the text detection and text recognition steps and trains a task for each step. You need to write a large amount of code to process intermediate data that is generated between the two coupled steps. In this case, the recognition accuracy and speed of the OCR algorithm may be lowered. To resolve the preceding issue, EasyVision provides the end-to-end OCR algorithm. The end-to-end OCR algorithm simplifies the algorithm training and deployment, improves the recognition accuracy by about 2%, and increases the recognition speed by 170%. End-to-end text recognition models has greater advantages than the traditional detection and recognition algorithms. The following figure shows the algorithm framework of end-to-end text recognition models.



Step 1: Prepare data

1. Download a dataset.

   EasyVision provides the receipt dataset that is converted to TFRecord files. You can download the dataset to the *data* folder by running the following command:

   ```
   ossutil cp -r oss://pai-vision-data-hz/data/recipt_text/end2end_tfrecords/ data/recipt_text/end2end_tfrecords/
   ```

2. Download the TextEnd2End model provided by EasyVision to the *pretrained_models* folder by running the following command. The TextEnd2End model which is pretrained based on both Chinese and English data.

   ```
   ossutil cp -r oss://pai-vision-data-hz/pretrained_models/resnet_v1d_50/ pretrained_models/resnet_v1d_50
   ```

Step 2: Train the model

EasyVision allows you to train a model by referencing a configuration file or setting parameters.

- **Reference a configuration file**

  You can set parameters related to training and evaluation in a configuration file and reference the file in the Python code. For more information about sample configuration files, see End-to-End text recognition. If you want to know the configuration details, submit a ticket to contact the customer service representatives of Machine Learning Platform for AI (PAI). Run the following Python code that references a configuration file to start training and evaluation:

  ```
  import easy_vision
  easy_vision.train_and_evaluate(easy_vision.TEXT_END2END_SAMPLE_CONFIG_RECIPT)
  ```

  If you want to use a custom configuration file, replace `easy_vision.TEXT_END2END_SAMPLE_CONFIG_RECIPT` in the preceding code with the path of the custom configuration file.

- **Set parameters**

  You can set the parameters related to training and evaluation in the Python code. Run the following Python code that contains related parameters to start training and evaluation:

```
import easy_vision
param_config = """
  --model_type TextEnd2End
  --backbone  resnet_v1_50
  --num_classes 1
  --use_pretrained_model true
  --train_batch_size 1
  --test_batch_size 1
  --image_min_sizes 960
  --image_max_sizes 1440
  --initial_learning_rate 0.0001
  --optimizer adam
  --lr_type exponential_decay
  --decay_epochs 40
  --decay_factor 0.5
  --num_epochs 10
  --staircase true
  --predict_text_direction true
  --text_direction_trainable true
  --text_direction_type smart_unified
  --feature_gather_type fixed_height_pyramid
  --train_data data/recipt_text/end2end_tfrecords/train_*.tfrecord
  --test_data data/recipt_text/end2end_tfrecords/test.tfrecord
  --model_dir experiments/recipt_text/text_end2end_krcnn_resnet50_attn"""
easy_vision.train_and_evaluate_with_param_config(param_config)
```

In the preceding code, the param_config parameter contains the parameters that are used for training and evaluation. The format of the param_config parameter is the same as that of the ArgumentParser() object in Python. The following table describes the parameters.

> **Note** Do not enclose a parameter value of the STRING type in double quotation marks (") or single quotation marks (').

Parameters in the param_config structure

Parameter	Required	Description	Value type or sample value	Default value
model_type	Yes	The type of the model to train. Set this parameter to TextEnd2End when you train a model for end-to-end text recognition.	STRING	N/A
backbone	Yes	The name of the backbone network that is used by the model. Valid values: ○ resnet_v1_50 ○ resnet_v1_101	STRING	N/A
weight_decay	No	The value of L2 regularization.	FLOAT	1e-4
num_classes	No	The number of categories. By default, the value is obtained by analyzing the dataset that is used for the training.	21	-1

Parameter	Required	Description	Value type or sample value	Default value
anchor_scales	No	The size of the anchor box. The size of the anchor box is the same as that of the input image where the anchor box resides after the image is resized. Set this parameter to the size of the anchor box in the layer that has the highest resolution. The total number of layers is five. The size of the anchor box in a layer is twice as that in the previous layer. For example, if the size of the anchor box in the first layer is 32, the sizes of the anchor boxes in the next four layers are 64, 128, 256, and 512.	FLOAT list. Sample value: 32 (single scale).	24
anchor_ratios	No	The ratios of the width to the height of the anchor boxes.	FLOAT list	0.2 0.5 1 2 5
predict_text_direction	No	Specifies whether to predict the text orientation.	BOOL	false
text_direction_trainable	No	Specifies whether to train the model to predict the text orientation.	BOOL	false
text_direction_type	No	The type of the prediction of text orientation. Valid values: ◦ normal: greedy prediction. ◦ unified: The orientation of most text lines is determined as the text orientation. ◦ smart_unified: The orientation of most text lines excluding the lines of which the height is twice as the width is determined as the text orientation.	STRING	normal
feature_gather_type	No	The type of the extractor that is used to extract features of the text lines. Valid values: ◦ fixed_size: extracts features based on the specified width and height. ◦ fixed_height: extracts features based on the specified height and the specified ratio of the width to the height. ◦ fixed_height_pyramid: extracts features from multi-scale features based on the specified height and the specified ratio of the width to the height.	STRING	fixed_height

Parameter	Required	Description	Value type or sample value	Default value
feature_gather_aspect_ratio	No	The ratio of the width to the height of the text lines. If you set feature_gather_type to fixed_size, this parameter specifies the ratio of the specified height to width after the features are resized. If you set feature_gather_type to fixed_height, this parameter specifies the maximum ratio of the specified height to the custom width after the features are resized.	FLOAT	40
feature_gather_batch_size	No	The size of the current batch of the text lines that are used to train the model.	INT	160
recognition_norm_type	No	The type of the normalization that is used by the encoder and feature extractor. Valid values: ○ batch_norm ○ group_norm	STRING	group_norm
recognition_bn_trainable	No	Specifies whether the batch normalization value that is obtained by the encoder and feature extractor can be used for the training. This parameter takes effect only when norm_type is set to batch_norm.	BOOL	false
encoder_type	No	The type of the encoder. Valid values: ○ crnn: hybrid Convolutional Neural Network (CNN)-Recurrent Neural Network (RNN) encoder. ○ cnn_line: CNN encoder. ○ cnn_spatial: the encoder that uses spatial attention CNN.	STRING	crnn
encoder_cnn_name	No	The type of CNN that is used by the encoder. Valid values: ○ conv5_encoder ○ senet5_encoder	STRING	senet5_encoder
encoder_num_layers	No	The number of layers in the encoder, which refer to RNN layers. CNN layers are not counted.	INT	2
encoder_rnn_type	No	The type of RNN that is used by the encoder. Valid values: ○ bi: bidirectional RNN. ○ uni: unidirectional RNN.	STRING	uni

Parameter	Required	Description	Value type or sample value	Default value
encoder_hidden_size	No	The number of neurons in the hidden layer of the encoder.	INT	512
encoder_cell_type	No	The type of RNN cells in the encoder. Valid values: ○ basic_lstm ○ gru ○ layer_norm_basic_lstm ○ nas	STRING	basic_lstm
decoder_type	No	The type of the decoder. Valid values: ○ attention ○ ctc	STRING	attention
decoder_num_layers	No	The number of layers in the decoder.	INT	2
decoder_hidden_size	No	The number of neurons in the hidden layer of the decoder.	INT	512
decoder_cell_type	No	The type of RNN cells in the decoder. Valid values: ○ basic_lstm ○ gru ○ layer_norm_basic_lstm ○ nas	STRING	basic_lstm
embedding_size	No	The embedding size of the dictionary.	INT	64
beam_width	No	The beam width of the beam search.	INT	0
length_penalty_weight	No	The length penalty score of the beam search. This prevents shorter sentences from receiving higher scores.	FLOAT	0.0
attention_mechanism	No	The type of the attention mechanism of the decoder. Valid values: ○ luong ○ scaled_luong ○ bahdanau ○ normed_bahdanau	STRING	normed_bahdanau
aspect_ratio_min_jitter_coef	No	The minimum ratio of the width to the height at which images can be resized during the training. A value of 0 indicates that the ratios of the width to the height of images remain unchanged during the training.	FLOAT	0.8

Parameter	Required	Description	Value type or sample value	Default value
aspect_ratio_max_jitter_coef	No	The maximum ratio of the width to the height at which images can be resized during the training. A value of 0 indicates that the ratios of the width to the height of images remain unchanged during the training.	FLOAT	1.2
random_rotation_angle	No	The maximum angle to which images can be randomly rotated during the training, in the clockwise or anticlockwise direction. A value of 0 indicates that images are not randomly rotated during the training.	FLOAT	10
random_crop_min_area	No	The minimum ratio of the size of an image after it is randomly cropped to the size of the original image. A value of 0 indicates that images are not randomly cropped during the training.	FLOAT	0.1
random_crop_max_area	No	The maximum ratio of the size of an image after it is randomly cropped to the size of the original image. A value of 0 indicates that images are not randomly cropped during the training.	FLOAT	1.0
random_crop_min_aspect_ratio	No	The minimum ratio of the width to the height of images after they are randomly cropped during the training. A value of 0 indicates that images are not randomly cropped during the training.	FLOAT	0.2
random_crop_max_aspect_ratio	No	The maximum ratio of the width to the height of images after they are randomly cropped during the training. A value of 0 indicates that images are not randomly cropped during the training.	FLOAT	5
image_min_sizes	No	The length of the shorter side of images after they are resized. If you specify multiple lengths for the shorter sides of images in the value of this parameter, the last length but one is used to train the model, whereas the last one is used to evaluate the model. If you specify only one length for the shorter sides of images, this length is used for both training and evaluation.	FLOAT list	800

Parameter	Required	Description	Value type or sample value	Default value
image_max_sizes	No	The length of the longer side of images after they are resized. If you specify multiple lengths for the longer sides of images in the value of this parameter, the last length but one is used to train the model, whereas the last one is used to evaluate the model. If you specify only one length for the longer sides of images, this length is used for both training and evaluation.	FLOAT list	1200
random_distort_color	No	Specifies whether to randomly change the brightness, contrast, and saturation of images during the training.	BOOL	true
optimizer	No	The type of the optimizer. Valid values: ◦ momentum: stochastic gradient descent (SGD) with momentum ◦ adam	STRING	momentum

Parameter	Required	Description	Value type or sample value	Default value
lr_type	No	The policy that is used to adjust the learning rate. Valid values: ◦ exponential_decay: the exponential decay. ◦ polynomial_decay: the polynomial decay. If you set the lr_type parameter to polynomial_decay, the num_steps parameter is automatically set to the total number of training iterations. The value of the end_learning_rate parameter is automatically set to one thousandth of the value of the initial_learning_rate parameter. ◦ manual_step: The learning rate of each epoch is manually adjusted. If you set the lr_type parameter to manual_step, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rate. You must also set the learning_rates parameter to specify the learning rate as needed. ◦ cosine_decay The learning rate of each epoch is adjusted based on the cosine curve. For more information, see SGDR: Stochastic Gradient Descent with Warm Restarts. If you set the lr_type parameter to cosine_decay, you must set the decay_epochs parameter to specify the epochs for which you want to adjust the learning rates.	STRING	exponential_decay
initial_learning_rate	No	The initial learning rate.	FLOAT	0.01

Parameter	Required	Description	Value type or sample value	Default value
decay_epochs	No	If you set the lr_type parameter to exponential_decay, the decay_epochs parameter is equivalent to the decay_steps parameter of tf.train.exponential.decay. In this case, the decay_epochs parameter specifies the epoch interval at which you want to adjust the learning rate. The system automatically converts the value of the decay_epochs parameter to the value of the decay_steps parameter based on the total number of training data entries. Typically, you can set the decay_epochs parameter to half of the total number of epochs. For example, if the total number of epochs is 20, you can set this parameter to 10. If you set the lr_type parameter to manual_step, the decay_epochs parameter specifies the epochs for which you want to adjust the learning rate. For example, a value of 16 18 indicates that you want to adjust the learning rate for the 16th and 18th epochs. Typically, if the total number of epochs is N, you can set the two values of the decay_epochs parameter to 8/10 × N and 9/10 × N.	INTEGER list. Sample value: 20 20 40 60.	20
decay_factor	No	The decay rate. This parameter is equivalent to the decay_factor parameter of tf.train.exponential.decay.	FLOAT	0.95
staircase	No	Specifies whether the learning rate changes based on the decay_epochs parameter. This parameter is equivalent to the staircase parameter of tf.train.exponential.decay.	BOOL	true
power	No	The power of the polynomial. This parameter is equivalent to the power parameter of tf.train.polynomial.decay.	FLOAT	0.9

Parameter	Required	Description	Value type or sample value	Default value
learning_rates	No	The learning rate that you want to set for the specified epochs. This parameter is required when you set the lr_type parameter to manual_step. If you want to adjust the learning rate for two epochs, specify two learning rates in the value. For example, if the decay_epoches parameter is set to 20 40, you must specify two learning rates in the learning_rates parameter, such as 0.001 0.0001. This indicates that the learning rate of the 20th epoch is adjusted to 0.001 and the learning rate of the 40th epoch is adjusted to 0.0001. We recommend that you adjust the learning rate to one tenth, one hundredth, and one thousandth of the initial learning rate in sequence.	FLOAT list	N/A
lr_warmup	No	Specifies whether to warm up the learning rate.	BOOL	false
lr_warm_up_epochs	No	The number of epochs for which you want to warm up the learning rate.	FLOAT	1
train_data	Yes	The Object Storage Service (OSS) path of the data that is used to train the model.	oss://path/to/train_*.tfrecord	N/A
test_data	Yes	The OSS path of the data that is evaluated during the training.	oss://path/to/test_*.tfrecord	N/A
train_batch_size	Yes	The size of the data that is used to train the model in the current batch.	INT. Sample value: 32.	N/A
test_batch_size	Yes	The size of the data that is evaluated in the current batch.	INT. Sample value: 32.	N/A
train_num_readers	No	The number of concurrent threads that are used to read the training data.	INT	4
model_dir	Yes	The OSS path of the model.	oss://path/to/model	N/A
pretrained_model	No	The OSS path of the pretrained model. If this parameter is specified, the actual model is finetuned based on the pretrained model.	oss://pai-vision-data-sh/pretrained_models/inception_v4.ckpt	""
use_pretrained_model	No	Specifies whether to use a pretrained model.	BOOL	true
num_epochs	Yes	The number of training iterations. A value of 1 indicates that all data is iterated once for the training.	INT. Sample value: 40.	N/A

Parameter	Required	Description	Value type or sample value	Default value
num_test_example	No	The number of data entries that are evaluated during the training. A value of -1 indicates that all training data is evaluated.	INT. Sample value: 2000.	-1
num_visualizations	No	The number of data entries that can be visualized during the evaluation.	INT	10
save_checkpoint_epochs	No	The epoch interval at which a checkpoint is saved. A value of 1 indicates that a checkpoint is saved each time an epoch is complete.	INT	1
save_summary_epochs	No	The epoch interval at which a summary is saved. The value of 0.01 indicates that a summary is saved each time 1% of the training data is iterated.	FLOAT	0.01
num_train_images	No	The total number of data entries that are used for the training. If you use custom TFRecord files to train the model, this parameter is required.	INT	0
label_map_path	No	The category mapping file. If you use custom TFRecord files to train the model, this parameter is required.	STRING	""

4.1.11.5. Examples of configuration files

4.1.11.5.1. Image classification

EasyVision that is provided by Machine Learning Platform for AI (PAI) allows you to set the training parameters in a configuration file to train a model. This topic provides examples of the configuration files for commonly used image classification models. You can use the complete configurations or modify specific parameters as required to train the image classification models.

Background information

The following models are involved in this topic:

- resnet_v1d_50
- mobilenet_v3
- efficientnet_b0
- darknet53

resnet_v1d_50

```
##-*- encoding:utf-8 -*-
# classfication_resnet50.config:  config resnet50 model for configuration
model_config {
  model_class: 'Classification'
  classification {
    backbone {
      class_name: 'resnet_v1d_50'
```

```
      weight_decay: 0.0001
    }
    num_classes: 1000
    loss {
      weighted_softmax {
      }
    }
    label_id_offset: 1
  }
}
train_config: {
  optimizer {
    momentum_optimizer: {
       learning_rate: {
         manual_step_learning_rate {
           initial_learning_rate: 0.0
           schedule {
             step: 5000
             learning_rate: 0.1
           }
           schedule {
             step: 200000
             learning_rate: 0.01
           }
           schedule {
             step: 400000
             learning_rate: 0.001
           }
            schedule {
             step: 600000
             learning_rate: 0.0001
           }
           warmup: true
         }
       }
       momentum_optimizer_value: 0.9
    }
  }
  #gradient_clipping_by_norm : 10.0
  #distribute training setting
  sync_replicas: true
  #using 8 gpu
  replicas_to_aggregate:8
  num_worker_replicas: 8
  num_steps: 800000
  model_dir: 'experiments/imagenet_resnet50_dis/train'
}
train_data: {
  input_path: "data/imagenet_tfrecord/train-*"
  batch_size: 32
  num_readers: 4
  shuffle: true
  read_block_length: 32
  classification_decoder_config{
  }
  data_augmentation_options {
    random_distort_color {
      color_ordering: 0
      fast_mode: true
```

```
    }
  }
  data_augmentation_options {
    vgg_preprocessing {
      is_training: true
    }
  }
}
eval_config: {
  num_examples: 50000
  # Note: The below line limits the evaluation process to 10 evaluations.
  # Remove the below line to evaluate indefinitely.
  # max_evals: 10
  metrics_set :  'classification_metric'
}
eval_data : {
  input_path: "data/imagenet_tfrecord/validation-*"
  batch_size: 32
  shuffle: false
  num_readers: 1
  classification_decoder_config{
    label_map_path: 'data/imagenet_tfrecord/imagenet_labelmap.pbtxt'
  }
  data_augmentation_options {
    vgg_preprocessing {
      is_training: false
    }
  }
}
```

mobilenet_v3

```
##-*- encoding:utf-8 -*-
# classfication_resnet50.config:  config resnet50 model for configuration
model_config {
  model_class: 'Classification'
  classification {
    backbone {
      class_name: 'mobilenet_v3'
      weight_decay: 0.00004
      depth_multiplier: 1.0
    }
    num_classes: 1001
    loss {
      weighted_softmax {
      }
    }
    label_id_offset: 0
  }
}
train_config: {
  optimizer {
    momentum_optimizer: {
       learning_rate: {
         manual_step_learning_rate {
           initial_learning_rate: 0.1
           schedule {
             step: 200000
             learning_rate: 0.01
```

```
              learning_rate: 0.01
            }
            schedule {
              step: 400000
              learning_rate: 0.001
            }
             schedule {
              step: 550000
              learning_rate: 0.0001
            }
          }
        }
        momentum_optimizer_value: 0.9
    }
  }
  #gradient_clipping_by_norm : 10.0
  num_steps: 800000
  model_dir: 'experiments/imagenet/output/imagenet_mobilenet_v3'
}
train_data: {
  input_path: "data/imagenet/tfrecords/train-*"
  batch_size: 256
  num_readers: 4
  shuffle: true
  read_block_length : 256
  classification_decoder_config{
  }
  data_augmentation_options {
    inception_preprocessing {
      is_training: true
    }
  }
}
eval_config: {
  num_examples: 50000
  # Note: The below line limits the evaluation process to 10 evaluations.
  # Remove the below line to evaluate indefinitely.
  # max_evals: 10
  num_visualizations: 0
  metrics_set :  'classification_metric'
}
eval_data : {
  input_path: "data/imagenet/tfrecords/validation-*"
  batch_size: 100
  shuffle: false
  num_readers: 1
  classification_decoder_config{
    label_map_path: 'data/imagenet/tfrecords/imagenet_labelmap.pbtxt'
  }
  data_augmentation_options {
    inception_preprocessing {
      is_training: false
    }
  }
}
```

efficientnet_b0

```
##-*- encoding:utf-8 -*-
```

```
model_config {
  model_class: 'Classification'
  classification {
    backbone {
      class_name: 'efficientnet-b0'
      weight_decay: 0.00001
      connect_survival_prob: 0.8
    }
    num_classes: 1000
    loss {
      weighted_softmax {
      }
    }
    label_id_offset: 1
  }
}
train_config: {
  optimizer {
    rms_prop_optimizer: {
       learning_rate: {
         # decay by 0.97 every 2.4 epochs
         exponential_decay_learning_rate {
           initial_learning_rate: 0.032
           decay_steps: 6000
           decay_factor: 0.97
          }
       }
    }
    use_moving_average: false
  }
  num_steps: 900000     # total 150 epochs
  save_checkpoints_steps: 5000
  model_dir: 'experiments/imagenet/output/imagenet_efficientnet_b0/'
  sync_replicas: true
  replicas_to_aggregate:8
  num_worker_replicas: 8
}
train_data: {
  input_path: "data/imagenet/tfrecords/train-*"
  batch_size: 64
  num_readers: 8
  shuffle: true
  read_block_length : 32
  classification_decoder_config{
  }
  data_augmentation_options {
    efficientnet_preprocessing {
      model_name: 'efficientnet-b0' # use default image size for the model
      is_training: true
    }
  }
}
eval_config: {
  num_examples: 50000
  metrics_set :  'classification_metric'
}
eval_data : {
  input_path: "data/imagenet/tfrecords/validation-*"
  batch size: 100
```

```
  read_block_length : 100
  shuffle: false
  num_readers: 1
  classification_decoder_config{
    label_map_path: 'data/imagenet/imagenet_labelmap.pbtxt'
  }
  data_augmentation_options {
    efficientnet_preprocessing {
      model_name: 'efficientnet-b0' # use default image size for the model
      is_training: false
    }
  }
}
```

darknet53

```
##-*- encoding:utf-8 -*-
model_config {
  model_class: 'Classification'
  classification {
    backbone {
      class_name: 'darknet53'
      weight_decay: 0.00001
    }
    num_classes: 1000
    loss {
      weighted_softmax {
      }
    }
    label_id_offset: 1
  }
}
train_config: {
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        # decay by 0.97 every 2.4 epochs
        exponential_decay_learning_rate {
          # initial_learning_rate: 0.032
          initial_learning_rate: 0.0
          decay_steps: 6000
          decay_factor: 0.97
         }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  num_steps: 1 # 900000     # total 150 epochs
  save_checkpoints_steps: 5000
  model_dir: 'experiments/imagenet/output/imagenet_darknet53/'
}
train_data: {
  input_path: "data/imagenet/tfrecords/train-*"
  batch_size: 64
  num_readers: 8
  shuffle: true
  read_block_length : 32
```

```
  classification_decoder_config{
  }
  data_augmentation_options {
    classification_random_crop {
    }
  }
  data_augmentation_options {
    resize_image {
      new_height: 256
      new_width: 256
    }
  }
  data_augmentation_options {
    normalize_image {
      original_minval: 0.0
      original_maxval: 255.0
      target_minval: 0.0
      target_maxval: 1.0
    }
  }
}
eval_config: {
  num_examples: 50000
  metrics_set :  'classification_metric'
}
eval_data : {
  input_path: "data/imagenet/tfrecords/validation-*"
  batch_size: 100
  read_block_length : 100
  shuffle: false
  num_readers: 1
  classification_decoder_config{
    label_map_path: 'data/imagenet/imagenet_labelmap.pbtxt'
  }
  data_augmentation_options {
    classification_central_crop {
    }
  }
  data_augmentation_options {
    resize_image {
      new_height: 224 #256
      new_width: 224 #256
    }
  }
  data_augmentation_options {
    normalize_image {
      original_minval: 0.0
      original_maxval: 255.0
      target_minval: 0.0
      target_maxval: 1.0
    }
  }
}
```

4.1.11.5.2. Object detection

EasyVision that is provided by Machine Learning Platform for AI (PAI) allows you to set the training parameters in a configuration file to train a model. This topic provides examples of the configuration files for commonly used object detection models. You can use the complete configurations or modify specific parameters as required to train the object detection models.

Background information

The following models are involved in this topic:

- faster_rcnn_r50
- faster_rcnn_r50_fpn
- ssd_r50
- ssd_r50_fpn
- ssd_mobilenet
- yolo3

faster_rcnn_r50

```
#-*- encoding:utf-8 -*-
# simple_rpn.config: encode the configs used in a simple rpn model
train_config: {
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        manual_step_learning_rate {
          initial_learning_rate: 0.00001
          schedule {
            step: 100
            learning_rate: 0.001
          }
          schedule {
            step: 90000
            learning_rate: .0001
          }
          schedule {
            step: 120000
            learning_rate: .00001
          }
          warmup: true
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  fine_tune_checkpoint: "pretrained_models/resnet_v1d_50/model.ckpt"
  num_steps: 150000
  model_dir: "pascal_resnet50_frcnn_model"
}
train_data: {
  # [0-7] evenly split into 8 parts
  input_path: "data/voc0712_tfrecord/voc0712_part_*.tfrecord"
  batch_size: 2
  num_readers: 4
  read_block_length: 1
  shuffle: true
  voc_decoder_config {
    label_map_path: "data/voc0712_tfrecord/pascal_label_map.pbtxt"
```

```
  }
  # note the augmentation order is important, so it cannot be changed
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    random_resize_to_range {
      min_sizes: 600
      max_sizes: 1024
    }
  }
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
  use_diff: false
}
eval_config: {
  num_examples: 4952
  max_evals: 1000
  num_visualizations: 100
  #metrics_set: 'coco_detection_metrics'
  metrics_set: 'pascal_voc_detection_metrics'
  metrics_set: 'pascal_voc07_detection_metrics'
}
eval_data: {
  input_path: "data/voc0712_tfrecord/VOC2007_test.tfrecord"
  batch_size: 1
  shuffle: false
  num_readers: 1
  voc_decoder_config {
    label_map_path: "data/voc0712_tfrecord/pascal_label_map.pbtxt"
  }
  # note the augmentation order is important, so it cannot be changed
  data_augmentation_options {
    random_resize_to_range {
      min_sizes: 600
      max_sizes: 1024
    }
  }
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
  use_diff: true
}
export_config {
  batch_size: 1
}
model_config: {
```

```
model_class: 'FasterRcnn'
faster_rcnn {
  backbone {
    class_name: 'resnet_v1d_50'
    batchnorm_trainable: false
    weight_decay: 0.0001
    output_stride: 16
  }
  rpn_head {
    input_layer: 'resnet_v1d_50/block3'
    box_predictor {
      convolutional_box_predictor {
        conv_hyperparams {
          op: CONV
          regularizer {
            l2_regularizer {
              weight: 0.0001
            }
          }
          initializer {
            truncated_normal_initializer {
              stddev: 0.01
            }
          }
        }
        min_depth: 512
        max_depth: 512
        num_layers_before_predictor: 1
        kernel_size: 3
      }
    }
    first_stage_minibatch_size: 256
    first_stage_positive_balance_fraction: 0.5
    first_stage_nms_iou_threshold: 0.7
    first_stage_max_proposals: 300
    rpn_min_size: 16
    first_stage_anchor_generator {
      # the default base anchor size is 256
      grid_anchor_generator {
        scales: [0.5, 1.0, 2.0]
        aspect_ratios: [0.5, 1.0, 2.0]
        height_stride: 16
        width_stride: 16
      }
    }
  }
  region_feature_extractor {
    resnet_block {
      class_name: 'resnet_v1d_50' #the name of backbone
      block_name: 'block4'  #the last residual block of resnet_v1d_50
      stride: 1
      weight_decay: 0.0001
    }
  }
  rcnn_head {
    input_layer: 'resnet_v1d_50/block3'
    initial_crop_size: 14
    maxpool_kernel_size: 2
    maxpool_stride: 2
```

```
      num_classes: 20
      second_stage_box_predictor {
        mask_rcnn_box_predictor {
          fc_hyperparams {
            op: FC
            regularizer {
              l2_regularizer {
                weight: 0.0001
              }
            }
            initializer {
              xavier_initializer {
              }
            }
          }
          agnostic: true
        }
      }
      nms_config {
        score_threshold: 0.0
        iou_threshold: 0.3
        max_detections_per_class: 400
        max_total_detections: 400
      }
      second_stage_batch_size: 128
      second_stage_balance_fraction: 0.25
    }
  }
}
```

faster_rcnn_r50_fpn

```
#-*- encoding:utf-8 -*-
# simple_rpn.config: encode the configs used in a simple rpn model
train_config: {
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        manual_step_learning_rate {
          initial_learning_rate: 0.00001
          schedule {
            step: 100
            learning_rate: 0.001
          }
          schedule {
            step: 90000
            learning_rate: .0001
          }
          schedule {
            step: 120000
            learning_rate: .00001
          }
          warmup: true
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
```

```
  }
  fine_tune_checkpoint: "pretrained_models/resnet_v1d_50/model.ckpt"
  num_steps: 150000
  model_dir: "pascal_resnet50_frcnn_model_fpn"
  log_step_count_steps: 1
}
train_data: {
  input_path: "data/voc0712_tfrecord/voc0712_part_*.tfrecord"
  batch_size: 2
  num_readers: 4
  read_block_length: 1
  shuffle: true
  voc_decoder_config {
    label_map_path: "data/voc0712_tfrecord/pascal_label_map.pbtxt"
  }
  # note the augmentation order is important, so it cannot be changed
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    random_resize_to_range {
      min_sizes: 600
      max_sizes: 1024
    }
  }
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
  use_diff: false
}
eval_config: {
  num_examples: 4952
  max_evals: 1000
  num_visualizations: 100
  #metrics_set: 'coco_detection_metrics'
  metrics_set: 'pascal_voc_detection_metrics'
  metrics_set: 'pascal_voc07_detection_metrics'
}
eval_data: {
  input_path: "data/voc0712_tfrecord/VOC2007_test.tfrecord"
  batch_size: 1
  shuffle: false
  num_readers: 1
  voc_decoder_config {
    label_map_path: "data/voc0712_tfrecord/pascal_label_map.pbtxt"
  }
  # note the augmentation order is important, so it cannot be changed
  data_augmentation_options {
    random_resize_to_range {
      min_sizes: 600
      max_sizes: 1024
    }
  }
```

```
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
  use_diff: true
}
export_config {
  batch_size: 1
}
model_config: {
  model_class: 'FasterRcnn'
  faster_rcnn {
    backbone {
      class_name: 'resnet_v1d_50'
      batchnorm_trainable: false
      weight_decay: 0.0001
    }
    fpn {
      input: 'resnet_v1d_50/block1'
      input: 'resnet_v1d_50/block2'
      input: 'resnet_v1d_50/block3'
      input: 'resnet_v1d_50/block4'
      fea_dim: 256
      extra_conv_layers: 1
      roi_min_level: 2
      roi_max_level: 5
      conv_hyperparams {
        op: CONV
        regularizer {
          l2_regularizer {
            weight: 0.0001
          }
        }
        initializer {
          truncated_normal_initializer {
            stddev: 0.01
          }
        }
      }
    }
    rpn_head {
      # if input_layer is not specified, will use fpn features,
      # which all have "FPN/" prefix
      box_predictor {
        weight_shared_convolutional_box_predictor {
          conv_hyperparams {
            op: CONV
            regularizer {
              l2_regularizer {
                weight: 0.0001
              }
            }
            initializer {
              truncated_normal_initializer {
                stddev: 0.01
```

```
            }
          }
        }
        depth: 512
        num_layers_before_predictor: 1
        kernel_size: 3
      }
    }
    first_stage_minibatch_size: 256
    first_stage_positive_balance_fraction: 0.5
    first_stage_nms_iou_threshold: 0.7
    first_stage_max_proposals: 300
    rpn_min_size: 16
    first_stage_anchor_generator {
      # anchor_size = anchor_scale * feature_map_stride
      multiscale_anchor_generator {
        min_level: 2
        max_level: 6
        anchor_scale: 8
        aspect_ratios: 0.5
        aspect_ratios: 1
        aspect_ratios: 2
        normalize_coordinates: false
        scales_per_octave: 1
      }
    }
  }
  rcnn_head {
    initial_crop_size: 14
    maxpool_kernel_size: 2
    maxpool_stride: 2
    num_classes: 20
    second_stage_box_predictor {
      mask_rcnn_box_predictor {
        depth: 1024
        num_layers_before_predictor: 2
        fc_hyperparams {
          op: FC
          regularizer {
            l2_regularizer {
              weight: 0.0001
            }
          }
          initializer {
            xavier_initializer {
            }
          }
        }
        agnostic: true
      }
    }
    nms_config {
      score_threshold: 0.0
      iou_threshold: 0.3
      max_detections_per_class: 400
      max_total_detections: 400
    }
    second_stage_batch_size: 128
    second_stage_balance_fraction: 0.25
```

```
    }
  }
}
```

ssd_r50

```
##-*- encoding:utf-8 -*-
# SSD with Resnet50 configuration for VOC Dataset.
# Users should configure the fine_tune_checkpoint field in the train config as
# well as the label_map_path and input_path fields in the train_input_reader and
# eval_input_reader. Search for "PATH_TO_BE_CONFIGURED" to find the fields that
# should be configured.
model_config {
  model_class: 'SSD'
  ssd {
    backbone {
      class_name: "resnet_v1d_50"
      output_stride: 16
    }
    ssd_head {
      num_classes: 20
      ssd_featuremap_layout {
        from_layer: 'resnet_v1d_50/block3'
        from_layer: 'resnet_v1d_50/block4'
        from_layer: ''
        from_layer: ''
        from_layer: ''
        from_layer: ''
        layer_depth: -1
        layer_depth: -1
        layer_depth: 512
        layer_depth: 512
        layer_depth: 256
        layer_depth: 256
      }
      #min_depth: 16
      #depth_multiplier: 1.0
      conv_hyperparams {
        activation: RELU,
        regularizer {
          l2_regularizer {
            weight: 0.0005
          }
        }
        initializer {
          truncated_normal_initializer {
            stddev: 0.03
            mean: 0.0
          }
        }
        batch_norm {
          train: true,
          scale: true,
          center: true,
          decay: 0.9997,
          epsilon: 0.001,
        }
      }
      box_coder {
```

```
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
      argmax_matcher {
        matched_threshold: 0.5
        unmatched_threshold: 0.5
        ignore_thresholds: false
        negatives_lower_than_unmatched: true
        force_match_for_each_row: true
      }
    }
    similarity_calculator {
      iou_similarity {
      }
    }
    anchor_generator {
      ssd_anchor_generator {
        num_layers: 6
        #min_scale: 0.2
        #max_scale: 0.9
        #use caffe anchor scale, the last one is [0.88, 1.0]
        scales: 0.1
        scales: 0.2
        scales: 0.37
        scales: 0.54
        scales: 0.71
        scales: 0.88
        scales: 1.0
        aspect_ratios: 1.0
        aspect_ratios: 2.0
        aspect_ratios: 0.5
        aspect_ratios: 3.0
        aspect_ratios: 0.3333
        reduce_boxes_in_lowest_layer: true
        reduce_boxes_in_larger_layers: true
        interpolate_in_all_layers: true
      }
    }
    box_predictor {
      convolutional_box_predictor {
        min_depth: 0
        max_depth: 0
        num_layers_before_predictor: 0
        kernel_size: 3
        box_code_size: 4
        conv_hyperparams {
          #activation: RELU_6,
          activation: NONE,
          regularizer {
            l2_regularizer {
              weight: 0.0005
            }
          }
          initializer {
```

```
            initializer {
              #truncated_normal_initializer {
              #  stddev: 0.03
              #  mean: 0.0
              #}
              xavier_initializer {
                uniform : false
              }
            }
          }
        }
      }
      post_processing {
        batch_non_max_suppression {
          score_threshold: 0.01
          iou_threshold: 0.45
          max_detections_per_class: 100
          max_total_detections: 200
        }
        score_converter: SOFTMAX
      }
      normalize_loss_by_num_matches: true
      loss {
        classification_loss {
          weighted_softmax {
          }
        }
        localization_loss {
          weighted_smooth_l1 {
          }
        }
        hard_example_miner {
          num_hard_examples: 3000
          iou_threshold: 0.99
          loss_type: BOTH
          max_negatives_per_positive: 3
          min_negatives_per_image: 0
        }
        classification_weight: 3.0
        localization_weight: 1.0
      }
    }
  }
}
train_config: {
  optimizer {
    momentum_optimizer: {
       learning_rate: {
         manual_step_learning_rate {
           initial_learning_rate: 0.001
           schedule {
             step: 80000
             learning_rate: 0.0001
           }
           schedule {
             step: 100000
             learning_rate: 0.00001
           }
         }
       }
```

```
        }
        momentum_optimizer_value: 0.9
      }
    }
    #gradient_clipping_by_norm : 10.0
    fine_tune_checkpoint: "pretrained_models/resnet_v1d_50/model.ckpt"
    num_steps: 120000
    model_dir: 'experiments/ssd_resnet50/train'
  }
  train_data: {
    input_path: "data/voc0712_tfrecord/VOC2007_train.tfrecord"
    input_path: "data/voc0712_tfrecord/VOC2012_train.tfrecord"
    input_path: "data/voc0712_tfrecord/VOC2007_val.tfrecord"
    input_path: "data/voc0712_tfrecord/VOC2012_val.tfrecord"
    batch_size: 32
    num_readers: 4
    shuffle: true
    read_block_length : 32
    voc_decoder_config {
      label_map_path: "data/voc0712_tfrecord/pascal_label_map.pbtxt"
    }
    #data argumentation
    data_augmentation_options {
      ssd_random_crop {
      }
    }
    data_augmentation_options {
      random_adjust_brightness {
        max_delta:0.125
      }
    }
    data_augmentation_options {
      random_adjust_contrast {
        min_delta : 0.5
        max_delta : 1.5
      }
    }
    data_augmentation_options {
      random_adjust_hue {
        max_delta : 0.046875
      }
    }
    data_augmentation_options {
      random_adjust_saturation {
        min_delta : 0.5
        max_delta : 1.5
      }
    }
    data_augmentation_options {
      random_horizontal_flip {
      }
    }
    data_augmentation_options {
      resize_image {
        new_height: 300
        new_width: 300
        method: BILINEAR
      }
    }
    data_augmentation_options {
```

```
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/weiliu89/caffe/blob/ssd/examples/ssd/ssd_pascal.py#L177
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
}
eval_config: {
  num_examples: 4592
  # Note: The below line limits the evaluation process to 10 evaluations.
  # Remove the below line to evaluate indefinitely.
  # max_evals: 10
  metrics_set :  'pascal_voc07_detection_metrics'
  #metrics_set :  'coco_detection_metrics'
  visualize_groundtruth_boxes : true
  # num of visualizations to be displayed on tensorboard
  num_visualizations : 10
  # all the evaluation results will be saved to this dir if not ''
  visualization_export_dir: ''
  max_num_boxes_to_visualize: 20
  min_score_threshold: 0.5
}
eval_data : {
  input_path: "data/voc0712_tfrecord/VOC2007_test.tfrecord"
  batch_size: 1
  shuffle: false
  num_readers: 1
  voc_decoder_config {
    label_map_path: "data/voc0712_tfrecord/pascal_label_map.pbtxt"
  }
  data_augmentation_options {
    resize_image {
      new_height: 300
      new_width: 300
      method: BILINEAR
    }
  }
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/weiliu89/caffe/blob/ssd/examples/ssd/ssd_pascal.py#L177
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
}
export_config {
  batch_size: 1
}
```

ssd_r50_fpn

```
##-*- encoding:utf-8 -*-
# SSD with Resnet50 configuration for VOC Dataset.
# Users should configure the fine_tune_checkpoint field in the train config as
# well as the label_map_path and input_path fields in the train_input_reader and
# eval_input_reader. Search for "PATH_TO_BE_CONFIGURED" to find the fields that
```

```
# should be configured.
model_config {
  model_class: 'SSD'
  ssd {
    backbone {
      class_name: "resnet_v1d_50"
    }
    ssd_head {
      num_classes: 20
      fpn_featuremap_layout {
        from_layer: 'resnet_v1d_50/block2'
        from_layer: 'resnet_v1d_50/block3'
        from_layer: 'resnet_v1d_50/block4'
        layer_depth: 256
        extra_conv_layers: 2
      }
      conv_hyperparams {
        activation: RELU,
        regularizer {
          l2_regularizer {
            weight: 0.0005
          }
        }
        initializer {
          truncated_normal_initializer {
            stddev: 0.03
            mean: 0.0
          }
        }
        batch_norm {
          train: true,
          scale: true,
          center: true,
          decay: 0.9997,
          epsilon: 0.001,
        }
      }
      box_coder {
        faster_rcnn_box_coder {
          y_scale: 10.0
          x_scale: 10.0
          height_scale: 5.0
          width_scale: 5.0
        }
      }
      matcher {
        argmax_matcher {
          matched_threshold: 0.5
          unmatched_threshold: 0.5
          ignore_thresholds: false
          negatives_lower_than_unmatched: true
          force_match_for_each_row: true
        }
      }
      similarity_calculator {
        iou_similarity {
        }
      }
      anchor_generator {
```

```
    multiscale_anchor_generator{
      min_level: 3
      max_level: 7
      anchor_scale: 4.0
      aspect_ratios: [0.5, 1.0, 2.0, 0.333, 3.0]
      scales_per_octave: 2
      normalize_coordinates: true
    }
  }
  box_predictor {
    weight_shared_convolutional_box_predictor{
      depth: 256
      num_layers_before_predictor: 4
      kernel_size: 3
      box_code_size: 4
      conv_hyperparams {
        #activation: RELU_6,
        activation: NONE,
        regularizer {
          l2_regularizer {
            weight: 0.0005
          }
        }
        initializer {
          xavier_initializer {
            uniform : false
          }
        }
        batch_norm {
           scale: true,
           decay: 0.9997,
           epsilon: 0.001,
        }
      }
    }
  }
  post_processing {
    batch_non_max_suppression {
      score_threshold: 0.01
      iou_threshold: 0.45
      max_detections_per_class: 100
      max_total_detections: 200
    }
    score_converter: SOFTMAX
  }
  normalize_loss_by_num_matches: true
  loss {
    classification_loss {
      weighted_softmax {
      }
    }
    localization_loss {
      weighted_smooth_l1 {
      }
    }
    hard_example_miner {
      num_hard_examples: 3000
      iou_threshold: 0.99
      loss_type: BOTH
```

```
          max_negatives_per_positive: 3
          min_negatives_per_image: 0
        }
        classification_weight: 3.0
        localization_weight: 1.0
      }
    }
  }
}
train_config: {
  optimizer {
    momentum_optimizer: {
       learning_rate: {
         manual_step_learning_rate {
           initial_learning_rate: 0.001
           schedule {
             step: 80000
             learning_rate: 0.0001
           }
           schedule {
             step: 100000
             learning_rate: 0.00001
           }
         }
       }
       momentum_optimizer_value: 0.9
    }
  }
  #gradient_clipping_by_norm : 10.0
  fine_tune_checkpoint: "pretrained_models/resnet_v1d_50/model.ckpt"
  num_steps: 120000
  model_dir: 'experiments/ssd_resnet50_fpn'
}
train_data: {
  input_path: "data/voc0712_tfrecord/VOC2007_train.tfrecord"
  input_path: "data/voc0712_tfrecord/VOC2012_train.tfrecord"
  input_path: "data/voc0712_tfrecord/VOC2007_val.tfrecord"
  input_path: "data/voc0712_tfrecord/VOC2012_val.tfrecord"
  batch_size: 32
  num_readers: 4
  shuffle: true
  read_block_length : 32
  voc_decoder_config {
    label_map_path: "data/voc0712_tfrecord/pascal_label_map.pbtxt"
  }
  #data argumentation
  data_augmentation_options {
    ssd_random_crop {
    }
  }
  data_augmentation_options {
    random_adjust_brightness {
      max_delta:0.125
    }
  }
  data_augmentation_options {
    random_adjust_contrast {
      min_delta : 0.5
      max_delta : 1.5
```

```
    }
  }
  data_augmentation_options {
    random_adjust_hue {
      max_delta : 0.046875
    }
  }
  data_augmentation_options {
    random_adjust_saturation {
      min_delta : 0.5
      max_delta : 1.5
    }
  }
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    resize_image {
      new_height: 300
      new_width: 300
      method: BILINEAR
    }
  }
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/weiliu89/caffe/blob/ssd/examples/ssd/ssd_pascal.py#L177
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
}
eval_config: {
  num_examples: 4592
  # Note: The below line limits the evaluation process to 10 evaluations.
  # Remove the below line to evaluate indefinitely.
  # max_evals: 10
  metrics_set :  'pascal_voc07_detection_metrics'
  #metrics_set :  'coco_detection_metrics'
  visualize_groundtruth_boxes : true
  # num of visualizations to be displayed on tensorboard
  num_visualizations : 10
  # all the evaluation results will be saved to this dir if not ''
  visualization_export_dir: ''
  max_num_boxes_to_visualize: 20
  min_score_threshold: 0.5
}
eval_data : {
  input_path: "data/voc0712_tfrecord/VOC2007_test.tfrecord"
  batch_size: 1
  shuffle: false
  num_readers: 1
  voc_decoder_config {
    label_map_path: "data/voc0712_tfrecord/pascal_label_map.pbtxt"
  }
  data_augmentation_options {
    resize_image {
      new_height: 300
```

```
      new_width: 300
      method: BILINEAR
    }
  }
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/weiliu89/caffe/blob/ssd/examples/ssd/ssd_pascal.py#L177
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
}
export_config {
  batch_size: 1
}
```

ssd_mobilenet

```
##-*- encoding:utf-8 -*-
# ssd_vgg16.config: encode the configs used in a ssd-vgg16 model
#
# SSD with VGG16 configuration for VOC Dataset.
# Users should configure the fine_tune_checkpoint field in the train config as
# well as the label_map_path and input_path fields in the train_input_reader and
# eval_input_reader. Search for "PATH_TO_BE_CONFIGURED" to find the fields that
# should be configured.
model_config {
  model_class: 'SSD'
  ssd {
    backbone {
      class_name: "mobilenet_v1"
      depth_multiplier: 1.0
      weight_decay: 5e-4
      output_stride: 16
    }
    ssd_head {
      num_classes: 20
      ssd_featuremap_layout {
        from_layer: 'Conv2d_11_pointwise'
        from_layer: 'Conv2d_13_pointwise'
        from_layer: ''
        from_layer: ''
        from_layer: ''
        from_layer: ''
        layer_depth: -1
        layer_depth: -1
        layer_depth: 512
        layer_depth: 512
        layer_depth: 256
        layer_depth: 256
      }
      conv_hyperparams {
        activation: RELU,
        regularizer {
          l2_regularizer {
            weight: 5e-4
          }
```

```
    }
    initializer {
      xavier_initializer {
        uniform : true
      }
    }
    batch_norm {
      decay: 0.9997
       center: true
       scale: true
       epsilon: 0.001
    }
  }
  box_coder {
    faster_rcnn_box_coder {
      y_scale: 10.0
      x_scale: 10.0
      height_scale: 5.0
      width_scale: 5.0
    }
  }
  matcher {
    argmax_matcher {
      matched_threshold: 0.5
      unmatched_threshold: 0.5
      ignore_thresholds: false
      negatives_lower_than_unmatched: true
      force_match_for_each_row: true
    }
  }
  similarity_calculator {
    iou_similarity {
    }
  }
  anchor_generator {
    ssd_anchor_generator {
      num_layers: 6
      #min_scale: 0.2
      #max_scale: 0.9
      #use caffe anchor scale, the last one is [0.88, 1.0]
      scales: 0.1
      scales: 0.2
      scales: 0.37
      scales: 0.54
      scales: 0.71
      scales: 0.88
      scales: 1.0
      aspect_ratios: 1.0
      aspect_ratios: 2.0
      aspect_ratios: 0.5
      aspect_ratios: 3.0
      aspect_ratios: 0.3333
      reduce_boxes_in_lowest_layer: true
      reduce_boxes_in_larger_layers: true
      interpolate_in_all_layers: true
    }
  }
  box_predictor {
    convolutional_box_predictor {
```

```
          min_depth: 0
          max_depth: 0
          num_layers_before_predictor: 0
          kernel_size: 3
          box_code_size: 4
          conv_hyperparams {
            activation: NONE,
            regularizer {
              l2_regularizer {
                weight: 5e-4
              }
            }
            initializer {
             xavier_initializer {
                uniform : true
              }
            }
          }
        }
      }
      post_processing {
        batch_non_max_suppression {
          score_threshold: 0.01
          iou_threshold: 0.45
          max_detections_per_class: 200
          max_total_detections: 200
        }
        score_converter: SOFTMAX
      }
      normalize_loss_by_num_matches: true
      loss {
        classification_loss {
          weighted_softmax {
          }
        }
        localization_loss {
          weighted_smooth_l1 {
          }
        }
        hard_example_miner {
          num_hard_examples: 3000
          iou_threshold: 0.99
          loss_type: BOTH
          max_negatives_per_positive: 3
          min_negatives_per_image: 0
        }
        classification_weight: 3.0
        localization_weight: 1.0
      }
    }
  }
}
train_config: {
  optimizer {
    momentum_optimizer: {
       learning_rate: {
         manual_step_learning_rate {
           initial_learning_rate: 0.001
           schedule {
```

```
              step: 80000
              learning_rate: 0.0001
            }
            schedule {
              step: 100000
              learning_rate: 0.00001
            }
          }
        }
        momentum_optimizer_value: 0.9
    }
  }
  #gradient_clipping_by_norm : 10.0
  fine_tune_checkpoint: "pretrained_models/mobilenet_v1_1.0_224/mobilenet_v1_1.0_224.ckpt"
  num_steps: 120000
  model_dir: 'experiments/ssd_mobilenet_v1_512'
  summary_model_vars: true
}
train_data: {
  input_path: "data/voc0712_tfrecord/VOC2007_train.tfrecord"
  input_path: "data/voc0712_tfrecord/VOC2012_train.tfrecord"
  input_path: "data/voc0712_tfrecord/VOC2007_val.tfrecord"
  input_path: "data/voc0712_tfrecord/VOC2012_val.tfrecord"
  batch_size: 32
  num_readers: 4
  shuffle: true
  read_block_length : 32
  voc_decoder_config {
    label_map_path: "data/voc0712_tfrecord/pascal_label_map.pbtxt"
  }
  #data argumentation
  data_augmentation_options {
    random_distort_color {
      color_ordering: 0
      fast_mode: true
    }
  }
  data_augmentation_options {
    random_pad_image{
      min_height_ratio: 1.1
      min_width_ratio: 1.1
      max_height_ratio: 4.0
      max_width_ratio: 4.0
      pad_color: 123.68
      pad_color: 116.779
      pad_color: 103.939
    }
  }
  data_augmentation_options {
    ssd_random_crop {
    }
  }
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    resize_image {
      new_height: 512
```

```
      new_width: 512
      method: BILINEAR
    }
  }
  data_augmentation_options {
    normalize_image {
      original_minval: 0.0
      original_maxval: 255.0
      target_minval: -1.0
      target_maxval: 1.0
    }
  }
}
eval_config: {
  num_examples: 4952
  # Note: The below line limits the evaluation process to 10 evaluations.
  # Remove the below line to evaluate indefinitely.
  # max_evals: 10
  metrics_set :  'pascal_voc07_detection_metrics'
  #metrics_set :  'coco_detection_metrics'
  visualize_groundtruth_boxes : true
  # num of visualizations to be displayed on tensorboard
  num_visualizations : 10
  # all the evaluation results will be saved to this dir if not ''
  visualization_export_dir: ''
  max_num_boxes_to_visualize: 20
  min_score_threshold: 0.5
  matching_iou_threshold: 0.5
  coco_analyze: true
}
eval_data : {
  input_path: "data/voc0712_tfrecord/VOC2007_test.tfrecord"
  batch_size: 1
  shuffle: false
  num_readers: 1
  voc_decoder_config {
    label_map_path: "data/voc0712_tfrecord/pascal_label_map.pbtxt"
  }
  data_augmentation_options {
    resize_image {
      new_height: 512
      new_width: 512
      method: BILINEAR
    }
  }
  data_augmentation_options {
    normalize_image {
      original_minval: 0.0
      original_maxval: 255.0
      target_minval: -1.0
      target_maxval: 1.0
    }
  }
}
export_config {
  batch_size: 1
}
```

yolo3

```
##-*- encoding:utf-8 -*-
# yolo3.config: encode the configs used in a yolo-v3 model
model_config {
  model_class: 'YOLO3'
  yolo {
    backbone {
      class_name: "darknet53"
    }
    yolo_head {
      num_classes: 20
      yolo_featuremap_layout {
        from_layer: 'conv4_res'
        from_layer: 'conv5_res'
        from_layer: 'conv6'
      }
      conv_hyperparams {
        activation: LEAKY_RELU
        regularizer {
          l2_regularizer {
            weight: 0.0005
          }
        }
        initializer {
          xavier_initializer {
            uniform : true
          }
        }
        batch_norm {
          decay: 0.997
          center: true
          scale: true
          epsilon: 1e-05
        }
      }
      anchor_generator {
        yolo_anchor_generator {
          anchor_group {
            anchor_size {
              width: 10
              height: 13
            }
            anchor_size {
              width: 16
              height: 30
            }
            anchor_size {
              width: 33
              height: 23
            }
          }
          anchor_group {
            anchor_size {
              width: 30
              height: 61
            }
            anchor_size {
              width: 62
```

```
            height: 45
          }
          anchor_size {
            width: 59
            height: 119
          }
        }
        anchor_group {
          anchor_size {
            width: 116
            height: 90
          }
          anchor_size {
            width: 156
            height: 198
          }
          anchor_size {
            width: 373
            height: 326
          }
        }
      }
      box_predictor {
        yolo_box_predictor {
          conv_hyperparams {
            activation: LEAKY_RELU
            regularizer {
              l2_regularizer {
                weight: 0.0005
              }
            }
            initializer {
             xavier_initializer {
                uniform : true
              }
            }
            batch_norm {
              decay: 0.997
              center: true
              scale: true
              epsilon: 1e-05
            }
          }
        }
      }
      post_processing {
        batch_non_max_suppression {
          score_threshold: 0.01
          iou_threshold: 0.45
          max_detections_per_class: 200
          max_total_detections: 200
        }
        score_converter: SIGMOID
      }
      loss {
        classification_loss {
          weighted_sigmoid {
          }
```

```
        }
        localization_loss {
          weighted_l2 {
          }
        }
        classification_weight: 1.0
        localization_weight: 1.0
      }
      ignore_threshold: 0.5
    }
  }
}
train_config: {
  optimizer {
    momentum_optimizer: {
       learning_rate: {
         manual_step_learning_rate {
           initial_learning_rate: 0.0000001
           schedule {
             step: 1000
             learning_rate: 0.001
           }
           schedule {
             step: 40000
             learning_rate: 0.0001
           }
           schedule {
             step: 45000
             learning_rate: 0.00001
           }
           warmup: true
         }
       }
       momentum_optimizer_value: 0.9
    }
  }
  #gradient_clipping_by_norm : 10.0
  num_steps: 50200
  fine_tune_checkpoint: "pretrained_models/darknet53/model.ckpt"
  model_dir: 'experiments/yolo/output/yolo3_voc'
  summary_model_vars: false
  sync_replicas: false
  train_distribute: "mirrored"
  num_gpus_per_worker: 8
}
train_data: {
  input_path: "data/voc0712_tfrecord/voc0712_part_*.tfrecord"
  batch_size: 8 #16
  num_readers: 4
  shuffle: true
  read_block_length : 32
  bucket_sizes: 10
  voc_decoder_config {
    label_map_path: "data/voc0712_tfrecord/pascal_label_map.pbtxt"
  }
  data_augmentation_options {
    random_distort_color {
    }
  }
```

```
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    random_pad_image {
      max_height_ratio: 1.6
      max_width_ratio: 1.6
    }
  }
  data_augmentation_options {
    random_crop_image {
      min_aspect_ratio: 0.25
      max_aspect_ratio: 4.0
      min_area: 0.1
      max_area: 1.0
    }
  }
  data_augmentation_options {
    normalize_image {
      original_minval: 0.0
      original_maxval: 255.0
      target_minval: 0.0
      target_maxval: 1.0
    }
  }
  data_augmentation_options {
    random_resize_image {
      new_heights: [320, 352, 384, 416, 448, 480, 512, 544, 576, 608]
      new_widths: [320, 352, 384, 416, 448, 480, 512, 544, 576, 608]
      method: BICUBIC
    }
  }
}
eval_config: {
  num_examples: 4952
  metrics_set :  'pascal_voc07_detection_metrics'
  visualize_groundtruth_boxes : true
  # num of visualizations to be displayed on tensorboard
  num_visualizations : 32
  # all the evaluation results will be saved to this dir if not ''
  visualization_export_dir: ''
  max_num_boxes_to_visualize: 32
  min_score_threshold: 0.5
  matching_iou_threshold: 0.5
}
eval_data : {
  input_path: "data/voc0712_tfrecord/VOC2007_test.tfrecord"
  batch_size: 1
  shuffle: false
  num_readers: 1
  voc_decoder_config {
    label_map_path: "data/voc0712_tfrecord/pascal_label_map.pbtxt"
  }
  data_augmentation_options {
    normalize_image {
      original_minval: 0.0
      original_maxval: 255.0
      target minval: 0.0
```

```
      target_maxval: 1.0
    }
  }
  data_augmentation_options {
    resize_image {
      new_height: 416
      new_width: 416
      method: BICUBIC
    }
  }
}
export_config {
  batch_size: 1
}
```

4.1.11.5.3. Image semantic segmentation

EasyVision that is provided by Machine Learning Platform for AI (PAI) allows you to set the training parameters in a configuration file to train a model. This topic provides examples of the configuration files for commonly used image semantic segmentation models. You can use the complete configurations or modify specific parameters as required to train the image semantic segmentation models.

Background information

The following models are involved in this topic:

- deeplab_v3+_r101_stage1
- deeplab_v3+_r101_stage2

deeplab_v3+_r101_stage1

```
#-*- encoding:utf-8 -*-
# step1 of deeplab configuration
train_config: {
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        poly_decay_learning_rate {
          learning_rate_base: 0.007
          total_steps: 30000
          power: 0.9
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  save_checkpoints_steps: 5000
  #fine_tune_checkpoint: "xception/model.ckpt"
  fine_tune_checkpoint: "pretrained_models/resnet_v1d_101/model.ckpt"
  num_steps: 30000
  model_dir: "pascal_deeplab_model"
}
train_data: {
  input_path: "data/pascal_voc_seg_aug/voc_ev_train.tfrecord"
  batch_size: 6
  num_readers: 4
  read_block_length: 1
```

```
  shuffle: true
  seg_decoder_config { }
  # note the augmentation order is important, so it cannot be changed
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
  data_augmentation_options {
    deeplab_random_crop {
      crop_size: 513
    }
  }
  data_augmentation_options {
    deeplab_random_horizontal_flip {
    }
  }
}
eval_config: {
#  num_examples: 100
  max_evals: 1000
  num_visualizations: 100
}
eval_data: {
  input_path: "data/pascal_voc_seg_aug/voc_ev_val.tfrecord"
  batch_size: 1
  shuffle: false
  num_readers: 1
  seg_decoder_config {
  }
  # note the augmentation order is important, so it cannot be changed
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
  num_epochs: 1
}
model_config: {
  model_class: 'DeepLab'
  deeplab {
    backbone {
        # class_name: 'xception_65'
        class_name: 'resnet_v1d_101'
        batchnorm_trainable: true
        weight_decay: 0.0005
        output_stride: 16
    }
    aspp_input_layer: 'resnet_v1d_101/block4'
    aspp_block {
      image_level_features: true
      batchnorm_trainable: true
      weight_decay: 1e-5
```

```
      feature_depth: 256
      atrous_rates: 6
      atrous_rates: 12
      atrous_rates: 18
      keep_prob: 0.9
    }
    seg_decoder_head {
      weight_decay: 1e-5
      batchnorm_trainable: true
      # input_layer: 'xception_65/entry_flow/block2/unit_1/xception_module/separable_conv2_pointwise
'
      input_layer: 'resnet_v1d_101/block1'
      decoder_depth: 256
      output_stride: 4
      num_classes: 21
    }
  }
}
```

deeplab_v3+_r101_stage2

```
#-*- encoding:utf-8 -*-
# step2 of deeplab configuration
train_config: {
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        poly_decay_learning_rate {
          learning_rate_base: 0.0002
          total_steps: 30000
          power: 0.9
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  save_checkpoints_steps: 5000
  fine_tune_checkpoint: "pascal_deeplab_model/model.ckpt-30000"
  num_steps: 30000
  model_dir: "pascal_deeplab_model_finetune"
}
train_data: {
  input_path: "pascal_voc_seg/train-00000-of-00004.tfrecord"
  input_path: "pascal_voc_seg/train-00001-of-00004.tfrecord"
  input_path: "pascal_voc_seg/train-00002-of-00004.tfrecord"
  input_path: "pascal_voc_seg/train-00003-of-00004.tfrecord"
  batch_size: 2
  num_readers: 4
  read_block_length: 1
  shuffle: true
  seg_decoder_config { }
  # note the augmentation order is important, so it cannot be changed
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
      means: 123.68
      means: 116.779
```

```
      means: 103.939
    }
  }
  data_augmentation_options {
    deeplab_random_crop {
      crop_size: 513
    }
  }
  data_augmentation_options {
    deeplab_random_horizontal_flip {
    }
  }
}
eval_config: {
#  num_examples: 100
  max_evals: 1000
  num_visualizations: 100
}
eval_data: {
  input_path: "pascal_voc_seg/val-00000-of-00004.tfrecord"
  input_path: "pascal_voc_seg/val-00001-of-00004.tfrecord"
  input_path: "pascal_voc_seg/val-00002-of-00004.tfrecord"
  input_path: "pascal_voc_seg/val-00003-of-00004.tfrecord"
  batch_size: 1
  shuffle: false
  num_readers: 1
  seg_decoder_config {
  }
  # note the augmentation order is important, so it cannot be changed
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
  num_epochs: 1
}
model_config: {
  model_class: 'DeepLab'
  deeplab {
    backbone {
        # class_name: 'xception_65'
        class_name: 'resnet_v1d_101'
        batchnorm_trainable: false
        weight_decay: 0.0005
        output_stride: 8
    }
    aspp_input_layer: 'resnet_v1d_101/block4'
    aspp_block {
      image_level_features: true
      batchnorm_trainable: false
      weight_decay: 1e-5
      feature_depth: 256
      atrous_rates: 12
      atrous_rates: 24
      atrous_rates: 36
      keep_prob: 0.9
```

```
    }
    seg_decoder_head {
      weight_decay: 1e-5
      batchnorm_trainable: false
      # input_layer: 'xception_65/entry_flow/block2/unit_1/xception_module/separable_conv2_pointwise
'
      input_layer: 'resnet_v1d_101/block1'
      decoder_depth: 256
      output_stride: 4
      num_classes: 21
    }
  }
}
```

4.1.11.5.4. Instance segmentation

EasyVision that is provided by Machine Learning Platform for AI (PAI) allows you to set the training parameters in a configuration file to train a model. This topic provides examples of the configuration files for commonly used instance segmentation models. You can use the complete configurations or modify specific parameters as required to train the instance segmentation models.

Background information

The following models are involved in this topic:

- mask_rcnn_r50
- mask_rcnn_r50_fpn

mask_rcnn_r50

```
#-*- encoding:utf-8 -*-
# mask_rcnn.config: mscoco mask rcnn model config
train_config: {
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        manual_step_learning_rate {
          initial_learning_rate: 0.00002
          schedule {
            step: 100
            learning_rate: 0.001
          }
          schedule {
            step: 240000
            learning_rate: .0001
          }
          schedule {
            step: 320000
            learning_rate: .00001
          }
          warmup: true
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  fine_tune_checkpoint: "pretrained_models/resnet_v1d_50/model.ckpt"
```

```
  num_steps: 360000
  model_dir: "experiments/coco_resnet50_maskrcnn_model"
}
train_data: {
  input_path: "data/coco_wmask/coco_train_*.tfrecord"
  batch_size: 1
  num_readers: 8
  read_block_length: 1
  shuffle: true
  shuffle_buffer_size: 512
  prefetch_size: 256
  voc_decoder_config {
    label_map_path: "data/coco_wmask/mscoco_label_map.pbtxt"
    load_instance_masks: true
    mask_format: PNG_MASK_FORMAT
  }
  # note the augmentation order is important, so it cannot be changed
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    random_resize_to_range {
      min_sizes: [640, 672, 704, 736, 768, 800]
      max_sizes: [1333, 1333, 1333, 1333, 1333, 1333]
    }
  }
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
  use_diff: false
}
eval_config: {
  num_examples: 5000
  num_visualizations: 16
  metrics_set: 'coco_detection_metrics'
  metrics_set: 'coco_mask_metrics'
  visualize_groundtruth_boxes: true
}
eval_data: {
  input_path: "data/coco_wmask/coco_val.tfrecord"
  batch_size: 1
  shuffle: false
  prefetch_size: 256
  num_readers: 1
  voc_decoder_config {
    label_map_path: "data/mscoco/mscoco_label_map.pbtxt"
    load_instance_masks: true
    mask_format: PNG_MASK_FORMAT
  }
  # note the augmentation order is important, so it cannot be changed
  data_augmentation_options {
    random_resize_to_range {
      min_sizes: 800
```

```
      max_sizes: 1333
    }
  }
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
  use_diff: true
}
export_config {
  batch_size: 1
}
model_config: {
  model_class: 'FasterRcnn'
  faster_rcnn {
    backbone {
      class_name: 'resnet_v1d_50'
      batchnorm_trainable: false
      weight_decay: 0.0001
      output_stride: 16
    }
    rpn_head {
      input_layer: 'resnet_v1d_50/block3'
      box_predictor {
        convolutional_box_predictor {
          conv_hyperparams {
            op: CONV
            regularizer {
              l2_regularizer {
                weight: 0.0001
              }
            }
            initializer {
              truncated_normal_initializer {
                stddev: 0.01
              }
            }
          }
          min_depth: 512
          max_depth: 512
          num_layers_before_predictor: 1
          kernel_size: 3
        }
      }
      first_stage_minibatch_size: 256
      first_stage_positive_balance_fraction: 0.5
      first_stage_nms_iou_threshold: 0.7
      first_stage_max_proposals: 2000
      rpn_min_size: 0
      first_stage_anchor_generator {
        # the default base anchor size is 256
        grid_anchor_generator {
          scales: [0.125, 0.25, 0.5, 1.0, 2.0]
          aspect_ratios: [0.5, 1.0, 2.0]
          height_stride: 16
```

```
        width_stride: 16
      }
    }
  }
  region_feature_extractor {
    resnet_block {
      class_name: 'resnet_v1d_50' #the name of backbone
      block_name: 'block4'  #the last residual block of resnet_v1d_50
      stride: 1
      weight_decay: 0.0001
    }
  }
  rcnn_head {
    input_layer: 'resnet_v1d_50/block3'
    initial_crop_size: 14
    maxpool_kernel_size: 2
    maxpool_stride: 2
    num_classes: 90
    second_stage_box_predictor {
      mask_rcnn_box_predictor {
        fc_hyperparams {
          op: FC
          regularizer {
            l2_regularizer {
              weight: 0.0001
            }
          }
          initializer {
            xavier_initializer {
            }
          }
        }
       agnostic: true
     }
    }
    nms_config {
      score_threshold: 0.05
      iou_threshold: 0.5
      max_detections_per_class: 100
      max_total_detections: 100
    }
    second_stage_batch_size: 512
    second_stage_balance_fraction: 0.25
  }
  mrcnn_head {
    input_layer: 'resnet_v1d_50/block3'
    initial_crop_size: 14
    maxpool_kernel_size: 2
    maxpool_stride: 2
    num_classes: 90
    third_stage_mask_predictor {
      mask_rcnn_mask_predictor {
        conv_hyperparams {
          op: CONV
          regularizer {
            l2_regularizer {
              weight: 0.0001
            }
          }
```

```
            initializer {
              variance_scaling_initializer {
              }
            }
          }
          mask_height: 14
          mask_width: 14
          mask_prediction_conv_depth: 256
          mask_prediction_num_conv_layers: 1
          convolve_then_upsample_masks: true
        }
      }
    }
  }
}
```

mask_rcnn_r50_fpn

```
#-*- encoding:utf-8 -*-
# mask_rcnn_fpn.config: coco_wmask mask rcnn fpn model config
train_config: {
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        manual_step_learning_rate {
          initial_learning_rate: 0.00002
          schedule {
            step: 100
            learning_rate: 0.001
          }
          schedule {
            step: 240000
            learning_rate: .0001
          }
          schedule {
            step: 320000
            learning_rate: .00001
          }
          warmup: true
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  fine_tune_checkpoint: "pretrained_models/resnet_v1d_50/model.ckpt"
  num_steps: 360000
  model_dir: "experiments/coco_resnet50_maskrcnn_model_fpn"
}
train_data: {
  input_path: "data/coco_wmask/coco_train_*.tfrecord"
  batch_size: 1
  num_readers: 8
  read_block_length: 1
  shuffle: true
  shuffle_buffer_size: 512
  prefetch_size: 256
  voc_decoder_config {
```

```
    label_map_path: "data/coco_wmask/mscoco_label_map.pbtxt"
    load_instance_masks: true
    mask_format: PNG_MASK_FORMAT
  }
  # note the augmentation order is important, so it cannot be changed
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    random_resize_to_range {
      min_sizes: [640, 672, 704, 736, 768, 800]
      max_sizes: [1333, 1333, 1333, 1333, 1333, 1333]
    }
  }
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
  use_diff: false
}
eval_config: {
  num_examples: 5000
  num_visualizations: 16
  metrics_set: 'coco_detection_metrics'
  metrics_set: 'coco_mask_metrics'
  visualize_groundtruth_boxes: true
}
eval_data: {
  input_path: "data/coco_wmask/coco_val.tfrecord"
  batch_size: 1
  shuffle: false
  prefetch_size: 256
  num_readers: 1
  voc_decoder_config {
    label_map_path: "data/coco_wmask/mscoco_label_map.pbtxt"
    load_instance_masks: true
    mask_format: PNG_MASK_FORMAT
  }
  # note the augmentation order is important, so it cannot be changed
  data_augmentation_options {
    random_resize_to_range {
      min_sizes: 800
      max_sizes: 1333
    }
  }
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
  use_diff: true
```

```
}
export_config {
  batch_size: 1
}
model_config: {
  model_class: 'FasterRcnn'
  faster_rcnn {
    backbone {
      class_name: 'resnet_v1d_50'
      batchnorm_trainable: false
      weight_decay: 0.0001
    }
    fpn {
      input: 'resnet_v1d_50/block1'
      input: 'resnet_v1d_50/block2'
      input: 'resnet_v1d_50/block3'
      input: 'resnet_v1d_50/block4'
      fea_dim: 256
      extra_conv_layers: 1
      roi_min_level: 2
      roi_max_level: 5
      conv_hyperparams {
        op: CONV
        regularizer {
          l2_regularizer {
            weight: 0.0001
          }
        }
        initializer {
          truncated_normal_initializer {
            stddev: 0.01
          }
        }
      }
    }
    rpn_head {
      # if input_layer is not specified, will use fpn features,
      # which all have "FPN/" prefix
      box_predictor {
        weight_shared_convolutional_box_predictor {
          conv_hyperparams {
            op: CONV
            regularizer {
              l2_regularizer {
                weight: 0.0001
              }
            }
            initializer {
              truncated_normal_initializer {
                stddev: 0.01
              }
            }
          }
          depth: 256
          num_layers_before_predictor: 1
          kernel_size: 3
        }
      }
      first_stage_minibatch_size: 256
```

```
    first_stage_positive_balance_fraction: 0.5
    first_stage_nms_iou_threshold: 0.7
    first_stage_max_proposals: 2000
    rpn_min_size: 16
    first_stage_anchor_generator {
      # anchor_size = anchor_scale * feature_map_stride
      multiscale_anchor_generator {
        min_level: 2
        max_level: 6
        anchor_scale: 8
        aspect_ratios: 0.5
        aspect_ratios: 1
        aspect_ratios: 2
        normalize_coordinates: false
        scales_per_octave: 1
      }
    }
  }
  rcnn_head {
    initial_crop_size: 14
    maxpool_kernel_size: 2
    maxpool_stride: 2
    num_classes: 90
    second_stage_box_predictor {
      mask_rcnn_box_predictor {
        num_layers_before_predictor: 2
        depth: 1024
        fc_hyperparams {
          op: FC
          regularizer {
            l2_regularizer {
              weight: 0.0001
            }
          }
          initializer {
            xavier_initializer {
            }
          }
        }
       agnostic: true
     }
    }
    nms_config {
      score_threshold: 0.05
      iou_threshold: 0.5
      max_detections_per_class: 100
      max_total_detections: 100
    }
    second_stage_batch_size: 512
    second_stage_balance_fraction: 0.25
  }
  mrcnn_head {
    initial_crop_size: 28
    maxpool_kernel_size: 2
    maxpool_stride: 2
    num_classes: 90
    third_stage_mask_predictor {
      mask_rcnn_mask_predictor {
        conv_hyperparams {
```

```
            op: CONV
            regularizer {
              l2_regularizer {
                weight: 0.0001
              }
            }
            initializer {
              variance_scaling_initializer {
              }
            }
          }
          mask_height: 28
          mask_width: 28
          mask_prediction_conv_depth: 256
          mask_prediction_num_conv_layers: 5
          convolve_then_upsample_masks: true
        }
      }
    }
  }
}
```

4.1.11.5.5. Text detection

EasyVision that is provided by Machine Learning Platform for AI (PAI) allows you to set the training parameters in a configuration file to train a model. This topic provides examples of the configuration files for the commonly used text detection model. You can use the complete configurations or modify specific parameters as required to train the text detection model.

text_krcnn_r50_fpn

```
#-*- encoding:utf-8 -*-
# text_krcnn_resnet50.config:
# icdar text krcnn model training config
train_config: {
  optimizer {
    adam_optimizer: {
      learning_rate: {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.00001
          decay_steps: 150000
          decay_factor: 0.5
          min_learning_rate: 0.0000001
        }
      }
    }
    use_moving_average: false
  }
  # gradient_clipping_by_norm: 0.0
  fine_tune_checkpoint: "pretrained_models/resnet_v1d_50/model.ckpt"
  num_steps: 400000
  model_dir: "experiments/icdar_ch4/text_krcnn_resnet50_fpn"
  save_checkpoints_steps: 2000
  save_summary_steps: 100
  log_step_count_steps: 100
  summary_model_vars: false
}
train_data: {
```

```
train_data: {
  input_path: "data/icdar_detection_tfrecords/icdar_training_*.tfrecord"
  batch_size: 1
  shuffle: true
  shuffle_buffer_size: 64
  prefetch_size: 64
  num_readers: 8
  text_detection_decoder_config {
    label_map_path: "data/icdar_detection_tfrecords/label_map.pbtxt"
  }
  data_augmentation_options {
    random_jitter_aspect_ratio {
      min_jitter_coef: 0.8
      max_jitter_coef: 1.2
    }
  }
  data_augmentation_options {
    random_rotation {
      min_angle: -10
      max_angle: 10
      use_keypoints_calc_boxes: true
    }
  }
  data_augmentation_options {
    random_resize_to_range {
      min_sizes: 640
      max_sizes: 2000
      min_sizes: 800
      max_sizes: 2000
      min_sizes: 960
      max_sizes: 2000
      min_sizes: 1120
      max_sizes: 2000
    }
  }
  data_augmentation_options {
    random_distort_color {
    }
  }
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
  use_diff: false
}
eval_config: {
  num_examples: 500
  num_visualizations : 16
  metrics_set: "icdar_detection_metrics"
  visualization_export_dir: ''
}
eval_data: {
  input_path: "data/icdar_detection_tfrecords/icdar-ch4-test.tfrecord"
  batch_size: 1
  shuffle: false
  prefetch_size: 32
```

```
  prefetch_size: 32
  text_detection_decoder_config {
    label_map_path: "data/icdar_detection_tfrecords/label_map.pbtxt"
  }
  # note the augmentation order is important, so it cannot be changed
  data_augmentation_options {
    random_resize_to_range {
      min_sizes: 960
      max_sizes: 2000
    }
  }
  data_augmentation_options {
    subtract_channel_mean {
      # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
  use_diff: true
}
export_config {
  batch_size: 1
}
model_config: {
  model_class: 'TextKRCNN'
  text_krcnn {
    backbone {
        class_name: 'resnet_v1d_50'
        batchnorm_trainable: false
        weight_decay: 0.0001
    }
    fpn {
      input: 'resnet_v1d_50/block1'
      input: 'resnet_v1d_50/block2'
      input: 'resnet_v1d_50/block3'
      input: 'resnet_v1d_50/block4'
      fea_dim: 256
      extra_conv_layers: 1
      roi_min_level: 2
      roi_max_level: 5
      roi_canonical_scale: 168
      roi_canonical_level: 4
      conv_hyperparams {
        op: CONV
        regularizer {
          l2_regularizer {
            weight: 0.0001
          }
        }
        initializer {
          truncated_normal_initializer {
            stddev: 0.01
          }
        }
      }
    }
    rpn_head {
      # if input_layer is not specified, will use fpn features,
      # which all have "FPN/" prefix
```

```
    # which all have "RPN" prefix
    box_predictor {
      weight_shared_convolutional_box_predictor {
        conv_hyperparams {
          op: CONV
          regularizer {
            l2_regularizer {
              weight: 0.0001
            }
          }
          initializer {
            truncated_normal_initializer {
              stddev: 0.01
            }
          }
        }
        depth: 256
        num_layers_before_predictor: 1
        kernel_size: 3
      }
    }
    first_stage_minibatch_size: 256
    first_stage_positive_balance_fraction: 0.5
    first_stage_nms_iou_threshold: 0.7
    first_stage_max_proposals: 300
    rpn_min_size: 8
    first_stage_anchor_generator {
      multiscale_anchor_generator {
        min_level: 2
        max_level: 6
        anchor_scale: 6
        aspect_ratios: [0.2, 0.5, 1, 2, 5]
        normalize_coordinates: false
        scales_per_octave: 1
      }
    }
  }
  rcnn_head {
    initial_crop_size: 14
    maxpool_kernel_size: 2
    maxpool_stride: 2
    num_classes: 1
    second_stage_box_predictor {
      mask_rcnn_box_predictor {
        num_layers_before_predictor: 2
        depth: 1024
        fc_hyperparams {
          op: FC
          regularizer {
            l2_regularizer {
              weight: 0.0001
            }
          }
          initializer {
            truncated_normal_initializer {
              stddev: 0.01
            }
          }
        }
        agnostic: true
```

```
        agnostic: true
      }
    }
    hard_example_miner {
      num_hard_examples: 128
      iou_threshold: 0.99
      loss_type: BOTH
    }
    nms_config {
      score_threshold: 0.7
      iou_threshold: 0.3
      max_detections_per_class: 400
      max_total_detections: 400
    }
    second_stage_batch_size: 128
    second_stage_balance_fraction: 0.25
  }
  keypoint_head {
    keypoint_predictor {
      text_resnet_keypoint_predictor {
        conv_hyperparams {
          op: CONV
          regularizer {
            l2_regularizer {
              weight: 0.0001
            }
          }
          initializer {
            variance_scaling_initializer {
            }
          }
        }
        fc_hyperparams {
          op: FC
          regularizer {
            l2_regularizer {
               weight: 0.0001
            }
          }
          initializer {
            variance_scaling_initializer {
            }
          }
        }
      }
    }
    initial_crop_size: 28
    maxpool_kernel_size: 2
    maxpool_stride: 2
    num_keypoints: 4
    predict_direction: false
    direction_trainable: false
  }
 }
}
```

4.1.11.5.6. Text recognition

EasyVision that is provided by Machine Learning Platform for AI (PAI) allows you to set the training parameters in a configuration file to train a model. This topic provides examples of the configuration files for commonly used text recognition models. You can use the complete configurations or modify specific parameters as required to train the text recognition models.

Background information

The following models are involved in this topic:

- crnn_ctc_r15
- crnn_attention_r15
- crnn_mono_attention_r15
- cnn_spatial_attention_r15
- transformer_ocr

crnn_ctc_r15

```
##-*- encoding:utf-8 -*-
# text crnn ctc config for recipt text
train_config: {
  optimizer {
    adam_optimizer: {
      learning_rate: {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.0001
          decay_steps: 30000
          decay_factor: 0.7
        }
      }
    }
    use_moving_average: false
  }
  # gradient_clipping_by_norm : 10.0
  num_steps: 1000000
  model_dir: 'experiments/recipt_text/crnn_ctc_resnet15_fixed_height_wopretrain'
  save_checkpoints_steps: 2000
  save_summary_steps: 100
  log_step_count_steps: 100
  summary_model_vars: false
  # for distributed training only
  # sync_replicas: false
  # replicas_to_aggregate: 8
  # num_worker_replicas: 8
}
train_data: {
  input_path: "data/recipt_text/recognition_tfrecords/train_*.tfrecord"
  batch_size: 64
  shuffle: true
  num_readers: 8
  text_recognition_decoder_config {
    char_dict_path: "data/recipt_text/recognition_tfrecords/char_dict"
    min_input_ratio: 0.125
    max_input_ratio: 38
    num_buckets: 10
  }
  data_augmentation_options {
    resize_image_with_fixed_height {
      new_height: 32
```

```
    }
  }
  data_augmentation_options {
    random_distort_color {
    }
  }
  data_augmentation_options {
    rgb_to_gray {
    }
  }
  data_augmentation_options {
    normalize_image {
      original_minval: 0
      original_maxval: 255
      target_minval: 0
      target_maxval: 1
    }
  }
}
eval_config: {
  num_visualizations : 16
}
eval_data: {
  input_path: "data/recipt_text/recognition_tfrecords/test.tfrecord"
  batch_size: 64
  shuffle: false
  text_recognition_decoder_config {
    char_dict_path: "data/recipt_text/recognition_tfrecords/char_dict"
    min_input_ratio: 0.125
    max_input_ratio: 100
    num_buckets: 10
  }
  data_augmentation_options {
    resize_image_with_fixed_height {
      new_height: 32
    }
  }
  data_augmentation_options {
    rgb_to_gray {
    }
  }
  data_augmentation_options {
    normalize_image {
      original_minval: 0
      original_maxval: 255
      target_minval: 0
      target_maxval: 1
    }
  }
  num_epochs: 1
}
export_config {
  batch_size: -1
}
model_config {
  model_class: 'TextRecognition'
  text_recognition {
    backbone {
      class_name: 'text_resnet15'
```

```
    }
    ctc_head {
      input_layer: 'text_resnet15/conv5_0'
      crnn_encoder {
        num_layers: 2
        basic_lstm {
          num_units: 512
        }
        encoder_type: UNI
      }
      ctc_decoder {
      }
    }
  }
}
```

crnn_attention_r15

```
##-*- encoding:utf-8 -*-
# text recognition config for recipt text
train_config: {
  optimizer {
    adam_optimizer: {
      learning_rate: {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.0001
          decay_steps: 30000
          decay_factor: 0.7
        }
      }
    }
    use_moving_average: false
  }
  # gradient_clipping_by_norm : 10.0
  num_steps: 1000000
  fine_tune_checkpoint: "pretrained_models/general_crnn_attn_resnet15/model.ckpt"
  model_dir: 'experiments/recipt_text/crnn_attn_resnet15_fixed_height'
  save_checkpoints_steps: 2000
  save_summary_steps: 100
  log_step_count_steps: 100
  summary_model_vars: false
  # for distributed training only
  # sync_replicas: false
  # replicas_to_aggregate: 8
  # num_worker_replicas: 8
}
train_data: {
  input_path: "data/recipt_text/recognition_tfrecords/train_*.tfrecord"
  batch_size: 64
  shuffle: true
  num_readers: 8
  text_recognition_decoder_config {
    char_dict_path: "data/recipt_text/recognition_tfrecords/char_dict"
    min_input_ratio: 0.125
    max_input_ratio: 38
    num_buckets: 10
  }
  data_augmentation_options {
    resize_image_with_fixed_height {
```

```
    resize_image_with_fixed_height {
      new_height: 32
    }
  }
  data_augmentation_options {
    random_distort_color {
    }
  }
  data_augmentation_options {
    rgb_to_gray {
    }
  }
  data_augmentation_options {
    normalize_image {
      original_minval: 0
      original_maxval: 255
      target_minval: 0
      target_maxval: 1
    }
  }
}
eval_config: {
  num_visualizations : 16
}
eval_data: {
  input_path: "data/recipt_text/recognition_tfrecords/test.tfrecord"
  batch_size: 64
  shuffle: false
  text_recognition_decoder_config {
    char_dict_path: "data/recipt_text/recognition_tfrecords/char_dict"
    min_input_ratio: 0.125
    max_input_ratio: 100
    num_buckets: 10
  }
  data_augmentation_options {
    resize_image_with_fixed_height {
      new_height: 32
    }
  }
  data_augmentation_options {
    rgb_to_gray {
    }
  }
  data_augmentation_options {
    normalize_image {
      original_minval: 0
      original_maxval: 255
      target_minval: 0
      target_maxval: 1
    }
  }
  num_epochs: 1
}
export_config {
  batch_size: -1
}
model_config {
  model_class: 'TextRecognition'
  text_recognition {
    backbone {
```

```
    backbone {
      class_name: 'text_resnet15'
    }
    attention_head {
      input_layer: 'text_resnet15/conv5_0'
      crnn_encoder {
        num_layers: 2
        basic_lstm {
          num_units: 512
        }
        encoder_type: UNI
      }
      attention_decoder {
        embedding_size: 64
        num_layers: 2
        basic_lstm {
          num_units: 512
        }
        attention_mechanism: "normed_bahdanau"
        # visualize_type: "line"
      }
    }
  }
}
```

crnn_mono_attention_r15

```
##-*- encoding:utf-8 -*-
# text recognition config for recipt text
train_config: {
  optimizer {
    adam_optimizer: {
      learning_rate: {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.0001
          decay_steps: 30000
          decay_factor: 0.7
        }
      }
    }
    use_moving_average: false
  }
  # gradient_clipping_by_norm : 10.0
  num_steps: 1000000
  fine_tune_checkpoint: "pretrained_models/general_crnn_mono_norm_attn_resnet15_xxlarge/model.ckpt"
  model_dir: 'experiments/recipt_text/crnn_mono_norm_attn_resnet15_fixed_height_xxlarge_dict'
  save_checkpoints_steps: 2000
  save_summary_steps: 100
  log_step_count_steps: 100
  summary_model_vars: false
  # for distributed training only
  # sync_replicas: false
  # replicas_to_aggregate: 8
  # num_worker_replicas: 8
}
train_data: {
  input_path: "data/recipt_text/recognition_tfrecords/train_*.tfrecord"
  batch_size: 64
  shuffle: true
```

```
  num_readers: 8
  text_recognition_decoder_config {
    char_dict_path: "data/recipt_text/recognition_tfrecords/char_dict_xxlarge"
    min_input_ratio: 0.125
    max_input_ratio: 38
    num_buckets: 10
  }
  data_augmentation_options {
    resize_image_with_fixed_height {
      new_height: 32
    }
  }
  data_augmentation_options {
    random_distort_color {
    }
  }
  data_augmentation_options {
    random_rgb_to_gray {
      probability: 0.2
    }
  }
  data_augmentation_options {
    subtract_channel_mean {
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
}
eval_config: {
  num_visualizations : 16
}
eval_data: {
  input_path: "data/recipt_text/recognition_tfrecords/test.tfrecord"
  batch_size: 64
  shuffle: false
  text_recognition_decoder_config {
    char_dict_path: "data/recipt_text/recognition_tfrecords/char_dict_xxlarge"
    min_input_ratio: 0.125
    max_input_ratio: 100
    num_buckets: 10
  }
  data_augmentation_options {
    resize_image_with_fixed_height {
      new_height: 32
    }
  }
  data_augmentation_options {
    subtract_channel_mean {
      means: 123.68
      means: 116.779
      means: 103.939
    }
  }
  num_epochs: 1
}
export_config {
  batch_size: -1
}
```

```
model_config {
  model_class: 'TextRecognition'
  text_recognition {
    backbone {
      class_name: 'text_resnet15'
      batchnorm_trainable: true
      weight_decay: 0.00001
    }
    attention_head {
      input_layer: 'text_resnet15/conv5_0'
      crnn_encoder {
        num_layers: 2
        layer_norm_basic_lstm {
          num_units: 512
        }
        encoder_type: UNI
      }
      attention_decoder {
        embedding_size: 256
        num_layers: 2
        layer_norm_basic_lstm {
          num_units: 512
        }
        attention_mechanism: "monotonic_normed_bahdanau"
        # visualize_type: "line"
      }
    }
  }
}
```

cnn_spatial_attention_r15

```
##-*- encoding:utf-8 -*-
# text cnn spatial attention config for recipt text
train_config: {
  optimizer {
    adam_optimizer: {
      learning_rate: {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.0001
          decay_steps: 30000
          decay_factor: 0.7
        }
      }
    }
    use_moving_average: false
  }
  # gradient_clipping_by_norm : 10.0
  num_steps: 1000000
  model_dir: 'experiments/recipt_text/cnn_spatial_attn_resnet15_fixed_height_wopretrain'
  save_checkpoints_steps: 2000
  save_summary_steps: 100
  log_step_count_steps: 100
  summary_model_vars: false
  # for distributed training only
  # sync_replicas: false
  # replicas_to_aggregate: 8
  # num_worker_replicas: 8
```

```
}
train_data: {
  input_path: "data/recipt_text/recognition_tfrecords/train_*.tfrecord"
  batch_size: 24
  shuffle: true
  num_readers: 8
  text_recognition_decoder_config {
    char_dict_path: "data/recipt_text/recognition_tfrecords/char_dict"
    min_input_ratio: 0.125
    max_input_ratio: 38
    num_buckets: 10
  }
  data_augmentation_options {
    resize_image_with_fixed_height {
      new_height: 64
    }
  }
  data_augmentation_options {
    random_distort_color {
    }
  }
  data_augmentation_options {
    rgb_to_gray {
    }
  }
  data_augmentation_options {
    normalize_image {
      original_minval: 0
      original_maxval: 255
      target_minval: 0
      target_maxval: 1
    }
  }
}
eval_config: {
  num_visualizations : 16
}
eval_data: {
  input_path: "data/recipt_text/recognition_tfrecords/test.tfrecord"
  batch_size: 24
  shuffle: false
  text_recognition_decoder_config {
    char_dict_path: "data/recipt_text/recognition_tfrecords/char_dict"
    min_input_ratio: 0.125
    max_input_ratio: 100
    num_buckets: 10
  }
  data_augmentation_options {
    resize_image_with_fixed_height {
      new_height: 64
    }
  }
  data_augmentation_options {
    rgb_to_gray {
    }
  }
  data_augmentation_options {
    normalize_image {
      original_minval: 0
```

```
      original_maxval: 255
      target_minval: 0
      target_maxval: 1
    }
  }
  num_epochs: 1
}
export_config {
  batch_size: -1
}
model_config {
  model_class: 'TextRecognition'
  text_recognition {
    backbone {
      class_name: 'text_resnet15'
    }
    attention_head {
      input_layer: 'text_resnet15/conv5_0'
      cnn_spatial_encoder {
      }
      attention_decoder {
        embedding_size: 64
        num_layers: 2
        basic_lstm {
          num_units: 512
        }
        attention_mechanism: "normed_bahdanau"
        pass_hidden_state: false
        visualize_type: "spatial"
      }
    }
  }
}
```

transformer_ocr

```
#-*- encoding:utf-8 -*-
# transformer_f1024_e12d4.config:
# Synth90k text recognition model (Transformer) training config
train_config: {
  optimizer {
    adam_optimizer: {
      learning_rate: {
        transformer_learning_rate {
          learning_rate_base: 2
          hidden_size: 512
          warmup_steps: 8000
        }
      }
    }
    use_moving_average: false
  }
  # gradient_clipping_by_norm: 10.0
  num_steps: 1000000
  model_dir: "experiments/synth90k/output/transformer_f1024_e12d4"
  save_checkpoints_steps: 2000
  save_summary_steps: 100
  log_step_count_steps: 100
```

```
  summary_model_vars: false
}
train_data: {
  input_path: "data/Synth90k_tfrecords/Synth90k_train_*.tfrecord"
  batch_size: 512
  shuffle: true
  num_readers: 8
  text_recognition_decoder_config {
    char_dict_path: "data/Synth90k_tfrecords/char_dict"
  }
  data_augmentation_options {
    resize_image {
      new_height: 32
      new_width: 100
    }
  }
  data_augmentation_options {
    random_distort_color {
    }
  }
  data_augmentation_options {
    rgb_to_gray {
    }
  }
  data_augmentation_options {
    normalize_image {
      original_minval: 0
      original_maxval: 255
      target_minval: -1
      target_maxval: 1
    }
  }
}
eval_config: {
  num_examples: 31232
  num_visualizations : 16
}
eval_data: {
  input_path: "data/Synth90k_tfrecords/Synth90k_test.tfrecord"
  batch_size: 512
  shuffle: false
  text_recognition_decoder_config {
    char_dict_path: "data/Synth90k_tfrecords/char_dict"
  }
  data_augmentation_options {
    resize_image {
      new_height: 32
      new_width: 100
    }
  }
  data_augmentation_options {
    rgb_to_gray {
    }
  }
  data_augmentation_options {
    normalize_image {
      original_minval: 0
      original_maxval: 255
      target_minval: -1
```

```
      target_maxval: 1
    }
  }
}
export_config {
  batch_size: -1
}
model_config {
  model_class: 'TextRecognition'
  text_recognition {
    transformer_head {
      input_layer: 'image'
      transformer_encoder {
        num_layers: 12
        hidden_size: 512
        num_heads: 8
        filter_size: 1024
        layer_postprocess_dropout: 0.1
        attention_dropout: 0.1
        relu_dropout: 0.1
        pooling_rate: 4
      }
      transformer_decoder {
        num_layers: 4
        hidden_size: 512
        num_heads: 8
        filter_size: 1024
        layer_postprocess_dropout: 0.1
        attention_dropout: 0.1
        relu_dropout: 0.1
      }
    }
  }
}
```

4.1.11.5.7. End-to-End text recognition

EasyVision that is provided by Machine Learning Platform for AI (PAI) allows you to set the training parameters in a configuration file to train a model. This topic provides examples of the configuration files for the commonly used end-to-end text recognition model. You can use the complete configurations or modify specific parameters as required to train the end-to-end text recognition model.

text_end2end_krcnn_attention

```
#-*- encoding:utf-8 -*-
# text_end2end_krcnn_attn_dis.config:
# receipt text text end2end model training config
train_config: {
  optimizer {
    adam_optimizer: {
      learning_rate: {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.0001
          decay_steps: 50000
          decay_factor: 0.5
          min_learning_rate: 0.000001
        }
      }
```

```
    }
    use_moving_average: false
  }
  # gradient_clipping_by_norm: 0.0
  fine_tune_checkpoint: "pretrained_models/general_text_end2end_krcnn_attn_resnet50/model.ckpt"
  num_steps: 1000000
  model_dir: "experiments/recipt_text/text_end2end_krcnn_resnet50_attn"
  save_checkpoints_steps: 2000
  save_summary_steps: 100
  log_step_count_steps: 100
  summary_model_vars: false
  # for distributed training only
  # sync_replicas: false
  # replicas_to_aggregate: 8
  # num_worker_replicas: 8
}
train_data: {
  input_path: "data/recipt_text/end2end_tfrecords/train_*.tfrecord"
  batch_size: 1
  shuffle: true
  shuffle_buffer_size: 64
  prefetch_size: 64
  num_readers: 8
  text_end2end_decoder_config {
    char_dict_path: "data/recipt_text/end2end_tfrecords/char_dict"
    label_map_path: "data/recipt_text/end2end_tfrecords/label_map.pbtxt"
  }
  data_augmentation_options {
    random_jitter_aspect_ratio {
      min_jitter_coef: 0.8
      max_jitter_coef: 1.2
    }
  }
  data_augmentation_options {
    random_rotation90 {
    }
  }
  data_augmentation_options {
    random_rotation {
      min_angle: -10
      max_angle: 10
      use_keypoints_calc_boxes: true
    }
  }
  data_augmentation_options {
    random_resize_to_range {
      min_sizes: 640
      max_sizes: 1440
      min_sizes: 800
      max_sizes: 1440
      min_sizes: 960
      max_sizes: 1440
    }
  }
  data_augmentation_options {
    random_distort_color {
    }
  }
  data_augmentation_options {
```

```
    data_augmentation_options {
      subtract_channel_mean {
        # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
        means: 123.68
        means: 116.779
        means: 103.939
      }
    }
    use_diff: false
  }
  eval_config: {
    num_examples: 299
    num_visualizations : 16
    visualization_export_dir: ''
    metrics_set: "icdar_end2end_metrics"
  }
  eval_data: {
    input_path: "data/recipt_text/end2end_tfrecords/test.tfrecord"
    batch_size: 1
    shuffle: false
    prefetch_size: 32
    text_end2end_decoder_config {
      char_dict_path: "data/recipt_text/end2end_tfrecords/char_dict"
      label_map_path: "data/recipt_text/end2end_tfrecords/label_map.pbtxt"
    }
    # note the augmentation order is important, so it cannot be changed
    data_augmentation_options {
      random_resize_to_range {
        min_sizes: 800
        max_sizes: 1440
      }
    }
    data_augmentation_options {
      subtract_channel_mean {
        # see https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/fast_rcnn/config.py#L181
        means: 123.68
        means: 116.779
        means: 103.939
      }
    }
    use_diff: true
  }
  export_config {
    batch_size: 1
  }
  model_config: {
    model_class: 'TextEnd2End'
    text_end2end {
      backbone {
          class_name: 'resnet_v1_50'
          batchnorm_trainable: false
          weight_decay: 0.0001
      }
      fpn {
        input: 'resnet_v1_50/block1'
        input: 'resnet_v1_50/block2'
        input: 'resnet_v1_50/block3'
        input: 'resnet_v1_50/block4'
        fea_dim: 256
        extra_conv_layers: 1
```

```
    roi_min_level: 2
    roi_max_level: 5
    roi_canonical_scale: 168
    roi_canonical_level: 4
    conv_hyperparams {
      op: CONV
      regularizer {
        l2_regularizer {
          weight: 0.0001
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.01
        }
      }
    }
  }
  rpn_head {
    # if input_layer is not specified, will use fpn features,
    # which all have "FPN/" prefix
    box_predictor {
      weight_shared_convolutional_box_predictor {
        conv_hyperparams {
          op: CONV
          regularizer {
            l2_regularizer {
              weight: 0.0001
            }
          }
          initializer {
            truncated_normal_initializer {
              stddev: 0.01
            }
          }
        }
        depth: 256
        num_layers_before_predictor: 1
        kernel_size: 3
      }
    }
    first_stage_minibatch_size: 256
    first_stage_positive_balance_fraction: 0.5
    first_stage_nms_iou_threshold: 0.7
    first_stage_max_proposals: 300
    rpn_min_size: 8
    first_stage_anchor_generator {
      multiscale_anchor_generator {
        min_level: 2
        max_level: 6
        anchor_scale: 6
        aspect_ratios: [0.2, 0.5, 1, 2, 5]
        normalize_coordinates: false
        scales_per_octave: 1
      }
    }
  }
  rcnn_head {
    initial_crop_size: 14
```

```
    initial_crop_size: 14
    maxpool_kernel_size: 2
    maxpool_stride: 2
    num_classes: 1
    second_stage_box_predictor {
      mask_rcnn_box_predictor {
        num_layers_before_predictor: 2
        depth: 1024
        fc_hyperparams {
          op: FC
          regularizer {
            l2_regularizer {
              weight: 0.0001
            }
          }
          initializer {
            truncated_normal_initializer {
              stddev: 0.01
            }
          }
        }
        agnostic: true
      }
    }
    hard_example_miner {
      num_hard_examples: 128
      iou_threshold: 0.99
      loss_type: BOTH
    }
    nms_config {
      score_threshold: 0.7
      iou_threshold: 0.3
      max_detections_per_class: 400
      max_total_detections: 400
    }
    second_stage_batch_size: 128
    second_stage_balance_fraction: 0.25
  }
  keypoint_head {
    keypoint_predictor {
      text_resnet_keypoint_predictor {
        conv_hyperparams {
          op: CONV
          regularizer {
            l2_regularizer {
              weight: 0.0001
            }
          }
          initializer {
            variance_scaling_initializer {
            }
          }
        }
        fc_hyperparams {
          op: FC
          regularizer {
            l2_regularizer {
               weight: 0.0001
            }
          }
```

```
            }
            initializer {
              variance_scaling_initializer {
              }
            }
          }
        }
      }
      initial_crop_size: 28
      maxpool_kernel_size: 2
      maxpool_stride: 2
      num_keypoints: 4
      predict_direction: true
      direction_trainable: true
      unified_direction: true
    }
    fixed_height_feature_gather {
      input_layer: 'FPN/level_1'
      height: 8
      max_width: 300
      visualize_height: 32
      visualize_width: 100
      num_buckets: 5
    }
    attention_head {
      crnn_encoder {
        cnn_name: 'senet5_encoder'
        norm_type: GROUP
        weight_decay: 0.0
        num_layers: 2
        basic_lstm {
          num_units: 512
        }
        encoder_type: UNI
      }
      attention_decoder {
        embedding_size: 64
        num_layers: 2
        basic_lstm {
          num_units: 512
        }
        attention_mechanism: "normed_bahdanau"
        # visualize_type: "line"
      }
    }
  }
}
```

4.1.11.5.8. SavedModel file evaluation

EasyVision that is provided by Machine Learning Platform for AI (PAI) allows you to set the evaluation parameters in a configuration file to evaluate an exported SavedModel file. This topic provides examples of the configuration files for SavedModel file evaluation. You can use the complete configurations or modify specific parameters as required to evaluate the SavedModel files that are exported from commonly used models.

Background information

The following models are involved in this topic:

- detector
- classifier
- multilabel_classifier
- text_detector
- text_recognizer
- text_spotter
- text_pipeline_predictor

detector

```
# -*- encoding:utf-8 -*-
# detector evaluation config
predictor_name: "Detector"
model_path: "data/test/inference/rfcn"
eval_data: {
  input_path: "data/voc0712_tfrecord/VOC2007_test.tfrecord"
  batch_size: 1
  shuffle: false
  prefetch_size: 32
  voc_decoder_config {
    label_map_path: "data/voc0712_tfrecord/pascal_label_map.pbtxt"
  }
  # do not need augmentation
  use_diff: true
  num_epochs: 1
}
eval_config: {
  num_examples: 10
  metrics_set: 'coco_detection_metrics'
  metrics_set: 'pascal_voc_detection_metrics'
  metrics_set: 'pascal_voc07_detection_metrics'
}
```

classifier

```
# -*- encoding:utf-8 -*-
# text pipeline predictor evaluation config
predictor_name: "Classifier"
model_path: "data/test/inference/cifar10_resnet50"
eval_data : {
  input_path: "data/cifar10/cifar10_test.tfrecord"
  batch_size: 100
  shuffle: false
  num_readers: 1
  drop_remainder: false
  classification_decoder_config{
    label_map_path: 'data/cifar10/labelmap.pbtxt'
  }
}
eval_config: {
  num_visualizations : 16
  visualization_export_dir: ''
  metrics_set: "classification_metrics"
}
```

multilabel_classifier

```
# -*- encoding:utf-8 -*-
# text pipeline predictor evaluation config
predictor_name: "MultiLabelClassifier"
model_path: "data/test/inference/objects365_resnet101"
eval_data : {
  input_path: "data/objects365_tfrecord/objects365_test*.tfrecord"
  batch_size: 16
  shuffle: false
  num_readers: 2
  classification_decoder_config {
    label_map_path: "data/objects365_tfrecord/objects365_label_map.pbtxt"
     is_multi_label: true
  }
}
eval_config: {
  metrics_set: "multi_label_classification_metrics"
  include_metrics_per_category: true
}
```

text_detector

```
# -*- encoding:utf-8 -*-
# text detector evaluation config
predictor_name: "TextDetector"
model_path: "data/test/inference/text_krcnn"
eval_data: {
  input_path: "data/icdar_detection_tfrecords/icdar-ch4-test.tfrecord"
  batch_size: 1
  shuffle: false
  prefetch_size: 32
  text_detection_decoder_config {
    label_map_path: "data/icdar_detection_tfrecords/label_map.pbtxt"
  }
  # do not need augmentation
  use_diff: true
  num_epochs: 1
}
eval_config: {
  metrics_set: "icdar_detection_metrics"
}
```

text_recognizer

```
# -*- encoding:utf-8 -*-
# text pipeline predictor evaluation config
predictor_name: "TextRecognizer"
model_path: "data/test/inference/crnn_attn"
eval_data: {
  input_path: "data/recipt_text/recognition_tfrecords/test.tfrecord"
  batch_size: 64
  shuffle: false
  text_recognition_decoder_config {
    char_dict_path: "data/recipt_text/recognition_tfrecords/char_dict"
    min_input_ratio: 0.125
    max_input_ratio: 100
  }
  # do not need augmentation
  num_epochs: 1
}
eval_config: {
  metrics_set: "text_recognition_metrics"
}
```

text_spotter

```
# -*- encoding:utf-8 -*-
# text pipeline predictor evaluation config
predictor_name: "TextSpotter"
model_path: "data/test/inference/text_end2end"
eval_data: {
  input_path: "data/recipt_text/end2end_tfrecords/test.tfrecord"
  batch_size: 1
  shuffle: false
  prefetch_size: 32
  text_end2end_decoder_config {
    char_dict_path: "data/recipt_text/end2end_tfrecords/char_dict"
    label_map_path: "data/recipt_text/end2end_tfrecords/label_map.pbtxt"
  }
  # do not need augmentation
  use_diff: true
  num_epochs: 1
}
eval_config: {
  num_visualizations : 16
  visualization_export_dir: ''
  metrics_set: "icdar_end2end_metrics"
}
```

text_pipeline_predictor

```
# -*- encoding:utf-8 -*-
# text pipeline predictor evaluation config
predictor_name: "TextPipelinePredictor"
model_path: "data/test/inference/text_pipeline"
eval_data: {
  input_path: "data/recipt_text/end2end_tfrecords/test.tfrecord"
  batch_size: 1
  shuffle: false
  prefetch_size: 32
  text_end2end_decoder_config {
    char_dict_path: "data/recipt_text/end2end_tfrecords/char_dict"
    label_map_path: "data/recipt_text/end2end_tfrecords/label_map.pbtxt"
  }
  # do not need augmentation
  use_diff: true
  num_epochs: 1
}
eval_config: {
  metrics_set: "icdar_end2end_metrics"
}
```

4.1.11.6. API

4.1.11.6.1. easy_vision.python.main

This topic describes major Python methods that are provided by EasyVision.

train_and_evaluate

```
easy_vision.python.main.train_and_evaluate(pipeline_config_path, continue_train=False)
```

- Feature: creates, trains, and evaluates a CV estimator.
- Parameters
  - `pipeline_config_path`: the path of the proto.CvEstimator configuration file.
  - `train_config`: the configuration file that is used for training. The configuration file contains the configurations about the model, data, and evaluation process.
  - `continue_train`: specifies whether to restart training from the point when last training stopped.
- No result is returned for this method. After you call this method, the system saves the trained model to the `train_config.model_dir` folder.

train_and_evaluate_with_param_config

```
easy_vision.python.main.train_and_evaluate_with_param_config(param_config_str, continue_train=False,
 data_prefix='')
```

- Feature: trains and evaluates a model by referencing a configuration file.
- Parameters
  - `param_config_str`: the string that can be converted to the configuration file used for model training.
  - `continue_train`: specifies whether to restart training from the point when last training stopped.
  - `data_prefix`: the prefix of the path where the data of the pretrained model are stored. Examples: OSS path: *oss://pai-vision-data/*. On-premises path: *local path/home/user/data*.
- No result is returned for this method. After you call this method, the system saves the trained model to the `train_config.model_dir` folder.

evaluate

```
easy_vision.python.main.evaluate(pipeline_config_path, eval_checkpoint_path='', eval_data_path=None,
eval_result_filename='eval_result.txt')
```

- Feature: evaluates the evaluation metrics in the path specified by the `pipeline_config_path` parameter. Evaluation results are displayed on a TensorBoard dashboard.
- Parameters
  - `pipeline_config_path`: the path of the `proto.CvEstimator` configuration file that contains the configurations about the model, evaluation metrics, and estimator.
  - `eval_checkpoint_path`: If you set this parameter, the model in the path specified by this parameter, instead of the model in the path specified by the `pipeline_config_path` parameter, is used for evaluation.
  - `eval_data_path`: the path of the evaluation metrics. By default, the evaluation metrics in the path specified by the `pipeline_config_path` parameter are evaluated. You can set this parameter to a path or a list of paths.
- A dictionary of evaluation results is returned for this method. You can configure evaluation metrics in the path specified by the `pipeline_config_path` parameter.
- The AssertionError code indicates that the path specified by the `pipeline_config_path` parameter does not exist.

predict

Test for evaluation for eval_data in pipeline_config_path.

```
easy_vision.python.main.predict(pipeline_config_path, test_checkpoint_path='', test_filelist=None)
```

- Feature: tests the evaluation metrics in the path specified by the `pipeline_config_path` parameter.
- Parameters
  - `pipeline_config_path`: the path of the `proto.CvEstimator` configuration file that contains the configurations about the model, evaluation metrics, and estimator.
  - `test_checkpoint_path`: If you set this parameter, the model in the path specified by this parameter, instead of the model in the path specified by the `pipeline_config_path` parameter, is used for testing.
- A list of test results is returned for this method. Each piece of test result indicates the prediction result for an image.
- The AssertionError code indicates that the path specified by the `pipeline_config_path` parameter does not exist, or the `train_config.model_dir` folder does not exist.

export

Export model defined in pipeline_config_path.

```
easy_vision.python.main.export(export_dir, pipeline_config_path, checkpoint_path='')
```

- Feature: exports the model in the path specified by the `pipeline_config_path` parameter.
- Parameters
  - `export_dir`: the path of the folder to which the model is exported.
  - `pipeline_config_path`: the path of the `proto.CvEstimator` configuration file that contains the configurations about the model, evaluation metrics, and estimator.
  - `checkpoint_path`: If you set this parameter, the model in the path specified by this parameter, instead of the model in the path specified by the `pipeline_config_path` parameter, is used for testing.
- The path of the folder to which the model is exported is returned for this method.
- The AssertionError code indicates that the path specified by the `pipeline_config_path` parameter does not exist.

predictor_evaluate

```
easy_vision.python.main.predictor_evaluate(config_path)
```

- Feature: evaluates a predictor.
- `config_path`: the path of the `proto.PredictorEval` configuration file.
- A dictionary of evaluation results is returned for this method.
- The AssertionError code indicates that the path specified by the `config_path` parameter does not exist.

4.1.11.6.2. easy_vision.python.data_main

This topic describes the data conversion methods provided by EasyVision.

create_dataset

```
easy_vision.python.data_main.create_dataset(input_path, config_path, output_prefix)
```

- Feature: converts the files in the path specified by the `input_path` parameter to TFRecord files.
- Parameters
  - `input_path`: the path of the files to be converted.
  - `config_path`: the path of the configuration file used for data conversion.
  - `output_prefix`: the prefix of the path of the output TFRecord files.
- The AssertionError code indicates that the path specified by the `config_path` parameter does not exist, or the value of the `config_path.proc_num` parameter is less than or equal to 0.

create_dataset_with_param_config

```
easy_vision.python.data_main.create_dataset_with_param_config(input_path, param_config_str, output_prefix, data_prefix='')
```

- Feature: converts the files in the path specified by the `input_path` parameter to TFRecord files by setting parameters.
- Parameters
  - `input_path`: the path of the files to be converted.
  - `param_config_str`: the string that can be converted to the configuration file used for data conversion.
  - `output_prefix`: the prefix of the path of the output TFRecord files.
  - `data_prefix`:data path prefix for data and pretrained-models, e.g., osspath oss://pai-vision-data/, local path /home/user/data.
  - `data_prefix`: the prefix of the path where data is stored. Examples: OSS path: *oss://pai-vision-data/*. On-premises path: *local path/home/user/data*.
- The AssertionError code indicates that the path specified by the `config_path` parameter does not exist, or the value of the `config_path.proc_num` parameter is less than or equal to 0.

4.1.11.6.3. easy_vision.python.inference

This topic describes the prediction classes provided by EasyVision.

predictor.Classifier

```
class easy_vision.python.inference.predictor.Classifier(model_path, profiling_file=None, verbose_result=False)
```

The base class of the `easy_vision.python.inference.predictor.Classifier` class is `easy_vision.python.inference.predictor.Predictor`. The following table describes the methods contained in the easy_vision.python.inference.predictor.Classifier class.

Method	Description

Method	Description
`__init__(model_path, profiling_file=None, verbose_result=False)`	The initialization method. You need to set the following parameters: • `model_path`: the path of the model in the SavedModel format or the path of a Frozen Graph model in the PB format. • `profiling_file`: the output file that contains the prediction statistics. Default value: None. If you specify this parameter, the prediction function collects the statistics about the prediction time by using Timeline and returns the statistics in the JSON format in the output file. • `verbose_result`: If you set this parameter to True, the prediction evaluator returns detailed results.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`predict(**kwargs)`	A method that is contained in the base class. This method is not implemented.

predictor.Detector

```
class easy_vision.python.inference.predictor.Detector(model_path, profiling_file=None)
```

The base class of the `easy_vision.python.inference.predictor.Detector` class is `easy_vision.python.inference.predictor.Predictor`. The following table describes the methods contained in the easy_vision.python.inference.predictor.Detector class.

Method	Description
`__init__(model_path, profiling_file=None)`	The initialization method. You need to set the following parameters: • `model_path`: the path of the model in the SavedModel format or the path of a Frozen Graph model in the PB format. • `profiling_file`: Default value: None.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`predict(**kwargs)`	A method that is contained in the base class. This method is not implemented.

predictor.FeatureExtractor

```
class easy_vision.python.inference.predictor.FeatureExtractor(model_path, output_feature, profiling_file=None)
```

The base class of the `easy_vision.python.inference.predictor.FeatureExtractor` class is `easy_vision.python.inference.predictor.Predictor`. The following table describes the methods contained in the easy_vision.python.inference.predictor.FeatureExtractor class.

Method	Description
`__init__(model_path, output_feature, profiling_file=None)`	The initialization method. You need to set the following parameters: • `model_path`: the path of the model in the SavedModel format or the path of a Frozen Graph model in the PB format. • `output_feature`: the name of the output node.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`predict(**kwargs)`	A method that is contained in the base class. This method is not implemented.

predictor.MultiLabelClassifier

```
class easy_vision.python.inference.predictor.MultiLabelClassifier(model_path, profiling_file=None, verbose_result=False)
```

The base class of the `easy_vision.python.inference.predictor.MultiLabelClassifier` class is `easy_vision.python.inference.predictor.Predictor`. The following table describes the methods contained in the easy_vision.python.inference.predictor.MultiLabelClassifier class.

Method	Description
`__init__(model_path, profiling_file=None, verbose_result=False)`	The initialization method. You need to set the following parameters: • `model_path`: the path of the model in the SavedModel format or the path of a Frozen Graph model in the PB format. • `profiling_file`: the output file that contains the prediction statistics. Default value: None. If you specify this parameter, the prediction function collects the statistics about the prediction time by using Timeline and returns the statistics in the JSON format in the output file. • `verbose_result`: If you set this parameter to True, the predictive evaluator returns detailed results.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`predict(**kwargs)`	A method that is contained in the base class. This method is not implemented.

predictor.Predictor

```
class easy_vision.python.inference.predictor.Predictor(model_path, profiling_file=None, decode=True)
```

The base class of the `easy_vision.python.inference.predictor.Predictor` class is `easy_vision.python.inference.predictor.PredictorInterface`. The following table describes the methods contained in the easy_vision.python.inference.predictor.Predictor class.

Method	Description
`__init__(model_path, profiling_file=None, decode=True)`	The initialization method. You need to set the following parameters: • `model_path`: the path of the model in the SavedModel format or the path of a Frozen Graph model in the PB format. • `profiling_file`: the output file that contains the prediction statistics. Default value: None. If you specify this parameter, the prediction function collects the statistics about the prediction time by using Timeline and returns the statistics in the JSON format in the output file. • `decode`: Default value: True.
`batch(images)`	• Feature: This method packages multiple red-green-blue (RGB) images. • Parameter: The `images` parameter specifies the information about images and image shapes. • Return value: a feed dictionary that contains one or more key-value pairs. The feed dictionary is an input format in TensorFlow.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`get_class_name(cls_id)`	• Feature: This method returns the name of the class whose ID is the same as the value of the `cls_id` parameter. • Parameter: The `cls_id` parameter specifies the class ID. The value of this parameter is of the INT32 type. • Return value: the class name that is of the STRING type.
`get_input_names`	Returns a list that contains the names of available input nodes in the model.
`get_output_names()`	Returns a list that contains the names of available output nodes in the model.

Method	Description

`predict(data_list, output_names=None, batch_size=1)`	• Parameters:
	  ◦ `data_list`: the input data in numPy arrays. The input data is of the UINT8 type.
	  ◦ `output_names`: If the parameter is not empty, specific output fields are returned.
	  ◦ `batch_size`: the size of the data in a batch that is used for prediction. A value of -1 indicates that the actual batch size is used.
	• Return value: a list of dictionaries. Each dictionary contains one or more key-value pairs. In each key-value pair, the key is the name of an output field, and the value is the value of the output field.

predictor.PredictorImpl

```
class easy_vision.python.inference.predictor.PredictorImpl(model_path, profiling_file=None, decode=True)
```

The base class of the `easy_vision.python.inference.predictor.PredictorImpl` class is `object`. The following table describes the methods contained in the easy_vision.python.inference.predictor.PredictorImpl class.

Method	Description
`__init__(model_path, profiling_file=None, decode=True)`	The initialization method. You need to set the following parameters: • `model_path`: the path of the model in the SavedModel format or the path of a Frozen Graph model in the PB format. • `profiling_file`: the output file that contains the prediction statistics. Default value: None. If you specify this parameter, the prediction function collects the statistics about the prediction time by using Timeline and returns the statistics in the JSON format in the output file. • `decode`: Default value: True.

Method	Description

`predict(input_data_dict, output_names=None)`	• Parameters:
	  ◦ `input_data_dict`: a dictionary that contains all input data. The dictionary contains one or more key-value pairs. In each key-value pair, the key is the name of an input field, and the value is the value of the input field.
	  ◦ `output_names`: If the parameter is not empty, specific output fields are returned. If the parameter is empty, the specific output data is returned based on all the output information in the model signature.
	• Return value: a list of dictionaries. Each dictionary contains one or more key-value pairs. In each key-value pair, the key is the name of an output field, and the value is the value of the output field.

`search_pb(directory)`	• Feature: This method recursively searches for a PB file in the folders. If multiple PB files exist, an exception is thrown. • Return value: the folder that contains a PB file.

predictor.PredictorInterface

```
class easy_vision.python.inference.predictor.PredictorInterface(model_path, model_config=None)
```

The base class of the `easy_vision.python.inference.predictor.PredictorInterface` class is `object`. The following table describes the methods contained in the easy_vision.python.inference.predictor.PredictorInterface class.

Method	Description
`__init__(model_path, model_config=None)`	Initializes a session with Machine Learning Platform for AI (PAI)-TensorFlow and loads a TensorFlow model. You need to set the following parameters: • `model_path`: the path of the model that you want to initialize. • `model_config`: the configurations that you want to initialize for the model. The configurations are in a JSON string.
`check_signature(impl_cls)`	Checks whether the __init__ and predict methods are contained in the implementation classes that inherit this class.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.

Method	Description
`get_output_type()`	The type of the output data. If you want the output data to be returned as a dictionary, you can set the parameter to json. Then, the output data returned by the predictor is serialized to a JSON string. If the input data consists of images, the images are converted to binary images, and each binary image is written to Object Storage Service (OSS) as an object. The object name of each binary image is in the following format: output_dir/${key}/${input_filename}_${idx}.jpg. The system extracts a value from the image URL for the input_filename variable. The value of the key variable is the key in the output dictionary. If each image is an indexed element in a list, the system generates the value of the idx variable by referencing the index of the list element. If the input data is not an indexed list, the idx variable is empty. If the input data consists of videos, videos are converted to binary videos, and each binary video is written to OSS as an object. For example, {'image':'image','feature':'json'} indicates that each image in the output dictionary will be stored as an OSS object and the feature data in the output dictionary will be converted to the JSON format.

`predict(input_data, batch_size)`	• Feature: This method predicts a batch of sample data whose size is specified by the batch_size parameter in a session.
	• Parameters:
	  ◦ `input_data` : the input data in numPy arrays. Each array is a sample data entry to be predicted.
	  ◦ `batch_size` : the size of the data in a batch that is used for prediction. If you want to call the predict method and do not want to change the batch size during prediction, you can set the parameter to a fixed number and ignore the parameter in further configurations.
	• Return value: a list of dictionaries. Each dictionary is the prediction result of a sample data entry.
	
	For example, if the return value is {"output1":value1,"output2":value2}, the values can be of the INT, STRING, or FLOAT type in Python, or be numPy arrays in Python.

predictor.PredictorInterfaceV2

```
class easy_vision.python.inference.predictor.PredictorInterfaceV2(model_path, model_config=None)
```

The base class of the `easy_vision.python.inference.predictor.PredictorInterfaceV2` class is `easy_vision.python.inference.predictor.PredictorInterface`. The following table describes the methods contained in the easy_vision.python.inference.predictor.PredictorInterfaceV2 class.

Method	Description
`__init__(model_path, model_config=None)`	Initializes a session with PAI-TensorFlow and loads a TensorFlow model. You need to set the following parameters: • `model_path`: the path of the model that you want to initialize. • `model_config`: the configurations that you want to initialize for the model. The configurations are in a JSON string.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`get_output_type()`	The type of the output data. If you want the output data to be returned as a dictionary, you can set the parameter to json. Then, the output data returned by the predictor is serialized to a JSON string. If the input data consists of images, the images are converted to binary images, and each binary image is written to Object Storage Service (OSS) as an object. The object name of each binary image is in the following format: output_dir/${key}/${input_filename}_${idx}.jpg. The system extracts a value from the image URL for the input_filename variable. The value of the key variable is the key in the output dictionary. If each image is an indexed element in a list, the system generates the value of the idx variable by referencing the index of the list element. If the input data is not an indexed list, the idx variable is empty. If the input data consists of videos, videos are converted to binary videos, and each binary video is written to OSS as an object. For example, {'image':'image','feature':'json'}indicates that each image in the output dictionary will be stored as an OSS object and the feature data in the output dictionary will be converted to the JSON format.

Method	Description

`predict(input_data_dict_list, batch_size)`	• Feature: This method predicts a batch of sample data whose size is specified by the batch_size parameter in a session.
	• Parameters:
	  ◦ `input_data_dict_list`: the input data in dictionaries. Each dictionary is a sample data entry to be predicted.
	  ◦ `batch_size`: the size of the data in a batch that is used for prediction. If you want to call the predict method and do not want to change the batch size during prediction, you can set the parameter to a fixed number and ignore the parameter in further configurations.
	• Return value: a list of dictionaries. Each dictionary is the prediction result of a sample data entry.
	For example, if the return value is {"output1":value1,"output2":value2}, the values can be of the INT, STRING, or FLOAT type in Python, or be numPy arrays in Python.

predictor.PredictorV2

```
class easy_vision.python.inference.predictor.PredictorV2(model_path, profiling_file=None, decode=True)
```

The base class of the `easy_vision.python.inference.predictor.PredictorV2` class is `easy_vision.python.inference.predictor.PredictorInterfaceV2`. The following table describes the methods contained in the easy_vision.python.inference.predictor.PredictorV2 class.

Method	Description
`__init__(model_path, profiling_file=None, decode=True)`	The initialization method. You need to set the following parameters: • `model_path`: the path of the model in the SavedModel format or the path of a Frozen Graph model in the PB format. • `profiling_file`: the output file that contains the prediction statistics. Default value: None. If you specify this parameter, the prediction function collects the statistics about the prediction time by using Timeline and returns the statistics in the JSON format in the output file.
`batch(batch_data_dict_list)`	• Feature: This method packages multiple RGB images into a batch. • Parameter: The value of the `batch_data_dict_list` parameter is a list of dictionaries. • Return value: a feed dictionary. The feed dictionary is an input format in TensorFlow.

Method	Description
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`get_class_name(cls_id)`	• Feature: This method returns the name of the class whose ID is the same as the value of the `cls_id` parameter. • Parameter: The `cls_id` parameter specifies the class ID. The value of this parameter is of the INT32 type. • Return value: the class name that is of the STRING type.
`get_input_names()`	Returns a list that contains the names of available input nodes in the model.
`get_output_names()`	Returns a list that contains the names of available output nodes in the model.

`predict(input_data_dict_list, output_names=None, batch_size=1)`	• Feature: This method predicts a batch of sample data whose size is specified by the batch_size parameter in a session.
	• Parameters:
	  ○ `input_data_dict_list`: the input data in dictionaries. Each dictionary is a sample data entry to be predicted.
	  ○ `output_names`: If the parameter is not empty, specific output fields are returned.
	  ○ `batch_size`: the size of the data that is used for prediction in a batch. A value of -1 indicates that the batch size of the input data is used.
	• Return value: a list of dictionaries. Each dictionary is the prediction result of a sample data entry.

predictor.Segmentor

```
class easy_vision.python.inference.predictor.Segmentor(model_path, profiling_file=None)
```

The base class of the `easy_vision.python.inference.predictor.Segmentor` class is `easy_vision.python.inference.predictor.Predictor`. The following table describes the methods contained in the easy_vision.python.inference.predictor.Segmentor class.

Method	Description

Method	Description
`__init__(model_path, profiling_file=None)`	The initialization method. You need to set the following parameters: • `model_path`: the path of the model in the SavedModel format or the path of a Frozen Graph model in the PB format. • `profiling_file`: the output file that contains the prediction statistics. Default value: None. If you specify this parameter, the prediction function collects the statistics about the prediction time by using Timeline and returns the statistics in the JSON format in the output file.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`get_output_type()`	The type of the output data. If you want the output data to be returned as a dictionary, you can set the parameter to json. Then, the output data returned by the predictor is serialized to a JSON string. If the input data consists of images, the images are converted to binary images, and each binary image is written to OSS as an object. The object name of each binary image is in the following format: output_dir/${key}/${input_filename}_${idx}.jpg. The system extracts a value from the image URL for the input_filename variable. The value of the key variable is the key in the output dictionary. If each image is an indexed element in a list, the system generates the value of the idx variable by referencing the index of the list element. If the input data is not an indexed list, the idx variable is empty. If the input data consists of videos, videos are converted to binary videos, and each binary video is written to OSS as an object. For example, {'image':'image','feature':'json'}indicates that each image in the output dictionary will be stored as an OSS object and the feature data in the output dictionary will be converted to the JSON format.
`predict(**kwargs)`	A method that is contained in the base class. This method is not implemented.

predictor.TextDetector

```
class easy_vision.python.inference.predictor.TextDetector(model_path, profiling_file=None)
```

The base class of the `easy_vision.python.inference.predictor.TextDetector` class is `easy_vision.python.inference.predictor.Predictor`. The following table describes the methods contained in the easy_vision.python.inference.predictor.TextDetector class.

Method	Description

Method	Description
`__init__(model_path, profiling_file=None)`	The initialization method. You need to set the following parameters: • `model_path`: the path of the model in the SavedModel format or the path of a Frozen Graph model in the PB format. • `profiling_file`: the output file that contains the prediction statistics. Default value: None. If you specify this parameter, the prediction function collects the statistics about the prediction time by using Timeline and returns the statistics in the JSON format in the output file.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`predict(**kwargs)`	A method that is contained in the base class. This method is not implemented.

predictor.TextRecognizer

```
class easy_vision.python.inference.predictor.TextRecognizer(model_path, profiling_file=None)
```

The base class of the `easy_vision.python.inference.predictor.TextRecognizer` class is `easy_vision.python.inference.predictor.Predictor`. The following table describes the methods contained in the easy_vision.python.inference.predictor.TextRecognizer class.

Method	Description
`__init__(model_path, profiling_file=None)`	The initialization method. You need to set the following parameters: • `model_path`: the path of the model in the SavedModel format or the path of a Frozen Graph model in the PB format. • `profiling_file`: the output file that contains the prediction statistics. Default value: None. If you specify this parameter, the prediction function collects the statistics about the prediction time by using Timeline and returns the statistics in the JSON format in the output file.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`predict(**kwargs)`	A method that is contained in the base class. This method is not implemented.

predictor.TextSpotter

```
class easy_vision.python.inference.predictor.TextSpotter(model_path, profiling_file=None)
```

The base class of the `easy_vision.python.inference.predictor.TextSpotter` class is `easy_vision.python.inference.predictor.Predictor`. The following table describes the methods contained in the easy_vision.python.inference.predictor.TextSpotter class.

Method	Description
`__init__(model_path, profiling_file=None)`	The initialization method. You need to set the following parameters: • `model_path`: the path of the model in the SavedModel format or the path of a Frozen Graph model in the PB format. • `profiling_file`: the output file that contains the prediction statistics. Default value: None. If you specify this parameter, the prediction function collects the statistics about the prediction time by using Timeline and returns the statistics in the JSON format in the output file.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`predict(**kwargs)`	A method that is contained in the base class. This method is not implemented.

predictor.batch_images

```
easy_vision.python.inference.predictor.batch_images(images, use_bgr)
```

text_pipeline_predictor.TextPipelinePredictor

```
class easy_vision.python.inference.text_pipeline_predictor.TextPipelinePredictor(model_path)
```

The base class of the `easy_vision.python.inference.text_pipeline_predictor.TextPipelinePredictor` class is `easy_vision.python.inference.predictor.PredictorInterface`. The following table describes the methods contained in the easy_vision.python.inference.text_pipeline_predictor.TextPipelinePredictor class.

Method	Description
`__init__(model_path)`	Initializes the text detector and recognizer. The `model_path` parameter specified the path of the detection model and the recognition model.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`crop_roi_image(img, coords)`	• `img`: the image object. • `coords`: a 4 × 2 numPy array. The value of coords parameter is in the format of [:, y,x].

Method	Description

`predict(image_list, batch_size=None)`	• Parameters:
	  ◦ `image_list`: the image data in numPy arrays. The image data is of the UINT8 type.
	  ◦ `batch_size`: You do not need to specify a value. This parameter is reserved to improve compatibility.
	• Return value:
	A list of dictionaries is returned. Each dictionary consists of key-value pairs. The following keys are returned:
	  ◦ detection_boxes: The value of the key is a numPy array of the FLOAT32 type. The shape of the numPy array is [Number of detected objects, 4].
	  ◦ detection_keypoints: The value of the key is a numPy array of the FLOAT32 type. The shape of the numPy array is [Number of detected objects, 8].
	  ◦ detection_classes: The value of the key is a numPy array of the FLOAT32 type. The shape of the numPy array is [Number of detected objects].
	  ◦ detection_class_names: The value of the key is a numPy array of class names.
	  ◦ detection_scores: The value of the key is a numPy array of the FLOAT32 type. The shape of the numPy array is [Number of detected objects].
	  ◦ detection_texts: The value of the key is a numPy array of the STRING type. The shape of the numPy array is [Number of detected objects].
	  ◦ detection_texts_scores:The value of the key is a numPy array of the FLOAT32 type. The shape of the numPy array is [Number of detected objects].

video_predictor.I3DClassifier

```
class easy_vision.python.inference.video_predictor.I3DClassifier(model_path, resize_height=256, resize_width=340, crop_size=224, input_modal='rgb', profiling_file=None)
```

The base class of the `easy_vision.python.inference.video_predictor.I3DClassifier` class is `easy_vision.python.inference.video_predictor.VideoPredictor`. The following table describes the methods contained in the easy_vision.python.inference.video_predictor.I3DClassifier class.

Method	Description

Method	Description
`__init__(model_path, resize_height=256, resize_width=340, crop_size=224, input_modal='rgb', profiling_file=None)`	The initialization method. You need to set the following parameters: • `model_path`: the path of a model in the SavedModel format. The model can be an RGB model or a stream model. • `input_modal`: the input modal. A value of rgb indicates an RGB model. A value of flow indicates a stream model. • `profiling_file`: the output file that contains the prediction statistics. Default value: None. If you specify this parameter, the prediction function collects the statistics about the prediction time by using Timeline and returns the statistics in the JSON format in the output file.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`predict(**kwargs)`	A method that is contained in the base class. This method is not implemented.
`predict_inner(data_dict_list, batch_size=4)`	This class is abstracted as a dual-stream classifier. You need to set the following parameters: • `data_dict_list`: the input data in numPy arrays. The input data is of the UINT8 type. • `batch_size`: the size of a batch.
`preprocess(data_dict)`	Preprocesses the three-dimensional (3D) data. This method assumes that the video frame data is cropped in the time dimension and resized in the space dimension. This method only crops images in the space dimension and normalizes images.

video_predictor.I3DTwoStreamClassifier

```
class easy_vision.python.inference.video_predictor.I3DTwoStreamClassifier(model_path, resize_height=256, resize_width=340, crop_size=224, profiling_file=None)
```

The base class of the `easy_vision.python.inference.video_predictor.I3DTwoStreamClassifier` class is `easy_vision.python.inference.predictor.PredictorInterfaceV2`. The following table describes the methods contained in the easy_vision.python.inference.video_predictor.I3DTwoStreamClassifier class.

Method	Description

Method	Description
`__init__(model_path, resize_height=256, resize_width=340, crop_size=224, profiling_file=None)`	The initialization method. You need to set the following parameters: • `model_path`: the path of both an RGB model and a stream model. The models are in the SavedModel format. • `resize_height`: the height after resizing. • `resize_width`: the width after resizing. • `crop_size`: the size after random cropping. • `profiling_file`: the output file that contains the prediction statistics. Default value: None. If you specify this parameter, the prediction function collects the statistics about the prediction time by using Timeline and returns the statistics in the JSON format in the output file.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.

`predict(input_data_dict_list, batch_size=4)`	• Feature: This method predicts a batch of sample data whose size is specified by the batch_size parameter in a session.
	• Parameters:
	  ◦ `input_data_dict_list`: the input data in dictionaries. Each dictionary is a sample data entry to be predicted.
	  ◦ `batch_size`: the size of the data in a batch that is used for prediction. If you want to call the predict method and do not want to change the batch size during prediction, you can set the parameter to a fixed number and ignore the parameter in further configurations.
	• Return value: a list of dictionaries. Each dictionary is the prediction result of a sample data entry.
	For example, if the return value is {"output1":value1,"output2":value2}, the values can be of the INT, STRING, or FLOAT type in Python, or be numPy arrays in Python.

video_predictor.VideoClassifier

```
class easy_vision.python.inference.video_predictor.VideoClassifier(model_path, sample_duration=16, train_crop='corner', sample_size=112, output_feature=None, profiling_file=None)
```

The base class of the `easy_vision.python.inference.video_predictor.VideoClassifier` class is `easy_vision.python.inference.video_predictor.VideoPredictor`. The following table describes the methods contained in the easy_vision.python.inference.video_predictor.VideoClassifier class.

Method	Description
`__init__(model_path, sample_duration=16, train_crop='corner', sample_size=112, output_feature=None, profiling_file=None)`	The initialization method. You need to set the following parameters: • `model_path`: the path of the model in the SavedModel format or the path of a Frozen Graph model in the PB format. • `sample_duration`: the duration of the sample video. Default value: 16. • `train_crop`: the cropping method. Default value: corner. • `sample_size`: the size of an output frame. Default value: 112. • `output_feature`: the name of the extracted feature. Default value: None. • `profiling_file`: the output file that contains the prediction statistics. Default value: None. If you specify this parameter, the prediction function collects the statistics about the prediction time by using Timeline and returns the statistics in the JSON format in the output file.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`predict(**kwargs)`	A method that is contained in the base class. This method is not implemented.

video_predictor.VideoMultiLabelClassifier

```
class easy_vision.python.inference.video_predictor.VideoMultiLabelClassifier(model_path, sample_duration=16, train_crop='corner', sample_size=112, profiling_file=None)
```

The base class of the `easy_vision.python.inference.video_predictor.VideoMultiLabelClassifier` class is `easy_vision.python.inference.video_predictor.VideoPredictor`. The following table describes the methods contained in the easy_vision.python.inference.video_predictor.VideoMultiLabelClassifier class.

Method	Description

Method	Description
`__init__(model_path, sample_duration=16, train_crop='corner', sample_size=112, profiling_file=None)`	The initialization method for multi-label video classification. You need to set the following parameters: • `model_path`: the path of the model in the SavedModel format or the path of a Frozen Graph model in the PB format. • `sample_duration`: the duration of the sample video. Default value: 16. • `train_crop`: the cropping method. Default value: corner. • `sample_size`: the size of an output frame. Default value: 112. • `profiling_file`: the output file that contains the prediction statistics. Default value: None. If you specify this parameter, the prediction function collects the statistics about the prediction time by using Timeline and returns the statistics in the JSON format in the output file.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.
`predict(**kwargs)`	A method that is contained in the base class. This method is not implemented.

video_predictor.VideoPredictor

```
class easy_vision.python.inference.video_predictor.VideoPredictor(model_path, profiling_file=None, input_modal='rgb')
```

The base class of the `easy_vision.python.inference.video_predictor.VideoPredictor` class is `easy_vision.python.inference.predictor.PredictorV2`. The following table describes the methods contained in the easy_vision.python.inference.video_predictor.VideoPredictor class.

Method	Description

Method	Description
`__init__(model_path, profiling_file=None, input_modal='rgb')`	The initialization method. You need to set the following parameters: • `model_path`: the path of the model in the SavedModel format or the path of a Frozen Graph model in the PB format. • `profiling_file`: the output file that contains the prediction statistics. Default value: None. If you specify this parameter, the prediction function collects the statistics about the prediction time by using Timeline and returns the statistics in the JSON format in the output file. • `input_modal`: the input modal. A value of rgb indicates that video clips are used. If the preprocessed images are not exported, the data about video clips is a numPy array of the FLOAT32 type. The shape of the numPy array is [num_frame, h, w, c]. If the preprocessed images are exported, the data about video clips is a numPy array of the STRING type. The shape of the numPy array is [num_frame]. A value of flow indicates that video streams are used. If the preprocessed images are not exported, the data about video streams is a numPy array of the FLOAT32 type. The shape of the numPy array is [num_frame, h, w, c]. If the preprocessed images are exported, the data about video streams is a numPy array of the STRING type. The shape of the numPy array is [num_frame].
`batch(batch_data_dict_list)`	Packages multiple RGB images into a batch. The value of the `batch_data_dict_list` parameter is a list of dictionaries. This method returns a feed dictionary. The feed dictionary is an input format in TensorFlow.
`batch_clips(clips)`	Packages multiple clips. The value of the `clips` parameter is a list of numPy arrays. This method returns two numPy arrays. One numPy array contains a batch of data which is of the FLOAT type. The other numPy array contains the data about the original image shape and size. The data about the original image shape and size is of the INT32 type.
`create_class(name)`	Checks whether the class whose name is the same as the value of the name parameter is registered. If the class is registered, the class is returned. If the class is not registered, an exception is thrown.

4.1.12. Terms and acronyms

4.1.12.1. Terms

This topic describes the terms that are used in Apsara Stack Machine Learning Platform for AI.

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 in Machine Learning Platform for AI. For example, you can use components to import and export data, process data, analyze data, train models, and make predictions.

CAP

Cloud Application Access Platform (CAP) is a unified access platform that is provided to integrate the basic data in various big data tools to share data and models. The runtime support platform involves concepts such as tenants, namespaces, resources, and messages. By connecting various big data tools to the runtime support platform, you can use consistent definitions of these concepts throughout the data platform. This way, you can share resources among the tools and shield the differences among underlying platforms. The runtime support platform connects to the underlying platforms so that you can deploy the upper-level tool platforms that run based on the runtime support platform to different platforms and environments without modifications.

Dmscloud

Dmscloud is a set of data mining, modeling, and prediction tools. It is developed based on MaxCompute. Dmscloud provides you with an all-in-one algorithm service that supports algorithm development, sharing, model training, deployment, and monitoring. Dmscloud allows you to manage experiments on the graphic user interface (GUI) or by running Machine Learning Platform for AI commands. Machine Learning Platform for AI provides features such as data preprocessing, machine learning algorithms, and model evaluation and prediction.

Machine Learning Platform for AI runs on MaxCompute. After you deploy algorithms in MaxCompute clusters, you can call the algorithms from the Machine Learning Platform for AI console. This decouples algorithm applications from computing engines.

Machine Learning Platform for AI provides you with bountiful algorithms and reliable technical support to help resolve issues in various business scenarios. In the data technology (DT) era, you can use Machine Learning Platform for AI to develop data-driven business.

JCS

Job Control System (JCS) is a task control engine that is used to schedule and control distributed tasks in Machine Learning Platform for AI. JCS supports directed acyclic graph (DAG) parsing and breakpoint reruns.

4.1.12.2. Acronyms

This topic describes the acronyms that are used in PAI 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 a large amount of structured data that does not require real-time processing. Therefore, MaxCompute provides support for various data warehouse solutions as well as big data analysis and modeling.

Relational Database Service (RDS)

ApsaraDB RDS is a stable, reliable, and scalable online database service on Alibaba Cloud. ApsaraDB RDS is also an out-of-box service that supports MySQL, SQL Server, PostgreSQL, and PPAS. ApsaraDB RDS is highly compatible with Oracle. ApsaraDB RDS provides various features, including online scale-out, backup, rollback, and performance monitoring and analysis.

Open Cache Service (OCS)

OCS is a memory-based cache service that allows a large number of requests to access a small amount of data with high speed.

Source and destination tables in MaxCompute

A table is a data storage object 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 destination table is the output of an algorithm node.

5.DataHub

5.1. User Guide

5.1.1. 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.

5.1.2. 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.
String size	Up to 1 MB	The size of a string cannot exceed 1 MB.

Item	Limit	Description
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.

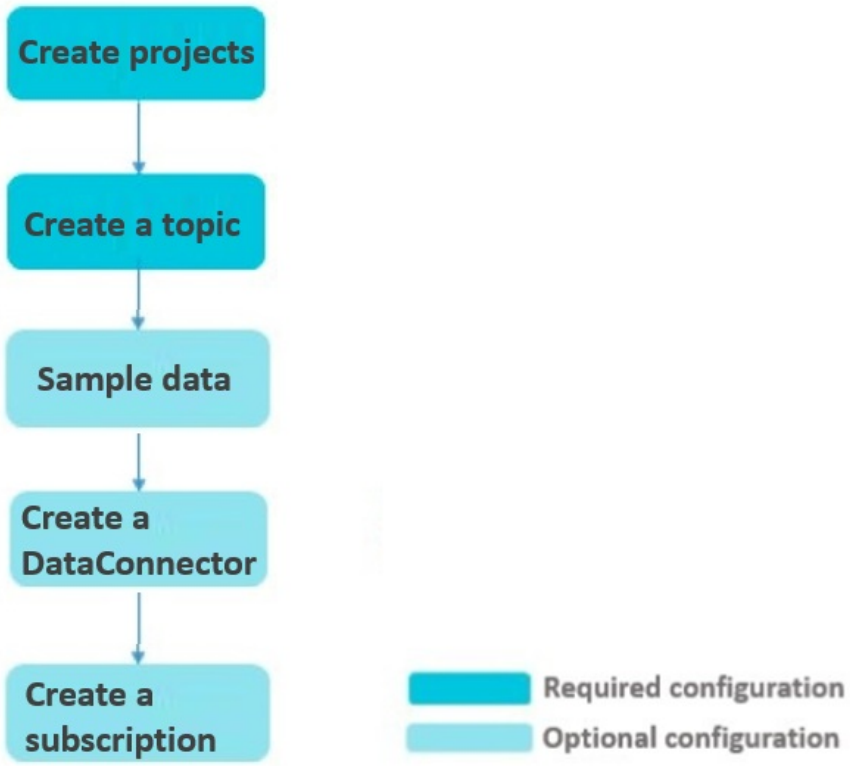
5.1.3. Quick Start

5.1.3.1. 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.

5.1.3.2. Log on to the DataHub console

This topic describes how to log on to the DataHub console. Google Chrome is used in this example.

Prerequisites

Before you log on to the DataHub console, make sure that the following requirements are met:

- You have obtained the URL of the Apsara Uni-manager Management Console.
- A browser is available. We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL of the Apsara Uni-manager Management Console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password from the operations administrator.

   > **Note** If you log on to the Apsara Uni-manager Management Console for the first time, you must change the password of your username. To ensure the security of your account, the password must be 8 to 20 characters in length and must contain at least two types of the following characters:
   >
   > - Uppercase or lowercase letters
   > - Digits
   > - Special characters: exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%)

3. Click **Login** to go to the Apsara Uni-manager Management Console.
4. In the top navigation bar, choose **Products > Big Data > DataHub** to go to the DataHub console. The **Overview** page appears.

5.1.3.3. 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 and bind a virtual private cloud (VPC) to the project in the DataHub console.

Prerequisites

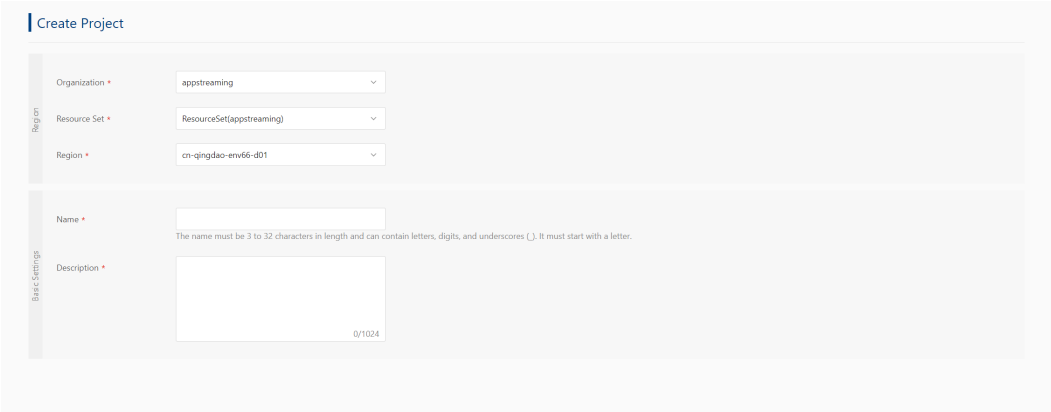
An Apsara Stack tenant account is created.

Considerations

- DataHub projects are independent of MaxCompute projects. Projects you created in MaxCompute cannot be used in DataHub.
- You can create up to 100 projects with each account.

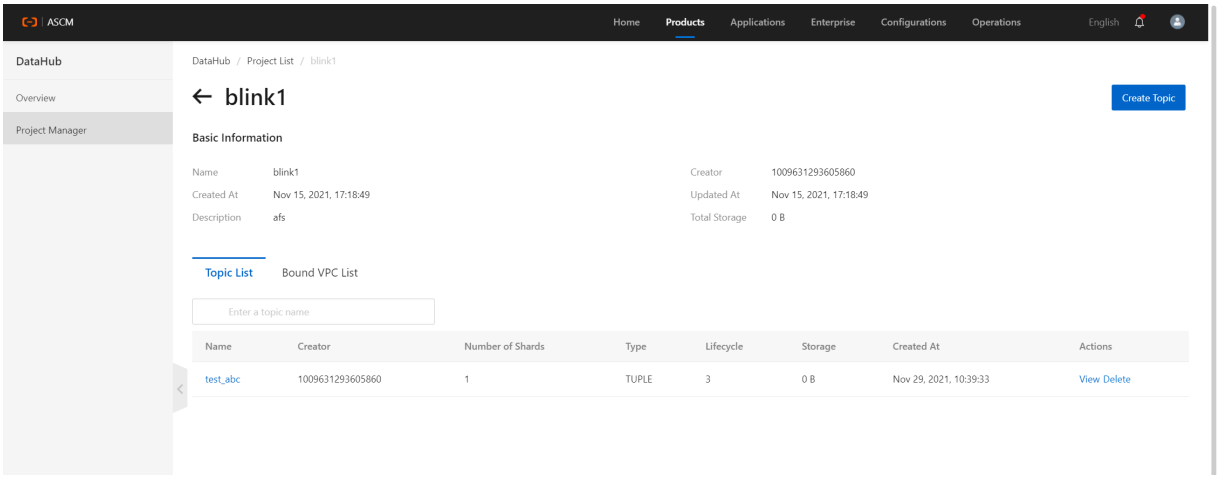
Procedure

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 the parameters in the Region and Basic Settings sections and click **Submit**.



3. After the project is created, you can click **View** in the Actions column to view the details of the created

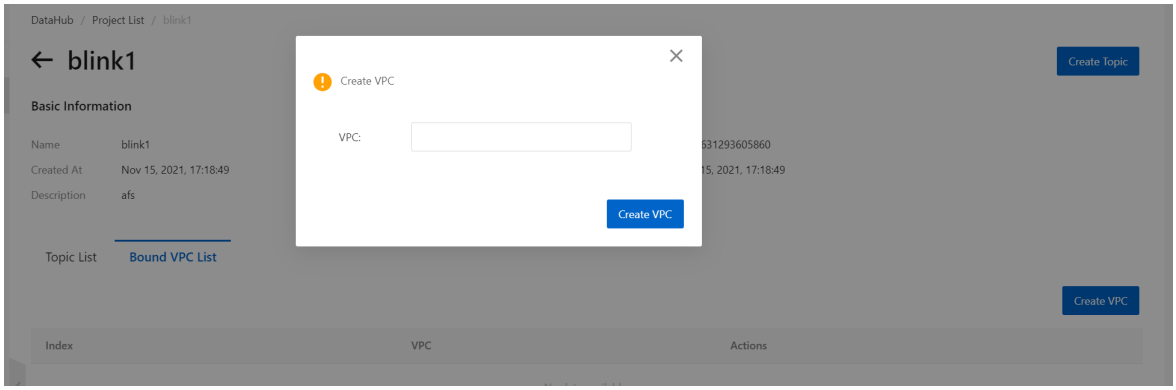
project.



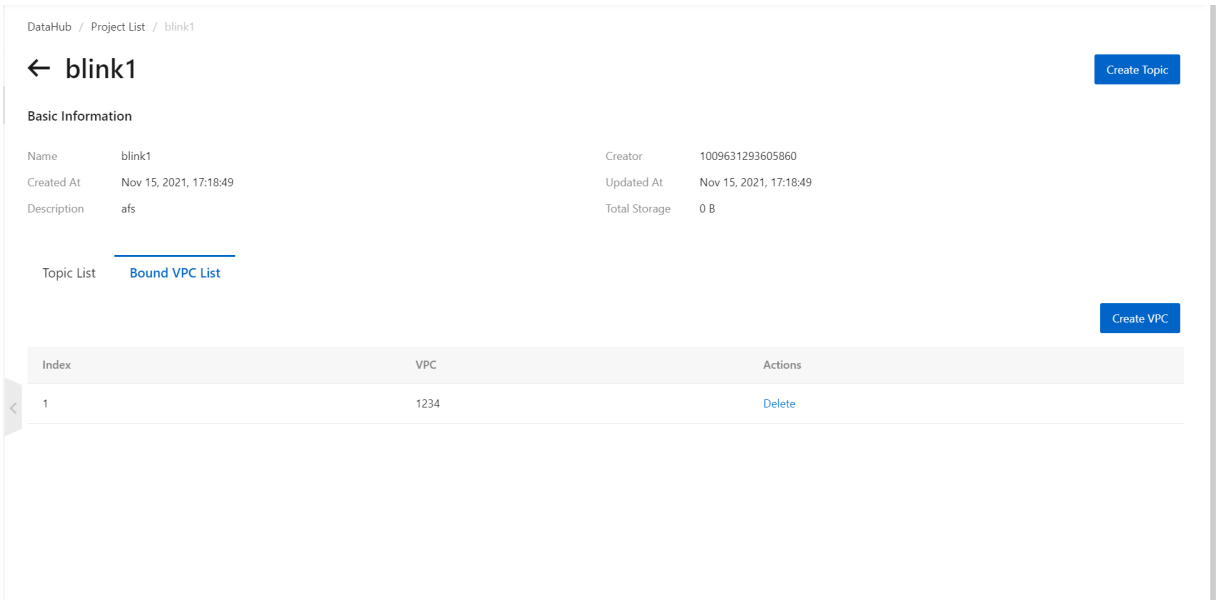
Bind a VPC to a DataHub project

You can bind a VPC to a DataHub project so that the DataHub project can be accessed only from IP addresses in this VPC. Perform the following operations to bind a VPC to a DataHub project:

1. On the project details page, click **Create VPC** on the **Bound VPC List** tab. In the dialog box that appears, enter the name of the VPC and click **Create VPC**.



2. To delete a VPC, click **Delete** in the Actions column.



5.1.3.4. 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

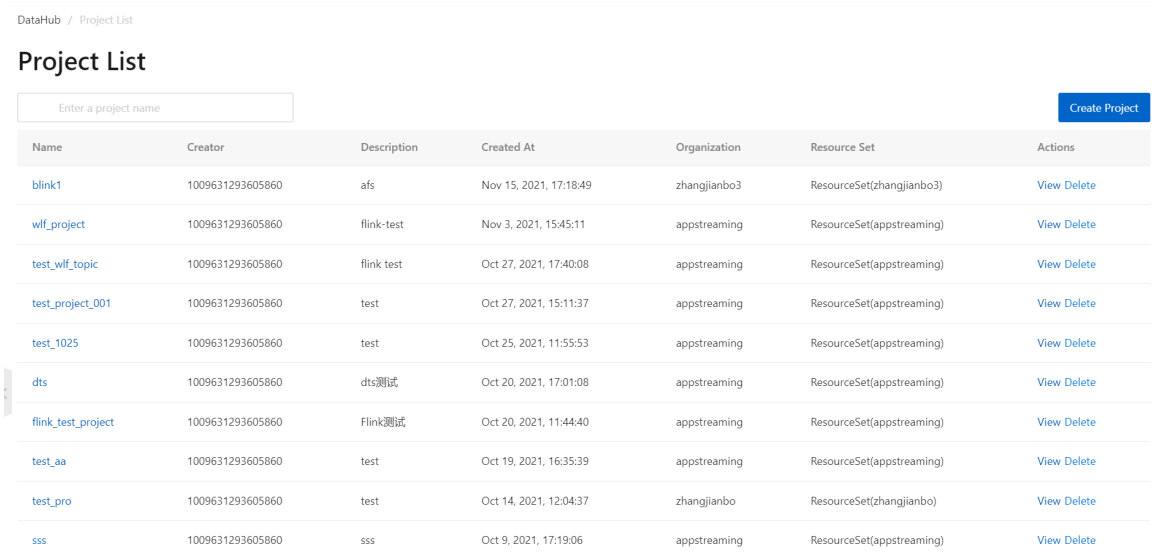
A project is created.

Considerations

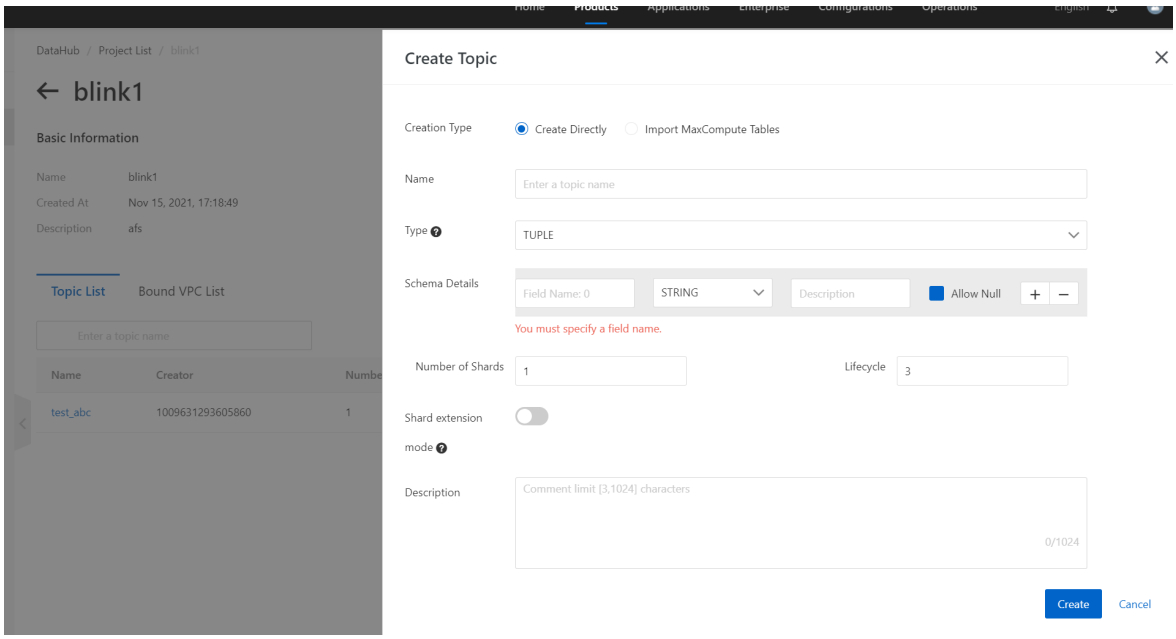
You can create up to 1,000 topics in each project. Contact the administrator if you need to create more topics.

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 project for which you want to create a topic and click **View** in the Actions column.



3. On the project details page, click **Create Topic**.
4. In the Create Topic panel, set relevant parameters and click **Create**.



Note

- After a topic is created, you can click **View** in the Actions column to view the details of the created topic.
- DataHub allows you to directly create a topic or create a topic by importing a table schema from MaxCompute.

5.1.3.5. 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.

5.1.3.6. 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**.

5.1.3.7. 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**.

5.1.4. Access Control

5.1.4.1. 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.

5.1.4.2. DataHub resources in RAM

RAM users can access the following resources in DataHub: projects, topics, and subscriptions. Subscription is the action that you specify an application to read and process the records in topics of a specific project. DataHub supports the RAM authentication of each project, topic, and subscription. RAM authentication is not supported at the shard level.

In RAM, each resource type has an Alibaba Cloud Resource Name (ARN) format to describe the specific object of the resource type. For example, the ARN format of a project that resides in a specific region is *acs:dhs:$region:$accountid:projects/$projectName*. The *$region, $accountid, and $projectName* fields indicate the region that the project resides, the user ID, and the project name.

ARN format for different resource types

Resource type	ARN format
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/*

5.1.4.3. 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

5.1.4.4. 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.

5.1.4.5. Sample RAM authorization policy content

5.1.4.5.1. 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"
    }
  ]
}
```

5.1.4.5.2. 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"
    }
  ]
}
```

5.1.5. Data Acquisition

5.1.5.1. 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).

5.1.5.2. 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/proj
ect/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

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.
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.

5.1.5.3. 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/proj
ect/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+(? <lo
g_level>\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.

Parameter	Description
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_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.

5.1.5.4. 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 fo
r installing 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 configur
ed.
INVENTORY_LOCATION=
#The UNIX group of the inventory directory. In this example, OGG is installed by using the o
gg_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 100m au
toextend 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, SUPPLEMENT
AL_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/log4
j.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 m
appings. At least one of them must be specified.-->
        <project>YOUR_DATAHUB_PROJECT</project>
        <! --The AccessKey ID for accessing DataHub, which can be overwritten by datahubAcce
ssId 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 datahub
AccessKey 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 overwri
tten by ctypeColumn in the column mappings.-->
        <ctypeColumn>optype</ctypeColumn>
        <! -- The column in DataHub that indicates the data update time, which can be overwr
itten by ctimeColumn in the column mappings.-->
```

```
        <ctimeColumn>readtime</ctimeColumn>
        <! -- The column in DataHub that indicates the sequence number of the updated data,
which can be overwritten by cidColumn in the column mappings. The sequence number increases
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:
mm:ss.-->
    <defaultDateFormat>yyyy-MM-dd HH:mm:ss</defaultDateFormat>
    <! --(Optional) Indicates whether the system needs to ignore dirty records. Default valu
e: false.-->
    <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 r
etries. 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 updated.
                isShardColumn: (Optional) Indicates whether the shard ID is generated based
on the hash key value, which can be overwritten by shardId. Default value: false.
                isDateFormat: Indicates whether the timestamp is converted into a string bas
ed on dateFormat. Default value: true. If you set the value to false, the data type in the s
ource database must be long.
                dateFormat: The format that the timestamp is converted into. If this paramet
```

```
er is left blank, the default format is used.
                -->
                <column src="id" dest="id" isShardColumn="true"  isDateFormat="false" dateFo
rmat="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}/dat
ahub-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 pi
   d_before, 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.

5.1.6. Data synchronization

5.1.6.1. 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.

5.1.6.2. Synchronize data to MaxCompute

5.1.6.2.1. 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.

5.1.6.2.2. 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.

5.1.7. 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.

5.1.8. Data subscription

5.1.8.1. 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.

5.1.8.2. 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.

5.1.8.3. 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 consumption of
fsets 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 supports binary data transmission.
                            new AliyunAccount(accessId, accessKey), true))
            .build();
    RecordSchema schema = datahubClient.getTopic(projectName, topicName).getRecordSchema();
    OpenSubscriptionSessionResult openSubscriptionSessionResult = datahubClient.openSubscriptionSess
ion(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 o
r the record 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 sequ
ence number, the record expired.
            cursor = datahubClient.getCursor(projectName, topicName, shardId, CursorType.SEQUENCE, n
extSequence).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).get
Cursor();
        }
    }
    // 2. Read records and save consumption offsets. In this example, you read tuple records and sav
e consumption 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, sha
rdId, 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.
                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 subscrip
tion is offline. The SessionChange exception indicates that the subscription is consumed by other cl
ients.
            e.printStackTrace();
            throw e;
        } catch (SubscriptionOffsetResetException e) {
            // The consumption offset is reset. You must obtain the version information of the consu
mption offset again.
            SubscriptionOffset offset = datahubClient.getSubscriptionOffset(projectName, topicName,
subId, shardIds).getOffsets().get(shardId);
            subscriptionOffset.setVersionId(offset.getVersionId());
            // After the consumption offset is reset, you must obtain the cursor of the record at th
e consumption offset again. The method for obtaining the cursor depends on the method of resetting t
he consumption offset.
            // If both the sequence number and timestamp are specified to reset the consumption offs
et, you 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 obt
ain the cursor only based on the sequence number.
            // If only the timestamp is specified to reset the consumption offset, you can obtain th
e cursor only based on the timestamp.
            // Generally, preferentially obtain the cursor based on the sequence number. If the curs
or failed to be obtained based on the sequence number or the timestamp, obtain the cursor of the ear
liest record.
            cursor = null;
            if (cursor == null) {
                try {
                    long nextSequence = offset.getSequence() + 1;
```

```
                    cursor = datahubClient.getCursor(projectName, topicName, shardId, CursorType.SEQ
UENCE, 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_T
IME");
                }
            }
            if (cursor == null) {
                try {
                    cursor = datahubClient.getCursor(projectName, topicName, shardId, CursorType.SYS
TEM_TIME, 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 OLDES
T");
                }
            }
            if (cursor == null) {
                try {
                    cursor = datahubClient.getCursor(projectName, topicName, shardId, CursorType.OLD
EST).getCursor();
                    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.

5.1.9. Collaborative consumption

5.1.9.1. Note

DataHub-client-library encapsulates the Java SDK and integrates the consumer for collaborative consumption and the producer for distributing data evenly among shards.

5.1.9.2. 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.

5.1.9.3. 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
```

5.1.9.4. 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 cons
umption 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 cons
umption feature, specify the subscription ID, the shards to be read by the consumer, and the consump
tion 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("field
2"));
            }
        }
    } 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();
```

5.1.9.5. 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.