ALIBABA CLOUD

Alibaba Cloud

Apsara Stack Agility

User Guide

Product Version: 2009, Internal: V3.4.0

Document Version: 20210106

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.Apsara Uni-manager console

1.1. What is the Apsara Uni-manager console?

The Apsara Uni-manager console is a service capability platform based on the Alibaba Cloud Apsara Stack platform and designed for government and enterprise customers. This platform improves IT management and troubleshooting and is dedicated to providing a leading service capability platform of the cloud computing industry. It provides large-scale and cost-efficient end-to-end cloud computing and big data services for customers in industries such as government, education, healthcare, finance, and enterprise.

Overview

The Apsara Uni-manager console simplifies the management and deployment of physical and virtual resources by building an Apsara Stack platform that supports various business types of government and enterprise customers. The console helps you build your business systems in a simple and quick manner, fully improve resource utilization, and reduce O&M costs. This allows you to shift your focus from O&M to business. The console brings the Internet economy model to government and enterprise customers, and builds a new ecosystem chain based on cloud computing.

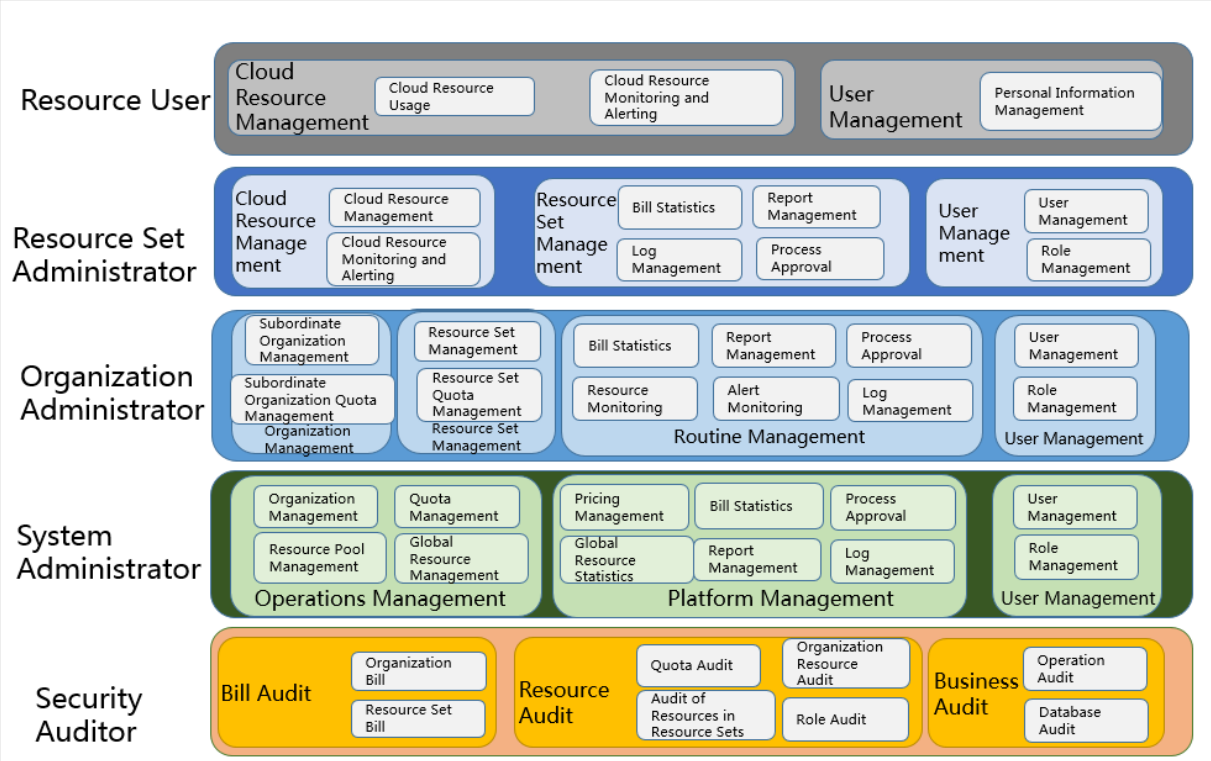
Workflow

Apsara Uni-manager console operations are divided into the following parts:

1. System initialization: This part is designed to complete basic system configurations, such as creating organizations, resource sets, and users, creating basic resources such as VPCs, and creating contacts and contact groups in Cloud Monitor.
2. Cloud resource creation: This part is designed to create resources.
3. Cloud resource management: This part is designed to complete resource management operations, such as starting, using, and releasing resources, and changing resource configurations and resource quotas.

1.2. User roles and permissions

This topic describes roles and their permissions.



Roles and permissions

Role	Permission
Resource user	This role has the permissions to view and modify resources in a resource set and create alert rules.
Resource set administrator	This role has the permissions to create, modify, and delete resources in a resource set and manage the users of the resource set.
Organization administrator	This role has the permissions to manage an organization and its subordinate organizations, create, modify, and delete the resources of organizations, create and view alert rules for resources, and export reports.
Operations administrator	This role has read and write permissions on all resources.
Security auditor	This role performs security audit on the Apsara Uni-manager console and has the read-only permissions on operation logs of the Apsara Uni-manager console.
Platform administrator	This role has the permissions to initialize the system and create operations administrators.
Resource auditor	This role has the read-only permissions on all resources in the Apsara Uni-manager console.

Role	Permission
Organization security administrator	This role manages the security of an organization, including the security of hosts, applications, and networks. This role has the read-only permissions on operation logs of the Apsara Uni-manager console and read and write permissions on ApsaraDB for RDS, ECS, and Apsara Stack Security.
Security system configuration administrator	This role configures system security features such as the upgrade center and global configurations. This role has read and write permissions on the upgrade, protection, and configuration features of Apsara Stack Security.
Global organization security administrator	This role manages the security of global tenants by using Cloud Security Operation Center (SOC). This role has read and write permissions on all features of Apsara Stack Security.
Platform security administrator	This role manages the security of the Apsara Uni-manager console by using SOC.
Global organization security auditor	This role checks the security conditions of all organizations by using SOC. This role has the read-only permissions on operation logs of the Apsara Uni-manager console and all features of Apsara Stack Security.
Platform security auditor	This role checks the security conditions of the Apsara Uni-manager console by using SOC. This role has the read-only permissions on operation logs of the Apsara Uni-manager console, Server Guard, Cloud Firewall, Sensitive Data Discovery and Protection, SOC, system configurations, and Web Application Firewall (WAF) configurations as well as read and write permissions on Anti-DDoS, Threat Detection, and Update Center of Apsara Stack Security.
Platform Security Configuration Administrator	This role configures and has read and write permissions on security services in the Apsara Uni-manager console, such as Server Guard and WAF.
Organization resource auditor	This role has the read-only permissions on all resources in an organization to which it belongs.

Roles and permissions

Role	Permission
Resource user	This role has the permissions to view and modify resources in a resource set and create alert rules.
Resource set administrator	This role has the permissions to create, modify, and delete resources in a resource set and manage the users of the resource set.

Role	Permission
Organization administrator	This role has the permissions to manage an organization and its subordinate organizations, create, modify, and delete the resources of organizations, create and view alert rules for resources, and export reports.
Operations administrator	This role has read and write permissions on all resources.
Platform administrator	This role has the permissions to initialize the system and create operations administrators.
Organization security administrator	This role manages the security of an organization, including the security of hosts, applications, and networks. This role has the read-only permissions on operation logs of the Apsara Uni-manager console and read and write permissions on ApsaraDB for RDS, ECS, and Apsara Stack Security.
Resource auditor	This role has the read-only permissions on all resources in the Apsara Uni-manager console.
Organization resource auditor	This role has the read-only permissions on all resources in an organization to which it belongs.

1.3. Log on to the Apsara Uni-manager console

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

Prerequisites

- The domain name of the Apsara Uni-manager console is obtained from the deployment personnel before you log on to the Apsara Uni-manager console.
- A browser is available. We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to log on to the Apsara Uni-manager console. 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.

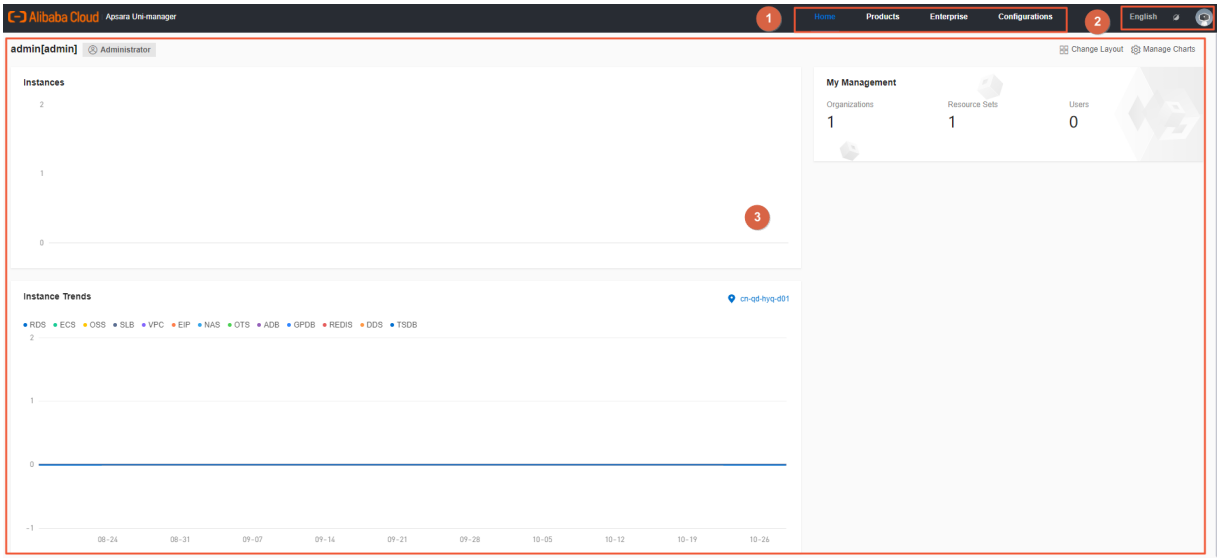
> **Note** When you log on to the Apsara Uni-manager console for the first time, you must change the password of your username. For security reasons, your password must meet the minimum complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:
>
> - Uppercase or lowercase letters.
> - Digits.
> - Special characters. Special characters include exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to go to the Apsara Uni-manager console homepage.

1.4. Web page introduction

The web page of the Apsara Uni-manager console consists of the top navigation bar, information section of the current logon user, and operation section.

Apsara Uni-manager console page



Functional sections of the web page

Section	Description

Section		Description
1	Top navigation bar	This section includes the following modules: • **Home**: uses charts to display the usage and monitoring data of existing system resources in each region. • **Products**: manages all types of basic cloud services and resources. • **Enterprise**: manages organizations, resource sets, roles, users, logon policies, user groups, and resource pools. • **Configurations**: manages password policies, specifications, menus, and RAM roles. • **Security**: provides operation logs and system logs.
2	Information section of the current logon user	• English: allows you to switch between English, simplified Chinese, and traditional Chinese. • : allows you to switch between day and night mode. • User Information: Click the icon of the current logon user. The **User Information** and **Exit** menu items are displayed. On the **User Information** page, you can perform the following operations: ◦ View basic information. ◦ Modify personal information. ◦ Change the logon password. ◦ View the AccessKey pair of your Apsara Stack tenant account. ◦ Switch the current role. ◦ Enable or disable alert notification.
3	Operation section	**Operation section**: the information display and operation section.

1.5. Initial configuration

1.5.1. Configuration description

Before you use the Apsara Uni-manager console, you must complete a series of basic configuration operations as an administrator, such as creating organizations, resource sets, users, and roles and initializing resources. This is the initial system configuration.

Based on the service-oriented principle, the Apsara Uni-manager console manages the organizations, resource sets, users, and roles of cloud data centers in a centralized manner to grant different resource access permissions to different users.

- Organization

After the Apsara Uni-manager console is deployed, a root organization is automatically generated. You can create other organizations under the root organization.

Organizations are displayed in a hierarchical structure. You can create subordinate organizations under each organization level.

- Resource Set

  A resource set is a container used to store resources. Each resource must belong to a resource set.

- User

  A user is a resource manager and user.

- User group

  A role is a set of access permissions. You can assign different roles to different users to meet different requirements for system access control.

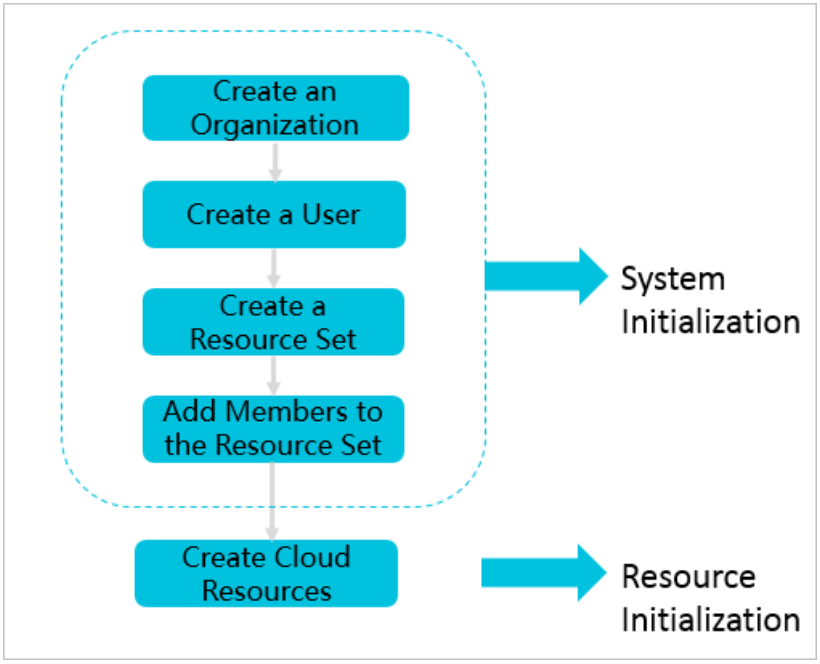
The following table describes the relationships among organizations, resource sets, users, roles, and cloud resources.

Relationship between two items	Relationship type	Description
Organization and resource set	One-to-many	An organization can have multiple resource sets, but each resource set can belong to only a single organization.
Organization and user	One-to-many	An organization can have multiple users, but each user can belong to only a single organization.
Resource set and user	Many-to-many	A user can have multiple resource sets, and a resource set can be assigned to multiple users under the same level-1 organization.
User and role	Many-to-many	A user can have multiple roles, and a role can be assigned to multiple users.
Resource set and resource	One-to-many	A resource set can have multiple resources, but each cloud resource can belong to only a single resource set.

1.5.2. Configuration process

This topic describes the initial configuration process.

Before you use the Apsara Uni-manager console, you must complete the initial system configurations as an administrator based on the process shown in the following figure.



1. Create an organization

   Create an organization to store resource sets and their resources.

2. Create a user

   Create a user and assign the user different roles to meet different requirements for system access control.

3. Create a resource set

   Create a resource set before you apply for resources.

4. Add a member to a resource set

   Add users to the resource set.

5. Create cloud resources

   Create instances in each service console based on project requirements. For more information about how to create cloud service instances, see the user guide of each cloud service.

1.6. Monitoring

1.6.1. View the workbench

The Apsara Uni-manager console uses charts to keep you up to date on the current usage of resources.

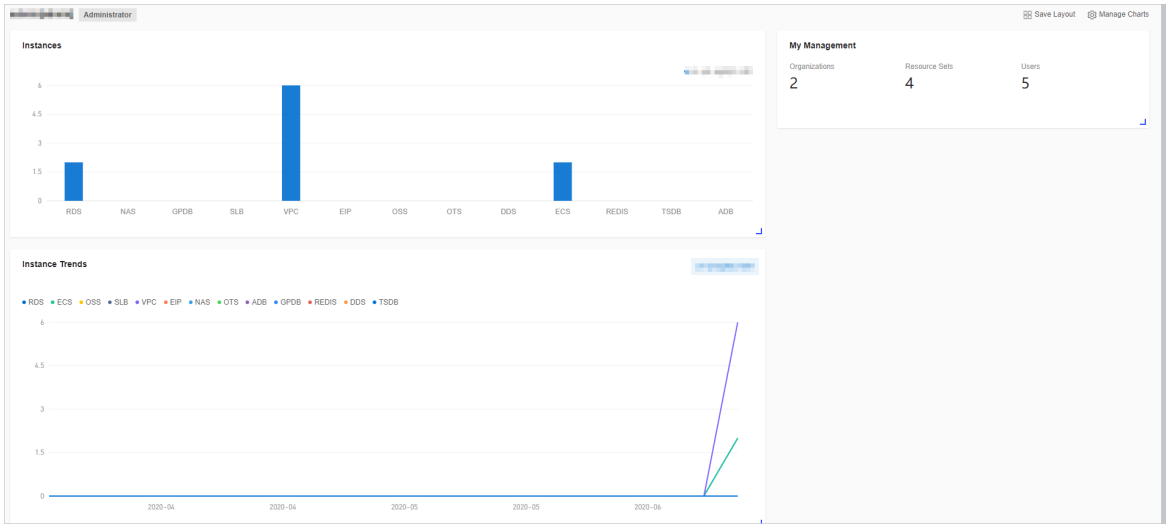
Context

> **Note** The resource types displayed may vary with region types. See your dashboard for available resource types.

Procedure

1. Log on to the Apsara Uni-manager console.By default, the **workbench** page appears when you

log on to the Apsara Uni-manager console. To return to the workbench page from other pages, click **Home** in the top navigation bar.



2. On the **workbench** page, you can view the instance summary information for all regions of the Apsara Stack environment.

   You can click **Manage Charts** in the upper-right corner of the page to select all or individual modules to view relevant information. You can also click **Change Layout** in the upper-right corner of the page and drag a specific module to a location.

   - **Instances**

     Shows the numbers of ECS instances, ApsaraDB for RDS instances, OSS buckets, and SLB instances in each region.

   - **Instance Trends**

     Shows the numbers of ECS instances, ApsaraDB for RDS instances, SLB instances, and OSS buckets for the last five days.

   - **My Management**

     Shows the numbers of organizations, resource sets, and users.

1.6.2. CloudMonitor

1.6.2.1. Cloud Monitor overview

Cloud Monitor provides real-time monitoring, alerting, and notification services for resources to protect your services and businesses.

Cloud Monitor can monitor metrics for ECS, ApsaraDB RDS for MySQL, and KVStore for Redis.

You can use the monitoring metrics of cloud services to configure alert rules and notification policies. This way, you can stay up to date on the running status and performance of your service instances and scale resources in a timely manner when resources are insufficient.

1.6.2.2. Metrics

This topic describes the monitoring metrics available for each service.

Cloud Monitor checks the availability of a service based on its monitored metrics. You can configure alert rules and notification policies for these metrics to stay up to date on the running status and performance of monitored service instances.

Cloud Monitor can monitor resources of Elastic Compute Service (ECS), ApsaraDB for RDS, and KVStore for Redis. The following tables list the metrics for each service.

Operating system metrics for ECS

Metric	Description	Unit
Host.cpu.total	The total number of CPU cores of an ECS instance.	%
Host.mem.usedutilization	The memory usage of an ECS instance.	%
Host.disk.utilization	The disk usage of an ECS instance.	%

Basic metrics for ECS

Metric	Description	Unit
CPU utilization	The CPU utilization of an ECS instance.	%
System disk BPS	The number of bytes read from and written to the system disk per second.	bytes/s
System disk IOPS	The number of reads from and writes to the system disk per second.	Count/Second

> **Note**
>
> For ECS instances, you must install a monitoring plug-in to collect metric data at the operating system level.
>
> Installation method: In the ECS instance list on the **CloudMonitor** page, select the instance to be monitored and click **Batch Install** below the instance list.
>
> The monitoring chart will begin to display data within 5 to 10 minutes of the monitoring plug-in being installed.

Metrics for ApsaraDB for RDS

Metric	Description	Apsara Stack service	Calculation formula

Metric	Description	Apsara Stack service	Calculation formula
CPU utilization	The CPU utilization of an ApsaraDB for RDS instance. Unit: %.	ApsaraDB for RDS	CPU utilization of an ApsaraDB for RDS instance/Total CPU cores of the ApsaraDB for RDS instance
Memory usage	The memory usage of an ApsaraDB for RDS instance. Unit: %.	ApsaraDB for RDS	Used memory of an ApsaraDB for RDS instance/Total memory of the ApsaraDB for RDS instance
Disk usage	The disk usage of an ApsaraDB for RDS instance. Unit: %.	ApsaraDB for RDS	N/A
IOPS utilization	The number of I/O requests for an ApsaraDB for RDS instance per second. Unit: %.	ApsaraDB for RDS	Number of I/O requests for an ApsaraDB for RDS instance/Statistical period
Connections utilization	The number of connections between an application and an ApsaraDB for RDS instance per second. Unit: %.	ApsaraDB for RDS	Number of connections between an application and an RDS instance per second/Statistical period

Metrics for KVStore for Redis

Metric	Description	Apsara Stack service	Unit
CPU utilization	The CPU utilization of a KVStore for Redis instance.	KVStore for Redis	%
Memory usage	The percentage of total memory in use.	KVStore for Redis	%
Used memory	The amount of memory that is in use.	KVStore for Redis	Bytes
Number of used connections	The total number of client connections.	KVStore for Redis	N/A
Percentage of connections in use	The percentage of total connections that are in use.	KVStore for Redis	%
Write bandwidth	The write traffic per second.	KVStore for Redis	Bytes/s
Read bandwidth	The read traffic per second.	KVStore for Redis	Bytes/s
Number of failed operations per second	The number of failed operations on a KVStore for Redis instance per second.	KVStore for Redis	N/A
Write bandwidth usage	The percentage of total bandwidth used by write operations.	KVStore for Redis	%
Read bandwidth usage	The percentage of total bandwidth used by read operations.	KVStore for Redis	%

Metric	Description	Apsara Stack service	Unit
Used QPS	The number of queries per second (QPS).	KVStore for Redis	N/A
QPS usage	The QPS utilization rate.	KVStore for Redis	%
Average response time	The average response time.	KVStore for Redis	ms
Maximum response time	The maximum response time.	KVStore for Redis	ms
Number of failed commands	The number of failed commands.	KVStore for Redis	N/A
Hit rate	The current hit rate.	KVStore for Redis	%
Inbound traffic	The inbound traffic to a KVStore for Redis instance.	KVStore for Redis	Bytes
Inbound bandwidth usage	The inbound bandwidth usage of a KVStore for Redis instance.	KVStore for Redis	%
Outbound traffic	The outbound traffic from a KVStore for Redis instance.	KVStore for Redis	Bytes
Outbound bandwidth usage	The outbound bandwidth usage of a KVStore for Redis instance.	KVStore for Redis	%

1.6.2.3. View monitoring charts

You can view monitoring charts to obtain up-to-date information about each instance.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, choose **Products > Monitoring and O&M > CloudMonitor**.
3. In the left-side navigation pane of the CloudMonitor page, click **Cloud Service Monitoring**.
4. Click a cloud service.
5. Click **Monitoring Charts** in the **Actions** column corresponding to an instance.On the Monitoring Charts page that appears, you can select a date and time to view the monitoring data of each metric.

1.6.3. Alerts

1.6.3.1. View alert overview

On the **Overview** page in Cloud Monitor, you can view the alert status statistics and alert logs.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.

2. In the top navigation bar, choose **Products > Monitoring and O&M > CloudMonitor**.
3. In the left-side navigation pane of the Cloud Monitor page, click **Overview**.
4. On the **Overview** page, view the alert status statistics and alert logs that are generated in the last 24 hours.

1.6.3.2. View alert logs

You can view alert information to stay up to date on the running status of instances.

Context

Alert information contains information for all items that do not comply with your configured alert rules.

> **Note** The system can retain up to one million alert items generated within the last three months.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, choose **Products > Monitoring and O&M > CloudMonitor**.
3. In the left-side navigation pane of the Cloud Monitor page, choose **Alerts > Alert History**.
4. On the **Alert Rule History List** page, filter alert information by rule ID, rule name, service, and date.The following table describes the fields in the query result.

Alert information fields

Field	Description
Product	The service for which the alert was triggered.
Fault Instance	The instance for which the alert was triggered.
Occurred At	The time when the alert was triggered.
Rule Name	The name of the alert rule.
Status	The status of the alert rule.
Notification Contact	The recipient of the alert notification.

1.6.3.3. Alert rules

1.6.3.3.1. View alert rules

After you create alert rules, you can view your alert rules on the Alert Rules page.

Procedure

1. Log on to the Apsara Uni-manager console.
2. In the top navigation bar, choose **Products > Monitoring and O&M > CloudMonitor**.

3. In the left-side navigation pane of the Cloud Monitor page, click **Cloud Service Monitoring**.
4. Click a cloud service.
5. Click **Alert Rules** in the **Actions** column corresponding to an instance to go to its **Alert Rules** page.On the **Alert Rules** page, view the detailed information of alert rules.

1.6.3.3.2. Create an alert rule

You can create an alert rule to monitor an instance.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, choose **Products > Monitoring and O&M > CloudMonitor**.
3. In the left-side navigation pane of the Cloud Monitor page, click **Cloud Service Monitoring**.
4. Click a cloud service.
5. Click **Alert Rules** in the **Actions** column corresponding to an instance.

   > **Note** You can also use the search function to query specific instances for which you want to create alert rules.

6. On the **Alert Rules** page, click **Create Alert Rule**.

Parameters for creating an alert rule

Parameter	Description
Product	The monitored cloud product.
Resource Range	The range of resources that is associated with the alert rule.
Rule Description	The description of the alert rule.
Add Rule Description	Click **Add Rule Description** to go to the rule configuration panel. For more information, see Parameters for adding rule description.
Effective Time	Only a single alert is sent during each mute duration, even if the metric value exceeds the alert rule threshold several times in a row.
Effective Period	An alert is sent only when the threshold is crossed during the effective period.
HTTP CallBack	The callback URL when the alert conditions are met.
Alert Contact Group	The group to which alerts are sent.

Parameters for adding rule description

Parameter	Description

Parameter	Description
Rule Name	The name of the alert rule. The name must be 1 to 64 characters in length and can contain letters and digits.
Metric Name	Different products have different monitoring metrics. For more information, see Metrics.
Comparison	The comparison between thresholds and observed values. The comparison operators include >, >=, <, and <=. When the comparison rule is satisfied, an alert rule is triggered.
Threshold And Alert Level	Different metrics have different reference thresholds.

7. Click **OK**.

1.6.3.3.3. Disable an alert rule

You can disable one or more alert rules.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, choose **Products > Monitoring and O&M > CloudMonitor**.
3. In the left-side navigation pane of the Cloud Monitor page, click **Cloud Service Monitoring**.
4. Click a cloud service.
5. Click **Alert Rules** in the **Actions** column corresponding to an instance.
6. On the **Alert Rules** page, choose **More > Disable** in the **Actions** column corresponding to the alert rule to be disabled.
7. In the Disable Alert Rule message, click **Confirm**.

1.6.3.3.4. Enable an alert rule

After an alert rule is disabled, you can re-enable it

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, choose **Products > Monitoring and O&M > CloudMonitor**.
3. In the left-side navigation pane of the Cloud Monitor page, click **Cloud Service Monitoring**.
4. Click a cloud service.
5. Click **Alert Rules** in the **Actions** column corresponding to an instance.
6. On the **Alert Rules** page, choose **More > Enable** in the **Actions** column corresponding to the alert rule to be enabled.
7. In the Enable Alert Rule message, click **Confirm**.

1.6.3.3.5. Delete an alert rule

You can delete alert rules that are no longer needed.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, choose **Products > Monitoring and O&M > CloudMonitor**.
3. In the left-side navigation pane of the Cloud Monitor page, click **Cloud Service Monitoring**.
4. Click a cloud service.
5. Click **Alert Rules** in the **Actions** column corresponding to an instance.
6. On the **Alert Rules** page, click **Delete** in the **Actions** column corresponding to the alert rule to be deleted.
7. In the Delete Alert message, click **Confirm**.

1.7. Enterprise

1.7.1. Organizations

1.7.1.1. Create an organization

You can create organizations to store resource sets and their resources.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Organizations**.
4. In the organization navigation tree, click a parent organization. In the Current Organization section, click **Add Organization**.
5. In the Add Organization dialog box, enter an organization name and click **OK**.

1.7.1.2. Query an organization

You can query an organization by name to view its resource sets, users, and user groups.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Organizations**.
4. In the search box below **Organizations**, enter an organization name.You can view the information about the corresponding organization.

1.7.1.3. View organization information

You can view information about an organization on the Organizations page.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Organizations**.
4. On the **Organizations** page, click an organization in the organization list.
5. In the right-side area, view the organization information.
   - In the **Resource Sets** section, you can view information such as the name, creation time, and creator of each resource set in the organization. Click the name of a resource set to view its details.
   - In the **Users** section, you can view information such as the name, status, and role of each user in the organization. Click a username to view the user details.
   - In the **User Groups** section, you can view the name, organization, role, members, and creation time of each user group in the organization.

1.7.1.4. Modify the name of an organization

Users that have operation permissions on an organization can modify the name of the organization.

Procedure

1. Log on to the Apsara Uni-manager console.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Organizations**.
4. In the organization navigation tree, click an organization name.
5. In the Current Organization section, click **Edit Organization**.
6. In the Edit Organization dialog box, modify the organization name.
7. Click **OK**.

1.7.1.5. Change organization ownership

Users that have operation permissions on organizations can change the ownership of organizations.

Prerequisites

- Make sure that each organization under the organization that you want to change the ownership has a unique name.
- The ownership of an organization cannot be changed cross level-1 organizations.

Context

Users can change the ownership of an organization cross parent organizations. This way, the ownership of subordinate organizations, users, and resources are also changed in a cascading manner.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.

2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Resource Sets**.
4. On the **Change Ownership** page, select an organization and click **Change Organization** in the upper-right corner.
5. In the **Change Organization** dialog box, select the destination organization and click **OK** to change the ownership of the organization along with that of its resources sets and users.

1.7.1.6. Obtain the AccessKey pair of an organization

An AccessKey pair consists of an AccessKey ID and an AccessKey secret. The AccessKey pair is used to implement symmetric encryption to verify the identity of the requester. The AccessKey ID is used to identify a user. The AccessKey secret is used to encrypt the signature string. This topic describes how to obtain the AccessKey pair of an organization.

Prerequisites

Only operations administrators and level-1 organization administrators can obtain the AccessKey pair of an organization.

Procedure

1. Log on to the Apsara Uni-manager console.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Organizations**.
4. In the organization navigation tree, click an organization name.
5. In the Current Organization section, click **AccessKey**.
6. In the AccessKey message, view the AccessKey pair of the organization.

1.7.1.7. Delete an organization

Administrators can delete organizations that are no longer needed.

Prerequisites

> **Note** Before you delete an organization, ensure that the organization does not contain users, resource sets, or subordinate organizations. Otherwise, the organization cannot be deleted.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Organizations**.
4. In the organization navigation tree, click an organization name. In the **Current Organization** section, click **Delete Organization**.
5. In the Confirm message, click **OK**.

1.7.2. Resource sets

1.7.2.1. Create a resource set

You must create a resource set before you apply for resources.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Resource Sets**.
4. In the upper-left corner of the **Resource Sets** page, click **Create Resource Set**.
5. In the **Create Resource Set** dialog box, set **Name** and **Organization**.
6. Click **OK**.

1.7.2.2. View the details of a resource set

When you need to use a cloud resource in your organization, you can view the details of the resource set that contains the resource, including all resource instances and members of the resource set.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Resource Sets**.
4. Select an organization from the **Organization** drop-down list, or enter a resource set name in the search bar.
5. Click **Search**.
6. Click the name of the resource set that you want to view to go to the **Resource Set Details** page.Click the **Resources** and **Members** tabs to view information about all resource instances and members of the resource set.

1.7.2.3. Modify the name of a resource set

An administrator can modify the name of a resource set to keep it up-to-date.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Resource Sets**.
4. Click **More** in the **Actions** column corresponding to a resource set, and choose **Edit Name** from the shortcut menu.
5. In the dialog box that appears, enter the new name.
6. Click **OK**.

1.7.2.4. Add a member to a resource set

You can add a member to a resource set so that the member can use the resources in the resource set.

Prerequisites

Before adding a member, make sure that the following prerequisites are met:

- A resource set is created. For more information, see Create a resource set.
- A user is created. For more information, see Create a user.

Context

Members of a resource set have the permissions to use resources in the resource set.

Deleting resources from a resource set does not affect the members of the resource set. Similarly, deleting members from a resource set does not affect the resources in the resource set.

You can delete a member that is no longer in use in a resource set. After the member is deleted, it will no longer be able to access the resource set.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Resource Sets**.
4. Click **More** in the **Actions** column corresponding to a resource set, and choose **Add Member** from the shortcut menu.
5. In the dialog box that appears, select a username.
6. Click **OK**.

1.7.2.5. Add or remove a user group of a resource set

You can add or remove a user group of a resource set to manage user group access to resources in the resource set.

Prerequisites

- A resource set is created. For more information, see Create a resource set.
- A user group is created. For more information, see Create a user group.

Context

User groups in a resource set have the permissions to use resources in the resource set.

Deleting resources from a resource set does not affect user groups of the resource set. Similarly, deleting user groups from a resource set does not affect the resources in the resource set.

You can delete a user group that is no longer in use in a resource set. After the user group is deleted, it will no longer be able to access the resource set.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Resource Sets**.

4. Click **More** in the **Actions** column corresponding to the target resource set.
5. Add or remove a user group.
   - Select **Add User Group**. In the dialog box that appears, select a user group. Click **OK** to add the user group.
   - Select **Delete User Group**. In the dialog box that appears, select a user group. Click **OK** to remove the user group.

1.7.2.6. Delete a resource set

You can delete resource sets that are not needed as an administrator.

Prerequisites

Ensure that the resource set to be deleted does not contain resources, users, or user groups.

> **Notice** A resource set cannot be deleted if it contains resources, users, or user groups.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Resource Sets**.
4. Click **More** in the **Actions** column corresponding to the target resource set, and select **Delete**.
5. In the message that appears, click **OK**.

1.7.3. Roles

1.7.3.1. Create a custom role

You can create custom roles in the Apsara Uni-manager console to more efficiently grant permissions to users so that different personnel can work with different functions.

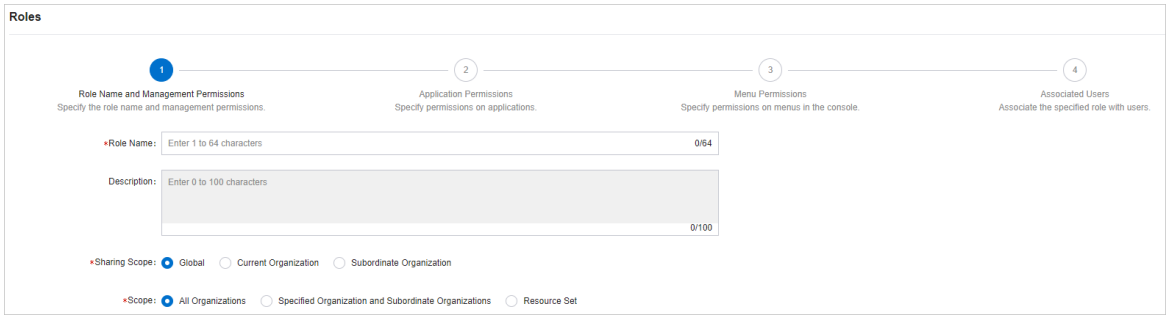
Context

A role is a set of access permissions. Each role has a range of permissions. A user can have multiple roles, which means that the user is granted all of the permissions defined for each role. A role can be used to grant the same set of permissions to a group of users.

The total number of custom and default roles cannot exceed 20.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the upper-right corner of the page, click **Create Custom Role**.
5. On the **Roles** page, set the role name and management permissions.



The following table describes the role parameters.

Role parameters

Parameter	Description
Role Name	The name of the RAM role. The name can be up to 15 characters in length and can contain only letters and digits.
Description	Optional. The description of the role. The description can be up to 100 characters in length and can contain letters, digits, commas (,), semicolons (;), and underscores (_).
Sharing Scope	◦ **Global** The role is visible and valid to all organizations involved. The default value is Global. ◦ **Current Organization** The role is visible and valid to the organization to which the user belongs. ◦ **Subordinate Organization** The role is visible and valid to the organization to which the user belongs and its subordinate organizations.
Scope	◦ **All Organizations** The permissions apply to all organizations involved. ◦ **Specified Organization and Subordinate Organizations** The permissions apply to the organization to which the user belongs and its subordinate organizations. ◦ **Resource Sets** The permissions apply to the resource sets that are assigned to the user.

6. Select the operation permissions that this role has, and click **Next**.
7. In the **Application Permissions** step, select the operation permissions that this role has on the cloud services, and click **Next**.
8. In the **Menu Permissions** step, select the operation permissions that this role has on the menus and the homepage template corresponding to the role, and click **Create Role**.
9. In the **Associated Users** step, select the users associated with the role from the drop-down list. The associated users are granted the permissions of the role.

1.7.3.2. View the details of a role

If you are not certain about the specific permissions of a role, go to the **Roles** page to view the role permissions.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. Click the name of the role that you want to view. On the **Roles** page, view information about the role.

1.7.3.3. Modify custom role information

You can modify the name and permissions of a custom role as an administrator.

Context

You cannot modify information about preset roles.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the role name list, choose **More** > **Modify** in the **Actions** column corresponding to a custom role.
5. On the **Roles** page, modify the custom role name, permissions, and associated users or user groups.
   - Modify the role name: Enter a new role name in the **Role Name** field.
   - Modify permissions: Click the **Management Permissions**, **Application Permissions**, or **Menu Permissions** tab, select or clear related permissions from the corresponding tab, and then click **Update**.
   - Bind a user to a role: Click the **Associated Users** tab and select a user from the **Select one or more users** drop-down list to add the user. To unbind the user from the role, click **Remove** in the **Actions** column.
   - Manage user groups: Click the **User Groups** tab, click **Add User Group**, select a user group from the drop-down list, and then click **OK** to bind the user group. To unbind the user group from the role, click **Remove** in the **Actions** column.

1.7.3.4. Copy a role

You can copy a preset role or a custom role to create a role that has the same permissions.

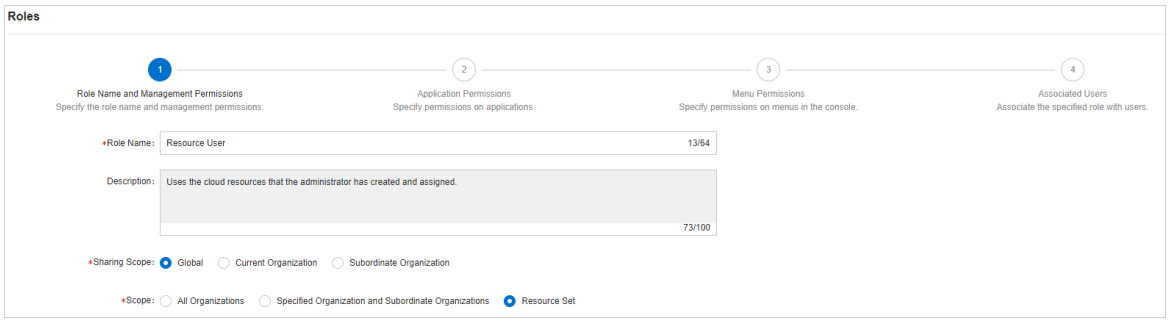
Context

Operations on the **Roles** page are the same as those for creating a custom role. You can add, modify, and remove the role permissions in the copied role. By default, if you do not modify the role permissions, the sharing scope, management permissions, application permissions, menu permissions,

and associated users of the copied role are all the same as those of the source role.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the role list, choose **More** > **Copy** in the **Actions** column corresponding to a role.
5. On the **Roles** page, set the new role name, sharing scope, and management permissions.



> **Note** The role name must be unique.

6. Select the operation permissions that this role has and click **Next**.
7. In the **Application Permissions** step, select the operation permissions that this role has on the cloud services and click **Next**.
8. In the **Menu Permissions** step, select the operation permissions that this role has on the menus and click **Create Role**.
9. In the **Associated Users** step, select the users that are associated with the role from the drop-down list.
   The associated users are granted the permissions of the role.

1.7.3.5. Disable a role

When you disable a role, the permissions of the role are disabled.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the role list, click **More** in the **Actions** column corresponding to a role and choose **Disable** from the shortcut menu.

1.7.3.6. Enable a role

When you enable a disabled role, the permissions of the role are restored.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the role list, click **More** in the **Actions** column corresponding to a disabled role and choose **Enable** from the shortcut menu.

1.7.3.7. Delete a custom role

You can delete a custom role that is no longer needed.

Prerequisites

- Default or preset roles cannot be deleted.
- To delete a role, you must unbind all user groups from the role.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. Choose **More** > **Delete** in the **Actions** column corresponding to a role.
5. In the Confirm message, click **OK**.

1.7.4. Users

1.7.4.1. System users

1.7.4.1.1. Create a user

You can create a user and assign the user different roles as an administrator to meet different requirements for system access control.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Users**.
4. Click **Create**.
5. In the Create User dialog box, configure the parameters.

Parameter	Description
Username	The Apsara Stack account name of the user. The name must be 3 to 30 characters in length, and can contain letters, digits, hyphens (-), underscores (_), periods (.), and at signs (@). It must start with a letter or digit.

Parameter	Description
Display Name	The display name of the user. The name must be 2 to 30 characters in length, and can contain letters, digits, hyphens (-), underscores (_), periods (.), and at signs (@).
Roles	The role to be assigned to the user.
Organization	The organization to which the user belongs.
Logon Policy	The logon policy that restricts the logon time and IP addresses of the user. The default policy is automatically bound to new users. **Note** The default policy does not restrict the time period and IP addresses for users to log on. To restrict the logon time and IP addresses of a user, you can modify the logon policy of the user or create a logon policy for the user. For more information, see Create a logon policy.
Mobile Number	The mobile number of the user. The mobile number is used by the system to notify users of resource application and usage. Ensure that the entered mobile number is correct. **Note** If the mobile number is changed, update it on the system in a timely manner.
Landline Number	Optional. The landline number of the user. The landline number must be 4 to 20 characters in length, and can contain only digits and hyphens (-).
Email	The email address of the user. Emails about the usage and requests for resources will be sent to the email address. Ensure that the specified email address is correct. **Note** If the email address is changed, update it on the system in a timely manner.
DingTalk Key	The key of the chatbot for the DingTalk group where the user is a member. For more information about how to configure the key, see DingTalk development documentation.
Notify User by SMS	After this option is selected, the Apsara Uni-manager console informs the user configured as the alert contact by SMS whenever an alert is generated. **Note** You must configure an SMS server to receive an SMS message each time an alert is triggered. For more information, contact on-site O&M engineers.

Parameter	Description
Notify User by Email	After this option is selected, the Apsara Uni-manager console will inform the user configured as the alert contact by email whenever an alert is generated. **Note** You must configure an email server to receive an email each time an alert is triggered. For more information, contact on-site O&M engineers.
Notify User by DingTalk	After this option is selected, the Apsara Uni-manager console will inform the user configured as the alert contact by DingTalk whenever an alert is generated.

6. Click **OK**.

1.7.4.1.2. Query a user

You can view user information such as name, organization, mobile number, email address, role, logon time, and initial password.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Users**.
4. Click the **System Users** tab.
5. Set **Username**, **Organization**, or **Role**, and then click **Search**.
6. Click **More** in the **Actions** column corresponding to a user, and choose **User Information** from the shortcut menu to view basic information about the user.

1.7.4.1.3. Change user roles

You can add, change, and delete roles for a user.

Change user roles by using user management

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Users**.
4. Click the **System Users** tab.
5. Choose **More** > **Authorize** in the **Actions** column corresponding to a user.
6. In the **Role** field, add, delete, or change user roles.
7. Click **OK**.

Change user roles by changing ownership

1. Log on to the Apsara Uni-manager console as an administrator.

2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Change Ownership**.
4. Click the ▸ icon to the left of an organization and click **Users**.
5. In the **Users** section on the right, set **Logon Policy** and **Role** or **Username**, and click **Search** to query a user.
6. Find the user and click **Change Ownership** in the **Actions** column.
7. In the **Organization to Change** dialog box, select the destination or original organization and select the role to be added or removed from the **Assigned Roles** drop-down list.

   > **Note**
   > - If you change only roles without changing the organization, select the original organization.
   > - Blue role names are the roles that are selected, and black role names are the roles that are not selected.

8. Click **OK**.

1.7.4.1.4. Change user roles

You can add, change, and delete roles for a user.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Users**.
4. Click the **System Users** tab.
5. Click **More** in the **Actions** column corresponding to a user, and choose **Authorize** from the shortcut menu.
6. In the Role field, add, delete, or change user roles as needed.
7. Click **OK**.

1.7.4.1.5. Modify a user logon policy

An administrator can modify a user's logon policy to restrict the permitted logon time and IP addresses of the user.

Prerequisites

A new logon policy is created. For more information about how to create a logon policy, see Create a logon policy.

Modify a user logon policy

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.

3. In the left-side navigation pane of the **Enterprise** page, click **Users**.
4. Click the **System Users** tab.
5. Click **More** in the **Actions** column corresponding to a user, and choose **Logon Policy** from the shortcut menu.
6. In the **Assign Logon Policy** dialog box, select a logon policy and click **OK**.

Modify multiple user logon policies at a time

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Users**.
4. Click the **System Users** tab.
5. Select multiple users.
6. In the upper-right corner of the page, click **Logon Policy**.
7. In the **Assign Logon Policies** dialog box, select a logon policy and click **OK**.

1.7.4.1.6. Modify the information of a user group

On the Users page, you can view the user group information and modify the ownership of users in user groups.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane, click **Users**.
4. Click the **System Users** tab, select a target user, and then click **More** in the **Actions** column.
   - Select **Add to User Group**. In the dialog box that appears, select the target user group and click **OK** to add the user to the user group.
   - Select **Remove from User Group**. In the dialog box that appears, select the target user group and click **OK** to remove the user from the user group.

1.7.4.1.7. View the initial password of a user

After a user is created, the system generates an initial password for the user.

Context

Organization administrators can view the initial passwords of all users in the organizations they manage.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. Use one of the following methods to view the initial password of a user on the **Enterprise** page:
   - In the left-side navigation pane, click **Users**. On the **System Users** tab of the **Users** page, select

a username.

- Click **View Initial Password** in the upper-right corner of the **Users** page to view the initial password.
- Choose **More** > **User Information** in the **Actions** column corresponding to the user. On the user information page, click **View Password** to view the initial password.

- In the left-side navigation pane, click **Organizations**. In the organization navigation tree on the **Organizations** page, click an organization name. In the **Users** section, click a username. On the user information page, click **View Password** to view the initial password.

1.7.4.1.8. Reset the password of a user

If users forget their logon passwords, the system administrator can reset the logon passwords for them.

Prerequisites

Only organization administrators can reset the password of a user.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. Use one of the following methods to go to the **User Information** page:
   - In the left-side navigation pane of the **Enterprise** page, click **Users**. On the **System Users** tab of the **Users** page, click a username.
   - In the left-side navigation pane of the **Enterprise** page, click **Organizations**. On the **Organizations** page, click a username in the **Users** section.
4. Click **Reset Password**.After the password is reset, a message is displayed, which indicates that the password has been reset. If you want to view the initial password after password reset, click **View Password**.

1.7.4.1.9. Disable or enable a user account

You can disable a user account to prevent the user account from logging on to the Apsara Uni-manager console. User accounts that are disabled must be re-enabled before they can be used to log on to the Apsara Uni-manager console again.

Context

By default, user accounts are enabled when they are created.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Users**.
4. Click the **System Users** tab.
5. You can perform the following operations on the current tab:
   - Select a user account whose **Status** is **Enabled**, choose **More** > **Disable** in the **Actions** column

- Select a user account whose **Status** is **Enabled**, choose **More** > **Disable** in the **Actions** column to disable the user account.
- Select a user whose **Status** is **Disabled**, choose **More** > **Enable** in the **Actions** column to enable the user account.

1.7.4.1.10. Delete a user

Administrators can delete users.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Users**.
4. Click the **System Users** tab.
5. Click **More** in the **Actions** column corresponding to a user, and choose **Delete** from the shortcut menu.
6. In the message that appears, click **OK**.

1.7.4.2. Historical users

1.7.4.2.1. Query historical users

You can check whether a user has been deleted and restore a user that has been deleted.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Users**.
4. Click the **Historical Users** tab.
5. Enter the username that you want to query in the **Username** search box.

   > **Note** You can search for usernames by fuzzy match.

6. Click **Search**.

1.7.4.2.2. Restore historical users

An administrator can restore a deleted user account from the **Historical Users** tab.

Context

The basic information such as logon password of a restored user will be the same as it was before the user was deleted, except for the organization and role.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.

2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Users**.
4. Click the **Historical Users** tab.
5. Find the user to be restored and click **Restore** in the **Actions** column.
6. In the **Restore User** dialog box that appears, select an organization and a role.
7. Click **OK**.

1.7.5. Logon policies

1.7.5.1. Create a logon policy

To improve the security of the Apsara Uni-manager console, you can create a logon policy as an administrator to control the logon time and IP addresses of a user.

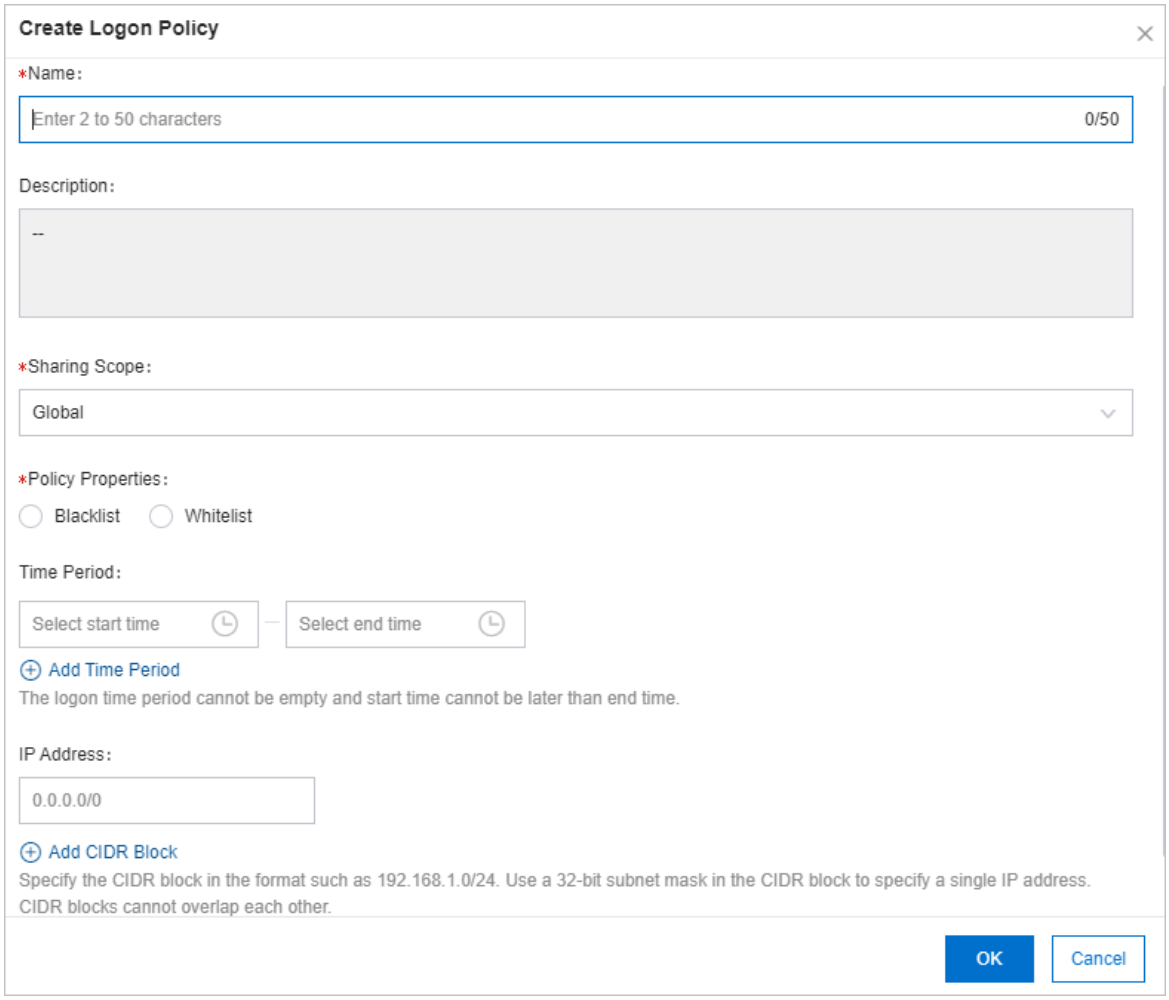
Context

Logon policies are used to control the time period and IP addresses for users to log on. After a user is bound to a logon policy, user logons are restricted based on the logon time and IP addresses specified in the policy.

A default policy without restrictions on the logon time and IP addresses is automatically generated in the Apsara Uni-manager console. The default policy cannot be deleted.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Logon Policies**.
4. In the upper-right corner of the page, click **Create**.
5. In the **Create Logon Policy** dialog box, set Name, Sharing Scope, Policy Properties, Time Period, and IP Address.



Parameters for creating a logon policy

Parameter	Description
Name	The name of the logon policy. The name must be 2 to 50 characters in length and can contain only letters and digits. The name must be unique in the system.
Description	The description of the logon policy.
Sharing Scope	The scope in which the role is visible. ○ **Global**: The role is globally visible. The default value is Global. ○ **Current Organization**: The role is visible only in the current organization and is invisible in subordinate organizations. ○ **Subordinate Organization**: The role is visible in the current organization and all its subordinate organizations.
Policy Properties	The authentication method of the logon policy. ○ **Whitelist**: Logon is allowed if the parameter settings are met. ○ **Blacklist**: Logon is denied if the parameter settings are met.

Parameter	Description
Time Period	The permitted logon time period. When this policy is configured, users can log on to the Apsara Uni-manager console only during the configured period. Specify the time in minutes in a 24-hour clock. Example: `16:32`. **Note** When the Policy Properties parameter is set to Whitelist, you can select No Time Limit.
IP Address	The permitted CIDR block. - If the **Policy Properties** parameter is set to **Whitelist**, IP addresses within this CIDR block are allowed to log on to the Apsara Uni-manager console. - If the **Policy Properties** parameter is set to **Blacklist**, IP addresses within this CIDR block are not allowed to log on to the Apsara Uni-manager console. **Note** When the Policy Properties parameter is set to Whitelist, you can select No CIDR Block Limit.

1.7.5.2. Query a logon policy

When the Apsara Uni-manager console provides services, it automatically generates a default policy without restrictions on the logon time and IP addresses.

Context

When the Apsara Uni-manager console provides services, it automatically generates a default policy without restrictions on the logon time and IP addresses.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Logon Policies**.
4. Enter the name of the policy that you want to view and click **Search**.
5. View the logon policy, including the permitted logon time and IP addresses.

1.7.5.3. Modify a logon policy

You can modify the policy name, policy properties, permitted logon time period, and IP addresses of a logon policy.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Logon Policies**.

4. Click **More** in the **Actions** column corresponding to a policy, and choose **Modify** from the shortcut menu.
5. In the **Modify Logon Policy** dialog box that appears, modify the logon policy information.
6. Click **OK**.

1.7.5.4. Disable a logon policy

You can disable logon policies that are no longer needed.

Procedure

1. Log on to the Apsara Uni-manager console.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Logon Policies**.
4. Click **More** in the **Actions** column corresponding to a policy, and choose **Disable** from the shortcut menu.

1.7.5.5. Enable a logon policy

You can re-enable disabled logon policies.

Procedure

1. Log on to the Apsara Uni-manager console.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Logon Policies**.
4. Click **More** in the **Actions** column corresponding to a policy, and choose **Enable** from the shortcut menu.

1.7.5.6. Delete a logon policy

You can delete logon policies that are no longer needed.

Prerequisites

The logon policy to be deleted is not bound to any users. If a logon policy is bound to a user, the logon policy cannot be deleted.

Context

> **Note** The default policy cannot be deleted.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Logon Policies**.
4. Click **More** in the **Actions** column corresponding to a policy, and choose **Delete** from the shortcut

menu.

5. In the message that appears, click **OK**.

1.7.6. User groups

1.7.6.1. Create a user group

You can create a user group in a selected organization and grant batch authorizations to users in the group.

Prerequisites

Before creating a user group, you must create an organization. For more information, see Create an organization.

Context

Relationship between user groups and users:

- A user group can contain zero or more users.
- You can add users to user groups as needed.
- You can add a user to multiple user groups.

Relationship between user groups and organizations:

- A user group can only belong to a single organization.
- You can create multiple user groups in an organization.

Relationship between user groups and roles:

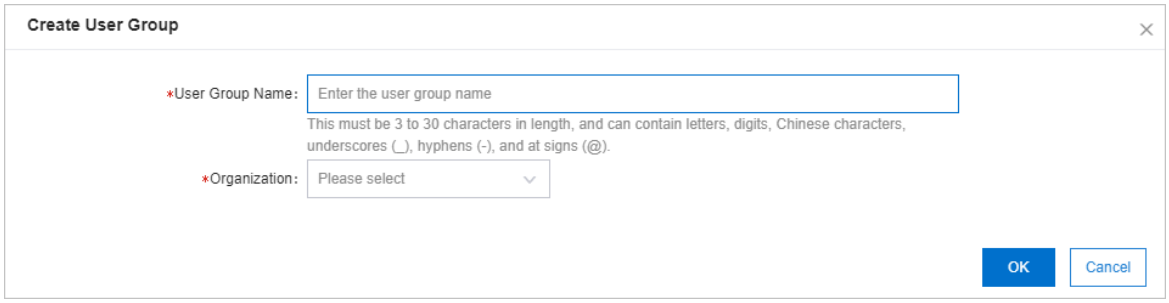
- A user group can only be bound to a single role.
- A role can be associated with multiple user groups.
- When a role is associated with a user group, the role permissions are automatically granted to users in the user group.

Relationship between user groups and resource sets:

- You can add zero or more user groups to a resource set.
- A user group can be added to multiple resource sets.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **User Groups**.
4. In the upper-right corner of the page, click **Create User Group**.
5. In the dialog box that appears, set **User Group Name** and **Organization**.



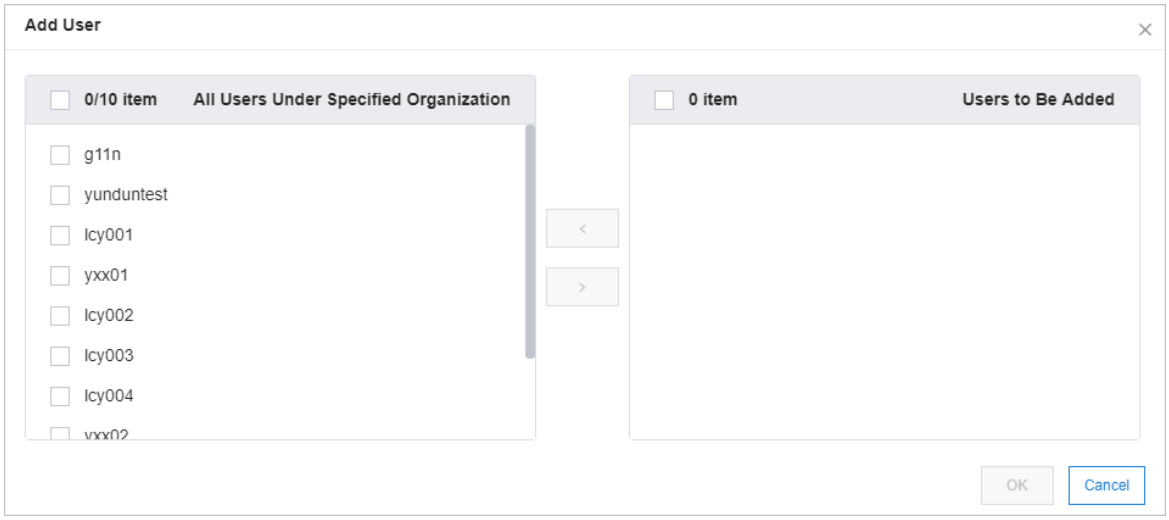
6. Click **OK**.

1.7.6.2. Add users to a user group

You can add users to a user group.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **User Groups**.
4. Click **Add User** in the **Actions** column corresponding to a user group.
5. Select the names of users to be added from the left list, and click the right arrow to move them to the right list.



6. Click **OK**.

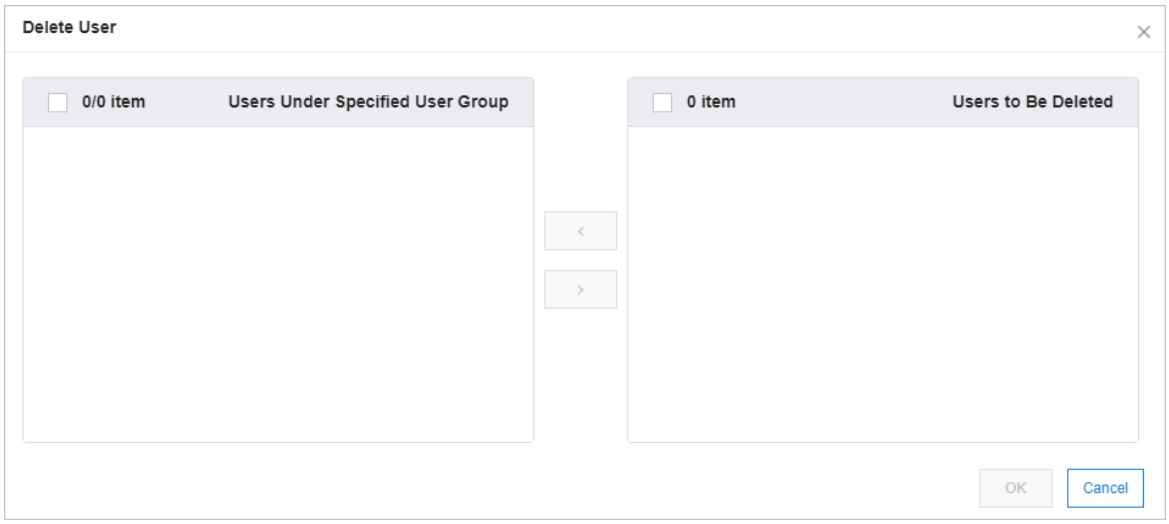
1.7.6.3. Delete users from a user group

You can delete users from a user group.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **User Groups**.

4. Click **Delete User** in the **Actions** column corresponding to a user group.
5. Select the names of users to be deleted from the **Users Under Specified User Group** list, and click the right arrow to move them to the **Users to Be Deleted** list.



6. Click **OK**.

1.7.6.4. Add a role

You can add a role to a user group and assign the role to all users in the group.

Context

> **Note** You can add only one role to a user group.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **User Groups**.
4. Click **Add Role** in the **Actions** column corresponding to a user group.
5. In the dialog box that appears, select a role.
6. Click **OK**.

1.7.6.5. Delete a role

You can delete existing roles.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **User Groups**.
4. Click **Delete Role** in the **Actions** column corresponding to a user group.

5. In the **Confirm** message that appears, click **OK**.

1.7.6.6. Modify the name of a user group

You can modify the names of user groups.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **User Groups**.
4. Click **Edit User Group** in the **Actions** column corresponding to a user group.
5. In the dialog box that appears, enter the new name.
6. Click **OK**.

1.7.6.7. Delete a user group

You can delete user groups that are no longer needed.

Prerequisites

The user group to be deleted is unbound from any roles. If a user group is bound to a role, the user group cannot be deleted.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **User Groups**.
4. Click **Delete User Group** in the **Actions** column corresponding to a user group.
5. In the **Confirm** message that appears, click **OK**.

1.7.7. Resource pools

1.7.7.1. Update associations

You can deploy the Apsara Uni-manager console in multiple regions. You can update the associations between organizations and regions.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Resource Pool Management**.
4. In the left-side organization navigation tree, click the name of an organization.
5. In the corresponding region list, select the names of regions to be associated.
6. Click **Update Association**.

1.7.8. Change ownership

You can change the ownership of instances in resource sets.

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Change Ownership**.
4. Click the ▸ icon to the left of an organization and click a resource set.
5. In the resource list on the right side of the page, set a service type and a resource type, enter an instance ID, and then click **Search** to query the instance.
6. You can change the ownership of a single instance or add multiple instances to the same resource set at a time.
   - Single change: Click **Change Ownership** in the **Actions** column corresponding to an instance.
   - Batch change: Select multiple instance IDs and click **Batch Change Ownership**.
7. In the **Change Resource Set** dialog box, select a resource set and click **OK**.

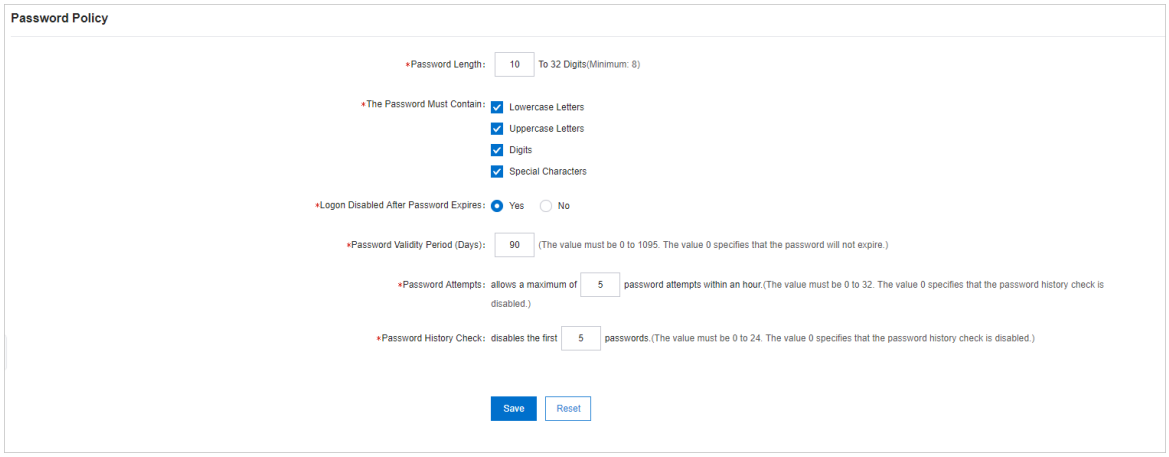
1.8. Configurations

1.8.1. Password policies

You can configure password policies for user logons.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Password Policies**.
4. On the **Password Policy** page, set the password policy parameters.



To restore to the default password policy, click **Reset**.

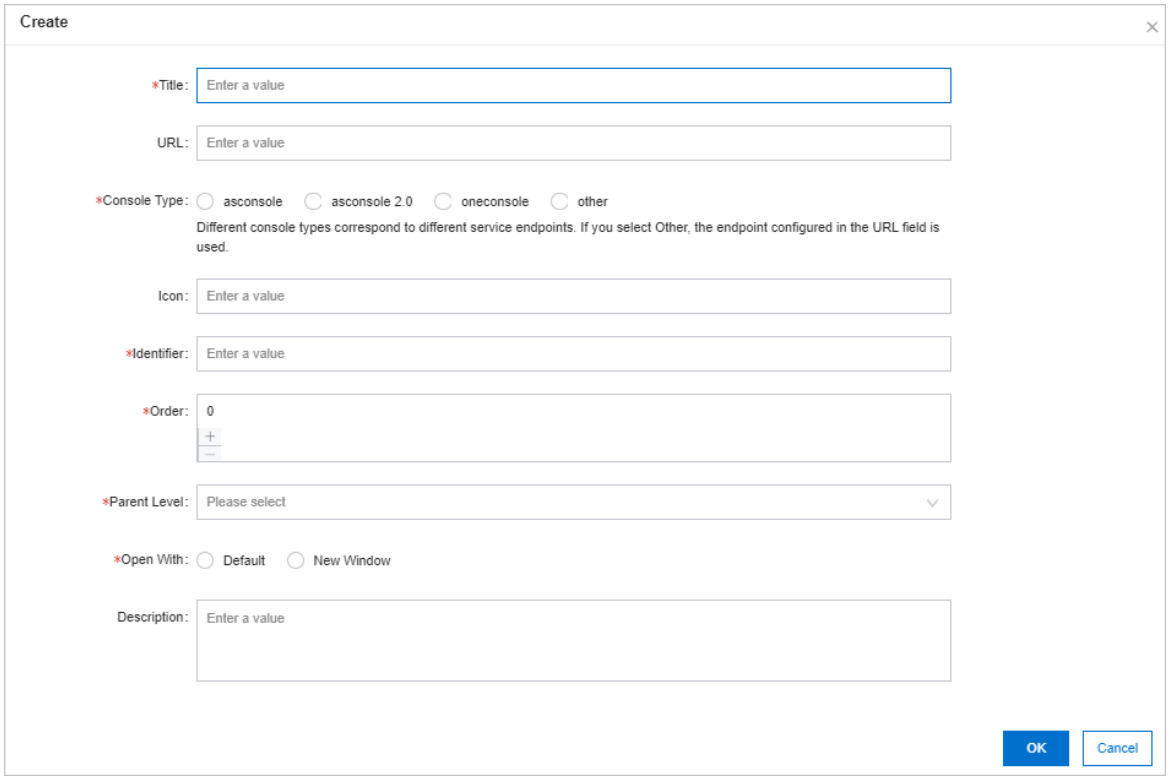
1.8.2. Menus

1.8.2.1. Create a menu

You can create a menu and add its URL to the Apsara Uni-manager console for quick access.

Procedure

1. Log on to the Apsara Uni-manager console as a platform administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Menu Settings**.
4. On the **Main Menu** page, click **Create** in the upper-right corner.
5. In the **Create** dialog box, set the menu parameters.



Menu parameters

Parameter	Description
Title	The display name of the menu.
URL	The URL of the menu.
Console Type	Different console types correspond to different domain names. ○ **oneconsole**: You need only to enter the path in the URL field. The domain name is automatically matched. ○ **asconsole**: You need only to enter the path in the URL field. The domain name is automatically matched. ○ **other**: You must enter the domain name in the URL field.

Parameter	Description
Icon	The icon displayed in the left-side navigation pane. The icon cannot be changed.
Identifier	The unique identifier of the menu in the system. This identifier can be used to indicate whether the menu is selected in the navigation bar. The identifier cannot be changed.
Order	The display order among the same-level menus. The larger the value, the lower the display order. Leave the Order field empty.
Parent Level	The displayed tree structure.
Open With	Specifies whether to open the menu in the current window or in a new window.
Description	The description of the menu.

1.8.2.2. Modify a menu

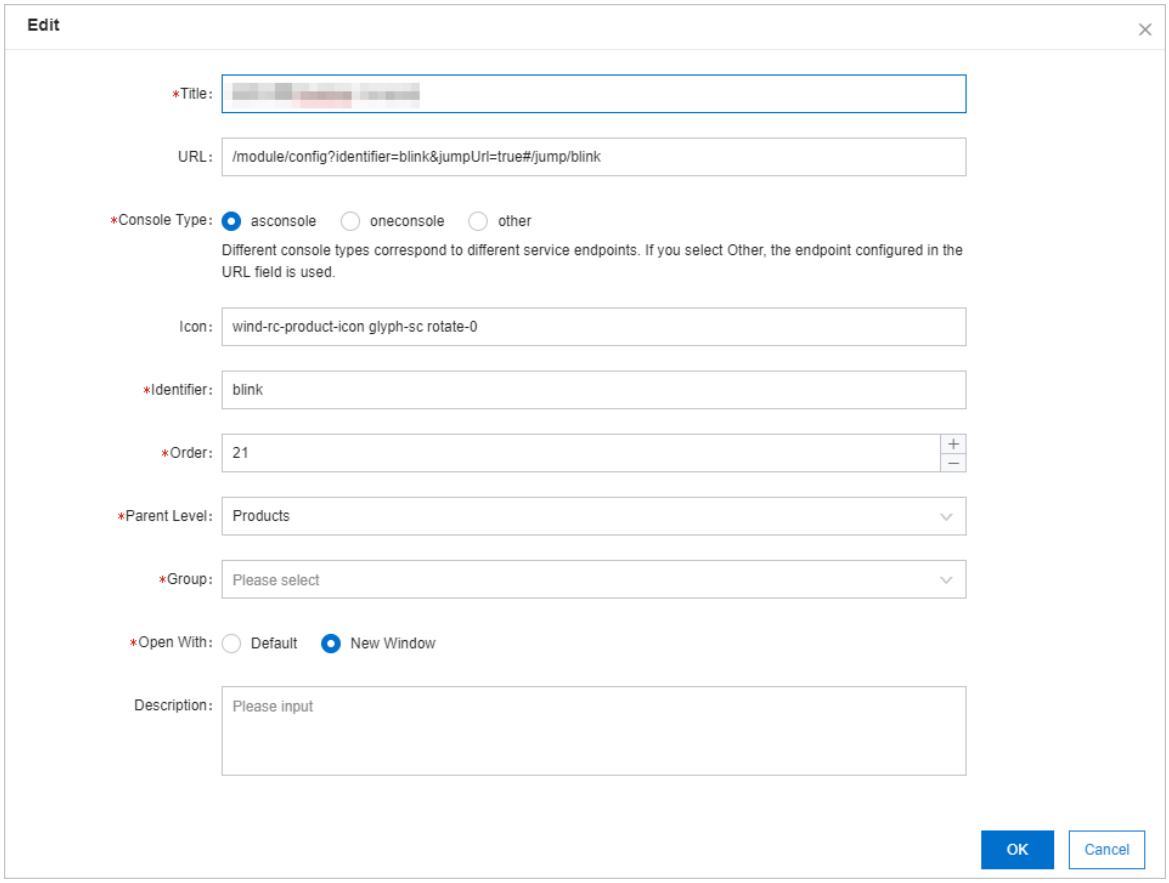
You can modify an existing menu, including the menu name, URL, icon, and menu order.

Prerequisites

Default menus cannot be modified.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Menu Settings**.
4. Click **Edit** in the **Actions** column corresponding to a menu.
5. In the **Edit** dialog box that appears, modify relevant information about the menu.



1.8.2.3. Delete a menu

You can delete menus that are no longer needed.

Prerequisites

Default menus cannot be deleted.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Menu Settings**.
4. Click **Delete** in the **Actions** column corresponding to a menu.
5. In the message that appears, click **OK**.

1.8.2.4. Display or hide menus

You can display or hide menus as follows:

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Menu Settings**.

4. Select or clear the check box in the **Displayed** column corresponding to a menu.

1.8.3. Specifications

1.8.3.1. Specification parameters

This topic describes the specification parameters of each resource type.

OSS

Parameter	Description
Specifications	The specifications that can be configured for Object Storage Service (OSS).
Specifications Description	The description of the specifications that can be configured for OSS.

NAT Gateway

Parameter	Description
Specifications	The specifications that can be configured for NAT Gateway.
Specifications Description	The description of the specifications that can be configured for NAT Gateway.

AnalyticDB for PostgreSQL

Parameter	Description
Specifications	The specifications that can be configured for AnalyticDB for PostgreSQL.
Specifications Name	The name of the specifications that can be configured for AnalyticDB for PostgreSQL.
CPU	The total number of CPU cores that can be configured for AnalyticDB for PostgreSQL.
Memory	The memory size that can be configured for AnalyticDB for PostgreSQL.
Storage Space	The total storage size that can be configured for AnalyticDB for PostgreSQL.
Version	The version of AnalyticDB for PostgreSQL.
Node	The number of nodes that can be configured for AnalyticDB for PostgreSQL.

AnalyticDB for MySQL

Parameter	Description
Specifications	The specifications that can be configured for AnalyticDB for MySQL.
Minimum Nodes and Maximum Nodes	The minimum and maximum number of nodes that can be configured for AnalyticDB for MySQL.
Storage Space	The storage space that can be configured for AnalyticDB for MySQL.

SLB

Parameter	Description
Specifications	The specifications that can be configured for Server Load Balancer (SLB).
Specifications Name	The name of the specifications that can be configured for SLB.
Maximum Connections	The maximum number of connections that can be configured for SLB.
New Connections	The number of new connections that can be configured for SLB.
QPS	The queries per second (QPS) that can be configured for SLB.
Description	The description of the specifications that can be configured for SLB.

ApsaraDB for RDS

Parameter	Description
Engine Type	The engine type that can be configured for ApsaraDB for RDS.
Minimum Storage (GB)	The minimum storage space that can be configured for ApsaraDB for RDS.
Maximum Storage (GB)	The maximum storage space that can be configured for ApsaraDB for RDS.
Specifications Name	The name of the specifications that can be configured for ApsaraDB for RDS.
Version	The version of ApsaraDB for RDS.

Parameter	Description
CPUs	The number of CPU cores that can be configured for ApsaraDB for RDS.
Maximum Connections	The maximum number of connections that can be configured for ApsaraDB for RDS.
Storage	The storage space that can be configured for ApsaraDB for RDS.
Memory (GB)	The memory size that can be configured for ApsaraDB for RDS.
Share Type	The share type that can be configured for ApsaraDB for RDS.

DRDS

Parameter	Description
Instance Type	The instance type that can be configured for Distributed Relational Database Service (DRDS).
Instance Type Name	The name of the instance type that can be configured for DRDS.
Specifications	The specifications that can be configured for DRDS.
Specifications Name	The name of the specifications that can be configured for DRDS.

ECS

Parameter	Description
Instance Family	The instance family that is divided into different instance types based on the scenarios for which they are suitable.
Specifications Level	The level of the specifications that can be configured for Elastic Compute Service (ECS).
vCPUs	The maximum number of vCPUs that can be configured for ECS.
Memory (GB)	The memory size that can be configured for ECS.
Instance Specifications	The instance type that can be configured for ECS.
GPU Type	The GPU type that can be configured for ECS.
GPUs	The number of GPUs that can be configured for ECS.

Parameter	Description
Supported ENIs	The number of Elastic Network Interfaces (ENIs) that can be configured for ECS.
Number Of Private IP Addresses	The number of private IP addresses that can be configured for ECS.

IPv6 Translation Service

Parameter	Description
Specifications	The specifications that can be configured for IPv6 Translation Service.
Specifications Name	The name of the specifications that can be configured for IPv6 Translation Service.

KVStore for Redis

Parameter	Description
Specifications Name	The name of the specifications that can be configured for KVStore for Redis.
Instance Specifications	The instance type that can be configured for KVStore for Redis.
Maximum Connections	The maximum number of connections that can be configured for KVStore for Redis.
Maximum Bandwidth	The maximum bandwidth that can be configured for KVStore for Redis.
CPUs	The number of CPU cores that can be configured for KVStore for Redis.
Version	The version of KVStore for Redis.
Architecture	The architecture of KVStore for Redis.
Node Type	The node type of KVStore for Redis.
Service Plan	The service plan that can be configured for KVStore for Redis.

ApsaraDB for MongoDB

Parameter	Description
Specifications	The specifications that can be configured for ApsaraDB for MongoDB.

Parameter	Description
Specifications Name	The name of the specifications that can be configured for ApsaraDB for MongoDB.
Engine Type	The engine type that can be configured for ApsaraDB for MongoDB.
Version	The version of ApsaraDB for MongoDB.
Serial Number	The serial number of ApsaraDB for MongoDB.
Sequence Description	The description of the serial number of ApsaraDB for MongoDB.
Maximum Connections	The maximum number of connections that can be configured for ApsaraDB for MongoDB.
IOPS	The input/output operations per second (IOPS) of ApsaraDB for MongoDB.
Storage Space	The storage space that can be configured for ApsaraDB for MongoDB.
Minimum Storage	The minimum storage space that can be configured for ApsaraDB for MongoDB.
Maximum Storage	The maximum storage space that can be configured for ApsaraDB for MongoDB.

TSDB

Parameter	Description
Specifications Name	The name of the specifications that can be configured for Time Series Database (TSDB).
Maximum Time Limit	The maximum duration of connections in TSDB.
TPS	The number of transactions that can be processed per second by TSDB.
Storage Space	The storage space that can be configured for TSDB.

1.8.3.2. Create specifications

You can customize specifications for each resource type.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Configurations**.

3. In the left-side navigation pane of the **Configurations** page, click **Specifications**.
4. Click the resource type for which you want to create specifications.
5. In the upper-right corner of the page, click **Create Specifications**.
6. In the dialog box that appears, set the parameters.For more information about specification parameters, see Specification parameters.
7. Click **OK**.

1.8.3.3. View specifications

You can view the specifications of each resource type.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Specifications**.
4. Click the resource type for which you want to view specifications.
5. On the **Resource Specifications** tab, set a **region**, **column**, and **value**. The corresponding information is displayed in the specifications list.
6. Click the **Existing Specifications** tab and view the existing specifications and their quantity.

1.8.3.4. Disable specifications

By default, the status of newly created specifications is Enabled.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Specifications**.
4. Select the resource type for which you want to disable specifications.
5. Click **Disable** in the **Actions** column corresponding to the target specifications.
6. In the message that appears, click **OK**.

1.8.3.5. View specifications of each resource type in previous versions

You can view specifications of each resource type in previous versions.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Specifications**.
4. On the Specifications page, click the resource type for which you want to view specifications.

5. Click the **Specifications History** tab. View the detailed information in the specifications list.

1.8.3.6. Export specifications

You can export specifications that you want to view and share.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Specifications**.
4. Click the resource type for which you want to create specifications.
5. In the upper-right corner of the page, click **Export**.
6. Save the specifications file to the target path.

1.9. Security

1.9.1. View operation logs

You can view operation logs to obtain up-to-date information about various resources and functional modules in the Apsara Uni-manager console. You can also export operation logs to your personal computer.

Procedure

1. Log on to the Apsara Uni-manager console as a security administrator.
2. In the top navigation bar, click **Security**.
3. You can filter logs by username, object, level, source IP address, details, start time, and end time.

   The following table describes the fields in the query result.

Fields in the query result

Log field	Description
Username	The name of the operator.
Object	The Apsara Stack service on which operations are performed. The operations include creating, modifying, deleting, querying, updating, binding, unbinding, enabling and disabling service instances, applying for and releasing service instances, and changing the ownership of service instances.
Level	The operation level. Valid values: INFO, DEBUG, and ERROR.
Source IP	The IP address of the operator.
Details	The brief introduction of the operation.
Start Time	The time when the operation started.

Log field	Description
End Time	The time when the operation ended.

4. (Optional) Click **Export** to export the logs displayed on the current page to your personal computer in the .xls format.

   The exported log file is named *log.xls* and stored in the *C:\Users\Username\Downloads* directory.

1.10. RAM

1.10.1. RAM introduction

Resource Access Management (RAM) is a resource access control service provided by Apsara Stack.

You can use RAM to manage users and control which resources are accessible to employees, systems, and applications.

RAM provides the following features:

- RAM role

  To authorize a cloud service in a level-1 organization to use other resources in the organization, you must create a RAM role. This role specifies the operations that the cloud service can perform on resources.

  Only system administrators and level-1 organization administrators can create RAM roles.

- User group

  You can create multiple users within an organization and grant them different operation permissions on cloud resources.

  You can create RAM user groups to classify and authorize RAM users within your Apsara Stack tenant account. This simplifies the management of RAM users and their permissions.

  You can create RAM permission policies to grant different operation permissions to different user groups.

1.10.2. Permission policy structure and syntax

This topic describes the structure and syntax used to create or update permission policies in Resource Access Management (RAM).

Policy characters and usage rules

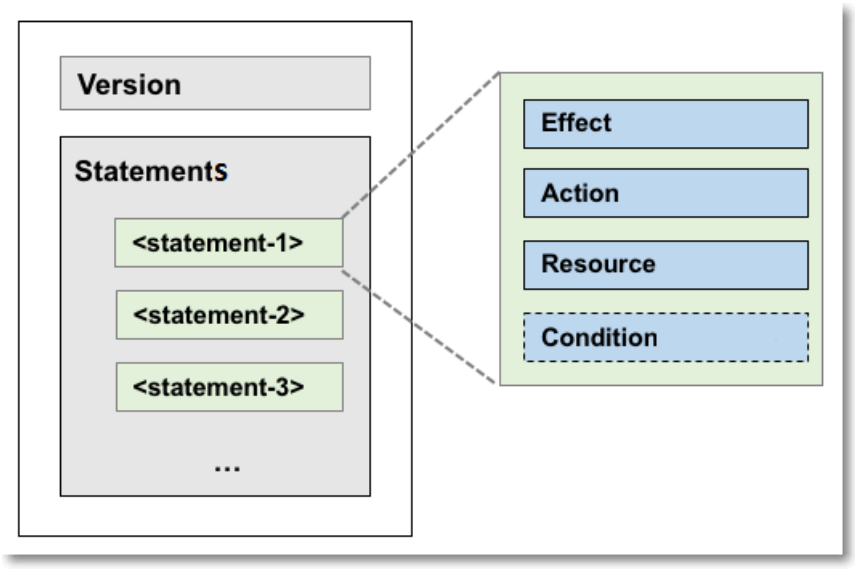
- Characters in a policy
  - The following characters are JSON tokens and are included in policies: `{}[]",:` .
  - The following characters are special characters in the syntax and are not included in policies: `=<>()|` .
- Use of characters

- If an element can have more than one value, you can perform the following operations:
  - Separate multiple values by using commas (,) as delimiters between each value and use an ellipsis (...) to describe the remaining values. Example: `[<action_string>, <action_string>, ...]`.
  - Include only one value. Examples: `"Action": [<action_string>]` and `"Action": <action_string>`.
- A question mark (?) following an element indicates that the element is optional. Example: `<condition_block?>`.
- A vertical bar (|) between elements indicates multiple options. Example: `("Allow" | "Deny")`.
- Elements that must be text strings are enclosed in double quotation marks (""). Example: `<version_block> = "Version" : ("1")`.

Policy structure

The policy structure includes the following components:

- The version number.
- A list of statements. Each statement contains the following elements: Effect, Action, Resource, and Condition. The Condition element is optional.



Policy syntax

```
policy ={
   <version_block>,
   <statement_block>
}
<version_block>= "Version" : ("1")
<statement_block>= "Statement" : [ <statement>, <statement>, ... ]
<statement> ={
  <effect_block>,
  <action_block>,
  <resource_block>,
  <condition_block? >
}
<effect_block>= "Effect" : ("Allow" | "Deny")
<action_block>= ("Action" | "NotAction") :
  ("*" | [<action_string>, <action_string>, ...])
<resource_block>= ("Resource" | "NotResource") :
  ("*" | [<resource_string>, <resource_string>, ...])
<condition_block>= "Condition" : <condition_map>
<condition_map>={
 <condition_type_string> : {
   <condition_key_string> : <condition_value_list>,
   <condition_key_string> : <condition_value_list>,
   ...
 },
 <condition_type_string> : {
   <condition_key_string> : <condition_value_list>,
   <condition_key_string> : <condition_value_list>,
   ...
 }, ...
}
<condition_value_list>= [<condition_value>, <condition_value>, ...]
<condition_value>= ("String" | "Number" | "Boolean")
```

Description:

- The current policy version is 1.
- The policy can have multiple statements.
  - The effect of each statement can be either `Allow` or `Deny`.

> **Note** In a statement, both the Action and Resource elements can have multiple values.

- Each statement can have its own conditions.

  > **Note** A condition block can contain multiple conditions with different operators and logical combinations of these conditions.

- You can attach multiple policies to a RAM user. If policies that apply to a request include an `Allow` statement and a `Deny` statement, the Deny statement overrides the Allow statement.
- Element value:
  - If an element value is a number or Boolean value, it must be enclosed in double quotation marks ("") in the same way as strings.
  - If an element value is a string, characters such as the asterisk (`*`) and question mark (`?`) can be used for fuzzy matching.
    - The asterisk (`*`) indicates any number (including zero) of allowed characters. For example, `ecs:Describe*` indicates all ECS API operations that start with `Describe` .
    - `The question mark (?)` indicates an allowed character.

Policy format check

Policies are stored in RAM as JSON documents. When you create or update a policy, RAM first checks whether the JSON format is valid.

- For more information about JSON syntax standards, see RFC 7159.
- We recommend that you use tools such as JSON validators and editors to check whether the policies meet JSON syntax standards.

1.10.3. RAM roles

1.10.3.1. View basic information about a RAM role

You can view basic information about a RAM role, including its user groups and existing permission policies.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. On the Roles page, click the name of the target RAM role.
5. In the basic information section, click the **User Groups** and **Permissions** tabs to view relevant information.

1.10.3.2. Create a RAM role

To authorize a cloud service in a level-1 organization to use other resources in the organization, you must create a RAM role. This role contains the operations that the cloud service can perform on resources.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the upper-right corner of the page, click **Create RAM Role**.
5. On the **Roles - Create RAM Role** page, set **Role Name**, **Description** and **Sharing Scope**.Valid values of the **Sharing Scope** parameter:
   - **Global**

     The role is visible and valid to all organizations involved. The default value is Global.
   - **Current Organization**

     The role is visible and valid to the organization to which the user belongs.
   - **Subordinate Organization**

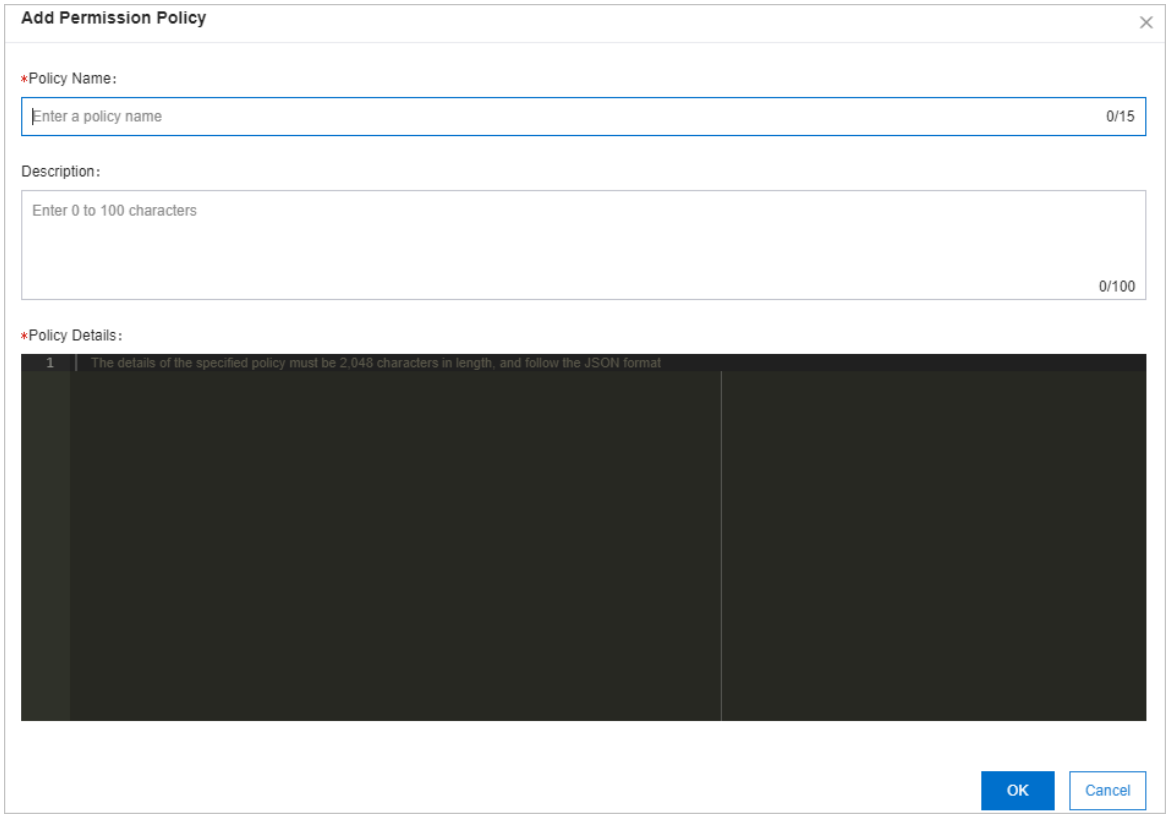
     The role is visible and valid to the organization to which the user belongs and its subordinate organizations.
6. Click **Create**.

1.10.3.3. Add a permission policy

To use a cloud service to access other cloud resources, you must create a permission policy and attach it to a user group.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the role name list, click **More** in the **Actions** column corresponding to a RAM role, and choose **Modify** from the shortcut menu to go to the **Roles** page.
5. Click the **Permissions** tab.
6. Click **Add Permission Policy**.
7. In the dialog box that appears, enter information about the permission policy.



For more information about how to enter the policy content, see Permission policy structure and syntax.

1.10.3.4. Modify the content of a RAM permission policy

You can modify the content of a RAM permission policy as needed.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the role list, click **More** in the **Actions** column corresponding to a RAM role, and choose **Modify** from the shortcut menu to go to the **Roles** page.
5. Click the **Permissions** tab.
6. Click the name of a permission policy in the **Permission Policy Name** column.
7. In the **Modify Permission Policy** dialog box that appears, modify the relevant information and click **OK**.For more information about how to modify the policy content, see Permission policy structure and syntax.

1.10.3.5. Modify the name of a RAM permission policy

You can modify the name of a RAM permission policy as needed.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the role name list, click **More** in the **Actions** column corresponding to a RAM role, and choose **Modify** from the shortcut menu to go to the **Roles** page.
5. Click the Permissions tab. Click the name of a permission policy in the **Permission Policy Name** column.
6. In the **Modify Permission Policy** dialog box that appears, modify the permission policy name.

1.10.3.6. Add a RAM role to a user group

You can bind RAM roles to user groups as needed.

Prerequisites

You must create a user group before RAM roles can be added. If no user groups have been created, see Add a role.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the role name list, click **More** in the **Actions** column corresponding to a RAM role, and choose **Modify** from the shortcut menu to go to the **Roles** page.
5. Click the **User Groups** tab.
6. Click **Add User Group**. In the dialog box that appears, select a user group.
7. Click **OK**.

1.10.3.7. Grant permissions to a RAM role

When you grant permissions to a RAM role, all users in the user groups that are assigned this role will share the granted permissions.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the role name list, click **More** in the **Actions** column corresponding to a RAM role, and choose **Modify** from the shortcut menu to go to the **Roles** page.
5. Click the **Permissions** tab.
6. Click **Select Existing Permission Policy**.
7. In the dialog box that appears, select a RAM permission policy and click **OK**.If no RAM permission policies are available, see Add a permission policy.

1.10.3.8. Remove permissions from a RAM role

You can remove permissions that are no longer needed from RAM roles.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the role name list, click **More** in the **Actions** column corresponding to a RAM role, and choose **Modify** from the shortcut menu to go to the **Roles** page.
5. Click the **Permissions** tab.
6. Click **Remove** in the **Actions** column corresponding to the permission policy that you want to remove.

1.10.3.9. Modify a RAM role name

Administrators can modify the names of RAM roles.

Context

> **Note** The name of a preset role cannot be modified.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the role name list, click **More** in the **Actions** column corresponding to a RAM role, and choose **Modify** from the shortcut menu to go to the **Roles** page.
5. Move the pointer over the role name and click to enter a new role name.

1.10.3.10. Delete a RAM role

This topic describes how to delete a RAM user.

Prerequisites

Before you delete a RAM role, make sure that no policies are attached to the RAM role.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the role name list, click **More** in the **Actions** column corresponding to a RAM role, and choose

Delete from the shortcut menu.

5. In the message that appears, click **OK**.

1.10.4. RAM authorization policies

1.10.4.1. Create a RAM role

You can create authorization policies and grant them to organizations as needed.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **RAM Roles**.
4. In the upper-right corner of the page, click **Create RAM User**.
5. On the **Create RAM User** page, set **Organization** and **Service**.
6. Click **OK**.

1.10.4.2. View the details of a RAM role

You can view the details of a RAM role, including its role name, creation time, description, and Alibaba Cloud Resource Name (ARN).

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **RAM Roles**.
4. On the **RAM Users** page, set **Role Name** or **Service Name**, and click **Search** in the upper-right corner.To perform another search, click **Clear**.
5. Find the RAM role that you want to view and click **Details** in the Actions column.
6. Click the **Role Details** tab to view the details of the RAM role.

1.10.4.3. View RAM authorization policies

You can view the details of a RAM authorization policy, including its policy name, policy type, default version, description, association time, and policy content.

Prerequisites

A RAM authorization policy is created. For more information, see Create a RAM role.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **RAM Roles**.

4. On the **RAM Users** page, set **Role Name** or **Service Name**, and click **Search** in the upper-right corner.To perform another search, click **Clear**.
5. Find the RAM role that you want to view and click **Details** in the **Actions** column.
6. Click the **Role Policy** tab to view information about the role authorization policy. Click **Details** in the Actions column to view the policy details.

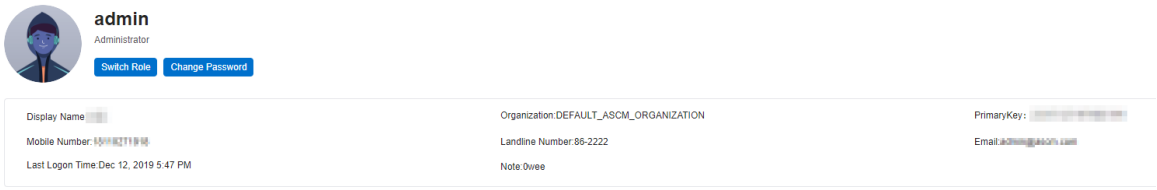
1.11. Personal information management

1.11.1. Modify personal information

You can modify your personal information to keep it up-to-date.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the upper-right corner of the homepage, move the pointer over the user profile picture and choose **User Information** from the shortcut menu.



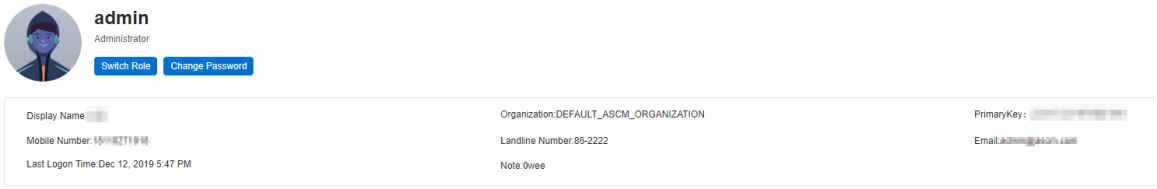
3. Click next to the item you want to modify.
4. In the Modify User Information dialog box that appears, modify the relevant information.
5. Click **OK**.

1.11.2. Change your logon password

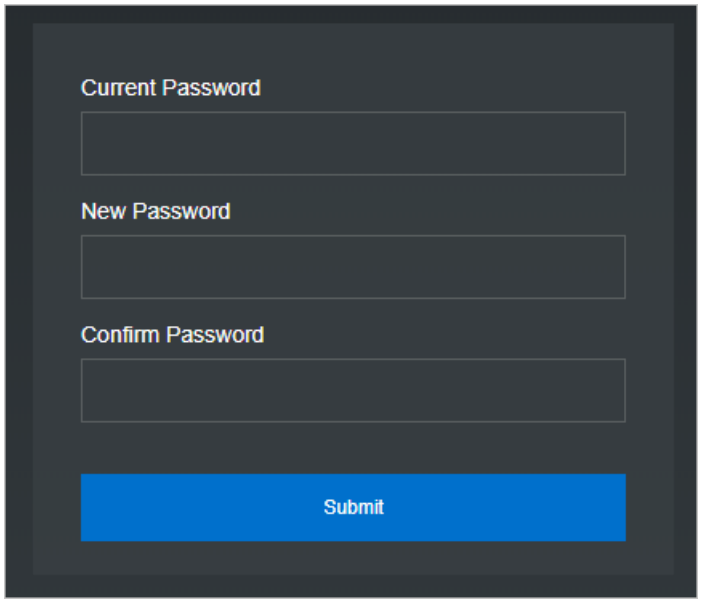
To improve security, you must change your logon password in a timely manner.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the upper-right corner of the homepage, move the pointer over the user profile picture and choose **User Information** from the shortcut menu.



3. Click **Change Password**. On the page that appears, set **Current Password**, **New Password**, and **Confirm Password**.



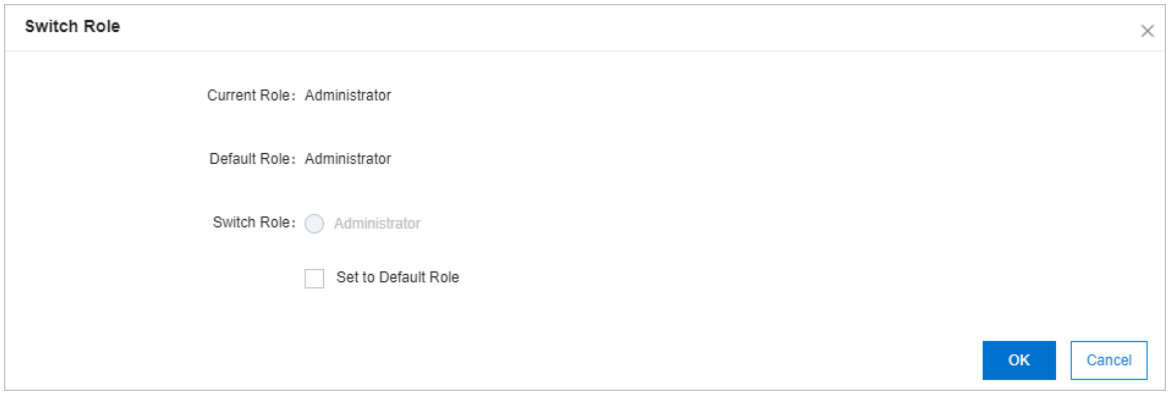
4. Click **Submit**.

1.11.3. Switch the current role

You can switch the scope of your current role.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the upper-right corner of the homepage, move the pointer over the user profile picture and choose **User Information** from the shortcut menu.
3. Click **Switch Role**.
4. In the **Switch Role** dialog box that appears, select the role that you want to switch to.



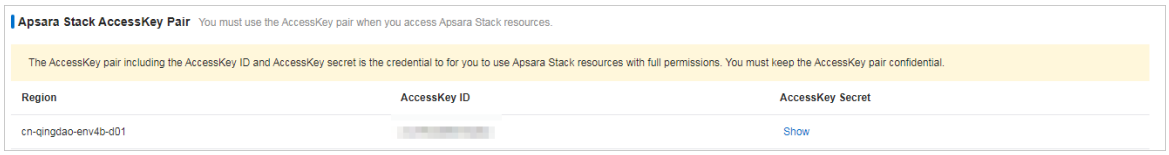
You can also switch back to the default role.

1.11.4. View the AccessKey pair of your Apsara Stack tenant account

To secure cloud resources, the system must verify the identity of visitors and ensure that they have the relevant permissions. You must obtain the AccessKey ID and AccessKey secret of your personal account to access cloud resources.

Procedure

1. Log on to the Apsara Uni-manager console as an administrator.
2. In the upper-right corner of the homepage, move the pointer over the user profile picture and choose **User Information** from the shortcut menu.
3. In the **Apsara Stack AccessKey Pair** section, view your AccessKey pair.



> **Note** The AccessKey pair is made up of the AccessKey ID and AccessKey secret. These credentials provide you full permissions on Apsara Stack resources. You must keep the AccessKey pair confidential.

2.Object Storage Service (OSS)

2.1. What is OSS?

Object Storage Service (OSS) is a secure, cost-effective, and highly reliable cloud storage service provided by Alibaba Cloud. It enables you to store a large amount of data in the cloud.

OSS is an immediately available storage solution that has unlimited storage capacity. Compared with user-created server storage, OSS has outstanding advantages in reliability, security, cost-effectiveness, and data processing capabilities. OSS enables you to store and retrieve a variety of unstructured data objects, such as texts, images, audios, and videos over the network at any time.

OSS is an object storage service based on key-value pairs. Files uploaded to OSS are stored as objects in buckets. You can obtain the content of an object based on the object key.

In OSS, you can:

- Create a bucket and upload objects to the bucket.
- Obtain an object URL from OSS to share or download the object.
- Modify the attributes or metadata of a bucket or an object, and configure ACL for the bucket or the object.
- Perform basic and advanced operations in the OSS console.
- Perform basic and advanced operations by using SDKs or calling RESTful API operations in your application.

2.2. Usage notes

Before you use OSS, you must understand the following content:

To allow other users to use all or part of OSS features, you must create RAM users and grant permissions to the users by configuring RAM policies.

Before you use OSS, you must also understand the following limits.

Item	Limit
Bucket	• You can create up to 100 buckets. • After a bucket is created, its name and region cannot be modified.
Upload objects	• Objects larger than 5 GB cannot be uploaded by using the following modes: console upload, simple upload, form upload, or append upload. To upload an object that is larger than 5 GB, you must use multipart upload. The size of an object uploaded by using multipart upload cannot exceed 48.8 TB. • If you upload an object that has the same name of an existing object in OSS, the new object will overwrite the existing object.

Item	Limit
Delete objects	• Deleted objects cannot be recovered. • You can delete up to 100 objects at a time in the OSS console. To delete more than 100 objects at a time, you must call an API operation or use an SDK.
Lifecycle	You can configure up to 1,000 lifecycle rules for each bucket.

2.3. Quick start

2.3.1. Log on to the OSS console

This topic describes how to log on to the OSS console.

Prerequisites

- The domain name of the Apsara Uni-manager console is obtained from the deployment personnel before you log on to the Apsara Uni-manager console.
- A browser is available. We recommend that you use Google Chrome.

Procedure

1. In the address bar, enter the URL used to access the Apsara Uni-manager console. Press Enter.
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 console for the first time, you must change the password of your username. For security reasons, your password must meet the minimum complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters.
   > - Digits.
   > - Special characters. Special characters include exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to go to the Apsara Uni-manager console homepage.
4. In the top navigation bar, choose **Products > Object Storage Service**.

2.3.2. Create buckets

Objects uploaded to OSS are stored in buckets. Before you upload an object to OSS, you must create a bucket.

Context

The attributes of a bucket include the region, ACL, and other metadata.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click **Create Bucket** if no buckets are available. On the **Create OSS Bucket** page, configure parameters for the bucket.In the left-side navigation pane, click the **+** icon next to **Buckets** if buckets exist in the region. The **Create OSS Bucket** page appears.

   The following table describes the parameters you can configure to create a bucket.

Parameters

Parameter	Description
Organization	Select an organization from the drop-down list for the bucket.
Resource Set	Select a resource set from the drop-down list for the bucket.
Region	Select a region from the drop-down list for the bucket. **Note** ○ After a bucket is created, the region cannot be changed. ○ If you want to access OSS from your ECS instance through the internal network, select the same region where your ECS instance is deployed.
Cluster	Select a cluster for the bucket. You can deploy two OSS clusters in Apsara Stack.
Bucket Name	Enter the name of the bucket. **Note** ○ The bucket name must comply with the naming conventions. ○ The bucket name must be globally unique among all the existing buckets in OSS. ○ The bucket name cannot be changed after the bucket is created.
Storage Class	Set the value to **Standard**. Only Standard is supported.
Capacity	Set the capacity of the bucket: ○ **Unlimited**: The capacity is unlimited. ○ **Custom**: Select this option to set the capacity of the bucket. Valid values: 0 to 2000000. Unit: TB or GB.

Parameter	Description
Access Control List (ACL)	Set the ACL for the bucket. You can select the following options: ○ **Private**: Only the owner or authorized users of this bucket can read and write objects in the bucket. Other users, including anonymous users cannot access objects in the bucket without authorization. ○ **Public Read**: Only the owner or authorized users of this bucket can read and write objects in the bucket. Other users, including anonymous users can only read objects in the bucket. ○ **Public Read/Write**: Any users, including anonymous users can read and write objects in the bucket. Fees incurred by such operations are paid by the owner of the bucket. Exercise caution when you configure this option. **Note** After a bucket is created, you can modify its ACL. For more information, see Modify bucket ACLs.
Server-side Encryption	**No**: Server-side encryption is not performed.

3. Click **Submit**.

2.3.3. Upload objects

After you create a bucket, you can upload objects to it.

Prerequisites

A bucket is created. For more information about how to create a bucket, see Create a bucket.

Context

You can upload an object of any format to a bucket. You can use the OSS console to upload an object up to 5 GB in size. To upload an object larger than 5 GB, use an SDK or call an API operation.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. Click the **Files** tab.
4. Click **Upload**. The **Upload** dialog box appears.
5. In the **Upload To** section, set the directory to which the object will be uploaded.
   - **Current**: Objects are uploaded to the current folder.
   - **Specified**: Objects are uploaded to the specified folder. OSS creates the specified folder automatically and uploads the object to it.

     Note For more information about folders, see Create folders.
6. In the **File ACL** section, select the ACL of the object to upload. By default, an object inherits the

ACL of the bucket to which it belongs.

7. Drag and drop one or more objects to upload to the **Upload** field, or click **Upload** to select one or more objects to upload.

   > Note
   > - If the uploaded object has the same name as an existing object in the bucket, the existing object will be overwritten.
   > - During object upload, do not refresh or close the page. Otherwise, the upload queue will be interrupted and cleared.
   > - The name of the uploaded object must comply with the following conventions:
   >   - The name can contain only UTF-8 characters.
   >   - The name is case-sensitive.
   >   - The name must be 1 to 1,023 bytes in length.
   >   - The name cannot start with a forward slash (/) or backslash (\).

8. After the object is uploaded, refresh the Files tab to view the uploaded object.

2.3.4. Obtain object URLs

You can obtain the URL of an object uploaded to a bucket. This URL can be used to share or download the object.

Prerequisites

Before you obtain an object URL, you must have a bucket and upload an object to it.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. Click the **Files** tab. The list of objects appears.
4. Click **View Details** in the Actions column corresponding to the target object. In the **Preview** dialog box that appears, click **Copy File URL** below the URL field. You can also choose **More > Copy File URL** in the Actions column corresponding to the bucket. In the dialog box that appears, click **Copy**.

What's next

You can send the URL to other users so that they can view or download the object.

2.4. Buckets

2.4.1. View bucket information

You can view the details of created buckets in the OSS console.

Prerequisites

A bucket is created. For more information about how to create a bucket, see Create a bucket.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. On the bucket details page that appears, click the **Overview** tab. View the bucket domain names and basic settings.

2.4.2. Delete buckets

You can delete buckets in the OSS console.

Prerequisites

All objects and parts stored in the bucket are deleted. For more information about how to delete objects and parts, see Delete objects and Manage parts.

> **Warning** Deleted objects, parts, and buckets cannot be recovered. Exercise caution when you delete objects, parts, and buckets.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. On the bucket details page that appears, click **Delete Bucket** in the upper right corner. In the message that appears, click **OK**.

2.4.3. Modify bucket ACLs

You can modify the access control list (ACL) of a bucket in the OSS console to control access to the bucket.

Prerequisites

A bucket is created. For more information about how to create a bucket, see Create buckets.

Context

OSS provides ACL to control access to buckets. By default, the ACL of a bucket is private when you create the bucket. You can modify the ACL of a bucket after the bucket is created.

OSS provides ACL for buckets. The following ACLs are available for a bucket:

- Private: Only the owner or authorized users of the bucket can read and write the object.
- Public read: Only the owner or authorized users of this bucket can write the object. Other users, including anonymous users can only read the object.
- Public read/write: Any users, including anonymous users can read and write the object. Fees incurred by such operations are paid by the owner of the bucket. Exercise caution when you configure this option.

Warning If you set ACL to public read or public read/write, other users can directly read the data in the bucket without authentication, resulting in security risks. For data security reasons, we recommend that you set the bucket ACL to private.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. On the bucket details page, click the **Basic Settings** tab. Find the **Access Control List (ACL)** section.
4. Click **Configure**. Modify the bucket ACL.
5. Click **Save**.

2.4.4. Configure static website hosting

You can configure static website hosting in the OSS console so that users can access the static website by using the bucket domain name.

Prerequisites

A bucket is created. For more information about how to create a bucket, see Create buckets.

Context

Static website hosting is not enabled if the default pages are not specified.

After the default homepage is configured, the default homepage is displayed if you access the root domain name of the static website or any URL that ends with a forward slash (/) under this domain name.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of the target bucket.
3. On the bucket details page, click the **Basic Settings** tab. Find the **Static Pages** section.
4. Click **Configure**. Configure the following parameters:
   - **Default Homepage**: Specify the name of the index document that links to the index page. The index page functions similar to index.html. Only HTML objects in the root folder can be used. The default homepage is disabled if you do not specify this parameter.
   - **Default 404 Page**: Specify the name of the error document that links to the error page displayed when the requested resource does not exist. Only HTML, JPG, PNG, BMP, or WebP objects in the root folder can be used. Default 404 Page is disabled if you do not specify this parameter.
5. Click **Save**.

2.4.5. Configure logging

You can enable or disable bucket logging in the OSS console.

Prerequisites

A bucket is created. For more information about how to create a bucket, see Create buckets.

Context

You can store access logs in the current bucket or in a new bucket.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of the target bucket to go to the bucket details page.
3. On the bucket details page that appears, click the **Basic Settings** tab. Find the **Logging** section.
4. Click **Configure**. Turn on Logging. Set **Destination Bucket** and **Log Prefix**.
   - **Destination Bucket**: Select the name of the bucket in which access logs are to store from the drop-down list. You must be the owner of the selected bucket and the bucket must be in the same region as the bucket for which logging is enabled.
   - **Log Prefix**: Enter the prefix and folder where the access logs are stored. If you specify *log/<TargetPrefix>*, the access logs are stored in the *log/* directory.
5. Click **Save**.

2.4.6. Configure hotlink protection

You can configure hotlink protection for a bucket in the OSS console to prevent unauthorized domain names from accessing the data in your bucket.

Prerequisites

A bucket is created. For more information about how to create a bucket, see Create buckets.

Context

OSS provides hotlink protection to prevent other domain names from accessing your data in OSS. You can configure the Referer field in the HTTP header to implement hotlink protection. You can configure a Referer whitelist for a bucket and configure whether to allow access requests that have an empty Referer field in the OSS console. For example, you can add `http://www.aliyun.com` to the Referer whitelist for a bucket named *oss-example*. Then, requests whose Referer field is set to `http://www.aliyun.com` can access the objects in the *oss-example* bucket.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of the target bucket to go to the bucket details page.
3. On the bucket details page, click the **Basic Settings** tab. Find the **Hotlink Protection** section.
4. Click **Configure**. Configure the following parameters:
   - **Referer Whitelist**: Add URLs to the whitelist. Referers are typically in URL format. Separate multiple Referers with break lines. You can use question marks (?) and asterisks (*) as wildcard characters.

- **Allow Empty Referer**: Specify whether to allow requests whose Referer field is empty. If you do not allow empty Referers, only HTTP or HTTPS requests which include the corresponding Referer field value can access the objects in the bucket.

5. Click **Save**.

2.4.7. Configure CORS

You can configure cross-origin resource sharing (CORS) in the OSS console to enable cross-origin access.

Prerequisites

A bucket is created. For more information about how to create a bucket, see Create a bucket.

Context

OSS provides CORS over HTML5 to implement cross-origin access. When OSS receives a cross-origin request (or an OPTIONS request) for a bucket, OSS reads the CORS rules of the bucket and checks the relevant permissions. OSS matches the rules one by one. When OSS finds the first match, OSS returns a corresponding header. If no match is found, OSS does not include any CORS header in the response.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of the target bucket to go to the bucket details page.
3. On the bucket details page, click the **Basic Settings** tab. Find the **Cross-Origin Resource Sharing (CORS)** section. Click **Configure**.
4. Click **Create Rule**. In the **Create Rule** dialog box that appears, configure the following parameters.

Parameter	Required	Description
Sources	Yes	Specifies the sources from which you want to allow cross-origin requests. You can configure multiple origins and separate them with break lines. Each origin can contain only one asterisk (*) wildcard. If Sources is set to asterisk (*), all cross-origin requests are allowed.
Allowed Methods	Yes	Specifies the cross-origin request methods that are allowed.
Allowed Headers	No	Specifies the allowed headers in a cross-origin request. Allowed headers are case-insensitive. You can configure multiple headers and separate them with break lines. Each allowed header can contain only one asterisk (*) wildcard. If there are no special header requirements, we recommend that you set Allowed Headers to asterisk (*) to allow all requests.

Parameter	Required	Description
Exposed Headers	No	Specifies the list of headers that can be exposed to the browser. The headers are the response headers that allow access from an application such as XMLHttpRequest in JavaScript. No asterisk (*) wildcards are allowed.
Cache Timeout (Seconds)	No	Specifies the time the browser caches the response for a prefetch (OPTIONS) request for specific resources.

> **Note** You can configure up to 10 rules for each bucket.

5. Click **OK**.

2.4.8. Manage lifecycle rules

You can define and manage lifecycle rules for a bucket in the OSS console.

Prerequisites

A bucket is created. For more information about how to create a bucket, see Create buckets.

Context

You can define a rule for a full set or a subset by specifying the prefix keyword of objects in a bucket. A rule applies to all objects that match the rule. You can manage lifecycle rules to perform operations, such as object management and automatic part deletion.

> **Notice**
>
> - If an object matches a rule, data of the object is deleted within two days from the effective date.
> - Data that is deleted based on a lifecycle rule cannot be recovered. Configure a rule only when necessary.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of the target bucket to go to the bucket details page.
3. Click the **Basic Settings** tab. Find the **Lifecycle** section. Click **Configure**.
4. Click **Create Rule**. In the **Create Rule** dialog box that appears, configure the following parameters:
   - **Status**: Configure the status of the rule: **Enabled** or **Disabled**.
   - **Applied To**: You can select **Files with Specified Prefix** or **Whole Bucket**. Files with Specified Prefix indicates that this rule applies to objects whose names contain a specified prefix. Whole Bucket indicates that this rule applies to all objects in the bucket.

> **Note** If you select **Files with Specified Prefix**, you can configure multiple lifecycle rules that have different prefixing configurations for objects. If you select **Whole Bucket**, only one lifecycle rule can be configured. In addition, if you have created a rule that has Files with Specified Prefix configured, you cannot create another rule that has Whole Bucket configured for the same bucket.

- **Prefix**: If you set **Applied To** to **Files with Specified Prefix**, you must enter the prefix of the objects to which to apply the rule. If you want to match objects whose names start with img, enter *img*.
- **File Lifecycle**: Configure operations to perform on expired objects. You can select **Validity Period (Days)**, **Expiration Date**, or **Disabled**.
  - **Validity Period (Days)**: Specify the number of days within which parts can be retained after they are last modified. After the validity period, expired parts are deleted. If you set Validity Period (Days) to 30, objects that are last modified on January 1, 2016 are scanned for by the backend application and deleted on January 31, 2016.
  - **Expiration Date**: Specify the date before which parts that are last modified expire and the operation to perform on these parts after they expire. If you select Delete and set Expiration Date to 2012-12-21, the backend application scans for objects that are last modified before December 21, 2012 and delete those objects.
  - **Disabled**: The automatic object deletion function is not enabled.
- **Delete**: If you select **Validity Period (Days)** or **Expiration Date** for **File Lifecycle**, you can select Delete to delete objects based on the validity period or expiration time. If you select **Disabled**, the rule becomes invalid.
- **Part Lifecycle**: Configure the delete operation to perform on expired parts. You can select **Validity Period (Days)**, **Expiration Date**, or **Disabled**.
  - **Validity Period (Days)**: Specify the number of days within which parts can be retained after they are last modified. After the validity period, expired parts are deleted. If you set Validity Period (Days) to 30, the backend application scans for parts that are last modified before January 1, 2016 and deletes them on January 31, 2016.
  - **Expiration Date**: Specify the date before which parts that are last modified expire and the operation to perform on these parts after they expire. If you set Expiration Date to 2012-12-21, parts that are last modified before this date are scanned for and deleted by the backend application.
  - **Disabled**: The automatic part deletion function is not enabled.
- **Delete**: If you select **Validity Period (Days)** or **Expiration Date** for **Part Lifecycle**, you can select Delete to delete parts based on the validity period or expiration time. If you select **Disabled**, the rule becomes invalid.

5. Click **OK**.

> **Notice**
>
> - Lifecycle rules are run after they are configured and saved. Check the configurations before you save them.
> - Object deletion is irreversible. Configure lifecycle rules as needed.

2.5. Objects

2.5.1. Search for objects

You can search buckets or folders for objects whose names contain a specified prefix in the OSS console.

Prerequisites

Before you search for objects, you must complete the procedure instructed in Create buckets and Upload objects, or at least one bucket exists in the current region and at least one object exists in the bucket.

Context

When you search for objects based on a prefix, the search string is case-sensitive and cannot contain a forward slash (/). The search range is limited to the root directory of the current bucket or the objects in the current folder (excluding subfolders and the objects in them).

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. On the bucket details page that appears, click the **Files** tab. The Files tab appears.
4. On the right side of the Files tab, enter a prefix in the search box and press Enter or click the **search** icon to search for the related objects.

   The system lists the names of objects and folders that match the prefix in the root directory of the bucket.

   > **Note** To search a specific folder for objects, open the folder and enter a prefix in the search box. The system lists the names of objects and subfolders in the folder that match the prefix.

2.5.2. Delete objects

You can delete uploaded objects in the OSS console.

Prerequisites

Before you delete objects, you must complete the procedure instructed in Create a bucket and Upload objects, or at least one bucket that contains at least one object must exist in the current region.

Context

You can delete one or more objects at a time. A maximum of 100 objects can be deleted at a time. You can use an SDK or call an API operation to delete a specific object or more than 100 objects.

> **Notice** Deleted objects cannot be recovered. Exercise caution when you delete objects.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of the target bucket to go to the bucket details page.
3. On the bucket details page that appears, click the **Files** tab.
4. Select one or more objects. Choose **Batch Operation > Delete**. You can also choose **More > Delete** in the Actions column corresponding to the target object.
5. In the **Delete File** message that appears, click **OK**.

2.5.3. Configure ACL for objects

You can configure ACL for an object in the OSS console to control access to the object.

Prerequisites

Before you configure ACL for an object, you must complete the procedure instructed in Create buckets and Upload objects, or at least one bucket exists in the current region and at least one object exists in the bucket.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. On the bucket details page that appears, click the **Files** tab.
4. On the Files tab, click the name of the target object. The **View Details** dialog box appears.
5. Click **Set ACL** on the right side of **File ACL**. The Set ACL dialog box appears. ACLs are described as follows:
   - **Inherited from Bucket**: The ACL of each object is the same as that of the bucket.
   - **Private**: Only the owner or authorized users of this bucket can read and write objects in the bucket. Other users, including anonymous users cannot access the objects in the bucket without authorization.
   - **Public Read**: Only the owner or authorized users of this bucket can write objects in the bucket. Other users, including anonymous users can only read objects in the bucket.
   - **Public Read/Write**: Any users, including anonymous users can read and write objects in the bucket. Fees incurred by such operations are paid by the owner of the bucket. Configure this option only when necessary.
6. Click **OK**.

2.5.4. Create folders

You can create a folder in a bucket in the OSS console.

Prerequisites

Before you create a folder, you must complete the procedure instructed in Create buckets, or at least one bucket exists in the current region.

Context

OSS does not use traditional folders. Folders are virtual in OSS. All elements are stored as objects. In the OSS console, a folder is an object whose size is 0 and has a name that ends with a forward slash (/). A folder is used to sort objects of the same type and process them at a time. The OSS console displays objects that end with a forward slash (/) as folders. These objects can be uploaded and downloaded. You can use OSS folders in the OSS console the way you use folders in Windows.

> **Note** The OSS console displays any objects that end with a forward slash (/) as folders, regardless of whether these objects contain data. You can download these objects only by calling an API operation or using an SDK.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. Click the **Files** tab. On the Files tab that appears, click **Create Folder**.
4. In the Create Folder dialog box that appears, set **Folder Name**. The folder name must comply with the following conventions:
   - The name can contain only UTF-8 characters. The name cannot contain emojis.
   - The name cannot start with a forward slash (/) or backslash (\). The name cannot contain consecutive forward slashes (/). Folders are separated with forward slashes (/). Subfolders in this path are automatically created.
   - The subfolder name cannot contain two consecutive periods (..) .
   - The name must be 1 to 254 characters in length.
5. Click **OK**.

2.5.5. Manage parts

When you upload an object in multipart upload mode, the object is split into several smaller parts. After all of the parts are uploaded to the OSS server, you can call CompleteMultipartUpload to combine them into a complete object. We recommend that you delete unnecessary parts on a regular basis.

Context

Parts are generated in multipart upload tasks and cannot be read until they are combined into a complete object. To save storage space in the bucket, you can configure lifecycle rules to manage unnecessary parts that are generated when multipart upload tasks fail. For more information, see Manage lifecycle rules.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of the target bucket to go to the bucket details page.
3. Click the **Files** tab. On the page that appears, click **Parts**.
4. In the **Parts** dialog box that appears, delete parts.
   - To delete all parts in the bucket, select all parts and click **Delete All**.

- To delete specified parts, select the parts that you want to delete and click **Delete**.

5. In the message that appears, click **OK**.

3.ApsaraDB for RDS

3.1. What is ApsaraDB for RDS?

ApsaraDB for RDS is a stable, reliable, and scalable online database service. Based on the distributed file system and high-performance storage, ApsaraDB for RDS allows you to easily perform database operations and maintenance with its set of solutions for disaster recovery, backup, restoration, monitoring, and migration.

ApsaraDB RDS for MySQL

Originally based on a branch of MySQL, ApsaraDB RDS for MySQL is a tried and tested solution for handling high-volume concurrent traffic during Double 11, providing excellent performance. ApsaraDB RDS for MySQL provides whitelist configuration, backup and restoration, transparent data encryption, data migration, and management for instances, accounts, and databases. ApsaraDB RDS for MySQL also provides the following advanced features:

- Read-only instance: In scenarios where RDS has a small number of write requests but a large number of read requests, you can enable read/write splitting to distribute read requests away from the primary instance. Read-only instances allow ApsaraDB RDS for MySQL 5.6 to automatically scale the reading capability and increase the application throughput when a large amount of data is being read.
- Data compression: ApsaraDB RDS for MySQL 5.6 allows you to use the TokuDB storage engine to compress data. Data transferred from the InnoDB storage engine to the TokuDB storage engine can be reduced by 80% to 90% in volume. 2 TB of data in InnoDB can be compressed to 400 GB or less in TokuDB. In addition to data compression, TokuDB supports transaction and online DDL operations. TokuDB is compatible with MyISAM and InnoDB applications.

3.2. Limits on ApsaraDB RDS for MySQL

Before using ApsaraDB RDS for MySQL, you must understand its limits and take precautions.

To ensure instance stability and security, ApsaraDB RDS for MySQL has some service limits, as listed in Limits on ApsaraDB RDS for MySQL.

Limits on ApsaraDB RDS for MySQL

Operation	Limit
Database parameter modification	Database parameters can be modified only by using the ApsaraDB RDS console or API operations. Due to security and stability considerations, only specific parameters can be modified.
Root permissions of databases	The root and SA permissions are not provided.
Database backup	• You can use the command line interface (CLI) or graphical user interface (GUI) to perform logical backup. • You can use the ApsaraDB for RDS console or API operations to perform physical backup.

Operation	Limit
Database restoration	• You can use the CLI or GUI to perform logical restoration. • You can use the ApsaraDB for RDS console or API operations to perform physical restoration.
Data import	• You can use the CLI or GUI to perform logical import. • You can use the MySQL CLI or DTS to import data.
ApsaraDB RDS for MySQL storage engine	• Only InnoDB and TokuDB are supported. Due to the inherent shortcomings of the MyISAM engine, some data may be lost. Only some existing instances use the MyISAM engine. MyISAM engine tables in newly created instances are automatically converted to InnoDB engine tables. • For performance and security considerations, we recommend that you use the InnoDB storage engine. • The Memory engine is not supported. Newly created Memory tables are automatically converted into InnoDB tables.
Database replication	ApsaraDB RDS for MySQL provides dual-node clusters based on a primary/secondary replication architecture. The secondary instances in this replication architecture are hidden and cannot be accessed directly.
RDS instance restart	Instances must be restarted by using the ApsaraDB for RDS console or API operations.
Account and database management	You can use the ApsaraDB for RDS console to manage accounts and databases for ApsaraDB RDS for MySQL instances. ApsaraDB RDS for MySQL also allows you to create a privileged account to manage users, passwords, and databases.
Standard account	• Custom authorization is not supported. • Both the account management and database management UIs are provided in the ApsaraDB for RDS console. • Instances that support standard accounts also support privileged accounts.
Privileged account	• Custom authorization is supported. • On the **Accounts** and **Databases** pages in the ApsaraDB for RDS console, the management feature is unavailable. Related operations can only be performed by using code • A privileged account cannot be reverted back to a standard account.

3.3. Log on to the ApsaraDB for RDS console

This topic describes how to log on to the ApsaraDB for RDS console.

Prerequisites

- The domain name of the Apsara Uni-manager console is obtained from the deployment personnel before you log on to the Apsara Uni-manager console.
- A browser is available. We recommend that you use the Google Chrome browser.

1. In the address bar, enter the URL used to log on to the Apsara Uni-manager console. 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 console for the first time, you must change the password of your username. For security reasons, your password must meet the minimum complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters.
   > - Digits.
   > - Special characters. Special characters include exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to go to the Apsara Uni-manager console homepage.
4. In the top navigation bar, choose **Products > ApsaraDB for RDS**.

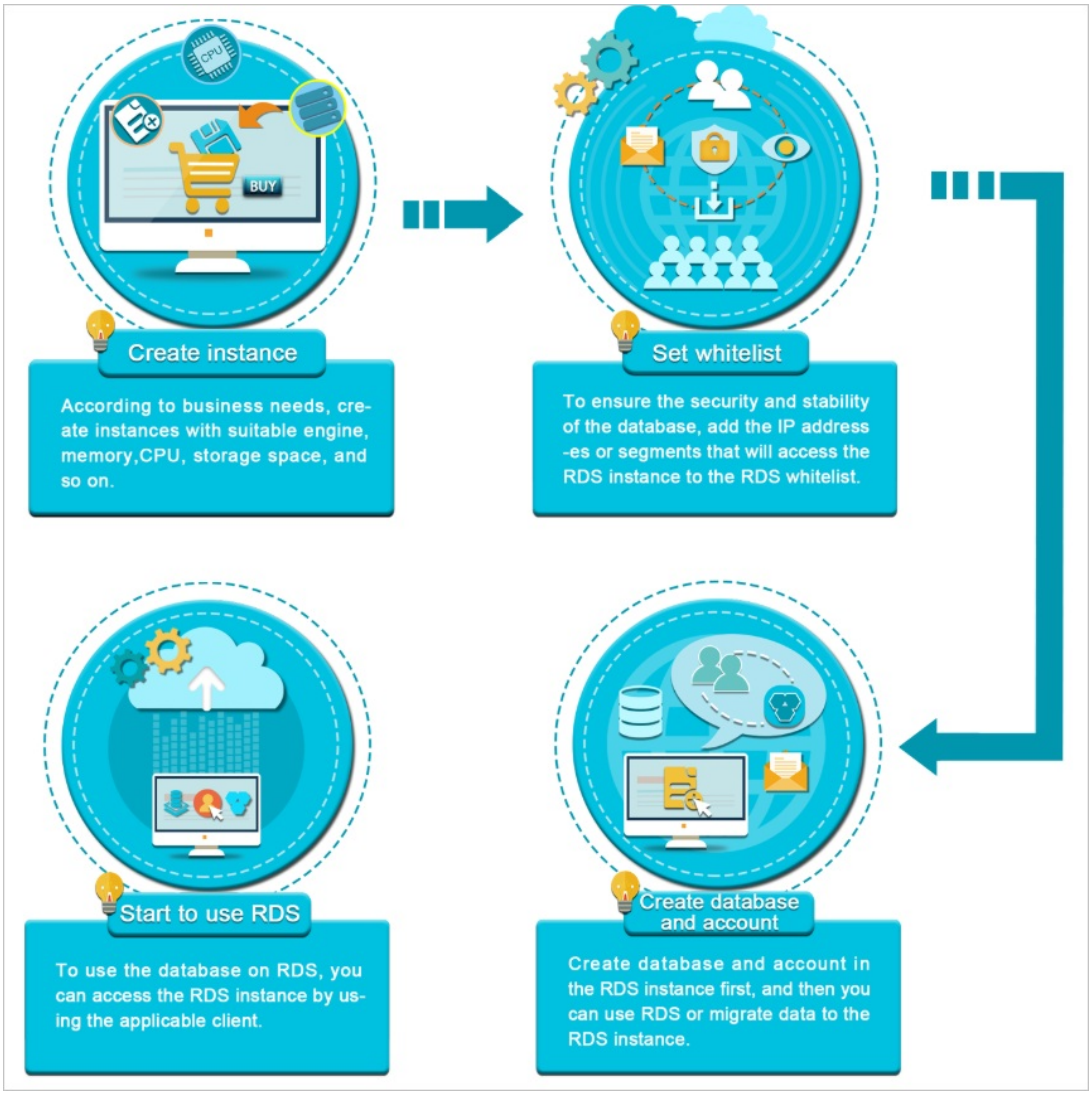
3.4. Quick Start

3.4.1. Procedure

ApsaraDB for RDS quick start covers the following topics: creating an RDS instance, configuring a whitelist, creating a database, creating an account, and connecting to the instance. This topic uses ApsaraDB RDS for MySQL as an example to describe how to use RDS. It provides all the necessary information to create an RDS instance.

Typically, you must complete several operations after instance creation to make it ready for use, as shown in Quick start flowchart.

Quick start flowchart



- Create an instance

  An instance is a virtualized database server on which you can create and manage multiple databases.

- Configure a whitelist

  After creating an RDS instance, you must configure its whitelist to allow access from external devices.

  The whitelist improves the access security of your RDS instance. We recommend that you maintain the whitelist on a regular basis. The whitelist configuration process does not affect the normal operations of the RDS instance.

- Create a database and an account

  Before using a database, you must first create the database and an account in the RDS instance. Different engines support different account modes. For more information, see the console UI and documentation.

- Connect to an ApsaraDB RDS for MySQL instance

  After creating an RDS instance, configuring a whitelist, and creating a database and an account, you can connect to the instance from a database client.

3.4.2. Create an instance

This topic describes how to create an instance in the ApsaraDB for RDS console.

Prerequisites

An Apsara Stack tenant account is obtained.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, click **Create Instance** in the upper-right corner.
3. Configure the following parameters.

Category	Parameter	Description
Basic Settings	Organization	The organization to which the instance belongs.
	Resource Set	The resource set to which the instance belongs.
Region	Region	The region where the instance resides. Services in different regions cannot communicate over the internal network. After a region is selected, it cannot be changed.
Specifications	Instance Name	The name of the instance. **Note** The name must be 2 to 64 characters in length and can contain letters, digits, underscores (_), hyphens (-), periods (.), colons (:), and commas (,). It must start with a letter and cannot start with http:// or https://.
	Database Engine	The engine of the database, which is automatically set to MySQL.
	Engine Version	The version of the database engine. Set the value to **5.6** or **5.7**.
	Edition	The edition of ApsaraDB for RDS. Select one from the drop-down list.
	Instance Type	The type of the instance. Select one from the drop-down list.
	Storage	The storage space of the instance, including the space for data, system files, binary binlog files, and transaction files.
Network Type	Network Type	The network type of the instance, which is automatically set to **Classic Network**. Classic Network: Cloud services in the classic network are not isolated. Unauthorized access to a cloud service is blocked only using security group or a whitelist policy of the service.

Category	Parameter	Description
IP Whitelist	The IP addresses that are allowed to connect to the ApsaraDB for RDS instance.	
Access Mode	Access Mode	The access mode of the instance, which is automatically set to **Standard**. Standard: ApsaraDB for RDS uses SLB to eliminate the impact of instance high-availability switchover on the application layer. This mode reduces the response time, but slightly increases the probability of transient connections and disables SQL interception.

4. After you configure the preceding parameters, click **Submit**.

3.4.3. Initialization

3.4.3.1. Configure a whitelist

To ensure database security and reliability, you must modify the whitelist of an ApsaraDB for RDS instance before you enable the instance. You must add the IP addresses or CIDR blocks that are used for database access to the whitelist.

Context

The whitelist improves the access security of your ApsaraDB for RDS instance. We recommend that you maintain the whitelist on a regular basis. The whitelist configuration process does not affect the normal operations of the ApsaraDB for RDS instance.

Precautions

- The default whitelist can be modified or cleared, but cannot be deleted.
- You can add up to 1,000 IP addresses or CIDR blocks to a whitelist. If you want to add a large number of IP addresses, we recommend that you merge them into CIDR blocks, such as 192.168.1.0/24.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Data Security**.

5. On the **Whitelist Settings** tab, click **Edit** corresponding to the **default** whitelist.

   > Note
   > - If you want to connect an ECS instance to an ApsaraDB for 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 click **Create Whitelist** to create a new whitelist.

6. In the dialog box that appears, specify the IP addresses or CIDR blocks used to access the instance and click **OK**.
   - If you specify the CIDR block 10.10.10.0/24, all IP addresses in the 10.10.10.X format are allowed to access the ApsaraDB for RDS instance.
   - If you want to add multiple IP addresses or CIDR blocks, separate each entry with a comma (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 and add them 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.

Create a whitelist

> Note You can create up to 50 whitelists for an instance.

Perform the following steps:

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Data Security**.
5. On the **Whitelist Settings** tab, click **Create Whitelist**.
6. In the **Create Whitelist** dialog box, configure the following parameters.

Parameter	Description
Whitelist Name	The name of the whitelist. - The whitelist name must be 2 to 32 characters in length. - It can contain lowercase letters, digits, and underscores (_). - It must start with a lowercase letter and end with a letter or digit.

Parameter	Description
IP Addresses	The IP addresses or CIDR blocks used to access the instance. ○ If you add the CIDR block 10.10.10.0/24, all IP addresses in the 10.10.10.X format are allowed to access the ApsaraDB for RDS instance. ○ If you want to add multiple IP addresses or CIDR blocks, separate each entry with a comma (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 and add them to the whitelist.

7. Click **OK**.

3.4.3.2. Create a privileged account

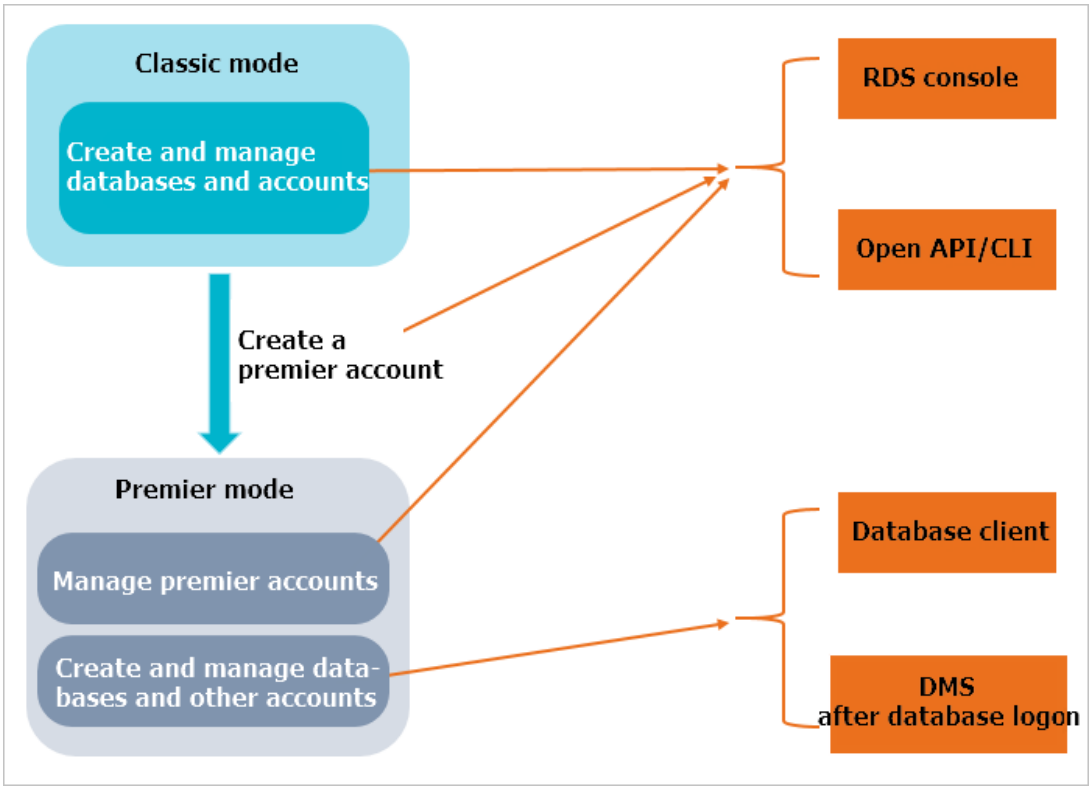
ApsaraDB for RDS supports two account management modes, which are classic and privileged. For ApsaraDB RDS for MySQL 5.6 instances, you can upgrade the account management mode from classic to privileged by creating a privileged account.

Context

Compared with the classic mode, the privileged mode enables more permissions and allows you to use SQL to directly manage databases and accounts. Therefore, we recommend that you use the privileged mode. After a privileged account is created for a primary instance, the privileged account is synchronized to read-only instances.

Comparison of account management modes shows the differences between the two modes in creating and managing databases and accounts for ApsaraDB RDS for MySQL 5.6 instances.

Comparison of account management modes



Note

- In an ApsaraDB RDS for MySQL 5.6 instance, the account management mode can be upgraded from classic to privileged, but cannot be downgraded from privileged to classic.
- The instance restarts when a privileged account is created, which causes a transient connection of less than 30 seconds. To avoid service impacts from transient connections, select an appropriate time and ensure that your applications are configured with automatic reconnection policies.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Accounts**.
5. Click **Create Account** in the upper-right corner.
6. Configure the following parameters.

Parameter	Description

Parameter	Description
Database Account	Required. Enter the account name. The account name must meet the following requirements: ○ The name must be 2 to 16 characters in length. ○ The name must start with a letter and end with a letter or digit. ○ The name can contain lowercase letters, digits, and underscores (_).
Account Type	Required. Select **Privileged Account**.
Password	Required. Enter an account password. The password must meet the following requirements: ○ The password must be 8 to 32 characters in length. ○ The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. ○ Special characters include: ! @#$%^&*()_+-=
Re-enter Password	Required. Enter the password again.
Description	Optional. Enter information about the account to facilitate subsequent management. The description can be up to 256 characters in length.

7. Click **Create**.

3.4.3.3. Create a standard account

After you create an ApsaraDB for RDS instance and configure its whitelist, you must create a database and an account in the instance. This topic describes how to create a standard account.

Context

To migrate a local database to ApsaraDB for RDS, you must create a database and an account in the ApsaraDB for RDS instance identical to those in the local database. Databases in an instance share all resources that belong to the instance.

Use service roles to create accounts and follow the principle of least privilege to assign appropriate read-only and read/write permissions to accounts. When necessary, you can split database accounts and databases into smaller units so that each database account can only access data for its own services.

Notice For database security, set strong account passwords and change the passwords on a regular basis.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to

the **Basic Information** page.

4. In the left-side navigation pane, click **Accounts**.
5. Click **Create Account** in the upper-right corner.
6. Configure the following parameters.

Parameter	Description
Database Account	Required. Enter the account name. The account name must meet the following requirements: ○ The name must be 2 to 16 characters in length. ○ The name must start with a letter and end with a letter or digit. ○ The name can contain lowercase letters, digits, and underscores (_).
Account Type	Required. Select **Standard Account**.
Authorized Databases	Optional. Grant permissions on one or more databases to the account. You do not have to configure this parameter at this time. You can authorize databases after the account is created. i. Select one or more databases from the Unauthorized Databases section and click **Add** to add them to the Authorized Databases box. ii. In the Authorized Databases section, select the **Read/Write**, **Read-only**, **DDL Only**, or **DML Only** permissions on each authorized database. If you want to grant the same permissions on multiple databases to the account, click the button in the upper-right corner of the section. The button may appear as **Set All to Read/Write**.
Password	Required. Enter an account password. The password must meet the following requirements: ○ The password must be 8 to 32 characters in length. ○ The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. ○ Special characters include: ! @#$%^&*()_+-=
Re-enter Password	Required. Enter the password again.
Description	Optional. Enter information about the account to facilitate subsequent management. The description can be up to 256 characters in length.

7. Click **Create**.

3.4.3.4. Create a database

After you create an ApsaraDB for RDS instance and configure its whitelist, you must create a database and an account in the instance.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Databases**.
5. Click **Create Database** in the upper-right corner.
6. Configure the following parameters.

Parameter	Description
Database Name	○ The database name must be 2 to 64 characters in length. ○ The name must start with a letter and end with a letter or digit. ○ The name can contain lowercase letters, digits, underscores (_), and hyphens (-). ○ Each database name must be unique in an instance.
Supported Character Sets	Select utf8, gbk, latin1, utf8mb4, or all. If you want to use other character sets, select **all**, and then select the required character set from the list.
Description	Optional. Enter information about the database to facilitate subsequent management. The description can be up to 256 characters in length.

7. Click **Create**.

3.4.4. Connect to an ApsaraDB RDS for MySQL instance

After completing the initial configurations, you can use an ECS instance or a database client to connect to an ApsaraDB RDS for MySQL instance.

Prerequisites

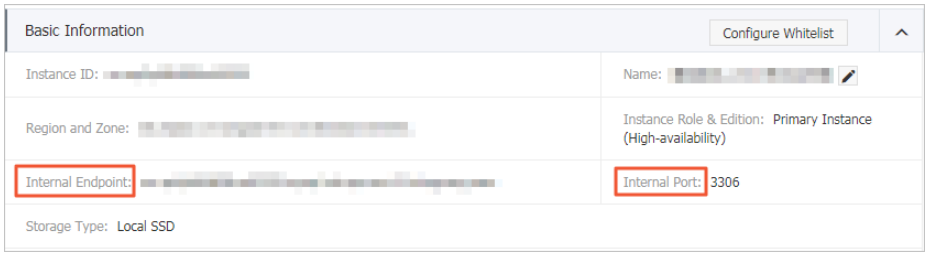
- An ApsaraDB RDS for MySQL instance is created. For more information about how to create an instance, see Create an instance.
- An account is created to connect to the MySQL instance. For more information about how to create an account, see Create a privileged account and Create a standard account.
- The IP address of the server that needs to connect to the MySQL instance is added to the whitelist. For more information about how to configure a whitelist, see Configure a whitelist.

If you need to connect an ECS instance to an ApsaraDB for RDS instance, you must make sure that both instances are in classic networks or in the same VPC, and the IP address of the ECS instance is correctly configured in the RDS whitelist.

Connect to an instance from a client

ApsaraDB RDS for MySQL is fully compatible with MySQL. You can connect to an ApsaraDB for RDS instance from a general database client in the similar way you connect to a MySQL database. The following example uses the HeidiSQL client:

1. Start the HeidiSQL client.
2. In the lower-left corner, click **Create**.
3. Enter information about the RDS instance you want to connect. The parameters are described as follows.

Parameter	Description
Network Type	The network type of the database you want to connect. Select **MariaDB or MySQL (TCP/IP)**.
Hostname/IP	Enter the internal or public IP address of the RDS instance. ○ If your client is deployed in an ECS instance, and the instance is in the same region and has the same network type as the destination RDS instance, you can use the internal IP address. For example, if your ECS and RDS instances are both in a VPC located in the China (Hangzhou) region, you can use the internal IP address provided to create a secure connection. ○ Use the public IP address for other situations. To view the internal and public endpoints together with the corresponding port numbers of the RDS instance, perform the following steps: i. Log on to the ApsaraDB for RDS console. ii. Find the target instance and click its ID. iii. On the **Basic Information** page that appears, you can view the internal endpoint and internal port number of the instance. 
User	Enter the name of the account used to access the RDS instance.
Password	The password of the account.
Port	The port number of the RDS instance. If you connect to the instance over the internal network, enter the internal port number of the instance. If you connect to the instance over the Internet, enter the public port number of the instance.

4. Click **Open**. If the connection information is correct, you can connect to the instance.

3.5. Instances

3.5.1. Create an instance

This topic describes how to create an instance in the ApsaraDB for RDS console.

Prerequisites

An Apsara Stack tenant account is obtained.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, click **Create Instance** in the upper-right corner.
3. Configure the following parameters.

Category	Parameter	Description
Basic Settings	Organization	The organization to which the instance belongs.
	Resource Set	The resource set to which the instance belongs.
Region	Region	The region where the instance resides. Services in different regions cannot communicate over the internal network. After a region is selected, it cannot be changed.
Specifications	Instance Name	The name of the instance. **Note** The name must be 2 to 64 characters in length and can contain letters, digits, underscores (_), hyphens (-), periods (.), colons (:), and commas (,). It must start with a letter and cannot start with http:// or https://.
	Database Engine	The engine of the database, which is automatically set to MySQL.
	Engine Version	The version of the database engine. Set the value to **5.6** or **5.7**.
	Edition	The edition of ApsaraDB for RDS. Select one from the drop-down list.
	Instance Type	The type of the instance. Select one from the drop-down list.
	Storage	The storage space of the instance, including the space for data, system files, binary binlog files, and transaction files.
Network Type	Network Type	The network type of the instance, which is automatically set to **Classic Network**. Classic Network: Cloud services in the classic network are not isolated. Unauthorized access to a cloud service is blocked only using security group or a whitelist policy of the service.

Category	Parameter	Description
IP Whitelist	The IP addresses that are allowed to connect to the ApsaraDB for RDS instance.	
Access Mode	Access Mode	The access mode of the instance, which is automatically set to **Standard**. Standard: ApsaraDB for RDS uses SLB to eliminate the impact of instance high-availability switchover on the application layer. This mode reduces the response time, but slightly increases the probability of transient connections and disables SQL interception.

4. After you configure the preceding parameters, click **Submit**.

3.5.2. View basic information about an instance

You can view the details of an instance, such as its basic information, internal network connection information, running status, and configurations.

1. Log on to the ApsaraDB for RDS console.
2. You can use one of the following methods to access the **Basic Information** page of an instance:
   - On the **RDS Management** page, click the ID of the instance to access its **Basic Information** page.
   - On the **RDS Management** page, click **Management** in the Actions column corresponding to the instance to access its **Basic Information** page.

3.5.3. Restart an instance

If the number of connections exceeds the threshold or any performance issue occurs on an instance, you can manually restart the instance.

Prerequisites

The instance is in the **Running** state.

> **Notice** A restart will disconnect the instance. We recommend that you make appropriate service arrangements before you restart an instance. Proceed with caution.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. Click **Restart Instance** in the upper-right corner.

> **Note** A restart will disconnect the instance. We recommend that you make appropriate service arrangements before you restart an instance. Proceed with caution.

5. In the message that appears, click **Confirm**.

3.5.4. Modify configurations

You can modify configurations of your instance, such as editions, instance types, and storage space, if the configurations do not meet the requirements of your application.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the **Configuration Information** section, click **Change Specifications**.
5. In the dialog box that appears, click **Next**.
6. On the **Change Specifications** page, set **Edition**, **Instance Type**, and **Storage**.
7. After you configure the preceding parameters, click **Submit**.

3.5.5. Set a maintenance window

You can set a maintenance window for an ApsaraDB for RDS instance based on your business requirements.

Context

To ensure the stability of ApsaraDB for RDS instances, the backend system performs maintenance on the instances at irregular intervals. The default maintenance window is from 02:00 to 06:00. You can set the maintenance window to the off-peak period of your business to avoid impact on business.

Precautions

- An instance enters the **Maintaining Instance** state before the maintenance window to ensure stability during the maintenance process. When the instance is in this state, access to data in the database and query operations such as performance monitoring are not affected. However, except for account and database management and IP address whitelist configuration, modification operations such as upgrade, downgrade, and restart are temporarily unavailable.
- During the maintenance window, the instance is disconnected once or twice. Make sure that your applications are configured with automatic reconnection policies to avoid service disruptions.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the **Configuration Information** section, click **Configure** to the right of **Maintenance**

Window.

5. Select a maintenance window and click **Save**.

> **Note** The maintenance window is in UTC+8.

3.5.6. Release an instance

You can manually release instances based on your business requirements.

Precautions

- You can manually release only instances that are in the running state.
- After an instance is released, the instance data is immediately cleared. We recommend that you back up your data before you release an instance.

1. Log on to the ApsaraDB for RDS console.
2. Find the target instance and choose **More** > **Release Instance** in the Actions column.
3. In the **Release Instance** message that appears, click **Confirm**.

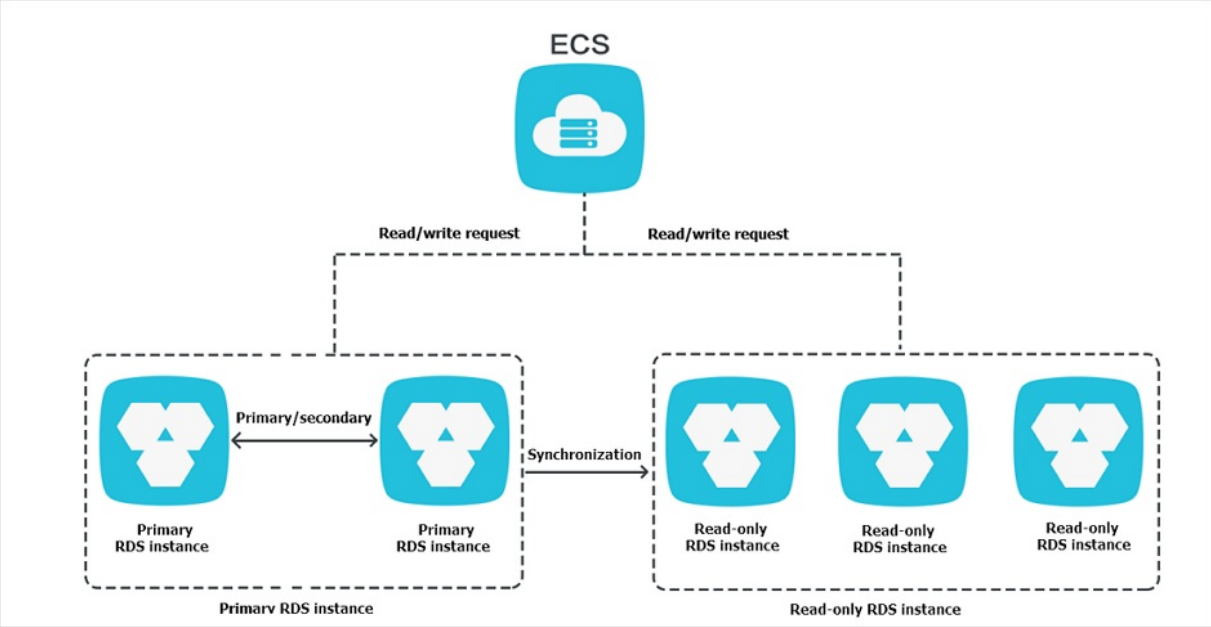
3.5.7. Read-only instances

3.5.7.1. Overview

RDS for MySQL 5.6 allows you to create read-only instances. In scenarios where there are a few write requests but a large number of read requests, you can create read-only instances to relieve the database access load on the primary instance. This topic describes the features and limits of read-only instances.

To scale the reading capability and distribute database access loads, you can create one or more read-only instances in a region. Read-only instances allow RDS to increase the application throughput when a large amount of data is being read.

A read-only instance with a single physical node and no backup node uses the native replication capability of MySQL to synchronize changes from the primary instance to all its read-only instances. Real-only instances must be in the same region as the primary instance but do not have to be in the same zone as the primary instance. The following figure shows the topology of read-only instances.



Read-only instances have the following features:

- Specifications of a read-only instance can be different from those of the primary instance and can be changed at any time, which facilitates elastic scaling.
- Read-only instances do not require account or database maintenance. Account and database information is synchronized from the primary instance.
- The whitelists of read-only instances can be configured independently.
- System performance monitoring is provided.

  RDS provides up to 20 system performance monitoring views, including those for disk capacity, IOPS, connections, CPU utilization, and network traffic. You can view the load of instances easily.
- RDS provides a variety of optimization recommendations, such as storage engine check, primary key check, large table check, and check for excessive indexes and missing indexes. You can optimize your databases based on the optimization recommendations and specific applications.

3.5.7.2. Create a read-only instance

You can create read-only instances of different specifications based on your business requirements.

Precautions

- A maximum of five read-only instances can be created for a primary instance.
- Backup settings and temporary backup are not supported.
- Instance restoration is not supported.
- Data migration to read-only instances is not supported.
- Database creation and deletion are not supported.
- Account creation, deletion, authorization, and password changes are not supported.
- After a read-only instance is created, you cannot restore data by directly overwriting the primary instance with a backup set.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the **Distributed by Instance Role** section on the right side of the page, click **Create Read-only Instance**.
5. On the **Create Read-only RDS Instance** page, configure the read-only instance parameters.

Category	Parameter	Description
Region	Region	The region where the ApsaraDB for RDS instance resides.
Specifications	Database Engine	The database engine of the read-only instance, which is automatically set to MySQL.
	Engine Version	The engine version of the read-only instance, which is the same as that of the primary instance.
	Edition	The edition of the read-only instance, which is automatically set to **Read-only Instance**.
	Instance Type	The type of the read-only instance. The type of the read-only instance can be different from that of the primary instance, and can be modified at any time to ensure flexible upgrade and downgrade.
	Storage	The storage space of the read-only instance. To ensure sufficient I/O throughput for data synchronization, we recommend that you select at least the same instance type and storage space as the primary instance for the read-only instance.
Network Type	Network Type	The network type of the read-only instance, which is automatically set to **Classic Network**.

6. After you configure the preceding parameters, click **Submit**.

3.5.7.3. View the details of read-only instances

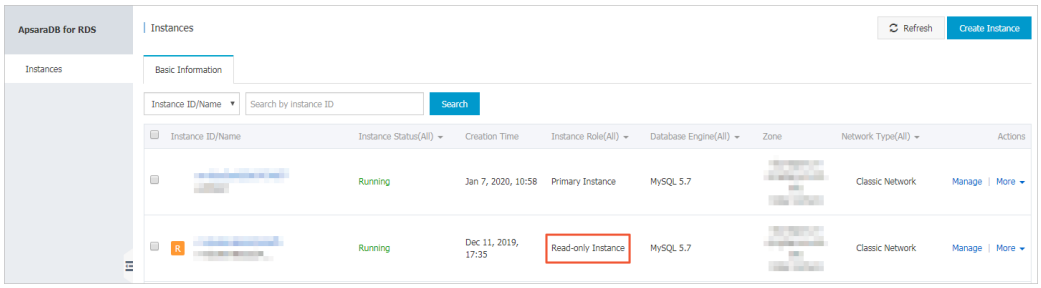
3.5.7.3.1. View instance details by using a read-only instance

You can go to the read-only instance management page from the Instances page or from the Read-only Instance section of the primary instance. Read-only instances are managed in the same way as primary instances. The read-only instance management page shows the management operations that can be performed. This topic describes how to go to the read-only instance management page from the Instances page.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, click the ID of a read-only instance. The Basic Information page appears. In the instance list, **Instance Role** of read-only instances is displayed as **Read-only Instance**, as shown in View read-only instances.

View read-only instances

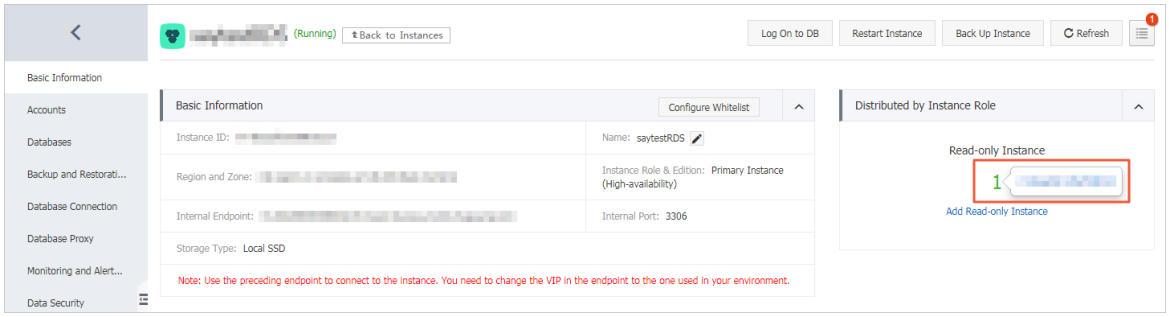


3.5.7.3.2. View instance details by using the primary instance

You can go to the read-only instance management page from the Instances page or from the Read-only Instance section of the primary instance. Read-only instances are managed in the same way as primary instances. The read-only instance management page shows the management operations that can be performed. This topic describes how to go to the read-only instance management page from the Read-only Instance section of the primary instance.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. On the **Basic Information** page, move the pointer over the number below **Read-only Instance** in the **Distributed by Instance Role** section. The ID of the read-only instance is displayed.



5. Click the ID of the read-only instance to go to the read-only instance management page.

3.6. Accounts

3.6.1. Create an account

This topic describes the functions and features of accounts in classic and privileged modes, and how to create accounts for both modes.

You must create an account in an ApsaraDB for RDS instance before you can use the database. ApsaraDB for RDS supports two account management modes: classic and privileged. The classic mode is a management mode retained from earlier versions of ApsaraDB for RDS. In the classic mode, databases and accounts cannot be managed by using SQL statements. The privileged mode is a management mode introduced in later versions that enables more permissions. In the privileged mode, databases and accounts can be managed by using SQL statements. We recommend that you use the privileged mode for personalized and fine-grained control over database permissions.

Account modes

In the classic mode, all accounts are created by using the ApsaraDB for RDS console or API operations, instead of by using SQL statements. All accounts are equal. There is not one account with more management permissions over others. You can use the ApsaraDB for RDS console to create and manage all accounts and databases.

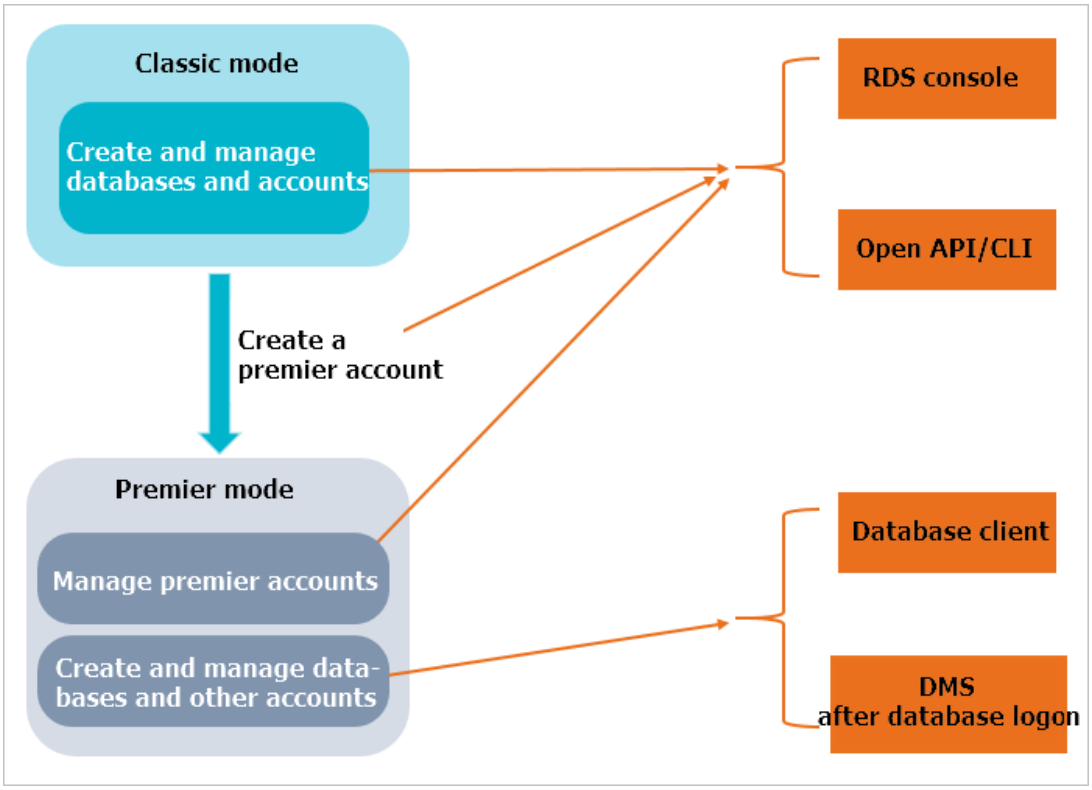
In the privileged mode, the first account that you create is the initial account. You must use the ApsaraDB for RDS console or API operations to create and manage the initial account. Log on to a database with your initial account. You can then use SQL statements or Data Management (DMS) to create and manage standard accounts. However, you cannot use the initial account to change the passwords of standard accounts. To change the password of a standard account, you must delete the account and create a new one. Run the following commands to log on to the database by using the initial account named root and create a standard account named jeffrey:

```
mysql -hxxxxxxxxx.mysql.rds.aliyuncs.com -uroot -pxxxxxx -e "
  CREATE USER 'jeffrey'@'%' IDENTIFIED BY 'password';
  CREATE DATABASE DB001;
  "
```

In the privileged mode, you cannot manage databases by using the ApsaraDB for RDS console or API operations. You must use SQL statements or DMS to create and manage databases.

For more information, see Difference between standard and privileged accounts.

Difference between standard and privileged accounts



How to create an account

> **Note**
>
> - Use service roles to create accounts and follow the principle of least privilege to assign appropriate read-only and read/write permissions to accounts. When necessary, you can split database accounts and databases into smaller units so that each database account can only access data for its own services. If an account does not need to write data to a database, assign read-only permissions to the account.
> - Use strong passwords for database accounts and change the passwords on a regular basis.

- For more information about how to create a standard account for ApsaraDB RDS for MySQL, see Create a standard account.
- For more information about how to create a privileged account for ApsaraDB RDS for MySQL, see Create a privileged account.

3.6.2. Reset the password

You can use the ApsaraDB for RDS console to reset the password of your database account.

> **Note** For data security, we recommend that you change your password on a regular basis.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.

4. In the left-side navigation pane, click **Accounts**.
5. Find the target account and click **Reset Password** in the Actions column.
6. In the dialog box that appears, enter and confirm the new password, and then click **OK**.

> **Note** The password must meet the following requirements:
> - The password must be 8 to 32 characters in length.
> - It must contain characters from at least three of the following categories: uppercase letters, lowercase letters, digits, and special characters.
> - Special characters include ! @ # $ % ^ & * ( ) _ + - =

3.6.3. Modify account permissions

You can modify the account permissions of your ApsaraDB for RDS instance at any time.

> **Note** You can modify the permissions of a standard account. The permissions of privileged accounts can only be reset to the default settings and cannot be changed to a specific set of permissions.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Accounts**.
5. Find the target account and click **Edit Permissions** in the Actions column.
6. Configure the following parameters.

Parameter	Description
Unauthorized Databases	Select a database and click **Add** or **Remove**.
Authorized Databases	In the Authorized Databases list, you can set the account permissions to Read/Write or Read-only. You can also click the button in the upper-right corner to set the permissions of the account on all authorized databases. The button shows **Set All to Read/Write**, **Set All to Read-only**, **Set All to DDL Only**, or **Set All to DML Only** as you click it. ○ **Read-only**: grants the database read-only permissions to the account. ○ **Read/Write**: grants the database read/write permissions to the account. ○ **DDL Only**: grants the database permissions of DDL operations to the account. ○ **DML Only**: grants the database permissions of DML operations to the account.

7. Click **OK**.

3.6.4. Delete an account

You can delete a database account in the ApsaraDB for RDS console.

You can use the console to delete privileged and standard accounts that are no longer needed.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Accounts**.
5. Find the target account and click **Delete** in the Actions column.
6. In the message that appears, click **Confirm**.

> **Note** Accounts in the **Processing** state cannot be deleted.

3.7. Databases

3.7.1. Create a database

After you create an ApsaraDB for RDS instance and configure its whitelist, you must create a database and an account in the instance.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Databases**.
5. Click **Create Database** in the upper-right corner.
6. Configure the following parameters.

Parameter	Description
Database Name	◦ The database name must be 2 to 64 characters in length. ◦ The name must start with a letter and end with a letter or digit. ◦ The name can contain lowercase letters, digits, underscores (_), and hyphens (-). ◦ Each database name must be unique in an instance.
Supported Character Sets	Select utf8, gbk, latin1, utf8mb4, or all. If you want to use other character sets, select **all**, and then select the required character set from the list.
Description	Optional. Enter information about the database to facilitate subsequent management. The description can be up to 256 characters in length.

7. Click **Create**.

3.7.2. Delete a database

You can delete databases that are no longer used in the ApsaraDB for RDS console.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Databases**.
5. Find the database you want to delete and click **Delete** in the Actions column.
6. In the message that appears, click **Confirm**.

3.8. Database connection

3.8.1. Change the endpoint of an instance

This topic describes how to change the endpoint of an instance.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Database Connection**.
5. Click **Change Endpoint** on the right side of the **Database Connection** section.
6. In the dialog box that appears, configure Connection Type, Endpoint, and Port, and click **OK**.

> Note
>
> - The prefix of the endpoint must be 8 to 64 characters in length and can contain letters, digits, and hyphens (-). It must start with a lowercase letter.
> - The port number must be in the range of 1000 to 65534.

3.9. Monitoring and alerts

The ApsaraDB for RDS console provides a variety of performance metrics for you to monitor the status of your instances.

3.9.1. View resource and engine monitoring data

The ApsaraDB for RDS console provides a variety of performance metrics to monitor the status of your instances.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Monitoring and Alerts**.
5. On the **Monitoring and Alerts** page, select **Resource Monitoring** or **Engine Monitoring**, and select a time range to view the corresponding monitoring data. The following table describes the metrics.

Category	Metric	Description
Resource Monitoring	Disk Space (MB)	The disk space usage of the instance. It consists of the following parts: ◦ Instance size ◦ Data usage ◦ Log size ◦ Temporary file size ◦ Other system file size Unit: MB.
	IOPS	The number of I/O requests per second for the instance.
	Total Connections	The total number of current connections to the instance, including the number of active connections and the total number of connections.
	CPU Utilization and Memory Usage	The CPU utilization and memory usage of the instance. These metrics do not include the CPU utilization and memory usage for the operating system.
	Network Traffic (KB)	The inbound and outbound traffic of the instance per second. Unit: KB.
	TPS/QPS	The average number of transactions per second and the average number of SQL statements executed per second.
	InnoDB Buffer Pool Read Hit Ratio, Usage Ratio, and Dirty Block Ratio (%)	The read hit ratio, usage ratio, and dirty block ratio of the InnoDB buffer pool.
	InnoDB Read/Write Volume (KB)	The amount of data that InnoDB reads and writes per second. Unit: KB.
	InnoDB Buffer Pool Read/Write Frequency	The number of read and write operations that InnoDB performs per second.
	InnoDB Log Read/Write/fsync	The average frequency of physical writes to log files per second by InnoDB, the log write request frequency, and the average frequency of fsync writes to log files.

Category	Metric	Description
Engine Monitoring	Temporary Tables Automatically Created on Hard Disk when MySQL Statements Are Executed	The number of temporary tables that are automatically created on the hard disk when the database executes SQL statements.
	MySQL_COMDML	The number of SQL statements that the database executes per second. The following SQL statements are included: ◦ Insert ◦ Delete ◦ Insert_Select ◦ Replace ◦ Replace_Select ◦ Select ◦ Update
	MySQL_RowDML	The numbers of operations that InnoDB performs per second. The following items are included: ◦ The average number of physical writes to log files per second ◦ The average number of rows that are read, updated, deleted, and inserted from InnoDB tables per second
	MyISAM Read/Write Frequency	The numbers of operations that MyISAM performs per second. The following items are included: ◦ The average number of MyISAM reads from the buffer pool per second ◦ The average number of MyISAM writes from the buffer pool per second ◦ The average number of MyISAM reads from the hard disk per second ◦ The average number of MyISAM writes from the hard disk per second
	MyISAM Key Buffer Read/Write/Usage Ratio (%)	The read hit ratio, write hit ratio, and usage of the MyISAM key buffer per second.

3.9.2. Set a monitoring frequency

The ApsaraDB for RDS console provides a variety of performance metrics for which you can set a monitoring frequency.

ApsaraDB for RDS provides the following monitoring frequencies:

- Every 5 seconds

- Every 60 seconds
- Every 300 seconds

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Monitoring and Alerts**.
5. Click **Set Monitoring Frequency** on the right side of the page.
6. In the **Set Monitoring Frequency** dialog box that appears, select the new monitoring frequency and click **OK**.

> **Note** If your instance does not support the selected monitoring frequency, a prompt appears in the Set Monitoring Frequency dialog box. Select a monitoring frequency supported by the instance.

3.10. Data security

3.10.1. Configure a whitelist

To ensure database security and reliability, you must modify the whitelist of an ApsaraDB for RDS instance before you enable the instance. You must add the IP addresses or CIDR blocks that are used for database access to the whitelist.

Context

The whitelist improves the access security of your ApsaraDB for RDS instance. We recommend that you maintain the whitelist on a regular basis. The whitelist configuration process does not affect the normal operations of the ApsaraDB for RDS instance.

Precautions

- The default whitelist can be modified or cleared, but cannot be deleted.
- You can add up to 1,000 IP addresses or CIDR blocks to a whitelist. If you want to add a large number of IP addresses, we recommend that you merge them into CIDR blocks, such as 192.168.1.0/24.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Data Security**.
5. On the **Whitelist Settings** tab, click **Edit** corresponding to the **default** whitelist.

> Note
>
> - If you want to connect an ECS instance to an ApsaraDB for 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 click **Create Whitelist** to create a new whitelist.

6. In the dialog box that appears, specify the IP addresses or CIDR blocks used to access the instance and click **OK**.
   - If you specify the CIDR block 10.10.10.0/24, all IP addresses in the 10.10.10.X format are allowed to access the ApsaraDB for RDS instance.
   - If you want to add multiple IP addresses or CIDR blocks, separate each entry with a comma (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 and add them 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.

Create a whitelist

> **Note** You can create up to 50 whitelists for an instance.

Perform the following steps:

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Data Security**.
5. On the **Whitelist Settings** tab, click **Create Whitelist**.
6. In the **Create Whitelist** dialog box, configure the following parameters.

Parameter	Description
Whitelist Name	The name of the whitelist. - The whitelist name must be 2 to 32 characters in length. - It can contain lowercase letters, digits, and underscores (_). - It must start with a lowercase letter and end with a letter or digit.

Parameter	Description
IP Addresses	The IP addresses or CIDR blocks used to access the instance. ○ If you add the CIDR block 10.10.10.0/24, all IP addresses in the 10.10.10.X format are allowed to access the ApsaraDB for RDS instance. ○ If you want to add multiple IP addresses or CIDR blocks, separate each entry with a comma (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 and add them to the whitelist.

7. Click **OK**.

3.10.2. SQL audit

You can use the SQL audit feature to audit SQL executions and check the details. SQL audit does not affect instance performance.

> **Note** You cannot view the logs that are generated before you enable SQL audit.

You can view the incremental data of your ApsaraDB RDS for MySQL instance in SQL audit logs or binlogs. However, these two methods differ in the following aspects:

- SQL audit logs are similar to audit logs in MySQL and record all DML and DDL operations by using network protocol analysis. SQL audit does not parse the actual parameter values. Therefore, a small amount of information may be lost if a large number of SQL statements are executed to query data. The incremental data obtained using this method may be inaccurate.
- Binlogs record all add, delete, and modify operations and the incremental data used for data restoration. Binlogs are temporarily stored in your ApsaraDB for RDS instance after they are generated. The system transfers full binlog files to OSS on a regular basis. OSS then stores the files for seven days. However, a binlog file cannot be transferred if data is being written to it. Such binlog files will fail to be uploaded to OSS after you click **Upload Binlogs**. Binlogs are not generated in real time, but you can obtain accurate incremental data from them.

  > **Note** You can click **Upload Binlogs** on the **Backup and Restoration** page.

Precautions

- SQL audit is disabled by default. SQL audit does not affect instance performance.
- SQL audit logs are retained for 30 days.
- Log files exported from SQL audit are retained for two days. The system clears files that are retained for more than two days.

Enable SQL audit

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to

the **Basic Information** page.

4. Click the **SQL Audit** tab.
5. Click **Enable SQL Audit**.
6. In the message that appears, click **Confirm**.

> **Note** After SQL audit is enabled, you can query SQL information based on conditions such as the time range, database, user, and keyword.

Disable SQL audit

> **Note** If SQL audit is disabled, all SQL audit logs are deleted. We recommend that you export and store audit logs locally before you disable SQL audit.

You can disable SQL audit to avoid charges when you do not need it. To disable SQL audit, perform the following operations:

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Data Security**.
5. Click the **SQL Audit** tab.
6. Click **Export File** to export and store the SQL audit content to your computer.
7. After the file is exported, click **Disable SQL Audit**. In the message that appears, click **Confirm**.

3.11. Service availability

3.11.1. Automatically or manually switch over services between primary and secondary instances

This topic describes how to automatically or manually switch over services between primary and secondary instances. After the switchover, the original primary instance becomes a secondary instance.

Context

- Automatic switchover: the default switchover mode. If the primary instance experiences a fault, your RDS services are automatically switched over to the secondary instance.
- Manual switchover: allows you to manually switch over services between primary and secondary instances. You can do so even if automatic switchover is enabled.

> **Note** Data is synchronized between the primary and secondary instances in real time. You can only connect to the primary instance. The secondary instance serves only as a backup and does not allow external access.

Precautions

- Services may be disconnected during a switchover. Make sure that your applications are configured with automatic reconnection policies to avoid service disruptions.
- If the primary instance is attached with read-only instances, data on the read-only RDS instances shows a latency of a few minutes after a switchover. This is because it takes time to re-establish replication connections and synchronize incremental data.

Manually switch over services between primary and secondary instances

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Service Availability**.
5. Click **Switch Primary/Secondary Instance** on the right side of the page.

   > **Note** Services may be disconnected once or twice during the switchover. Make sure that your applications are configured with automatic reconnection policies to avoid service disruptions.

6. In the message that appears, click **OK**.

FAQ

Can I connect to secondary instances?

No, you cannot connect to secondary instances. You can only connect to primary instances. Secondary instances only serve as a backup and do not allow external access.

3.11.2. Change the data replication mode

You can set the data replication mode between primary and secondary ApsaraDB for RDS instances to improve database availability.

Data Replication Mode

You can use the following methods to replicate data:

- **Semi-synchronous**:

  After an application-initiated update is completed on the primary instance, logs are synchronized to all secondary instances. This transaction is considered committed after at least one secondary instance receives the logs. This way, there is no need to wait for the logs to be applied.

  If the secondary instances are unavailable or a network exception occurs between the primary and secondary instances, semi-synchronous replication will degrade to the Asynchronous mode.

- **Asynchronous**:

  When an application initiates an update request to add, delete, or modify data, the primary instance responds to the application immediately after the operation is complete. The primary instance then replicates data to the secondary instances asynchronously. During asynchronous data replication, the unavailability of secondary instances does not affect the operations on the primary instance. Data remains consistent even if the primary instance is unavailable.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Service Availability**.
5. Click **Change Data Replication Mode** on the right side of the page.
6. In the dialog box that appears, select a data replication mode and click **OK**.

3.12. Database backup and restoration

3.12.1. Automatic backup

Automatic backup supports full physical backups. ApsaraDB for RDS automatically backs up data based on pre-configured policies. This topic describes how to configure a policy for automatic backup.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Backup and Restoration**.
5. Click the **Backup Settings** tab.
6. Click **Edit**.

   > **Note** To ensure data security, the system compares the new backup cycle and time with the original settings, and selects the most recent time point to back up the data. Therefore, the next backup may still be performed based on the original backup cycle and time. For example, if the backup time is set to 19:00-20:00 every Wednesday and you modify the time to 19:00-20:00 every Thursday before 19:00 this Wednesday, the system will still back up data during 19:00-20:00 this Wednesday.

7. Configure the following parameters.

Parameter	Description
Data Retention Period	The number of days for which data backup files are retained. Valid values: 7 to 730. Default value: 7.
Backup Cycle	The backup cycle. You can select one or multiple days within a week.

Parameter	Description
Backup Time	Any period of time within a day. Unit: hours. We recommend that you back up data during off-peak hours.
Log Backup	Specifies whether to enable log backup. **Notice** If you disable log backup, all the log backup files are deleted, and you cannot restore data to a saved point in time.
Log Retention Period	The number of days for which log backup files are retained. Valid values: 7 to 730. Default value: 7. **Note** The log backup retention period must be shorter than or equal to the data backup retention period.

8. Click **OK**.

3.12.2. Manual backup

Manual backup supports both full physical backups and full logical backups. This topic describes how to manually back up ApsaraDB for RDS data.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. Click **Back Up Instance** in the upper-right corner.
5. Specify the backup mode and backup policy, and click **OK**.

Note Two backup methods are available:

- Physical backup: directly backs up all files in all databases.
- Logical backup: extracts data from the databases through SQL and backs up the data in the text format. If you select logical backup, you must select a backup policy:
  - Instance Backup: backs up the entire instance.
  - Single-Database Backup: backs up one of the databases in the instance.

3.12.3. Restore data to a new instance (formerly known as cloning an instance)

A cloned instance is a new instance with the same content as the primary instance, including data and settings. This feature allows you to restore data of the primary instance or create multiple instances that are the same as the primary instance.

Prerequisites

- The primary instance is in the running state.
- The primary instance does not have an ongoing migration task.
- Data backup and log backup are enabled.
- The primary instance has at least one completed backup set before you clone the instance by backup set.

Features

You can specify a backup set or any point in time within the backup retention period to clone an instance.

> **Note**
>
> - A cloned instance copies only the content of the primary instance. The copied content includes database information, account information, and instance settings such as whitelist settings, backup settings, parameter settings, and alert threshold settings.
> - The database engine of a cloned instance must be the same as that of the primary instance. Other settings can be different, such as the instance edition, zone, network type, instance type, and storage space. If you want to clone an instance to restore the data of a primary instance, we recommend that you select an instance type that has higher specifications and more storage space than those of the primary instance to speed up the data restoration process.
> - The account type of a cloned instance must be the same as that of the primary instance. The account password of the cloned instance can be changed.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Backup and Restoration**.
5. In the backup list, find the target backup and click **Restore** in the Actions column.
6. In the dialog box that appears, select **Restore Database** and click **OK**.
7. On the **Restore RDS Instance** page, configure the following parameters.

Category	Parameter	Description
Region	Region	The region where the ApsaraDB for RDS instance resides.
	The restore mode	The restore mode. Select **By Time** or **By Backup Set** to restore the database.

Category	Parameter	Description
Database Restoration	Time	The point in time to which you want to restore the database. **Note** When **Restore Mode** is set to **By Time**, you must specify this parameter.
	Backup Set	The backup set for restoration. **Note** When **Restore Mode** is set to **By Backup Set**, you must specify this parameter.
Specifications	Instance Name	The name of the cloned instance.
	Database Engine	The engine of the database, which is the same as that of the primary instance and is automatically set to **MySQL**.
	Engine Version	The version of the database engine, which is automatically set to that of the primary instance.
	Edition	The edition of the database. Select **High-availability Edition** or **Read-only Instance**.
	Instance Type	The type of the cloned instance. **Note** We recommend that you select an instance type and storage space that are higher than those of the primary instance to speed up the data restoration process.
	Storage	The storage space of the instance, including the space for data, system files, binary log files, and transaction files.
Network Type	Network Type	The network type of the instance, which is automatically set to **Classic Network**.

8. After you configure the preceding parameters, click **Submit**.

3.12.4. Restore individual databases or tables for an ApsaraDB RDS for MySQL instance

ApsaraDB RDS for MySQL allows you to restore individual databases and tables. If you delete one or more databases or tables of an RDS instance by mistake, you can use a backup set to restore the databases or tables.

Prerequisites

- An instance has no more than 50,000 tables. This function is not available if an instance has more than 50,000 tables.
- Restoration of individual databases or tables is enabled. For more information, see Enable restoration of individual databases or tables.
- If you want to restore data to its original instance, the original instance must meet the following requirements:
  - It is running and is not locked.
  - It does not have an ongoing migration task.
  - To restore data by time, you must make sure that the log backup function is enabled.
  - To restore data by backup set, you must make sure that the original instance has at least one backup set.
- If you want to restore data to a new instance, the original instance must meet the following requirements:
  - It is running and is not locked.
  - To restore data by time, you must make sure that the log backup function is enabled.
  - To restore data by backup set, you must make sure that the original instance has at least one backup set.

Precautions

- A primary/secondary failover occurs when the data is restored to its original instance, which may result in transient connections of ApsaraDB for RDS. Make sure that your applications are configured with automatic reconnection policies.
- After you enable the restoration of individual databases or tables, backup files are converted from the TAR format to the xbstream format. The storage space occupied by backup files will slightly increase due to the change in the backup file format. Pay attention to the space used for backup.
- You can select a maximum of 50 databases or tables at a time.

Enable restoration of individual databases or tables

> **Note**
>
> - After you enable the restoration of individual databases or tables, this function cannot be disabled, and backup file formats are changed.
> - By default, this function is enabled for new instances and cannot be disabled.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Backup and Restoration**.
5. Click the **Backup Settings** tab and click **Edit**.
6. In the dialog box that appears, select **Enabled** to the right of **Restore Individual Database/Table**.

7. Click **OK**.

Restore individual databases or tables of an RDS instance

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Backup and Restoration**.
5. In the upper-right corner of the page, click **Restore Individual Database/Table**. In the dialog box that appears, configure the following parameters.

Parameter	Description
Restore To	**Current Instance**: restores databases or tables to the original instance.
Restore Method	○ **By Backup Set** ○ **By Time**: restores data to any point in time within the retention period of log backup. **Note** By Time is displayed only if the log backup function is enabled.
Backup Set	Select a backup set to restore databases or tables. **Note** This parameter is valid if you set **Restore Method** to By Backup Set.
Restore Time	Select the point in time to which you want to restore databases or tables. **Note** This parameter is valid if you set **Restore Method** to By Time.
Databases and Tables to Restore	Select the databases or tables you want to restore.
Selected Databases and Tables	○ The selected databases and tables are displayed. You can set new names for these databases and tables. ○ The total size of the selected databases and tables and the available storage space of the current instance are displayed. Check whether the available storage space is sufficient.

6. Click **OK**.

3.13. Logs

All ApsaraDB for RDS instances support log management. You can query details about the error logs and slow query logs of an ApsaraDB for RDS instance through the ApsaraDB for RDS console. The logs help you locate faults.

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, find the target instance.
3. Click the instance ID or click **Manage** in the **Actions** column corresponding to the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, click **Logs**.
5. On the Logs page, click the **Error Logs**, **Slow Log Details**, or **Primary/Secondary Instance Switching Logs** tab. Select a time range and click **Search**.

Tab	Description
Error Logs	Records database running errors that occurred within the last month.
Slow Log Details	Records SQL statements that took longer than one second to execute in the last month, and deletes redundant SQL statements. **Note** Slow query logs in the ApsaraDB for RDS console are updated once every minute. However, you can query real-time slow query logs from the mysql.slow_log table.
Primary/Secondary Switching Logs	This feature is suitable for ApsaraDB RDS for MySQL instances on High-availability Edition

3.14. Migrate data from an on-premises database to an ApsaraDB for RDS instance

3.14.1. Compress data

ApsaraDB RDS for MySQL 5.6 allows you to use the TokuDB storage engine to compress data. This topic describes how to compress data.

Context

Data transferred from the InnoDB storage engine to the TokuDB storage engine can be reduced by 80% to 90% in volume. 2 TB of data in InnoDB can be compressed to 400 GB or less in TokuDB. In addition to data compression, TokuDB supports transactions and online DDL operations. TokuDB is compatible with MyISAM and InnoDB applications.

> **Note**
> - The TokuDB storage engine does not support foreign keys.
> - The TokuDB storage engine is not applicable to scenarios that require frequent reading of large amounts of data.

Procedure

1. Run the following command to check the MySQL version:

```
SELECT version();
```

> **Note** For more information about how to log on to a database, see Connect to an ApsaraDB RDS for MySQL instance.

2. Run the following command and set loose_tokudb_buffer_pool_ratio to indicate the proportion of cache that TokuDB occupies in the shared cache of TokuDB and InnoDB:

```
select sum(data_length) into @all_size from information_schema.tables where engine='innodb';
select sum(data_length) into @change_size from information_schema.tables where engine='innodb' a
nd concat(table_schema, '.', table_name) in ('XX.XXXX', 'XX.XXXX', 'XX.XXXX');
select round(@change_size/@all_size*100);
```

> **Note** In the preceding command, XX.XXXX indicates the name of the database or table to be transferred to the TokuDB storage engine.

3. Restart the instance. For more information, see Restart an instance
4. Run the following command to change the storage engine:

```
ALTER TABLE XX.XXXX ENGINE=TokuDB
```

> **Note** In the preceding command, XX.XXXX indicates the name of the database or table to be transferred to the TokuDB storage engine.

3.14.2. Migrate MySQL data

3.14.2.1. Use mysqldump to migrate MySQL data

This topic describes how to use mysqldump to migrate local data to an ApsaraDB RDS for MySQL instance.

Prerequisites

An ECS instance is created.

Context

mysqldump is easy to use but requires extensive downtime. This tool is suitable for scenarios where the amount of data is small or extensive downtime is allowed.

ApsaraDB RDS for MySQL is fully compatible with the native database service. The procedure of migrating the original database to an ApsaraDB RDS for MySQL instance is similar to that of migrating data from one MySQL server to another.

Before you migrate data, create a migration account in the on-premises database, and grant read and write permissions on the database to the migration account.

Procedure

1. Run the following command to create a migration account in the on-premises database: `CREATE USER 'username'@'host' IDENTIFIED BY 'password';`

   Parameter description:

   - username: specifies the name of the account to be created.
   - host: specifies the host from which you log on to the database. As a local user, you can use localhost to log on to the database. To log on from any host, you can use the wildcard %.
   - password: specifies the logon password for the account.

   For example, if you want to create account William with password Changme123 for logging on to the on-premises database from any host, run the following command:

   ```
   CREATE USER 'William'@'%' IDENTIFIED BY 'Changme123';
   ```

2. Run the following command to grant permissions to the migration account in the on-premises database: `GRANT SELECT ON databasename.tablename TO 'username'@'host' WITH GRANT OPTION; GRANT REPLICATION SLAVE ON databasename.tablename TO 'username'@'host' WITH GRANT OPTION; GRANT REPLICATION SLAVE ON databasename.tablename TO 'username'@'host' WITH GRANT OPTION;`

   Parameter description:

   - privileges: specifies the operation permissions granted to the account, such as SELECT, INSERT, and UPDATE. To grant all permissions to the account, use ALL.
   - databasename: specifies the database name. To grant all database permissions to the account, use wildcard *.
   - tablename: specifies the table name. To grant all table permissions to the account, use the wildcard *.
   - username: specifies the name of the account to which you want to grant permissions.
   - host: specifies the host from which the account is authorized to log on to the database. As a local user, you can use localhost to log on to the database. To log on from any host, you can use the wildcard %.
   - WITH GRANT OPTION: an optional parameter that enables the account to use the GRANT command.

   For example, if you want to grant all of the database and table permissions to the William account and use the account to log on to the on-premises database from any host, run the following command:

   ```
   GRANT ALL ON *.* TO 'William'@'%';
   ```

3. Use the data export tool of mysqldump to export data from the database as a data file.

> **Notice** Do not update data during data export. This step only exports data. It does not export stored procedures, triggers, or functions.

```
mysqldump -h localIp -u userName -p --opt --default-character-set=utf8 --hex-blob dbName --skip-triggers > /tmp/dbName.sql
```

Parameter description:

- localIp: specifies the IP address of the on-premises database server.
- userName: specifies the migration account of the on-premises database.
- dbName: specifies the name of the database you want to migrate.
- /tmp/dbName.sql: specifies the name of the backup file.

4. Use mysqldump to export stored procedures, triggers, and functions.

> **Notice** Skip this step if no stored procedures, triggers, or functions are used in the database. When exporting stored procedures, triggers, and functions, you must remove the DEFINER to ensure compatibility with ApsaraDB RDS for MySQL.

```
mysqldump -h localIp -u userName -p --opt --default-character-set=utf8 --hex-blob dbName -R | sed -e 's/DEFINER[ ]*=[ ]*[^*]*\*/\*/' > /tmp/triggerProcedure.sql
```

Parameter description:

- localIp: specifies the IP address of the on-premises database server.
- userName: specifies the migration account of the on-premises database.
- dbName: specifies the name of the database you want to migrate.
- /tmp/triggerProcedure.sql: specifies the name of the backup file.

5. Upload the data file and stored procedure file to the ECS instance. The example in this topic shows how to upload files to the following paths:

```
/tmp/dbName.sql
```

```
/tmp/triggerProcedure.sql
```

6. Log on to the ECS console and import both the data file and stored procedure file to the target ApsaraDB RDS for MySQL instance.

```
mysql -h intranet4example.mysql.rds.aliyuncs.com -u userName -p dbName < /tmp/dbName.sql
```

```
mysql -h intranet4example.mysql.rds.aliyuncs.com -u userName -p dbName < /tmp/triggerProcedure.sql
```

Parameter description:

- intranet4example.mysql.rds.aliyuncs.com: the endpoint of the ApsaraDB RDS for MySQL instance. An internal endpoint is used as an example.
- userName: specifies the migration account of the ApsaraDB RDS for MySQL database.
- dbName: specifies the name of the database you want to import.
- /tmp/dbName.sql: specifies the name of the data file you want to import.
- /tmp/triggerProcedure.sql: specifies the name of the stored procedure file you want to import.

4.Data Transmission Service (DTS)

4.1. What is DTS?

Data Transmission Service (DTS) is a data service that is provided by Alibaba Cloud. DTS supports data transmission between various types of data sources, such as relational databases.

Features

DTS has the following advantages over traditional data migration and synchronization tools: high compatibility, high performance, security, reliability, and ease of use. DTS allows you to simplify data transmission and focus on business development.

Feature	Description
Data migration	You can use DTS to migrate data between homogeneous and heterogeneous data sources. This feature applies to the following scenarios: data migration to Alibaba Cloud, data migration between instances within Alibaba Cloud, and database splitting and scale-out.
Data synchronization	You can use DTS to synchronize data between data sources. This feature applies to the following scenarios: disaster recovery, data backup, load balancing, cloud BI systems, and real-time data warehousing.
Change tracking	You can use DTS to track data changes from databases in real time. This feature applies to the following scenarios: cache updates, business decoupling, asynchronous data processing, synchronization of heterogeneous data, and synchronization of extract, transform, and load (ETL) operations.

4.2. Log on to the DTS console

This topic describes how to log on to the Data Transmission Service (DTS) console. Google Chrome is used in this example.

Prerequisites

- The domain name of the Apsara Uni-manager console is obtained from the deployment personnel before you log on to the Apsara Uni-manager console.
- A browser is available. We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to log on to the Apsara Uni-manager console. 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 console for the first time, you must change the password of your username. For security reasons, your password must meet the minimum complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:
>
> - Uppercase or lowercase letters.
> - Digits.
> - Special characters. Special characters include exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to go to the Apsara Uni-manager console homepage.
4. In the top navigation bar, choose **Products > Data Transmission Service**.
5. On the **DTS** page, select an **organization** and **region**, and then click **DTS**.

4.3. Data migration

4.3.1. Database and migration types

You can use Data Transmission Service (DTS) to migrate data between homogeneous and heterogeneous data sources. Typical scenarios include data migration to Alibaba Cloud, data migration between instances within Alibaba Cloud, and database splitting and scale-out. This topic describes the database types, database versions, and migration types that are supported by data migration.

Database and migration types

Source database	Destination database	Migration type
User-created MySQL database Version 5.5, 5.6, 5.7, or 8.0	User-created MySQL database Version 5.5, 5.6, 5.7, or 8.0	• Schema migration • Full data migration • Incremental data migration
	User-created Kafka database Version 0.10.1.0 to 1.0.2	
User-created PostgreSQL database Version 9.4, 9.5, 9.6, or 10.x	User-created PostgreSQL database Version 9.4, 9.5, 9.6, or 10.x	
User-created Oracle database Version 9i, 10g, 11g, 12c, 18c, or 19c	AnalyticDB for PostgreSQL Version 4.3 or 6.0	

Source database	Destination database	Migration type
User-created MongoDB database Version 3.0, 3.2, 3.4, 3.6, or 4.0	User-created MongoDB database Version 3.0, 3.2, 3.4, 3.6, or 4.0	• Full data migration • Incremental data migration Note MongoDB and Redis are NoSQL databases that do not require schema migration.
User-created Redis database Version 2.8, 3.0, 3.2, or 4.0	User-created Redis database Version 2.8, 3.0, 3.2, or 4.0	

4.3.2. Data type mappings between heterogeneous databases

Heterogeneous databases support different data types. During schema migration, Data Transmission Service (DTS) converts the data types of the source database into those of the destination database. This topic lists the data type mappings for you to evaluate the impact of data migration on your business.

Data migration from a user-created Oracle database to an AnalyticDB for PostgreSQL instance

User-created Oracle database	AnalyticDB for PostgreSQL
VARCHAR2	varchar/text
BINARY_DOUBLE	double precision
BINARY_FLOAT	double precision
BINARY_INTEGER	integer
BLOB	bytea
CLOB	text
DATE	timestamp
DEC	decimal
DECIMAL	decimal
DOUBLE PRECISION	double precision
FLOAT	double precision
INT	int

User-created Oracle database	AnalyticDB for PostgreSQL

INTERGE	integer
LONG	text
LONG RAW	bytea
NCLOB	text
NUMBER	numeric
PLS_INTEGER	integer
RAW	bytea
REAL	real
ROWID	oid
SMALLINT	smallint
TIMESTAMP	timestamp
TIMESTAMP WITH LOCAL TIME ZONE	timestamp with time zone
TIMESTAMP WITH TIME ZONE	timestamp with time zone
XMLTYPE	xml

Note If an Oracle data type is not supported by AnalyticDB for PostgreSQL, DTS converts the data type into bytea. If the conversion fails, DTS sets the field value to null.

4.3.3. Create a data migration instance

Before you configure a task to migrate data, you must create a data migration instance. This topic describes how to create a data migration instance in the Data Transmission Service (DTS) console.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Migration**.
3. In the upper-right corner, click **Create Migration Task**.
4. In the Create DTS Instances dialog box, select a region, and enter the number of data migration instances that you want to create.

   Note In the Create DTS Instances dialog box, you can view the total number of instances, the number of existing instances, and the number of instances that can be created.

5. Click Create.

4.3.4. Configure data migration tasks

4.3.4.1. Migrate data between user-created MySQL databases

This topic describes how to migrate data between user-created MySQL databases by using Data Transmission Service (DTS). DTS supports schema migration, full data migration, and incremental data migration. You can select all of the supported migration types to ensure service continuity.

Prerequisites

- The version of the source MySQL database is 5.5, 5.6, 5.7, or 8.0.
- If you need to perform incremental data migration, you must enable the binary logging feature. The following requirements must be met:
  - The value of the `binlog_format` parameter is set to `row`.
  - The value of the `binlog_row_image` parameter is set to `full`.

Precautions

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- The source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, duplicate data may exist in the destination database.
- DTS uses the `ROUND(COLUMN,PRECISION)` function to retrieve values from columns of the FLOAT or DOUBLE data type. If you do not specify a precision, DTS sets the precision for the FLOAT data type to 38 digits and the precision for the DOUBLE data type to 308 digits. You must check whether the precision settings meet your business requirements.

Migration types

- Schema migration

  DTS migrates the schemas of objects to the destination database. DTS supports schema migration for tables, views, triggers, stored procedures, and functions.

  > Note
  >
  > - During schema migration, DTS changes the value of the SECURITY attribute from DEFINER to INVOKER for views, stored procedures, and functions.
  > - DTS does not migrate user information. Before a user can call views, stored procedures, and functions of the destination database, you must grant the read and write permissions to the user.

- Full data migration

DTS migrates historical data of the required objects from the source database to the destination database.

> **Note** During full data migration, concurrent INSERT operations cause fragmentation in the tables of the destination database. After full data migration is complete, the tablespace of the destination database is larger than that of the source database.

- Incremental data migration

  After full data migration is complete, DTS reads binary log files of the source database and synchronizes incremental data from the source database to the destination database.

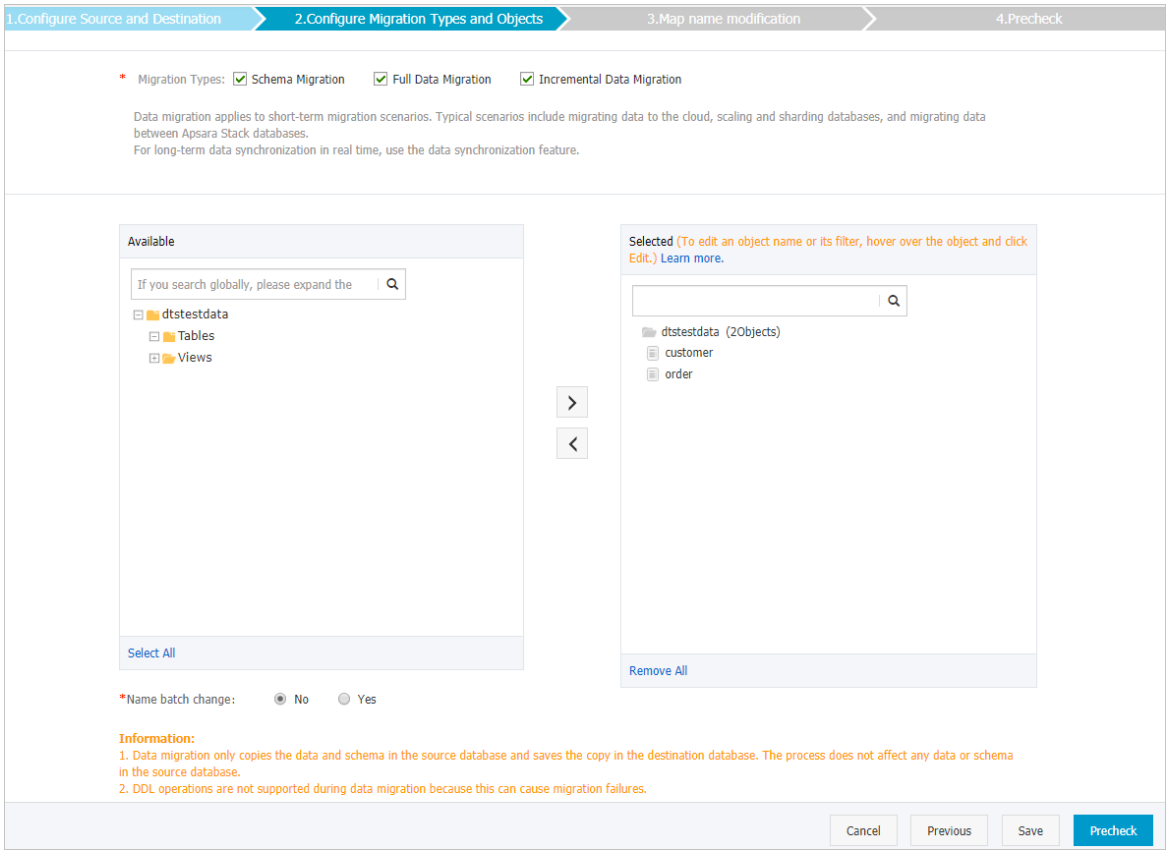
Procedure

1. Create a data migration instance.
2. In the migration task list, find the migration task that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	Select the region where the source database resides.
	Database Type	Select **MySQL**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the source database.
	Port Number	Enter the service port number of the source database. The default port number is **3306**.
	Database Account	Enter the account of the source database. ◦ If you need to migrate incremental data, the database account must have the following permissions: the SELECT permission on the objects to be migrated, the REPLICATION SLAVE permission, the REPLICATION CLIENT permission, and the SHOW VIEW permission. ◦ If you do not need to migrate incremental data, the database account must have the SELECT permission on the objects to be migrated.

Section	Parameter	Description
	Database Password	Enter the password of the source database account. **Note** After you specify the source database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the source database parameters based on the check results.
Destination Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	Database Type	Select **MySQL**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the destination database.
	Port Number	Enter the service port number of the destination database. The default port number is **3306**.
	Database Account	Enter the account of the destination database. The account must have the ALL permission on the destination database.
	Database Password	Enter the password of the destination database account. **Note** After you specify the destination database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the destination database parameters based on the check results.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the migration types and the objects to be migrated.



Setting	Description
Select the migration types	○ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ○ To ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.

Setting	Description
Select the objects to be migrated	Select objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** • You can select columns, tables, or databases as the objects to be migrated. • After an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. • If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.

6. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

7. After the task passes the precheck, click **Next**.

   Note
   - If **Incremental Data Migration** is not selected, wait until the migration task automatically stops.
   - If **Incremental Data Migration** is selected, wait until **Incremental Data Migration** and **The migration task is not delayed** appear in the progress bar. Then, manually stop the migration task.

4.3.4.2. Migrate data from a user-created MySQL database to a user-created Kafka cluster

Kafka is a distributed message queue service that features high throughput and high scalability. Kafka is widely used for big data analytics such as log collection, data aggregation, streaming processing, and online and offline analysis. It is important for the big data ecosystem. This topic describes how to migrate data from a user-created MySQL database to a user-created Kafka cluster by using Data Transmission Service (DTS). The data migration feature allows you to extend message processing capabilities.

Prerequisites

- The version of the source MySQL database is 5.5, 5.6, 5.7, or 8.0.
- If you need to perform incremental data migration, you must enable the binary logging feature. The following requirements must be met:
  - The value of the `binlog_format` parameter is set to `row`.
  - The value of the `binlog_row_image` parameter is set to `full`.
- A Kafka cluster is created and the Kafka version is 0.10.1.0 to 1.0.2.

Precautions

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- The source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
- You can select only tables as the objects to be migrated.

Data format

The data that is migrated to the Kafka cluster is stored in the Avro format. You must parse the migrated data based on the Avro schema. For more information, see DTS Avro schema.

Procedure

1. Create a data migration instance.
2. In the migration task list, find the migration task that you created, and click **Configure Migration Task** in the Actions column.
3. Configure a task name.DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
4. Configure the source and destination databases.

Section	Parameter	Description
	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	Select the region where the source database resides.
	Database Type	Select **MySQL**.

Section	Parameter	Description
Source Database	Hostname or IP Address	Enter the endpoint that is used to connect to the source database.
	Port Number	Enter the service port number of the source database. The default port number is **3306**.
	Database Account	Enter the account of the source database. The account must have the SELECT permission on the objects to be migrated, the REPLICATION CLIENT permission, the REPLICATION SLAVE permission, and the SHOW VIEW permission.
	Database Password	Enter the password of the source database account.
Destination Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	Database Type	Select **Kafka**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the Kafka cluster. In this example, enter the public IP address.
	Port Number	Enter the service port number of the Kafka cluster. The default port number is 9092.
	Database Account	Enter the username that is used to log on to the Kafka cluster. If no authentication is enabled for the Kafka cluster, you do not need to enter the username.
	Database Password	Enter the password of the username. If no authentication is enabled for the Kafka cluster, you do not need to enter the password.
	Topic	Click **Get Topic List**, and select a topic name from the drop-down list.
	Kafka Version	Select the version of the destination Kafka cluster.
	Encryption	Select **Non-encrypted** or **SCRAM-SHA-256** based on your business and security requirements.

5. In the lower-right corner of the page, click **Set Whitelist and Next**.
6. Select the migration types and the objects to be migrated.

Setting	Description
Select the migration types	Select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Notice** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.
Select the objects to be migrated	Select one or more tables from the **Available** section and click the icon to move the tables to the **Selected** section. **Note** DTS automatically maps the table names to the topic name that you selected in Step 4. For information about how to change the topic name, see Object name mapping.

7. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

8. After the task passes the precheck, click **Next**.

   Note
   - If **Incremental Data Migration** is not selected, wait until the migration task automatically stops.
   - If **Incremental Data Migration** is selected, wait until **Incremental Data Migration** and **The migration task is not delayed** appear in the progress bar. Then, manually stop the migration task.

4.3.4.3. Migrate data from a user-created Oracle database to an AnalyticDB for PostgreSQL instance

This topic describes how to migrate data from a user-created Oracle database to an AnalyticDB for PostgreSQL instance by using Data Transmission Service (DTS). The data migration feature allows you to transfer and analyze data with ease.

Prerequisites

- The version of the user-created Oracle database is 9i, 10g, 11g, 12c, 18c, or 19c.
- The user-created Oracle database is running in ARCHIVELOG mode. Archived log files are accessible and a suitable retention period is set for archived log files. For more information, see Managing Archived Redo Log Files.

- Supplemental logging, including SUPPLEMENTAL_LOG_DATA_PK and SUPPLEMENTAL_LOG_DATA_UI, is enabled for the user-created Oracle database. For more information, see Supplemental Logging.

Precautions

During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.

Migration types

Migration type	Description
Schema migration	DTS migrates the schemas of the required objects from the source database to the destination database. DTS supports schema migration for the following types of objects: table, index, constraint, function, sequence, and view. **Warning** • Oracle and AnalyticDB for PostgreSQL are heterogeneous databases. DTS does not ensure that the schemas of the source and destination databases are consistent after schema migration. We recommend that you evaluate the impact of data type conversion on your business. For more information, see Data type mappings between heterogeneous databases. • For partition tables, DTS discards the partition definitions. You must define partitions in the destination database.
Full data migration	DTS migrates historical data of the required objects from the source database to the destination database. **Note** During schema migration and full data migration, do not perform data definition language (DDL) operations on the required objects. Otherwise, the objects may fail to be migrated.
Incremental data migration	After full data migration is complete, DTS retrieves redo log files from the source Oracle database. Then, DTS migrates incremental data from the source Oracle database to the destination database in real time. DTS can synchronize the following SQL operations during incremental data migration: • Data manipulation language (DML) operations: INSERT, UPDATE, and DELETE • DDL operations: CREATE TABLE, ALTER TABLE, DROP TABLE, and TRUNCATE TABLE Incremental data migration allows you to ensure service continuity when you migrate data from an Oracle database to an AnalyticDB for PostgreSQL instance.

Procedure

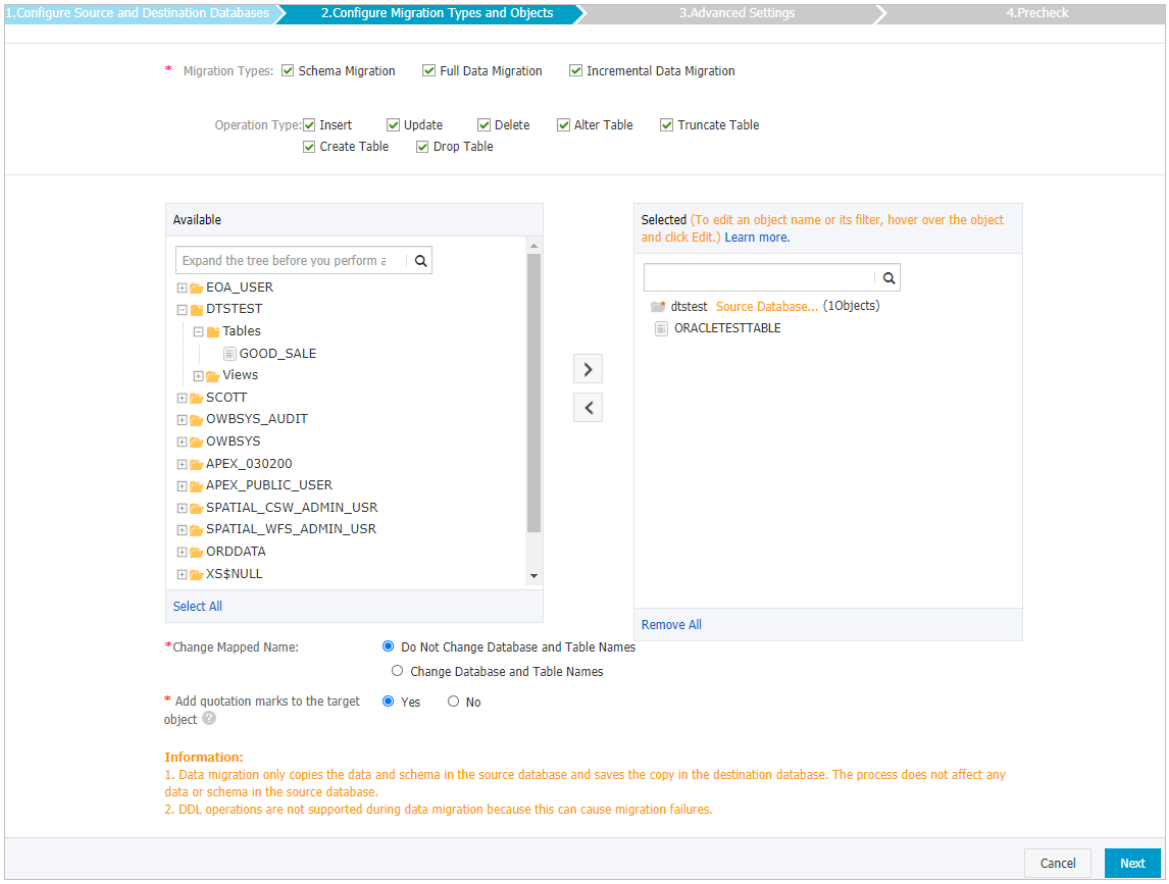
1. Create a data migration instance.

2. In the migration task list, find the migration task that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	Select the region where the source database resides.
	Database Type	Select **Oracle**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the source database.
	Port Number	Enter the service port number of the source database. The default port number is **1521**.
	Instance Type	◦ **Non-RAC Instance**: If you select this option, you must specify the **SID**. ◦ **RAC or PDB Instance**: If you select this option, you must specify the **Service Name**. In this example, select **Non-RAC Instance**.
	SID	Enter the system ID (SID) of the source database.
	Database Account	Enter the account of the source database. The account must have the database administrator (DBA) permission.
	Database Password	Enter the password of the source database account. **Note** After you specify the source database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the source database parameters based on the check results.
	Instance Type	Select **AnalyticDB for PostgreSQL**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	Instance ID	Select the ID of the destination instance.
	Database Name	Enter the name of the destination database.

Section	Parameter	Description
Destination Database	Database Account	Enter the database account of the destination instance. The account must have the read and write permissions on the destination database.
	Database Password	Enter the password of the destination database account. **Note** After you specify the destination database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the destination database parameters based on the check results.

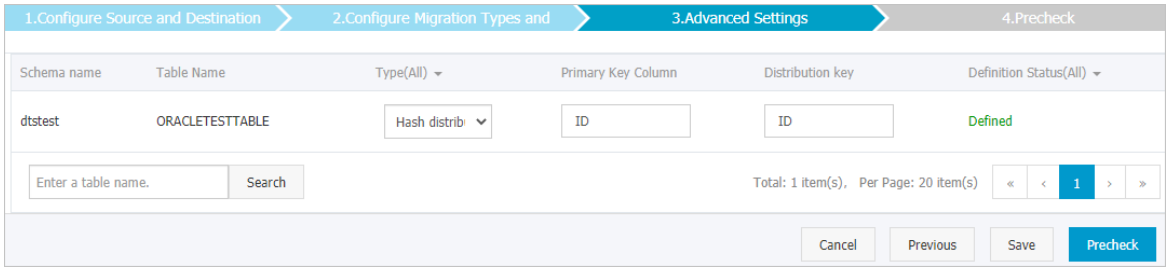
4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the migration types, the operation types, and the objects to be migrated.



Setting	Description

Setting	Description
Select the migration types	◦ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ◦ To ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.
Select the operation types	Select the types of operations that you want to synchronize during incremental data migration. All operation types are selected by default.
Select the objects to be migrated	Select objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** ◦ You can select columns, tables, or schemas as the objects to be migrated. ◦ After an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. ◦ If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.
Enclose object names in quotation marks	Specify whether you need to enclose object names in quotation marks. If you select **Yes** and the following conditions are met, DTS encloses object names in single or double quotation marks during schema migration and incremental data migration. ◦ The business environment of the source database is case-sensitive but the database name contains both uppercase and lowercase letters. ◦ A source table name does not start with a letter and contains characters other than letters, digits, and special characters. **Note** A source table name can contain only the following special characters: underscores (_), number signs (#), and dollar signs ($). ◦ The names of the schemas, tables, or columns that you want to migrate are keywords, reserved keywords, or invalid characters in the destination database.

6. Click **Next**.
7. Specify the primary key column and distribution key of the table that you want to migrate to the destination instance.



> **Note** If DTS identifies tables without primary keys, the option **Set Primary Keys and Distribution Keys of All Tables Without Primary Keys to ROWID** is displayed on the preceding page. If you select this option, DTS adds the ROWID field as the primary key and distribution key to the destination table.

8. In the lower-right corner of the page, click **Precheck**.

> **Note** You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.
>
> Troubleshoot the issues based on the causes and run a precheck again.

9. After the task passes the precheck, click **Next**.

> **Note**
>
> - If **Incremental Data Migration** is not selected, wait until the migration task automatically stops.
> - If **Incremental Data Migration** is selected, wait until **Incremental Data Migration** and **The migration task is not delayed** appear in the progress bar. Then, manually stop the migration task.

4.3.4.4. Migrate data between user-created PostgreSQL databases

This topic describes how to migrate data between user-created PostgreSQL databases by using Data Transmission Service (DTS). DTS supports schema migration, full data migration, and incremental data migration. You can select all of the supported migration types to ensure service continuity.

Prerequisites

The version of the source PostgreSQL database is 9.4, 9.5, 9.6, or 10.x.

Precautions

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- If you select a schema as the object to be migrated and create a table in the schema or run the

RENAME command to rename the table, you must run the `ALTER TABLE schema.table REPLICA IDENTITY FULL;` command before you write data to the table.

> **Note** Replace the `schema` and `table` in the preceding sample command with the actual schema name and table name.

- The source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.

Limits

- A single data migration task can migrate data from only one database. To migrate data from multiple databases, you must create a data migration task for each database.
- The name of the source database cannot contain hyphens (-), for example, dts-testdata.
- Incremental migration does not support resumable transmission after a switchover between primary and secondary PostgreSQL databases.
- During incremental data migration, DTS synchronizes only data manipulation language (DML) operations. DML operations include INSERT, DELETE, and UPDATE.
- DTS does not check the validity of metadata such as sequences. You must manually check the validity of metadata.
- After your workloads are switched to the destination database, newly written sequences do not increment from the maximum value of the sequences in the source database. Therefore, you must query the maximum value of the sequences in the source database before you switch your workloads to the destination database. Then, you must specify the queried maximum value as the starting value of the sequences in the destination database.

Before you begin

Before you configure an incremental data migration task, install the logical flow replication plug-in in the source database. PostgreSQL version 9.6 is used in this example.

> **Note** If the version of the source database is later than 10.0, you do not need to perform the following steps.

1. On the server of the source database, run the `wget` command to download the logical flow replication plug-in based on the database version. Download the plug-in from one of the following links:
   - Plug-in for PostgreSQL version 9.4
   - Plug-in for PostgreSQL version 9.5
   - Plug-in for PostgreSQL version 9.6
   - Plug-in for PostgreSQL version 10.0
2. Decompress the installation package.

   ```
   tar xvf ali_decoding_9.6.tar
   ```

3. Copy the *ali_decoding.so* file to the *lib* directory where PostgreSQL is installed.

   ```
   cp ali_decoding.so /usr/lib/postgresql/9.6/lib/
   ```

4. Copy the *ali_decoding.control* file to the *extension* directory where PostgreSQL is installed.

```
cp ali_decoding.control /usr/share/postgresql/9.6/extension/
```

5. Log on to the source database with the account that has the superuser permission.
6. Set the value of the `max_replication_slots` parameter to an integer greater than 1. In this example, set the value to `5`.

```
ALTER SYSTEM set max_replication_slots = '5';
```

> **Note** For more information, see PostgreSQL official documentation.

7. Set the value of the `wal_level` parameter to `logical`.

```
ALTER SYSTEM SET wal_level = logical;
```

8. Set the value of the `max_wal_senders` parameter to an integer greater than 1. In this example, set the value to `5`.

```
ALTER SYSTEM SET max_wal_senders = '5';
```

> **Note** The `max_wal_senders` parameter specifies the maximum number of concurrent tasks. We recommend that you set the value of this parameter to the same as the value of the `max_replication_slots` parameter.

9. Return to the shell of the server for the source database and run the following command to restart the PostgreSQL service:

```
service postgresql restart
```

10. Log on to the source database again and run the following command to check whether the replication slot can be created:

```
SELECT * FROM pg_create_logical_replication_slot('replication_slot_test', 'ali_decoding');
```

The following message indicates that the logical flow replication plug-in is installed.

```
postgres=# SELECT * FROM pg_create_logical_replication_slot('replication_slot_test', 'ali_decoding');
       slot_name       | xlog_position
-----------------------+---------------
 replication_slot_test | 0/5A[redacted]
(1 row)
```

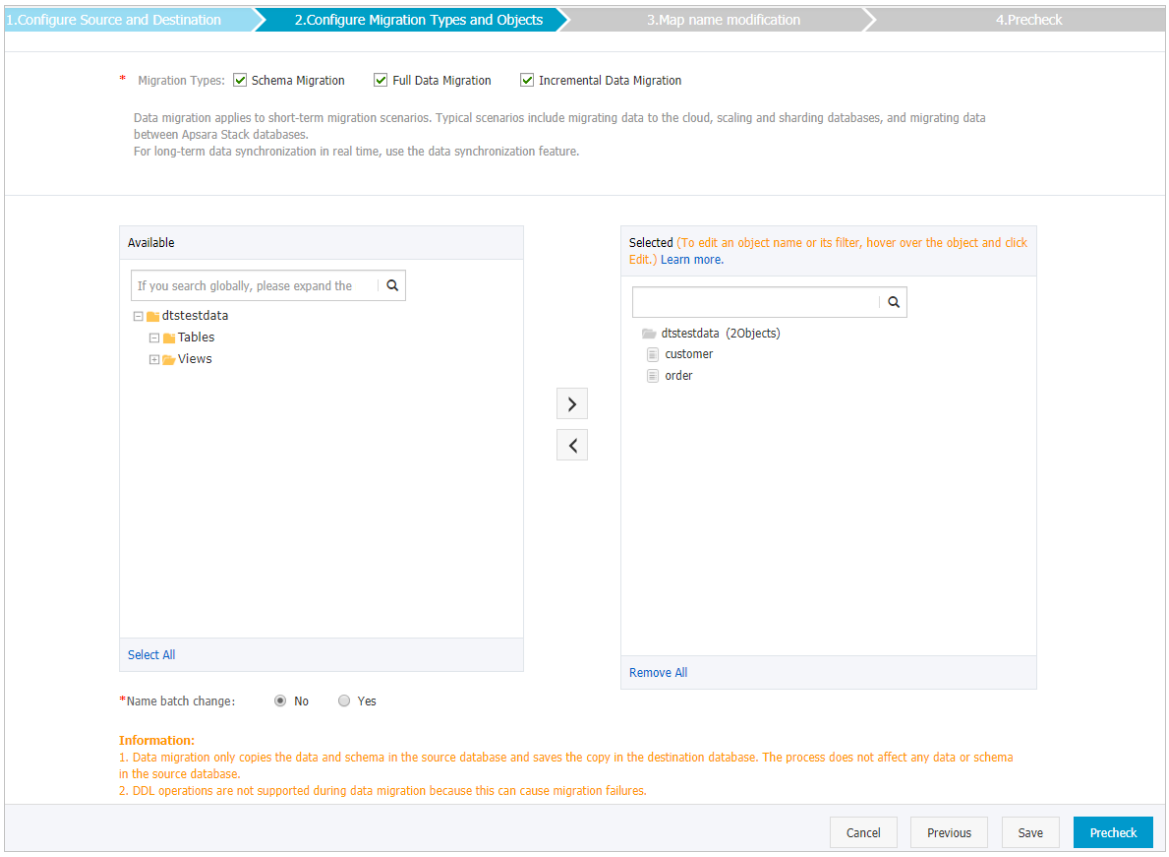
Procedure

1. Create a data migration instance.
2. In the migration task list, find the migration task that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	Select the region where the source database resides.
	Database Type	Select **PostgreSQL**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the source database.
	Port Number	Enter the service port number of the source database. The default port number is **5432**.
	Database Name	Enter the name of the source database.
	Database Account	Enter the account of the source database. The account must have the superuser permission.
	Database Password	Enter the password of the source database account. **Note** After you specify the source database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the source database parameters based on the check results.
Destination Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	Database Type	Select **PostgreSQL**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the destination database.
	Port Number	Enter the service port number of the destination database. The default port number is **5432**.
	Database Name	Enter the name of the destination database. The name can be different from the name of the source database.
	Database Account	Enter the account of the destination database. The account must have the owner permission on schemas.

Section	Parameter	Description
	Database Password	Enter the password of the destination database account. **Note** After you specify the destination database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the destination database parameters based on the check results.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the migration types and the objects to be migrated.



Setting	Description

Setting	Description
Select the migration types	○ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ○ To ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.
Select the objects to be migrated	Select objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** ○ You can select columns, tables, or databases as the objects to be migrated. ○ After an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. ○ If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.

6. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

7. After the task passes the precheck, click **Next**.

   Note
   - If **Incremental Data Migration** is not selected, wait until the migration task automatically stops.
   - If **Incremental Data Migration** is selected, wait until **Incremental Data Migration** and **The migration task is not delayed** appear in the progress bar. Then, manually stop the migration task.

4.3.4.5. Migrate data between user-created MongoDB databases

This topic describes how to migrate data between user-created MongoDB databases by using Data Transmission Service (DTS).

Prerequisites

The version of the source MongoDB database is 3.0, 3.2, 3.4, 3.6, or 4.0.

Precautions

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- If you need to migrate incremental data from a standalone MongoDB database, you must enable oplog. A standalone MongoDB database contains only a primary node. For more information, see Preparation for a standalone MongoDB database.
- If the source database uses the sharded cluster architecture, you must configure a data migration task for each shard.

Migration types

Migration type	Description
Full data migration	DTS migrates historical data of the required objects from the source database to the destination database. **Note** The following types of objects are supported: database, collection, and index.
Incremental data migration	After full data migration is complete, DTS migrates incremental data from the source database to the destination database in real time. **Note** • The create and delete operations that are performed on databases, collections, and indexes can be migrated. • The create, delete, and update operations that are performed on documents can be migrated.

Preparation for a standalone MongoDB database

If you need to migrate incremental data from a standalone MongoDB database, you must enable oplog. A standalone MongoDB database contains only a primary node. If you do not need to perform incremental data migration, skip the following steps.

Note To enable oplog, you must restart the MongoDB service. We recommend that you enable oplog during off-peak hours.

1. Use Mongo Shell to connect to the source database.
2. Run the following commands to shut down the MongoDB service:

```
use admin
db.shutdownServer()
```

3. Run the following command to start the MongoDB service from the backend as a replica set:

```
mongod --port 27017 --dbpath /var/lib/mongodb --logpath /var/log/mongodb/mongod.log --replSet rs0
--bind_ip 0.0.0.0 --auth --fork
```

> **Note**
>
> - The database path used by the preceding command is */var/lib/mongodb*. The log file path is */var/log/mongodb/mongod.log*. You must specify the paths based on your needs.
> - The command uses `0.0.0.0` as the associated IP address of the MongoDB service. This allows you to access the database by using all IP addresses. After the migration is complete, run the `kill` command to end the process, and start the MongoDB service by using the original configuration file.
> - The command enables authentication. You can access the database only after you pass the authentication.

4. Use Mongo Shell to connect to the source database again.
5. Run the following commands to initialize the replica set:

```
use admin
rs.initiate()
```

6. Wait until the role of the current node changes to primary, which indicates that oplog is enabled.

> **Note** You can run the `rs.printReplicationInfo()` command to view the status of oplog.

Procedure

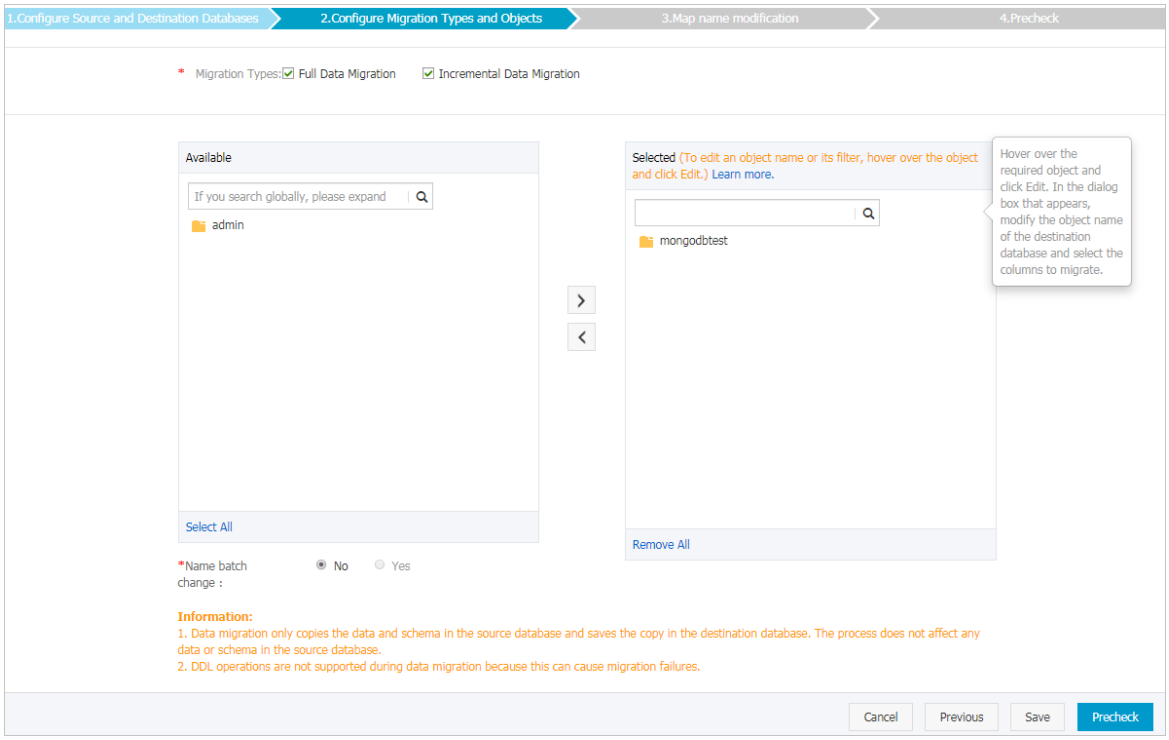
1. Create a data migration instance.
2. In the migration task list, find the migration task that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	Select the region where the source database resides.

Section	Parameter	Description
Source Database	Database Type	Select **MongoDB**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the source database.
	Port Number	Enter the service port number of the source database.
	Database Name	Enter the name of the authentication database. The database account is created in this database.
	Database Account	Enter the account of the source database. The account must have the read permissions on the source database, admin database, and local database.
	Database Password	Enter the password of the source database account. **Note** After you specify the source database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the source database parameters based on the check results.
Destination Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	Database Type	Select **MongoDB**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the destination database.
	Port Number	Enter the service port number of the destination database.
	Database Name	Enter the name of the authentication database. The database account is created in this database. **Note** If the database account is root, enter admin.
	Database Account	Enter the account of the destination database. The account must have the read and write permissions on the destination database.

Section	Parameter	Description
	Database Password	Enter the password of the destination database account. **Note** After you specify the destination database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the destination database parameters based on the check results.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the migration types and the objects to be migrated.



Setting	Description
Select the migration types	Select the migration types. For more information, see Migration types. ○ To perform only full data migration, select **Full Data Migration**. ○ To ensure service continuity during data migration, select **Full Data Migration** and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.

Setting	Description
Select the objects to be migrated	◦ Select objects from the **Available** section and click the › icon to move the objects to the **Selected** section. **Note** ■ You cannot migrate data from the admin or local database. ■ The config database is an internal database. We recommend that you do not migrate data from the config database. ◦ You can select databases, collections, or functions as the objects to be migrated. ◦ After an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping.

6. In the lower-right corner of the page, click **Precheck**.

Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

Troubleshoot the issues based on the causes and run a precheck again.

7. After the task passes the precheck, click **Next**.

Note

- If **Incremental Data Migration** is not selected, wait until the migration task automatically stops.
- If **Incremental Data Migration** is selected, wait until **Incremental Data Migration** and **The migration task is not delayed** appear in the progress bar. Then, manually stop the migration task.

4.3.4.6. Migrate data between user-created Redis databases

This topic describes how to migrate data between user-created Redis databases by using Data Transmission Service (DTS).

Prerequisites

- The version of the source Redis database is 2.8, 3.0, 3.2, or 4.0.
- The source Redis database uses the standalone architecture rather than the cluster architecture.
- The `PSYNC` or `SYNC` command can be executed on the source Redis database.

Precautions

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- If the data eviction policy (`maxmemory-policy`) of the destination database is not set to `noeviction` , data may become inconsistent between the source and destination databases. For more information, see Eviction policies.
- If you run the EVAL or EVALSHA command to call Lua scripts, DTS cannot identify whether these Lua scripts are executed on the destination database. During incremental data migration, the destination database does not explicitly return the execution results of Lua scripts.
- When you run the `PSYNC` or `SYNC` command to transmit data of the LIST type, DTS does not perform the `flush` operation on the existing data. Therefore, the destination database may contain duplicate data records.

Migration types

- Full data migration: DTS migrates historical data of the required objects from the source database to the destination database.
- Incremental data migration: DTS migrates incremental data from the source database to the destination database in real time.

Operations that can be synchronized during incremental data migration

- APPEND
- BITOP, BLPOP, BRPOP, and BRPOPLPUSH
- DECR, DECRBY, and DEL
- EVAL, EVALSHA, EXEC, EXPIRE, and EXPIREAT
- FLUSHALL and FLUSHDB
- GEOADD and GETSET
- HDEL, HINCRBY, HINCRBYFLOAT, HMSET, HSET, and HSETNX
- INCR, INCRBY, and INCRBYFLOAT
- LINSERT, LPOP, LPUSH, LPUSHX, LREM, LSET, and LTRIM
- MOVE, MSET, MSETNX, and MULTI
- PERSIST, PEXPIRE, PEXPIREAT, PFADD, PFMERGE, PSETEX, and PUBLISH
- RENAME, RENAMENX, RESTORE, RPOP, RPOPLPUSH, RPUSH, and RPUSHX
- SADD, SDIFFSTORE, SELECT, SET, SETBIT, SETEX, SETNX, SETRANGE, SINTERSTORE, SMOVE, SPOP, SREM, and SUNIONSTORE
- ZADD, ZINCRBY, ZINTERSTORE, ZREM, ZREMRANGEBYLEX, ZUNIONSTORE, ZREMRANGEBYRANK, and ZREMRANGEBYSCORE

Before you begin

To ensure that incremental data migration tasks run as expected, we recommend that you remove the limit on the replication output buffer for the source database. This topic uses a server that runs Linux as an example.

> **Note** If you perform only full data migration, skip the following steps.

1. Use the redis-cli program to connect to the source database.

   > **Note** You can use the redis-cli program after you install the Redis client. For more information, see Redis community official website.

   ```
   redis-cli -h <host> -p <port> -a <password>
   ```

   > **Note**
   > - <host>: the endpoint that is used to connect to the source database. You can use 127.0.0.1 in this example.
   > - <port>: the service port number of the source database. The default port number is 6379.
   > - <password>: the password of the source database.

   Example:

   ```
   redis-cli -h 127.0.0.1 -p 6379 -a Test123456
   ```

2. Run the following command to remove the limit on the replication output buffer:

   ```
   config set client-output-buffer-limit 'slave 0 0 0'
   ```

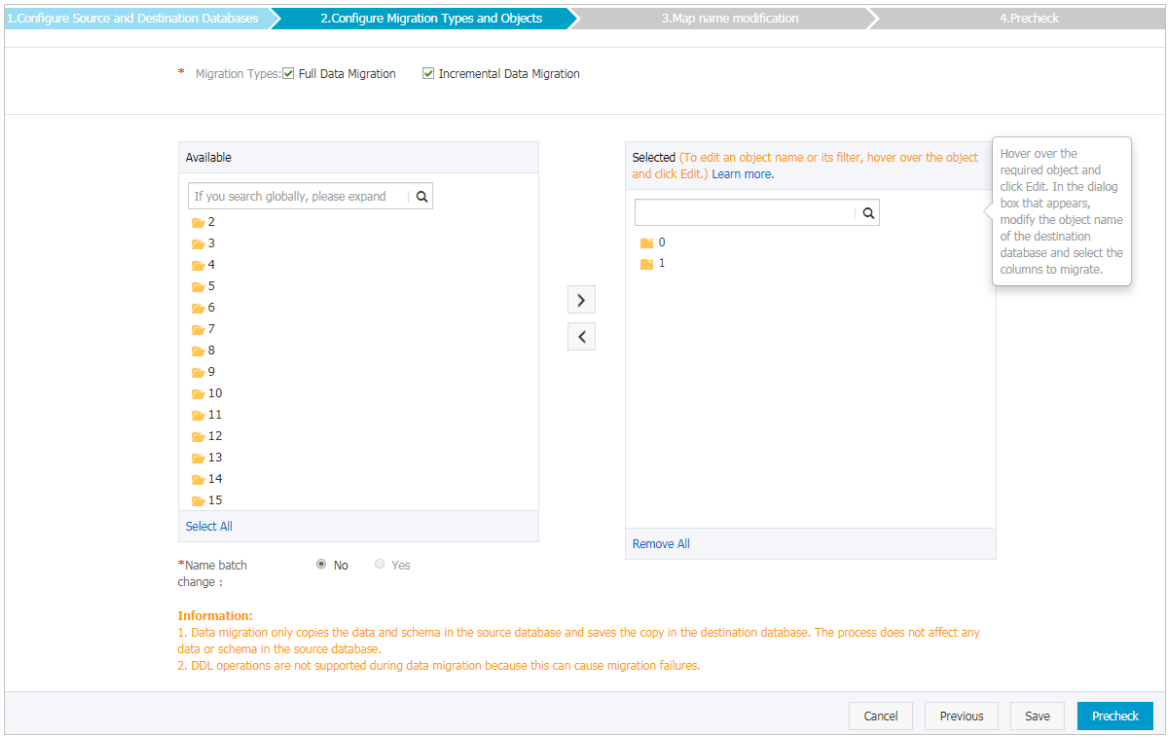
Procedure

1. Create a data migration instance.
2. In the migration task list, find the migration task that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	Select the region where the source database resides.
	Database Type	Select **Redis**.

Section	Parameter	Description
Source Database	Instance Mode	The value of this parameter is set to **Standalone** and cannot be changed to Cluster.
	Hostname or IP Address	Enter the endpoint that is used to connect to the source database.
	Port Number	Enter the service port number of the source database. The default port number is **6379**.
	Database Password	Enter the password of the source database. If password verification is disabled for the source database, you do not need to enter the password. **Note** After you specify the source database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the source database parameters based on the check results.
Destination Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	Database Type	Select **Redis**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the destination database.
	Port Number	Enter the service port number of the destination database. The default port number is **6379**.
	Database Password	Enter the password of the destination database. If password verification is disabled for the destination database, you do not need to enter the password. **Note** After you specify the destination database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the destination database parameters based on the check results.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the migration types and the objects to be migrated.



Setting	Description
Select the migration types	◦ To perform only full data migration, select only **Full Data Migration**. ◦ To migrate data with minimal downtime, select both **Full Data Migration** and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.
Select the objects to be migrated	Select objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** You can select databases as the objects to be migrated.

6. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

7. After the task passes the precheck, click **Next**.

> Note
>
> - If **Incremental Data Migration** is not selected, wait until the migration task automatically stops.
> - If **Incremental Data Migration** is selected, wait until **Incremental Data Migration** and **The migration task is not delayed** appear in the progress bar. Then, manually stop the migration task.

4.3.5. Precheck items

4.3.5.1. Source database connectivity

DTS checks whether DTS servers can connect to the source database. DTS creates a connection to the source database by using the JDBC protocol. If the connection fails, the migration task fails to pass the connectivity check.

The potential causes of connectivity check failures are described as follows:

- The database account or password specified in a data migration task is invalid.

  Troubleshooting:

  Find a server that can connect to the source database. On the server, enter the database account and password that are specified in the data migration task to check whether the account and password are valid. If the database account or password is invalid, the following error message is displayed: `Access deny`.

  Solution:

  Log on to the DTS console, modify the database account and password, and then perform a precheck again.

- The DTS servers are disallowed to access the source database.

  Troubleshooting:

  - Find a server that can connect to the source database. On the server, enter the database account and password that are specified in the data migration task to check whether the connection is successful. Only authorized DTS servers can connect to the source database. If the CIDR block of a DTS server is not included in the whitelist of the source database, the DTS server cannot connect to the source database.
  - If the source database is a MySQL database, use a MySQL client to connect to the database and run the `SELECT HOST FROM mysql.user WHERE user='Account', password='Password';` command. If the CIDR blocks of DTS servers are not included in the whitelist of the source database, the query result of the preceding command is not %.

  Solution:

  - If the source database is a MySQL database, run the `GRANT ALL ON . TO 'Account'@''%' IDENTIFIED BY 'Password';` command to authorize the database account. Replace Account and Password in the preceding command with your database account and password. After the account is authorized, perform a precheck again.

- A firewall is configured on the source database server.

Troubleshooting: If the server where the source database resides runs Linux, run the `iptables -L` command in the shell to check whether a firewall is configured for the server. If the server where the source database resides runs Windows, find Windows Defender Firewall from the Control Panel and check whether a firewall is configured for the server.

Solution:

Disable the firewall and perform a precheck again.

- The network between DTS servers and the source database is unavailable.

  If the failure persists, you can check whether the network between DTS servers and the source database is available. In this case, we recommend that you contact Alibaba Cloud engineers by submitting a ticket.

4.3.5.2. Check the destination database connectivity

This check item checks whether the DTS server can connect to the destination database for migration. DTS creates a connection to the destination database by using the JDBC protocol. If the connection fails, the check item fails.

The destination database connectivity precheck may fail for the following reasons:

- An incorrect account or password is provided when a migration task is created.

  Diagnostics:

  On any network-ready server that can connect to the destination database, use the account and password specified for creating the migration task to connect to the destination database through client software. Check whether the connection succeeds. If an error is reported for the connection and the error message contains Access deny, the account or password is incorrect.

  Troubleshooting:

  Modify the migration task in the DTS console, correct the account and password, and perform the precheck again.

- There is no connectivity between the DTS server and destination database.

  If you check that the password and account are correct, the check item may fail because there is no connectivity between the DTS server and the destination database. In this case, contact the DTS engineers on duty.

4.3.5.3. Binary logging configurations of the source database

Before you start incremental data migration between MySQL databases, DTS checks the binary logging configurations of the source database during the precheck. This topic describes how to troubleshoot the issues that are detected during the precheck for binary logging configurations.

Whether binary logging is enabled in the source database

This item is checked only when you migrate incremental data between MySQL databases. DTS checks whether binary logging is enabled in the source database. If binary logging is disabled in the source database, the check result is Failed.

Troubleshooting: Run the `log_bin=mysql_bin` command to modify the configuration file of the source database. Restart the source database and perform a precheck again.

Binary log format of the source database

This item is checked only when you migrate incremental data between MySQL databases. DTS checks whether the binary log format is set to ROW in the source database. If the binary log format is not set to ROW in the source database, the check result is Failed.

Troubleshooting: Run the `set global binlog_format=ROW` command in the source database and perform a precheck again. We recommend that you restart the MySQL process. Otherwise, data loss may occur because sessions will continue to be written in a non-ROW mode.

Binary log files in the source database

This item is checked only when you migrate incremental data between MySQL databases. DTS checks whether specific binary log files are removed from the source database. If binary log files in the source database are incomplete, the check result is Failed.

Troubleshooting: Run the `PURGE BINARY LOGS TO 'The name of the first binary log file that is not deleted'` command in the source database and perform a precheck again.

To find the binary log files that are removed from the source database, click the info icon next to the failed item. In the **View Details** dialog box, the names of deleted binary log files are displayed.

Parameter binlog_row_image of the source database

This item is checked only when you migrate incremental data between MySQL databases. DTS checks whether the value of the binlog_row_image parameter in the source database is set to FULL. This parameter indicates whether the full image is recorded. If the full image is not recorded in binary log files of the source database, the check result is Failed.

Troubleshooting: Run the `set global binlog_row_image=FULL` command in the source database and perform a precheck again.

4.3.5.4. Foreign key constraint

DTS checks whether all the parent and child tables that have foreign key dependencies are migrated. This ensures the integrity of foreign key constraints.

If the check result is Failed, an error message is displayed indicating that the parent table on which a child table depends is not migrated.

Troubleshooting:

- Do not migrate the child tables that cause the check failure. To do this, remove these child tables from the objects to be migrated and perform a precheck again.
- Migrate the parent tables rather than the child tables. To do this, add the parent tables to the objects to be migrated and perform a precheck again.
- Delete the foreign key dependencies between the parent and child tables in the source database and perform a precheck again.

4.3.5.5. Existence of FEDERATED tables

This item is checked only when you migrate incremental data between MySQL databases. DTS checks whether the source database contains storage engines that are not supported by incremental data migration. Incremental data migration does not support the FEDERATED and MRG_MYISAM storage engines.

If the check result is Failed, an error message is displayed indicating that **the FEDERATED storage engine is used by specific tables in the source database**.

If the check result is Failed, an error message is displayed indicating that **the MRG_MYISAM storage engine is used by specific tables in the source database**.

Troubleshooting:

Remove the tables that use the FEDERATED or MRG_MYISAM storage engine from the objects to be migrated. Then, create a separate migration task to perform schema migration and full data migration for these tables.

4.3.5.6. Permissions

Source database permissions

DTS checks whether the account of the source database has the required permissions to perform data migration. For information about the permissions that are required by each type of database, see the topics about how to configure data migration tasks.

Destination database permissions

DTS checks whether the account of the destination database has the required permissions to perform data migration. For information about the permissions that are required by each type of database, see the topics about how to configure data migration tasks.

4.3.5.7. Object name conflict

DTS checks whether the destination database contains objects that have the same names as the objects to be migrated. If the destination database contains objects that have the same names as the objects to be migrated, the check result is Failed. This causes migration failure.

If the check result is Failed, an error message is displayed indicating that an object in the destination database has the same name as an object to be migrated.

Troubleshooting:

- Use the object name mapping feature to map the conflicting object name to an object with a different name in the destination database.
- In the destination database, delete or rename the object that has the same name as the object to be migrated.
- Remove the conflicting object from the objects to be migrated.

4.3.5.8. Schema existence

This check item checks whether the database to be migrated exists in the destination RDS instance. If no, DTS creates one automatically. However, under the following circumstances, the automatic database creation fails, and this check item prompts a failure:

- The database name contains characters other than lowercase letters, digits, underscores (_), and hyphens (-).

The cause of the precheck failure is that the name of the **source database** does not comply with the requirements of RDS.

Troubleshooting: On the database management page of the RDS console, create a database that complies with the requirements of RDS and grant the migration account the read and write permissions on the new database. Use the database name mapping feature provided by DTS to map the source database to the new database. Then, perform the precheck again.

- The character set of the database is not UTF8, GBK, Latin1, or UTF-8MB4.

  The cause of the precheck failure is that the character set of the **source database** does not comply with the requirements of RDS.

  Troubleshooting: On the database management page of the RDS console, create a database that complies with the requirements of RDS and grant the migration account the read and write permissions on the new database. If the new database and the database to be migrated have different names, you can use the database name mapping feature of DTS to map the database to be migrated to the new database. Then re-run the precheck.

- The migration account of the destination database has no read and write permissions on the database to be migrated.

  The cause of the precheck failure is that you are not authorized to operate on the **source database**.

  Troubleshooting: On the database management page of the RDS console, click the Account Management tab. Grant the migration account the read and write permissions on the source database. Then, perform the precheck again.

4.3.5.9. Value of server_id in the source database

This item is checked only when you migrate incremental data between MySQL databases. DTS checks whether the value of **server-id** in the source database is set to an integer greater than 1.

If the check result is Failed, run the `set global server_id='An integer greater than 1'` command in the source database and perform a precheck again.

4.3.5.10. Source database version

DTS checks whether the version of the source database is supported. The table Source database types and versions lists the source database versions that are supported by DTS.

Source database types and versions

Source database type	Supported version
MySQL	5.0, 5.1, 5.5, 5.6, and 5.7. Only 5.1, 5.5, 5.6, and 5.7 are supported for incremental data migration.

If the check result is Failed, you must upgrade or downgrade the source database to a supported version before you perform a precheck again.

4.3.6. Manage data migration tasks

4.3.6.1. Object name mapping

Data Transmission Service (DTS) provides the object name mapping feature. You can use this feature to change the names of one or more objects that are migrated to the destination instance. This topic describes how to use the object name mapping feature when you configure a data migration task.

Limits

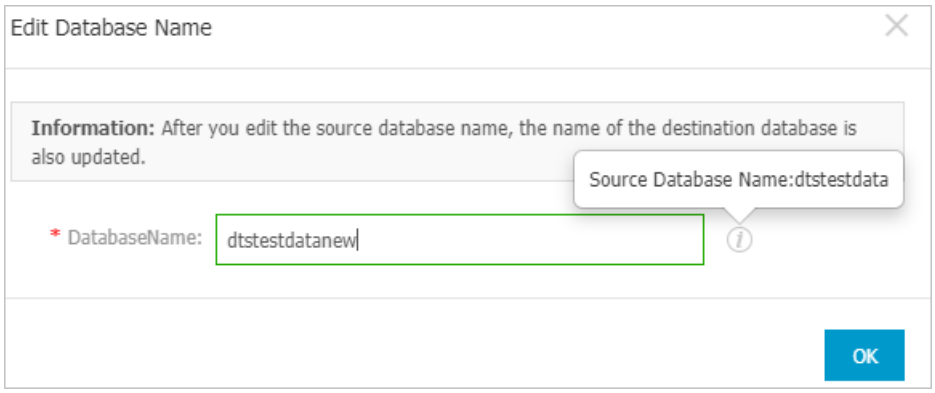
You can use the object name mapping feature only when a data migration task is configured and the current step is **Configure Migration Types and Objects**.

> **Note** Do not use the object name mapping feature after a data migration task is started. Otherwise, data may fail to be migrated.

Procedure

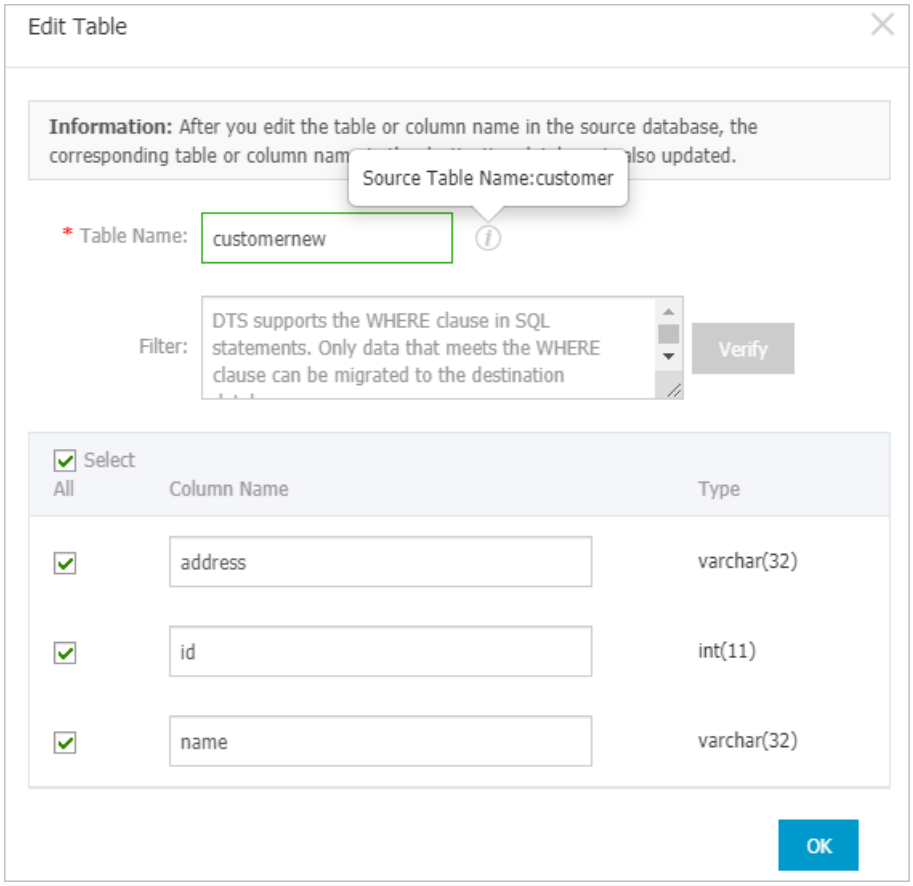
1. In the **Configure Migration Types and Objects** step, move the required objects to the **Selected** section, move the pointer over a database or table, and then click **Edit**.
2. In the dialog box that appears, specify a name for the object in the destination instance.
   - Database name mapping

     In the **Edit Database Name** dialog box that appears, enter the database name that you want to use in the destination instance.

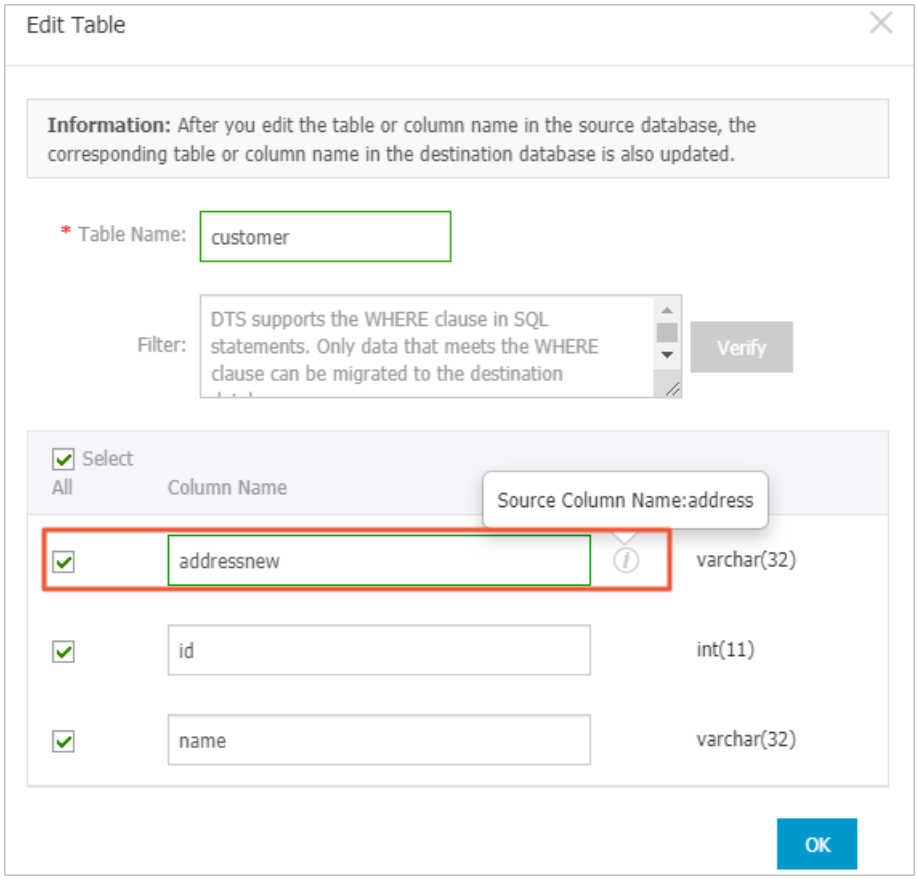
   - Table name mapping

     In the **Edit Table** dialog box that appears, enter the table name that you want to use in the destination instance.



- Column name mapping

  In the **Edit Table** dialog box that appears, enter a new name for each column.



Note In this step, you can clear the options of columns that do not need to be synchronized.

3. Click **OK**.
4. Configure other parameters that are required for the data migration task.

4.3.6.2. Specify an SQL condition to filter data

This topic describes how to specify an SQL condition to filter the data of a specific table when you configure a data migration task.

The SQL condition takes effect only within the table that you select. DTS migrates only the data that meets the SQL condition to the destination database. This feature is applicable to scenarios such as regular data migration and table partitioning.

Limits

An SQL condition applies only to full data migration. If you select **incremental data migration** as the migration type, the SQL condition does not filter incremental data.

Specify an SQL condition

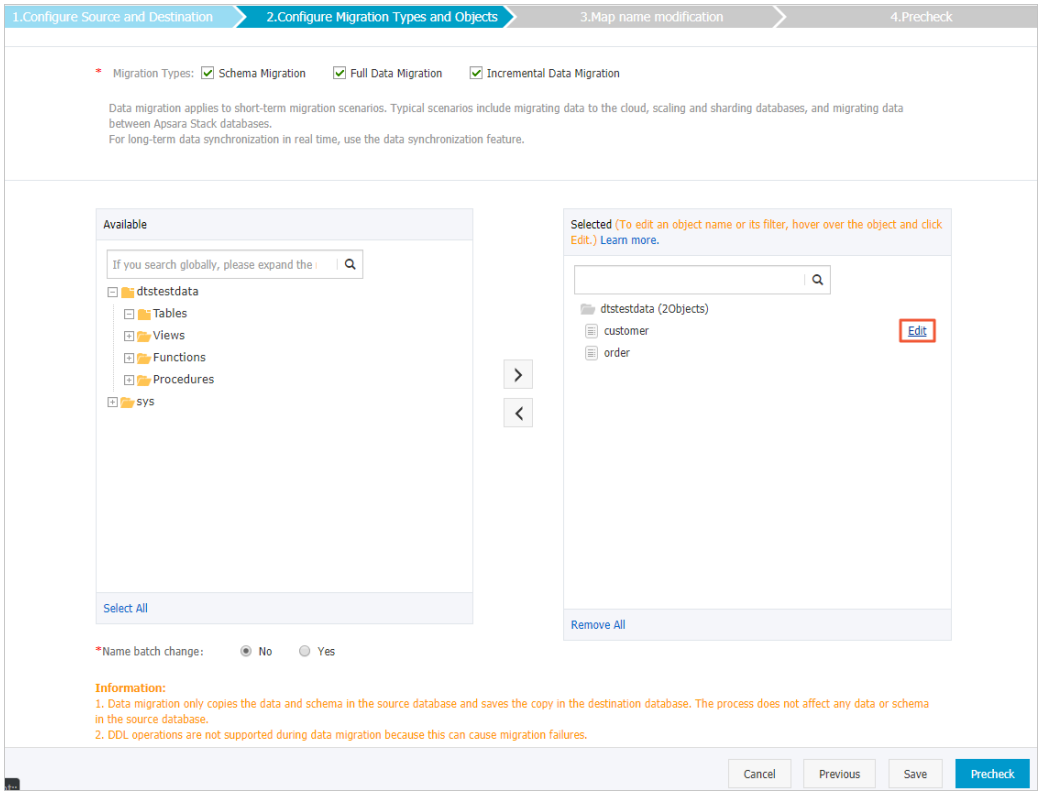
You can specify an SQL condition in the **Configure Migration Types and Objects** step when you configure a data migration task.

To filter the data of a specific table by using an SQL condition, you must select the table as the object that you want to migrate. You cannot select a database as the object. To specify an SQL condition, perform the following steps.

Procedure

1. In the **Configure Migration Types and Objects** step, move the pointer over a table in the **Selected** section. The **Edit** button appears, as shown in Edit button.

   Edit button



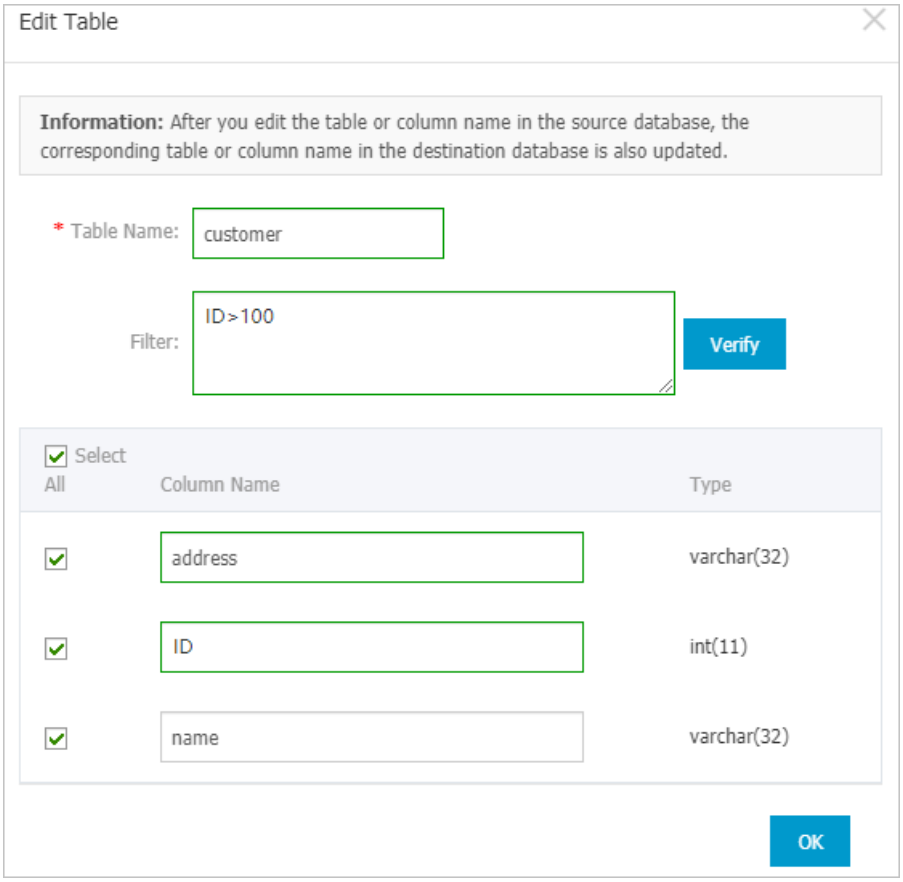
2. Click **Edit**. The Edit Table dialog box appears.

Modify an SQL condition

The SQL conditions in DTS are the same as the standard SQL WHERE conditions for databases. You can use SQL conditions to perform operations and run basic functions.

Enter an SQL condition in the text box. For example, you can enter id>1000 to migrate the records whose IDs are greater than 1,000 to the destination instance, as shown in Modify an SQL condition.

Modify an SQL condition



After the SQL condition is specified, click **OK**.

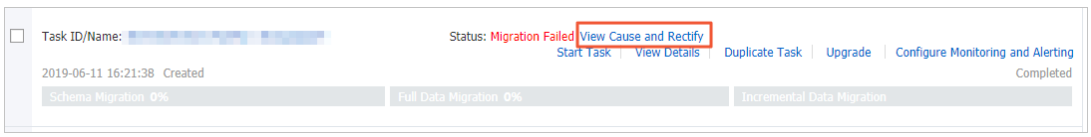
4.3.6.3. Troubleshoot a failed data migration task

This topic describes how to troubleshoot a failed data migration task. You can use this feature if your data migration task is in the **Migration Failed** state during schema migration or full data migration.

Troubleshoot a failed task during schema migration

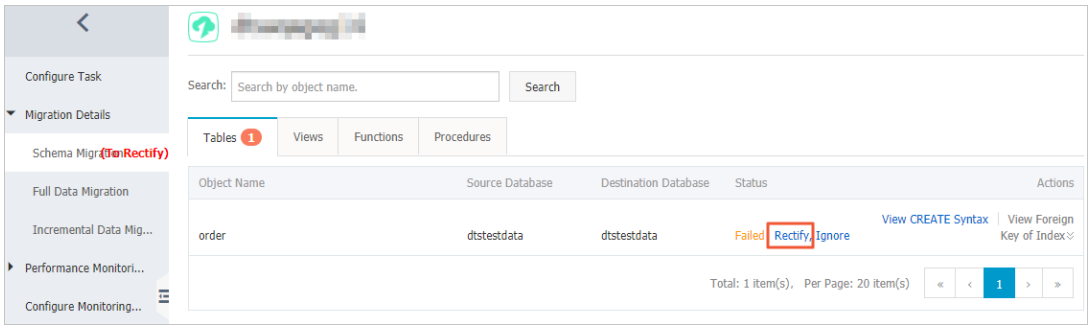
DTS supports data migration between heterogeneous data sources. However, if you migrate data of unsupported types to the destination instance during schema migration, the task may fail.

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Migration**.
3. Use one of the following methods to troubleshoot the failed task:
   - Method 1
     a. Find the target task and click **View Cause and Rectify**.



     b. Troubleshoot the issue based on the cause that is displayed in the View Cause and Rectify message. For example, you can troubleshoot an issue by modifying the schema syntax.
     c. Click **Restart Task**.

- Method 2
  a. Click the ID of the target task.
  b. In the left-side navigation pane, choose **Migration Details > Schema Migration**.
  c. On the **Schema Migration** page, find the object that causes the migration failure, and click **Rectify** in the Status column.



  d. Troubleshoot the issue based on the cause that is displayed in the **Rectify** dialog box. For example, you can troubleshoot an issue by modifying the schema syntax.
  e. Click **Rectify**.

> **Note**
> - If the failure persists, the **Rectify** dialog box does not close and shows the failure cause. You must continue troubleshooting based on the failure cause until the troubleshooting is successful.
> - If the troubleshooting is successful, the **Schema Migration** page appears and the status of the object changes to **Finished**.

4. If no objects are in the Failed state, DTS proceeds with the data migration task, for example, entering the full data migration process.

Troubleshoot a failed task during full data migration

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Migration**.
3. Find the target task and click **View Cause and Rectify**.

   DTS allows you to troubleshoot a task that fails during full data migration due to the following reasons.

> **Note** If a task fails during full data migration due to other reasons, DTS provides only the **Ignore** option. The object that causes the failure is not migrated to the destination database.

- The connection to the source or destination database failed or timed out.

  Troubleshoot the issue, make sure that the connection is successful, and then click **Restart Task**.

- The storage space of the destination instance is insufficient or the instance is locked.

  Upgrade the specification of the destination instance or clear the log space, and then click **Restart Task**.

- MyISAM tables in the source database are corrupted.

  Troubleshoot the issue in the source database, and then click **Restart Task**.

4. In the dialog box that appears, troubleshoot the issue based on the failure cause.
5. Click **Restart Task**.

4.4. Data synchronization

4.4.1. Database types, initial synchronization types, and synchronization topologies

You can use Data Transmission Service (DTS) to synchronize data between various data sources. This topic describes the database types, initial synchronization types, and synchronization topologies that are supported by DTS.

Source database	Destination database	Initial synchronization type	Synchronization topology
• User-created MySQL database Version 5.5, 5.6, 5.7, or 8.0 • RDS MySQL Version 5.6 or 5.7	User-created MySQL database Version 5.5, 5.6, 5.7, or 8.0	Initial schema synchronization Initial full data synchronization	One-way synchronization Two-way synchronization
	RDS MySQL Version 5.6 or 5.7	Initial schema synchronization Initial full data synchronization	One-way synchronization Two-way synchronization
PolarDB-X (formerly known as DRDS)	PolarDB-X	Initial full data synchronization	One-way synchronization

4.4.2. Create a data synchronization task

This topic describes how to create a data synchronization task in the DTS console.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Synchronization**.
3. In the upper-right corner of the page, click **Create Synchronization Task**.
4. In the Create DTS Instances dialog box, set parameters for the data synchronization task.

Parameter	Description
Source Instance Region	Select the region where the source instance resides.
Source Instance Type	Select the type of the source instance. ○ **MySQL**: a user-created MySQL database or an ApsaraDB RDS for MySQL instance ○ **Drds**: a Cloud Native Distributed Database PolarDB-X instance (formerly known as DRDS)
Destination Instance Region	Select the region where the destination instance resides.
Destination Instance Type	Select the type of the destination instance. ○ **MySQL**: a user-created MySQL database or an ApsaraDB RDS for MySQL instance ○ **Drds**: a Cloud Native Distributed Database PolarDB-X instance (formerly known as DRDS)
Synchronization Mode	Two-way synchronization is available only when you select **MySQL** as the type of both the source and destination instances.
Instances to Create	Set the number of data synchronization tasks that you want to create. The default value is 1.

Note In the Create DTS Instances dialog box, you can view the total number of instances, the number of existing instances, and the number of instances that can be created.

5. Click **Create**.

4.4.3. Configure data synchronization tasks

4.4.3.1. Configure one-way synchronization between ApsaraDB RDS for MySQL instances

This topic describes how to configure one-way data synchronization between ApsaraDB RDS for MySQL instances.

Prerequisites

The source and destination ApsaraDB RDS for MySQL instances are created.

Precautions

- DTS uses the read and write resources of the source and destination databases during initial full data synchronization. This may increase the load of the database server. Before you synchronize data, evaluate the impact of data synchronization on the performance of the source and destination

databases. We recommend that you synchronize data during off-peak hours.

- If you select one or more tables (not a database) as the required objects, do not use gh-ost or pt-online-schema-change to perform data definition language (DDL) operations on the tables during data synchronization. Otherwise, data may fail to be synchronized.
- The tables to be migrated in the source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
- During initial full data synchronization, concurrent INSERT operations cause fragmentation in the tables of the destination instance. After initial full data synchronization, the tablespace of the destination instance is larger than that of the source instance.

SQL operations that can be synchronized

Operation type	SQL statements
DML	INSERT, UPDATE, DELETE, and REPLACE
DDL	• ALTER TABLE and ALTER VIEW • CREATE FUNCTION, CREATE INDEX, CREATE PROCEDURE, CREATE TABLE, and CREATE VIEW • DROP INDEX and DROP TABLE • RENAME TABLE • TRUNCATE TABLE

Limits

- Incompatibility with triggers

  If the object to be synchronized is a database and the database contains a trigger that updates a synchronized table, data inconsistency may occur. To resolve this issue, you must delete the trigger in the destination database.

- Limits on RENAME TABLE operations

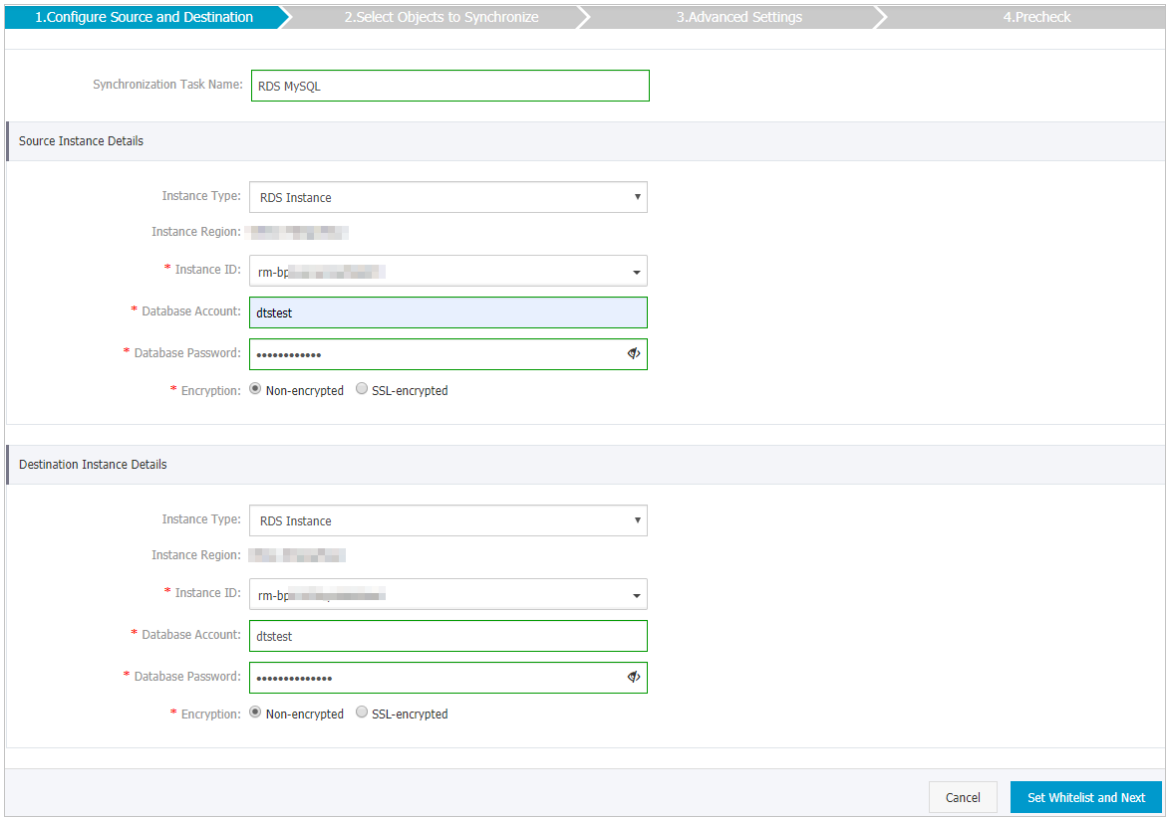
  RENAME TABLE operations may cause data inconsistency between the source and destination databases. For example, if a table to be synchronized is renamed during data synchronization, the data of this table is not synchronized to the destination database. To avoid this situation, you can select the database to which this table belongs as the object when you configure the data synchronization task.

Procedure

1. Create a data synchronization task.

   > **Note** When you create the data synchronization instance, set the type of both the source and destination instances to **MySQL** and set the synchronization topology to **One-Way Synchronization**.

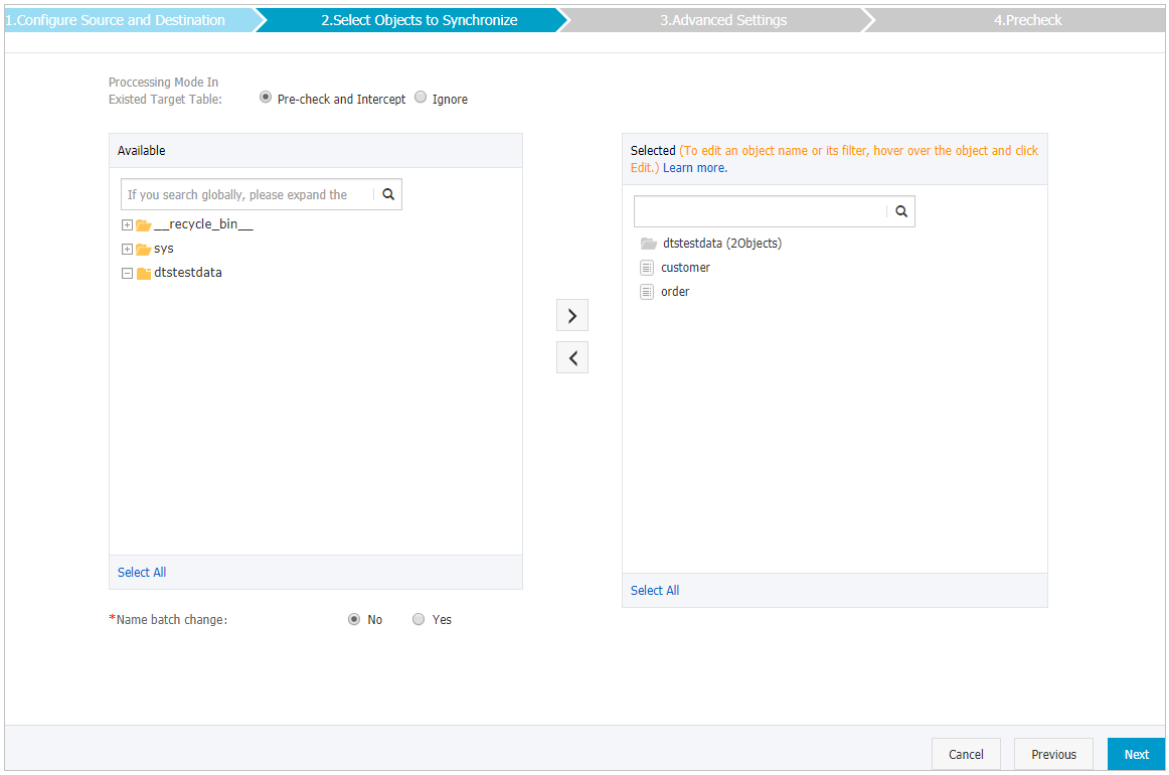
2. Find the data synchronization instance, and click **Configure Synchronization Channel** in the Actions column.
3. Configure the source and destination instances



Section	Parameter	Description
N/A	Synchronization Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Instance Details	Instance Type	Select **RDS Instance**.
	Instance Region	The region of the source instance. The region is the same as the region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	Instance ID	Select the ID of the source RDS instance.
	Database Account	Enter the database account of the source RDS instance. **Note** If the database engine of the source RDS instance is **MySQL 5.6**, you do not need to configure the **database account** or **database password**.
	Database Password	Enter the password of the source database account.
	Encryption	Select an encryption method. If you select **SSL-encrypted**, you must enable SSL encryption for the RDS instance before you configure the data synchronization task.

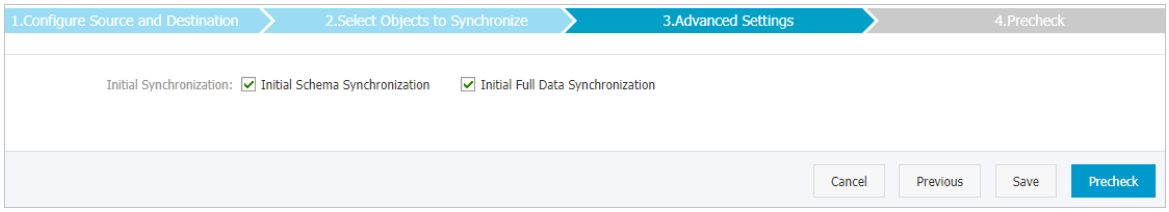
Section	Parameter	Description
Destination Instance Details	Instance Type	Select **RDS Instance**.
	Instance Region	The region of the destination instance. The region is the same as the region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	Instance ID	Select the ID of the destination RDS instance.
	Database Account	Enter the database account of the destination RDS instance. **Note** If the database engine of the destination RDS instance is **MySQL 5.6**, you do not need to configure the **database account** or **database password**.
	Database Password	Enter the password of the destination database account.
	Encryption	Select an encryption method. If you select **SSL-encrypted**, you must enable SSL encryption for the RDS instance before you configure the data synchronization task.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Configure the processing mode in existing destination tables and the objects to be synchronized.



Parameter	Description
Processing Mode In Existed Target Table	○ **Pre-check and Intercept**: checks whether the destination database contains tables that have the same names as tables in the source database. If the source and destination databases do not contain identical table names, the precheck is passed. Otherwise, an error is returned during precheck and the data synchronization task cannot be started. ○ **Ignore**: skips the precheck for identical table names in the source and destination databases. **Warning** If you select **Ignore**, data consistency is not guaranteed and your business may be exposed to potential risks. ■ DTS does not synchronize the data records that have the same primary keys as the data records in the destination database during initial data synchronization. This occurs if the source and destination databases have the same schema. However, DTS synchronizes these data records during incremental data synchronization. ■ If the source and destination databases have different schemas, initial data synchronization may fail. In this case, only specific columns are synchronized or the data synchronization task fails.
Objects	Select objects (tables or a database) from the **Available** section and click the › icon to move the objects to the **Selected** section. **Note** ○ If you select a database as the object to be synchronized, all schema changes in the database are synchronized to the destination database. ○ After an object is synchronized to the destination instance, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are synchronized to the destination instance. For more information, see Specify the name of an object in the destination instance.

6. In the lower-right corner of the page, click **Next**.
7. Configure initial synchronization.



> **Note** Initial synchronization includes initial schema synchronization and initial full data synchronization. If you select both **Initial Schema Synchronization** and **Initial Full Data Synchronization**, DTS synchronizes the schemas and historical data of the required objects before DTS synchronizes incremental data.

8. In the lower-right corner of the page, click **Precheck**.

> **Note** You can start the data synchronization task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details. Troubleshoot the issues based on the causes and run a precheck again.

9. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.** Then, DTS performs initial synchronization.

4.4.3.2. Configure two-way data synchronization between ApsaraDB RDS for MySQL instances

Data Transmission Service (DTS) supports two-way data synchronization between MySQL databases. This feature is applicable to scenarios such as disaster recovery and data backup. This topic describes how to configure two-way data synchronization between ApsaraDB RDS for MySQL instances.

Prerequisites

The source and destination ApsaraDB RDS for MySQL instances are created.

Limits

- DTS uses the read and write resources of the source and destination databases during initial full data synchronization. This may increase the load of the database server. Before you synchronize data, evaluate the impact of data synchronization on the performance of the source and destination databases. We recommend that you synchronize data during off-peak hours.
- If you select one or more tables (not a database) as the required objects, do not use gh-ost or pt-online-schema-change to perform data definition language (DDL) operations on the tables during data synchronization. Otherwise, data may fail to be synchronized.
- Incompatibility with triggers

  If the object to be synchronized is a database and the database contains a trigger that updates a synchronized table, data inconsistency may occur. To resolve this issue, you must delete the trigger in the destination database.

- Limits on RENAME TABLE operations

  RENAME TABLE operations may cause data inconsistency between the source and destination databases. For example, if a table to be synchronized is renamed during data synchronization, the data of this table is not synchronized to the destination database. To avoid this situation, you can select the database to which this table belongs as the object when you configure the data synchronization task.

- Limits on DDL synchronization direction

To ensure the stability of a two-way synchronization channel, you can synchronize DDL updates of a single table only in one direction. If DDL synchronization in a direction is configured, DDL synchronization in the opposite direction is not supported. Only data manipulation language (DML) operations can be synchronized in the opposite direction.

SQL operations that can be synchronized

Operation type	SQL statements
DML	INSERT, UPDATE, DELETE, and REPLACE
DDL	• ALTER TABLE and ALTER VIEW • CREATE FUNCTION, CREATE INDEX, CREATE PROCEDURE, CREATE TABLE, and CREATE VIEW • DROP INDEX and DROP TABLE • RENAME TABLE • TRUNCATE TABLE

Conflict detection

To ensure data consistency, make sure that data records with the same primary key, business primary key, or unique key are updated only on one of the synchronization nodes. If data records are updated on both nodes, DTS responds to conflicts based on the conflict resolution policy that you have specified for the data synchronization task.

DTS checks and fixes conflicts to maximize the stability of two-way synchronization instances. DTS can detect the following types of conflicts:

- Uniqueness conflicts caused by INSERT operations

  INSERT operations that do not comply with the uniqueness constraint cannot be synchronized. For example, if a record with the same primary key value is inserted into the two synchronization nodes at almost the same time, one of the inserted records fails to be synchronized. The synchronization fails because a record with the same primary key value already exists in the other node.

- Inconsistent records caused by UPDATE operations
  - If the records to be updated do not exist in the destination instance, DTS converts the UPDATE operation into an INSERT operation. However, uniqueness conflicts may occur.
  - The primary keys or unique keys of the records to be inserted may conflict with those of existing records in the destination instance.

- Non-existent records to be deleted

  The records to be deleted do not exist in the destination instance. In this case, DTS ignores the DELETE operation regardless of the conflict resolution policy that you have specified.

> **Notice**
>
> - During two-way synchronization, the system time of the source and destination instances may be different. Synchronization latency may occur. For these reasons, DTS does not guarantee that the conflict detection mechanism can prevent all data conflicts. To perform two-way synchronization, make sure that data records with the same primary key, business primary key, or unique key are updated only on one of the synchronization nodes.
> - DTS provides conflict resolution policies to prevent conflicts that may occur during data synchronization. You can select a conflict resolution policy when you configure a two-way data synchronization task.

Procedure

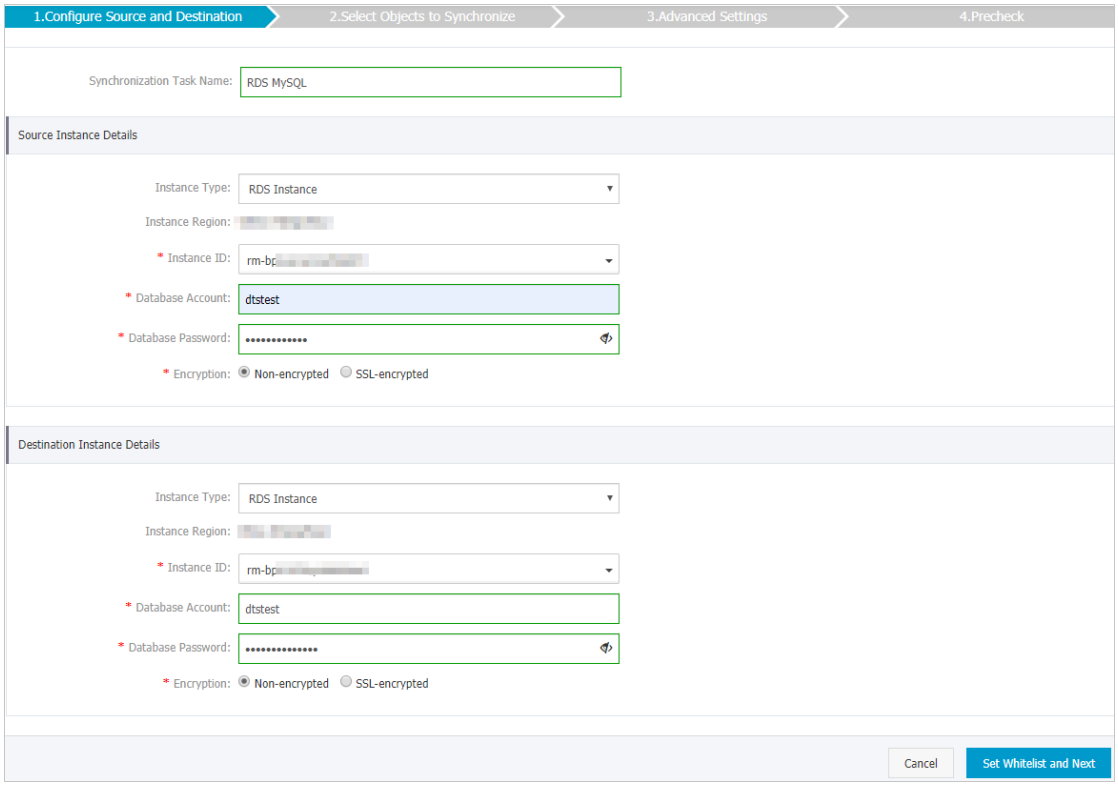
1. Create a data synchronization task.

   > **Note** When you create the data synchronization instance, set the type of both the source and destination instances to **MySQL** and set the synchronization topology to **Two-Way Synchronization**.

2. Configure the data synchronization task in one direction.
   i. Find the data synchronization instance, and click **Configure Synchronization Channel** in the Actions column of the first data synchronization task.

      > **Note** A two-way data synchronization instance contains two data synchronization tasks. You must configure a channel for each task. When you configure the second data synchronization task, find the task and click **Configure Synchronization Channel** in the **Actions** column.

   ii. Configure the source and destination instances

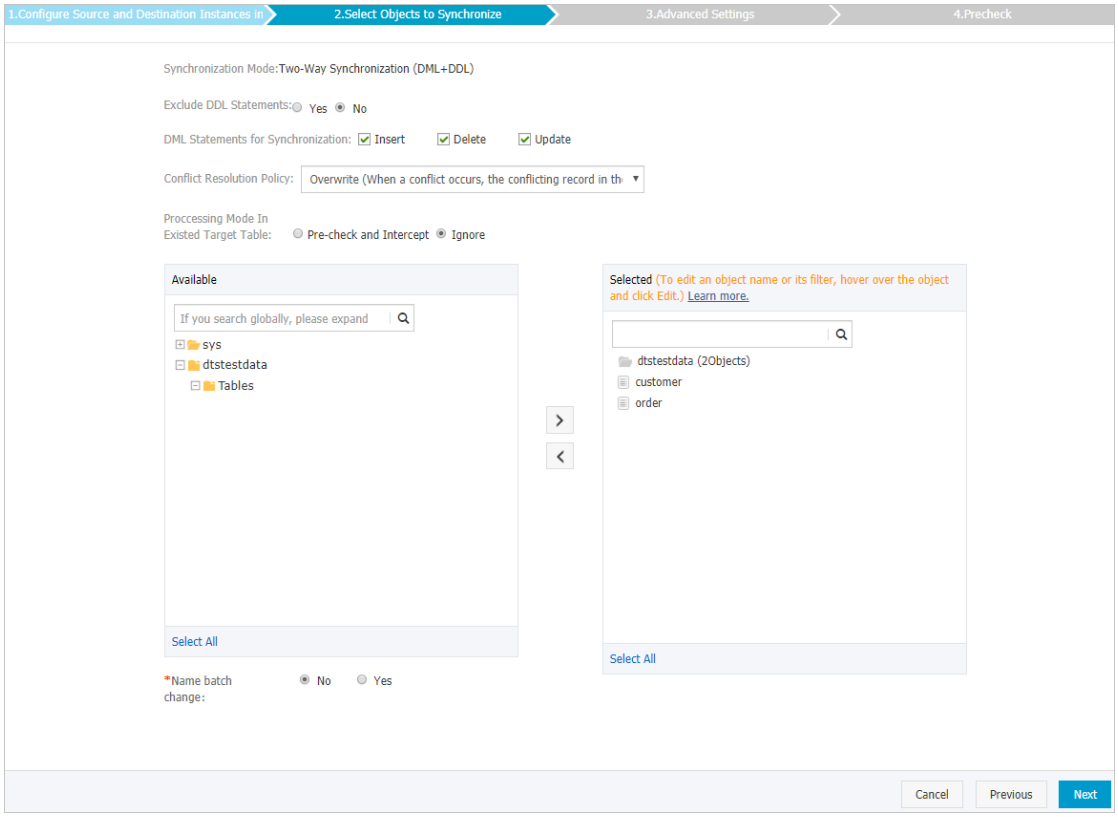


Section	Parameter	Description
N/A	Synchronization Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Instance Details	Instance Type	Select **RDS Instance**.
	Instance Region	The region of the source instance. The region is the same as the region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	Instance ID	Select the ID of the source RDS instance.
	Database Account	Enter the database account of the source RDS instance. **Note** If the database engine of the source RDS instance is **MySQL 5.6**, you do not need to configure the **database account** or **database password**.
	Database Password	Enter the password of the source database account.
	Encryption	Select an encryption method. If you select **SSL-encrypted**, you must enable SSL encryption for the RDS instance before you configure the data synchronization task.

Section	Parameter	Description
Destination Instance Details	Instance Type	Select **RDS Instance**.
	Instance Region	The region of the destination instance. The region is the same as the region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	Instance ID	Select the ID of the destination RDS instance.
	Database Account	Enter the database account of the destination RDS instance. **Note** If the database engine of the destination RDS instance is **MySQL 5.6**, you do not need to configure the **database account** or **database password**.
	Database Password	Enter the password of the destination database account.
	Encryption	Select an encryption method. If you select **SSL-encrypted**, you must enable SSL encryption for the RDS instance before you configure the data synchronization task.

iii. In the lower-right corner of the page, click **Set Whitelist and Next**.

iv. Configure the synchronization policy and objects.

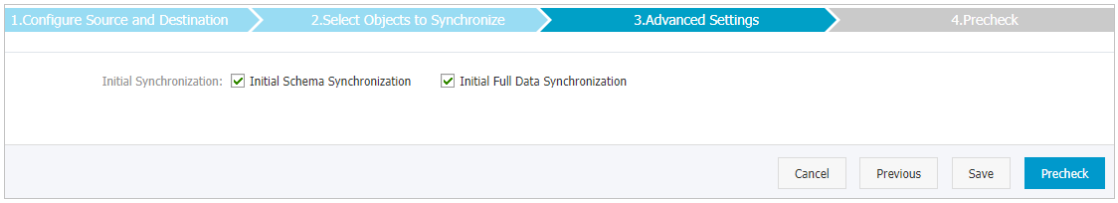


Section	Parameter	Description
Synchronization policy	Exclude DDL Statements	■ To exclude DDL operations, select **Yes**. ■ To include DDL operations, select **No**. **Note** If you select No, DTS does not synchronize the DDL operations that are performed on a table in the opposite direction.
	DML Statements for Synchronization	Select the types of DML operations that you want to synchronize. By default, the **INSERT**, **UPDATE**, and **DELETE** operations are selected. You can select the DML operation types based on your business requirements.
	Conflict Resolution Policy	Select the resolution policy for synchronization conflicts. By default, **TaskFailed** is selected. You can select a conflict resolution policy based on your business requirements. ■ **TaskFailed** The default conflict resolution policy. If a conflict occurs during data synchronization, the synchronization task reports an error and exits the process. The task enters a failed state and you must manually resolve the conflict. ■ **Ignore** If a conflict occurs during data synchronization, the synchronization task ignores the current statement and continues the process. The conflicting records in the destination database are used. ■ **Overwrite** If a conflict occurs during data synchronization, the conflicting records in the destination database are overwritten.

Section	Parameter	Description
	Processing Mode In Existed Target Table	■ **Pre-check and Intercept**: checks whether the destination database contains tables that have the same names as tables in the source database. If the source and destination databases do not contain identical table names, the precheck is passed. Otherwise, an error is returned during precheck and the data synchronization task cannot be started. ■ **Ignore**: skips the precheck for identical table names in the source and destination databases. **Warning** If you select **Ignore**, data consistency is not guaranteed and your business may be exposed to potential risks. ■ DTS does not synchronize the data records that have the same primary keys as the data records in the destination database during initial data synchronization. This occurs if the source and destination databases have the same schema. However, DTS synchronizes these data records during incremental data synchronization. ■ If the source and destination databases have different schemas, initial data synchronization may fail. In this case, only specific columns are synchronized or the data synchronization task fails.
Objects	N/A	Select objects (tables or a database) from the **Available** section and click the › icon to move the objects to the **Selected** section. ■ If you select a database as the object to be synchronized, all schema changes in the database are synchronized to the destination database. ■ After an object is synchronized to the destination instance, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are synchronized to the destination instance. For more information, see Specify the name of an object in the destination instance.

v. In the lower-right corner of the page, click **Next**.

vi. Configure initial synchronization.



During initial synchronization, DTS synchronizes the schemas and data of the required objects from the source instance to the destination instance. The schemas and data are the basis for subsequent incremental synchronization. Initial synchronization includes **initial schema synchronization** and **initial full data synchronization**. You must select both **Initial Schema Synchronization** and **Initial Full Data Synchronization** in most cases.

> **Note** If the tables to be synchronized in one direction are also included in the objects to be synchronized in the opposite direction, DTS does not synchronize these tables during initial synchronization.

vii. In the lower-right corner of the page, click **Precheck**.

> **Note** You can start the data synchronization task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details. Troubleshoot the issues based on the causes and run a precheck again.

3. Wait until the initial synchronization is complete and the data synchronization task is in the **Synchronizing** state.
4. Configure the data synchronization task in the opposite direction.
   i. Find the second data synchronization task, and click **Configure Synchronization Channel** in the Actions column.
   ii. Repeat substeps b to g that are described in step 2.

Result

Wait until both data synchronization tasks are in the **Synchronizing** state.

4.4.3.3. Synchronize data between PolarDB-X instances

PolarDB-X is formerly known as Distributed Relational Database Service (DRDS). It is compatible with the MySQL protocol and syntax, and supports automatic sharding, online smooth scaling, auto scaling, and transparent read/write splitting. This topic describes how to synchronize data between PolarDB-X instances by using Data Transmission Service (DTS).

Prerequisites

The tables that you want to synchronize contain primary keys.

Precautions

- DTS uses the read and write resources of the source and destination databases during initial full data synchronization. This may increase the load of the database server. Before you synchronize data, evaluate the impact of data synchronization on the performance of the source and destination

databases. We recommend that you synchronize data during off-peak hours.

- We recommend that you do not change the network type of the PolarDB-X instances during data synchronization.
- We recommend that you do not scale up or down the databases in the PolarDB-X instances. Otherwise, data may fail to be synchronized.

SQL operations that can be synchronized

INSERT, UPDATE, and DELETE

Before you begin

Create a database and tables in the destination instance based on the schemas of the objects in the source instance. This is because DTS does not support **initial schema synchronization** between PolarDB-X instances.

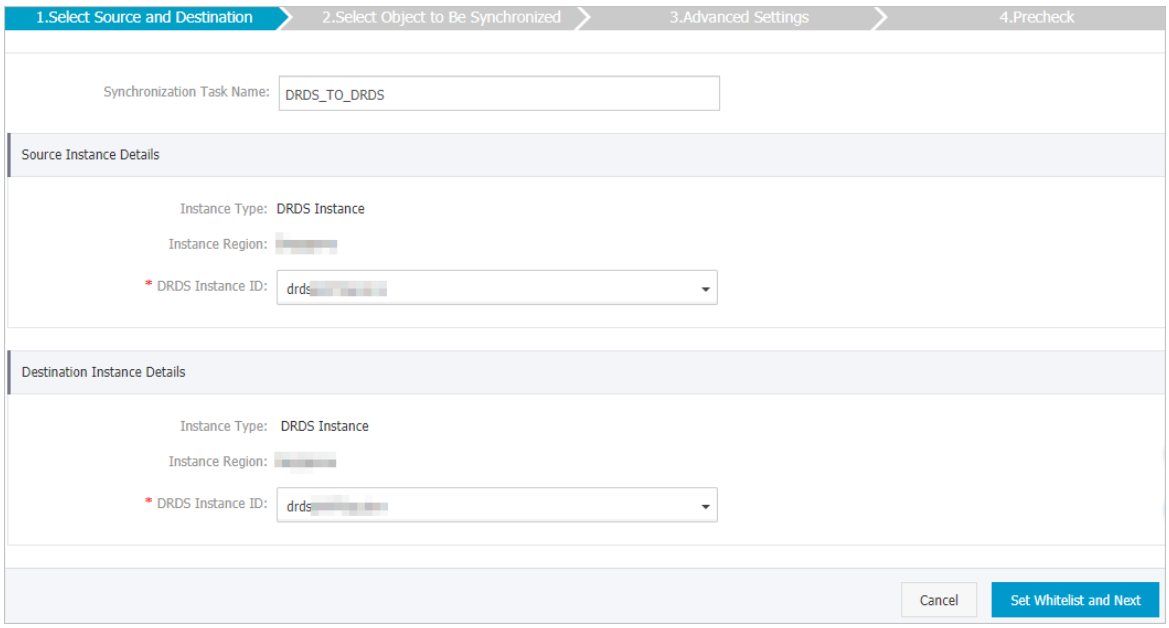
> **Note** During **initial schema synchronization**, DTS synchronizes the schemas of the required objects from the source database to the destination database.

Procedure

1. Create a data synchronization task.

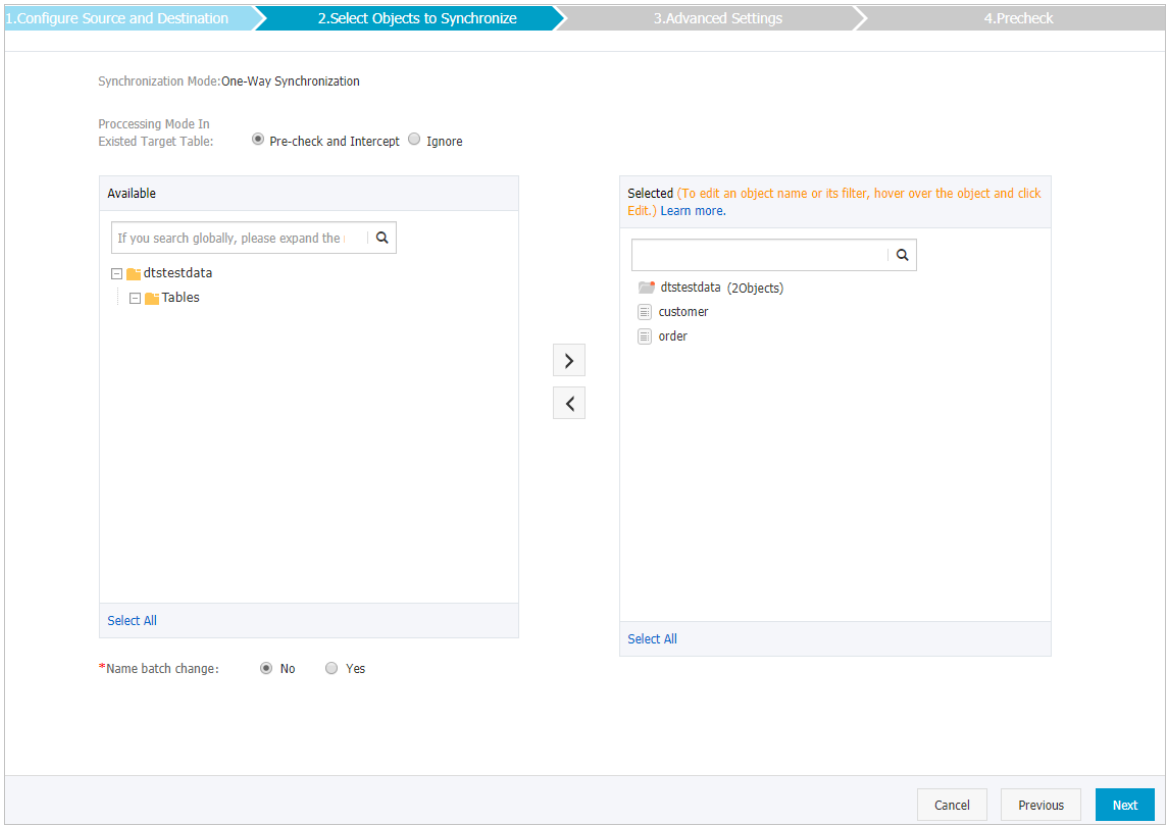
   > **Note** When you create the data synchronization instance, set both Source Instance Type and Destination Instance Type to **Drds**, and set Synchronization Mode to **One-Way Synchronization**.

2. Find the data synchronization instance, and click **Configure Synchronization Channel** in the **Actions** column.
3. Configure the source and destination instances.



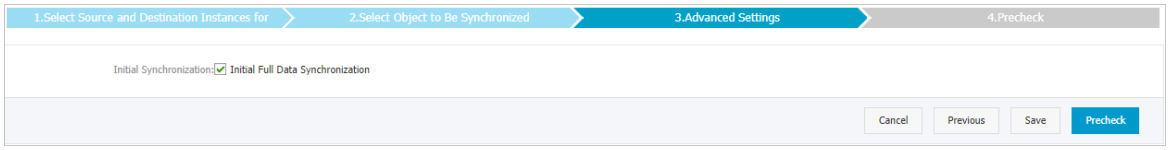
Section	Parameter	Description
N/A	Synchronization Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Instance Details	Instance Type	This parameter is set to **DRDS Instance** and cannot be changed.
	Instance Region	The region of the source instance. The region is the same as the source region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	DRDS Instance ID	Select the ID of the source PolarDB-X instance.
Destination Instance Details	Instance Type	This parameter is set to **DRDS Instance** and cannot be changed.
	Instance Region	The region of the source instance. The region is the same as the source region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	DRDS Instance ID	Select the ID of the destination PolarDB-X instance.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Configure the synchronization policy and objects.



Setting	Description
Select the processing mode of conflicting tables	○ **Pre-check and Intercept**: checks whether the destination tables are empty. If the destination tables are empty, the precheck is passed. If the tables are not empty, an error is returned during the precheck and the data synchronization task cannot be started. ○ **Ignore**: skips the check for empty destination tables. **Warning** If you select **Ignore**, data consistency is not guaranteed and your business may be exposed to potential risks. ■ If the source and destination databases have the same schema, DTS does not synchronize the data records that have the same primary keys as the data records in the destination database. ■ If the source and destination databases have different schemas, initial data synchronization may fail. In this case, only specific columns are synchronized or the data synchronization task fails.
Select the objects to be synchronized	Select tables from the **Available** section and click the icon to move the tables to the **Selected** section. **Note** ○ You can select only tables as the objects to be synchronized. ○ After an object is synchronized to the destination instance, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are synchronized to the destination instance. For more information, see Specify the name of an object in the destination instance.

6. Click **Next**.
7. Specify whether you want to perform initial full data synchronization.



Note During **initial full data synchronization**, DTS synchronizes the historical data of the required objects from the source database to the destination database. If you do not select Initial Full Data Synchronization, DTS does not synchronize the historical data.

8. In the lower-right corner of the page, click **Precheck**.

> **Note** You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.
>
> Troubleshoot the issues based on the causes and run a precheck again.

9. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.** Then, DTS performs initial synchronization.

4.4.4. Manage data synchronization instances

4.4.4.1. Specify the name of an object in the destination instance

After an object, such as a database or table, is synchronized from the source instance to the destination instance, the name of the object remains unchanged. You can use the object name mapping feature provided by DTS to specify a different name for the object in the destination instance.

Notes

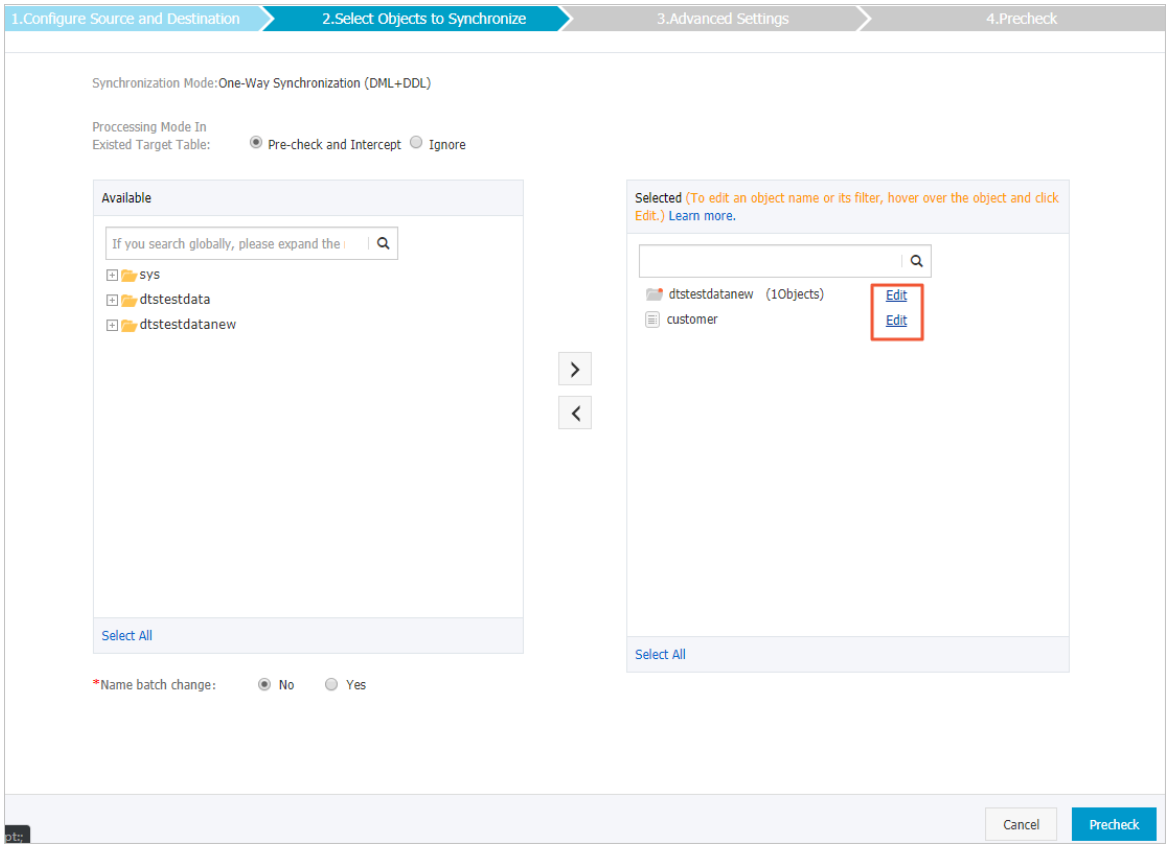
You can perform this operation only when a data synchronization task is configured and the current process is **Select Objects to Synchronize**.

> **Note** Do not perform this operation after the data synchronization task is started. Otherwise, the synchronization may fail.

Procedure

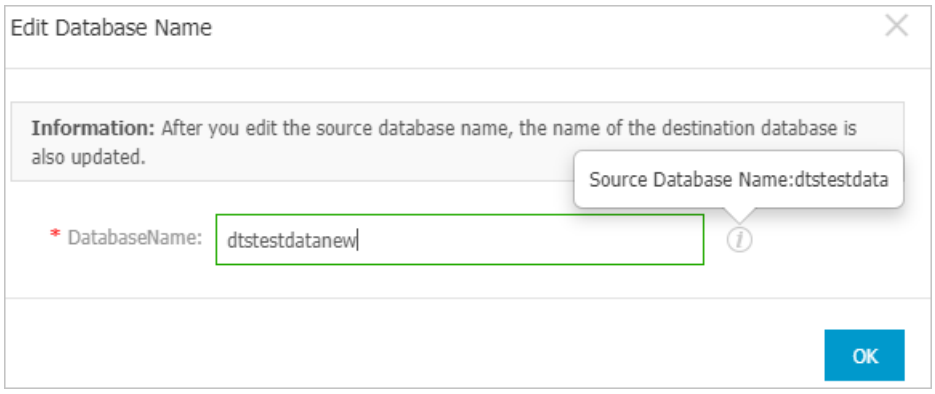
1. On the **Select Objects to Synchronize** page, move the required objects to the **Selected** section, move the pointer over a database or table, and then click **Edit**.

> **Note** Different database types support different objects. If **Edit** appears when you move the pointer over the target object, the operation is supported.



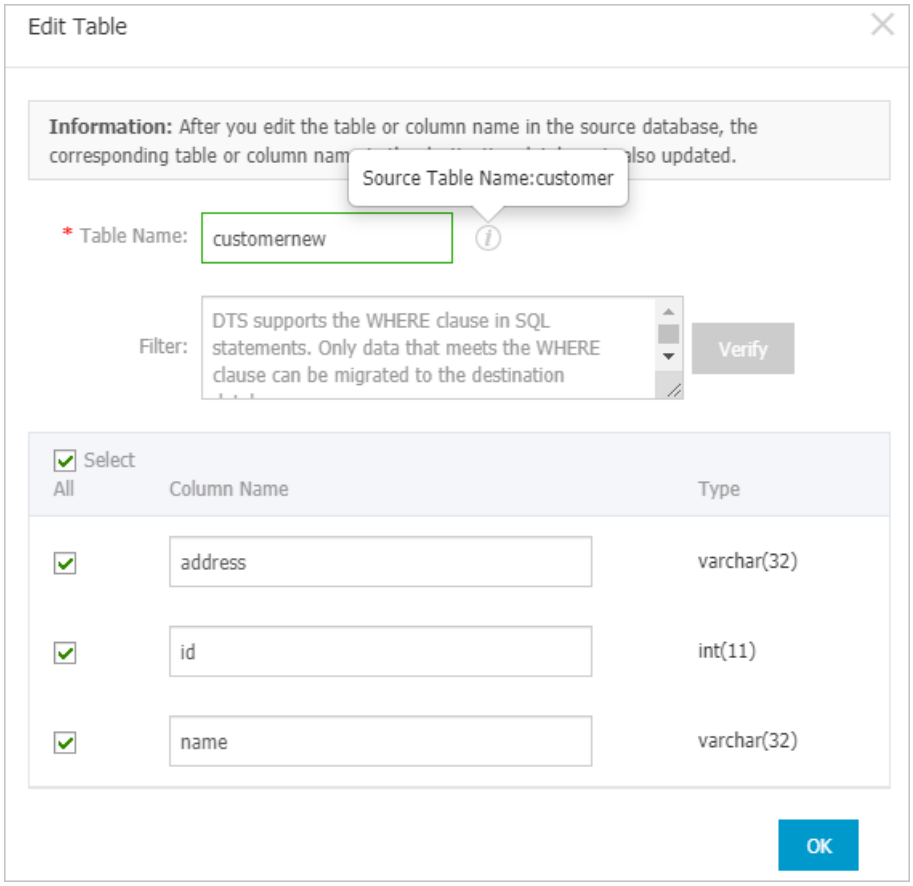
2. In the dialog box that appears, specify a name for the object in the destination instance.
   - Database name mapping

     In the **Edit Database Name** dialog box that appears, enter the database name that you want to use in the destination instance.



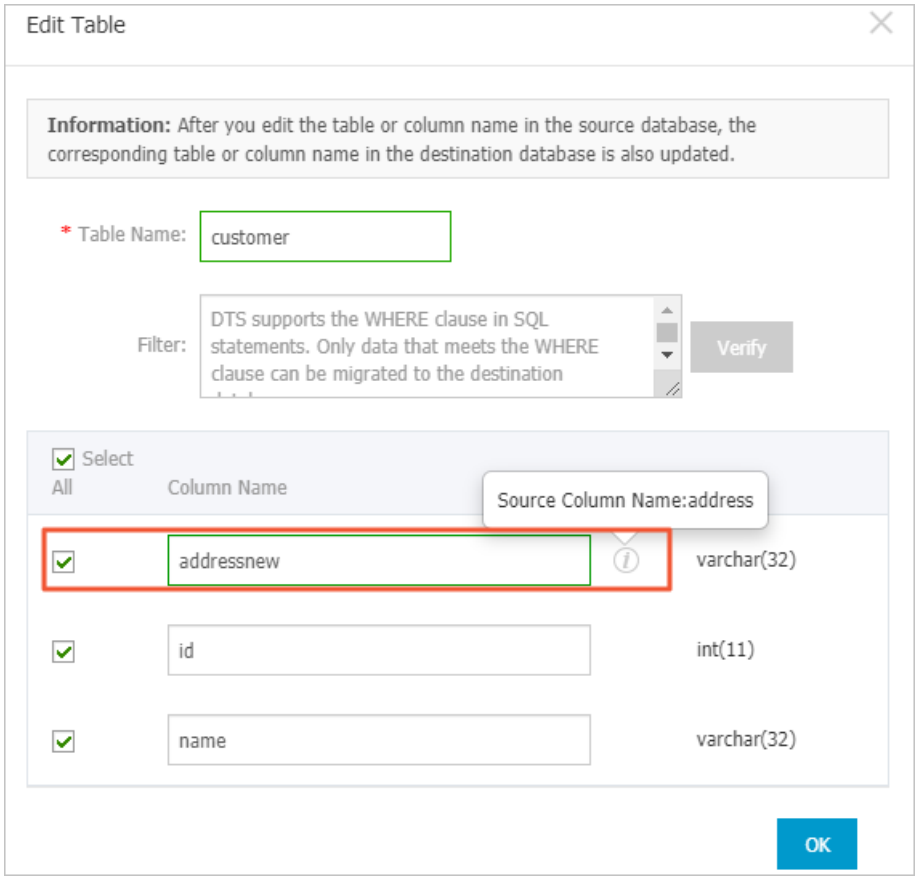
   - Table name mapping

     In the **Edit Table** dialog box that appears, enter the table name that you want to use in the destination instance.



- Column name mapping

  In the **Edit Table** dialog box that appears, enter a new name for each column.



Note In this step, you can deselect columns that do not need to be synchronized.

3. Click **OK**.
4. Configure other parameters that are required for the data synchronization task.

4.4.4.2. Check the synchronization performance

DTS provides the trend charts of data synchronization tasks based on three performance metrics: bandwidth, synchronization speed (TPS), and synchronization delay. You can view the running status of data synchronization tasks in the DTS console.

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Synchronization**.
3. On the Synchronization Tasks page, click the ID of the data synchronization task that you want to check.

   The task details page appears.
4. On the task details page, click **Synchronization Performance** in the left-side navigation pane.
5. View the trend charts of synchronization performance.

   DTS provides the trend charts of data synchronization tasks based on three performance metrics: bandwidth, synchronization speed (TPS), and synchronization delay.

   - Bandwidth: The bandwidth of data that the data writing module pulls from the data pulling module per second. Unit: MB/s.

- Synchronization speed (TPS): The number of transactions that DTS synchronizes to the destination instance per second.
- Synchronization delay: The difference between the timestamp of the latest synchronized data in the destination instance and the current timestamp in the source instance. Unit: milliseconds.

4.4.4.3. Add objects to a data synchronization task

When a data synchronization task is running, you can add objects to the task or remove objects from the task. This topic describes how to add objects to a data synchronization task in the DTS console.

Limits

You can modify the required objects only when the data synchronization task is in the **Synchronizing** or **Synchronization Failed** state.

Start time of data synchronization

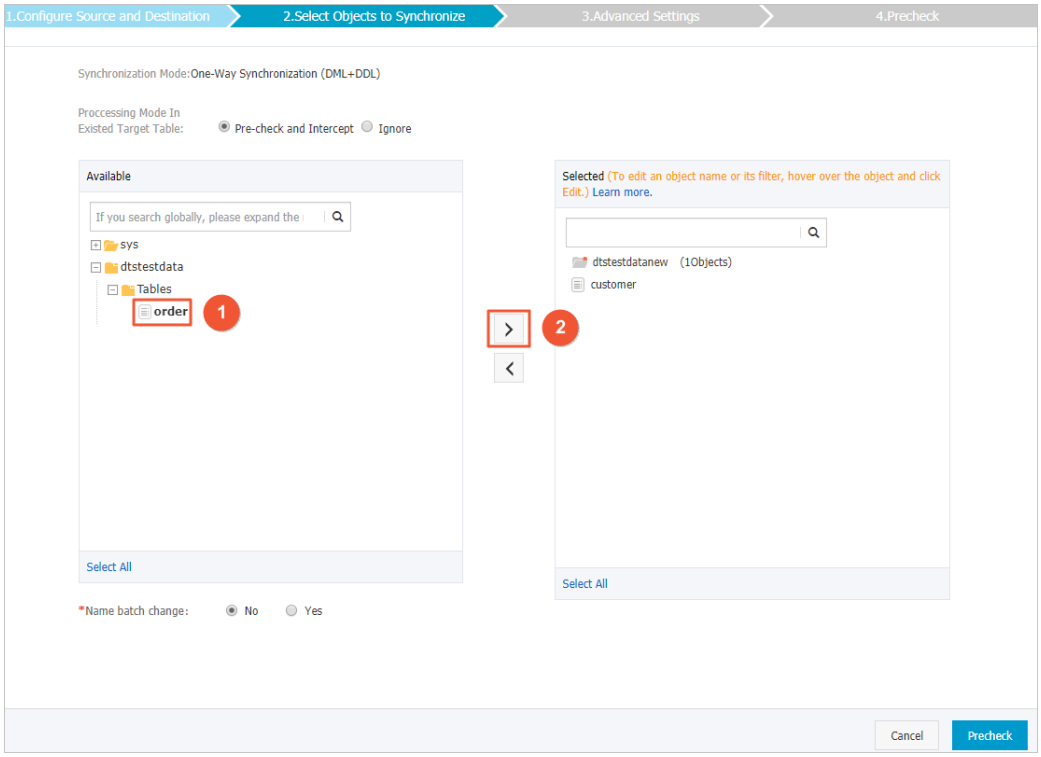
The time when DTS synchronizes data of new objects depends on whether initial synchronization is specified for the data synchronization task.

- If initial synchronization is specified, DTS synchronizes schemas and historical data, and then synchronizes incremental data.
- If initial synchronization is not specified, DTS synchronizes data after incremental data is generated on the source instance.

Procedure

1. For more information, see Log on to the DTS console.
2. In the left-side navigation pane, click **Data Synchronization**.
3. Find the data synchronization task and choose **More > Modify Objects to Synchronize** in the Actions column.
4. On the **Select Objects to Synchronize** tab, add objects based on your needs, as shown in Add objects to a data synchronization task.

   Add objects to a data synchronization task



5. Click **Precheck**.

   After the task passes the precheck, the objects are added to the data synchronization task.

   After the objects are added, if initial synchronization is specified for the data synchronization task, the task status changes from **Synchronizing** to **Synchronizing (The initial synchronization of the new objects is being performed.)**.

   > **Note** You can click **View More** to view the initial synchronization progress of the new objects. After the initial synchronization on the new objects is complete, the task status returns to **Synchronizing**.

4.4.4.4. Remove objects from a data synchronization task

When a data synchronization task is running, you can add objects to the task or remove objects from the task. This topic describes how to remove objects from a data synchronization task in the DTS console.

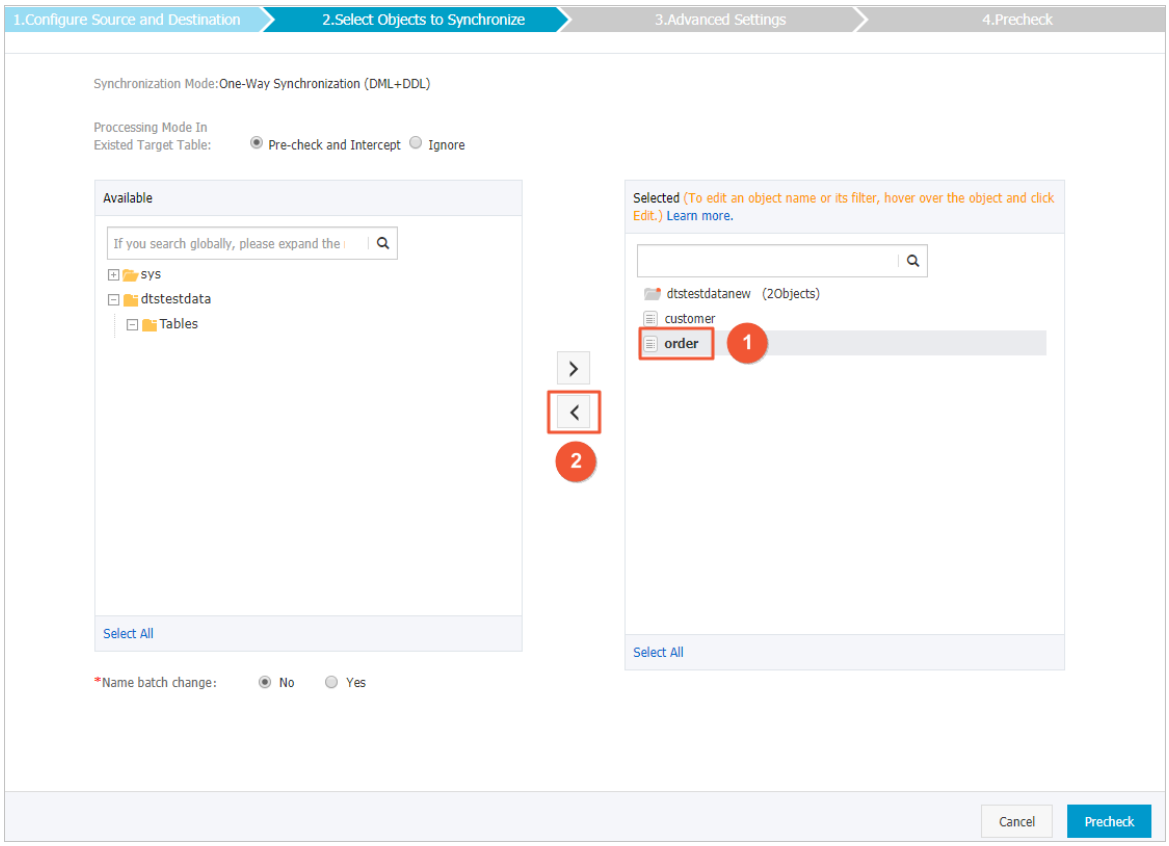
Restrictions on object modifications

You can modify the objects to be synchronized only when the synchronization task is in the **Synchronizing** or **Synchronization Failed** state.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Synchronization**.

3. Find the synchronization task to be modified, and click **View More > Modify Object to Be Synchronized** to modify the objects to be synchronized.
4. On the **Select Objects to Synchronize** tab, remove objects based on your needs.



5. Click **Precheck** to run a precheck.

4.4.4.5. Troubleshoot precheck failures

Before Data Transmission Service (DTS) runs a data synchronization task, DTS performs a precheck. This topic describes the precheck items and how to troubleshoot precheck failures.

Source database connectivity

- Description

  DTS checks whether DTS servers can connect to the source RDS instance. DTS creates a connection to the source RDS instance by using the JDBC protocol. If the connection fails, the task fails to pass the precheck.

- Cause of failure
  - DTS does not support data synchronization between RDS instances in the region where the source instance resides.
  - The database account or password of the source instance is invalid.

- Solution

  Submit a ticket and contact Alibaba Cloud technical support.

Destination database connectivity

- Description

  DTS checks whether DTS servers can connect to the destination RDS instance. DTS creates a connection to the destination RDS instance by using the JDBC protocol. If the connection fails, the task fails to pass the precheck.

- Cause of failure
  - DTS does not support data synchronization between RDS instances in the region where the destination instance resides.
  - The database account or password of the destination instance is invalid.

- Solution

  Submit a ticket and contact Alibaba Cloud technical support.

Source database version

- Description

  DTS checks whether:

  i. The database version of the source RDS instance is supported by the data synchronization feature.

  ii. The database version of the destination RDS instance is the same as the database version of the source RDS instance.

- Cause of failure
  - The database version of the source RDS instance is earlier than the supported database versions. The data synchronization feature supports the following database versions: MySQL 5.1, 5.5, 5.6, and 5.7.
  - The database version of the destination RDS instance is earlier than the database version of the source RDS instance.

- Solution
  - If the database version of the source RDS instance is earlier than the supported database versions, upgrade the source RDS instance to MySQL 5.6 or 5.7 in the RDS console. Then, create a data synchronization task again.
  - If the database version of the destination RDS instance is earlier than the database version of the source RDS instance, upgrade the destination RDS instance to MySQL 5.6 or 5.7 in the RDS console. Then, create a data synchronization task again.

Database existence

DTS checks whether the destination database already exists in the destination instance. If the destination database does not exist in the destination instance, DTS automatically creates a database. However, DTS fails to create the database and reports a failure under the following circumstances:

- The database name contains characters other than lowercase letters, digits, underscores (_), and hyphens (-).
- The character set of the database is not UTF-8, GBK, Latin1, or UTF-8MB4.
- The account of the destination database does not have the read and write permissions on the source database.

  If the data source is an RDS instance, the task passes the precheck.

Source database permissions

DTS checks whether the account of the source database has the required permissions. If the account does not have the required permissions, the task fails to pass the precheck. If the source database is an RDS instance, the task passes the precheck.

Destination database permissions

- Description

  DTS checks whether the account of the destination database has the required permissions. If the account does not have the required permissions, the task fails to pass the precheck.

- Cause of failure
  - DTS fails to create a database account in the destination RDS instance.
  - DTS fails to grant the read/write permissions to the database account of the destination RDS instance.

- Solution

  Submit a ticket and contact Alibaba Cloud technical support.

Object name conflict

- Description

  DTS checks object names only if you select initial synchronization for a data synchronization task. DTS checks whether an object that you want to synchronize has the same name as an object in the destination RDS instance.

- Cause of failure

  If an object in the destination RDS instance has the same name as the object that you want to synchronize, the task fails to pass the precheck.

- Solution
  - Remove the conflicting object from the destination database.
  - Then, create a data synchronization task again. Select both Initial Schema Synchronization and Initial Full Data Synchronization.

Value of server_id in the source database

DTS checks whether the value of the server_id parameter in the source database is set to an integer that is greater than or equal to 2. If the data source is an RDS instance, the task passes the precheck.

Whether binary logging is enabled for the source database

DTS checks whether the binary logging feature is enabled for the source database. If the binary logging feature is disabled for the source database, the task fails to pass the precheck. If the data source is an RDS instance, the task passes the precheck.

Whether the binlog format is ROW in the source database

DTS checks whether the binary log format of the source database is set to ROW. If the binary log format of the source database is not set to ROW, the task fails to pass the precheck. If the data source is an RDS instance, the task passes the precheck.

Integrity of the FOREIGN KEY constraints

- Description

  DTS checks whether the parent tables and child tables that have referential relationships with each other are all included in the required objects. The precheck allows DTS to protect the integrity of the FOREIGN KEY constraints.

- Cause of failure

  One or more child tables are included in the required objects. However, the parent tables that are referenced by the child tables are not included in the required objects. This impairs the integrity of the FOREIGN KEY constraints.

- Solution

  The following solutions are available:

  - Create a data synchronization task again and do not synchronize the child tables that fail to pass the precheck.
  - Create a data synchronization task again and add the parent tables to the required objects.
  - Remove the FOREIGN KEY constraints from the child tables that fail to pass the precheck. Then, create a data synchronization task again.

Storage engine

- Description

  DTS checks whether the required objects use the storage engines that are not supported by the data synchronization feature, such as FEDERATED, MRG_MyISAM, and TokuDB.

- Cause of failure

  If the storage engine of a source table is FEDERATED, MRG_MyISAM, or TokuDB, the task fails to pass the precheck.

- Solution

  Change the unsupported storage engine to InnoDB and create a data synchronization task again.

Character set

- Description

  DTS checks whether the required objects use the character sets that are not supported by the data synchronization feature, such as the UCS-2 character set.

- Cause of failure

  If the character sets used by the required objects are not supported by the data synchronization feature, the task fails to pass the precheck.

- Solution

  Change the unsupported character sets to UTF-8, GBK, or Latin1. Then, create a data synchronization task again.

Complicated topologies

- Description

DTS checks whether the topology that you specify for the source and destination RDS instances is supported.

- Cause of failure
  - The source RDS instance in the current task is being used as the destination instance of another task.
  - The destination RDS instance in the current task is being used as the source or destination instance of another task.
  - The objects that you want to synchronize in the current task are being synchronized by an existing task. The two tasks have the same source and destination RDS instances.
- Solution
  - If the task that you want to create has the same source and destination RDS instances as an existing task, you can add the required objects to the existing task. You do not need to create another task to synchronize these objects.
  - If the task that you want to create conflicts with an existing task, wait until the existing task is completed before you create a data synchronization task again.

Format of the MySQL database password

DTS checks whether the format of the password that is used to access the source database is no longer valid. If the data source is an RDS instance, the task passes the precheck.

4.5. Change tracking

4.5.1. Create a change tracking instance

Before you configure a task to track data changes, you must create a change tracking instance. This topic describes how to create a change tracking instance in the Data Transmission Service (DTS) console.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Change Tracking**.
3. In the upper-right corner, click **Create Change Tracking Task**.
4. In the Create DTS Instances dialog box, select a region, and enter the number of change tracking instances that you want to create.

   > **Note** In the Create DTS Instances dialog box, you can view the total number of instances, the number of existing instances, and the number of instances that can be created.

5. Click **Create**.

4.5.2. Track data changes from a user-created MySQL database

The change tracking feature allows you to track data changes in real time. It applies to scenarios such as lightweight cache updates, business decoupling, and synchronization of extract, transform, and load (ETL) operations. This topic describes how to track data changes from a user-created MySQL database.

Prerequisites

- A MySQL database of version 5.5, 5.6, 5.7, or 8.0 is created.
- If you need to perform incremental data migration, you must enable the binary logging feature. The following requirements must be met:
  - The value of the `binlog_format` parameter is set to `row`.
  - The value of the `binlog_row_image` parameter is set to `full`.

Precautions

- DTS does not track data definition language (DDL) operations that are performed by gh-ost or pt-online-schema-change. Therefore, the change tracking client may fail to write the consumed data to the destination tables due to schema conflict.
- If the source database is used in another task, for example, it is used in a running data migration task, DTS may track data changes of other objects. In this case, you must use the change tracking client to filter the tracked data.

Procedure

1. Create a change tracking instance.
2. Find the change tracking task and click **Configure Channel** in the **Actions** column.
3. Configure the source database and network type for the change tracking task.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Version	Select **New**.
	Instance Type	Select **User-Created Database with Public IP Address**.
	Database Type	The type of the source database. The type is the same as the **instance type** that you selected when you created the change tracking task. You cannot change the value of this parameter.
	Instance Region	The region of the source instance. The region is the same as the **region** that you selected when you created the change tracking task. You cannot change the value of this parameter.
	Hostname or IP Address	Enter the endpoint that is used to connect to the user-created MySQL database.

Section	Parameter	Description
	Port Number	Enter the service port number of the user-created MySQL database.
	Database Account	Enter the account of the user-created MySQL database.
	Database Password	Enter the password of the database account.
Consumer Network Type	Network Type	Select the network type of the change tracking instance. ○ **Classic** If you select **Classic**, no other configurations are required. ○ **VPC** If you select **VPC**, you must specify the **VPC** and **VSwitch**.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the data change types and objects.

Parameter	Description
Required Data Types	○ **Data Updates**: If you select Data Updates, DTS tracks data updates of the objects that you select. Data updates include the INSERT, DELETE, and UPDATE operations. ○ **Schema Updates**: If you select Schema Updates, DTS tracks the create, delete, and modify operations that are performed on all object schemas of the source instance. You must use the change tracking client to filter the tracked data. **Note** ○ If you select a database as the object, DTS tracks data changes of all objects, including new objects in the database. ○ If you select a table as the object, DTS tracks only data changes of this table. In this case, if you want to track data changes of another table, you must add the table to the required objects. For more information, see Modify objects for change tracking.
Required Objects	Select objects from the **Required Objects** section and click the › icon to move the objects to the **Selected** section. **Note** You can select tables and databases as the objects for change tracking.

6. In the lower-right corner of the page, click **Save and Precheck**.

> Note
>
> - Before you can start the change tracking task, a precheck is performed. You can start the change tracking task only after the task passes the precheck.
> - If the task fails to pass the precheck, click the icon next to each failed item to view details. Troubleshoot the issues based on the causes and run the precheck again.

7. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.**

After the change tracking task is configured, DTS performs initial change tracking, which takes about 1 minute. After the initial change tracking is complete, you can create one or more consumer groups to consume the tracked data.

What's next

Use a Kafka client to consume tracked data

4.5.3. Track data changes from a user-created Oracle database

You can use Data Transmission Service (DTS) to track data changes in real time. This feature applies to the following scenarios: lightweight cache updates, business decoupling, and synchronization of extract, transform, and load (ETL) operations. This topic describes how to track data changes from a user-created Oracle database.

Prerequisites

- The version of the user-created Oracle database is 9i, 10g, 11g, or 12c.
- Supplemental logging, including SUPPLEMENTAL_LOG_DATA_PK and SUPPLEMENTAL_LOG_DATA_UI, is enabled for the user-created Oracle database. For more information, see Supplemental Logging.
- The ARCHIVELOG mode is enabled for the user-created Oracle database. Archived log files are accessible and a suitable retention period is set for archived log files. For more information, see Managing Archived Redo Log Files.

Precautions

- DTS does not track data definition language (DDL) operations that are performed by gh-ost or pt-online-schema-change. Therefore, the change tracking client may fail to write the consumed data to the destination tables due to schema conflict.
- If the source database is used in another task, for example, it is used in a running data migration task, DTS may track data changes of other objects. In this case, you must use the change tracking client to filter the tracked data.

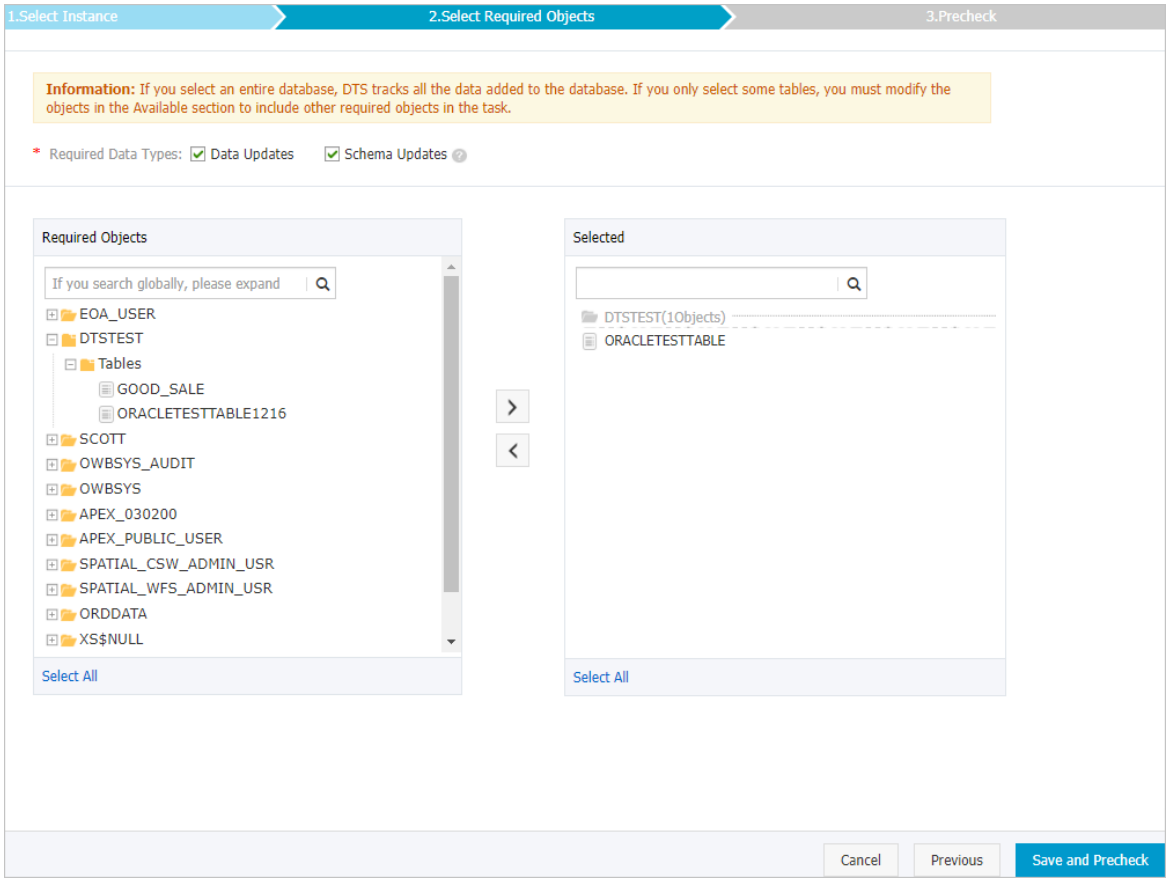
Procedure

1. Create a change tracking instance.
2. Find the change tracking instance that you created, and click **Configure Channel** in the **Actions** column.

3. Configure the source database and network type.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select an instance type based on the deployment of the source database. In this example, select **User-Created Database with Public IP Address**.
	Database Type	This parameter is set to Oracle and cannot be changed.
	Instance Region	The source region that you selected when you created the change tracking instance. You cannot change the value of this parameter.
	Hostname or IP Address	Enter the hostname or IP address of the user-created Oracle database.
	Port Number	Enter the service port number of the user-created Oracle database.
	SID	Enter the system ID (SID) of the user-created Oracle database.
	Database Account	Enter the account of the user-created Oracle database. **Note** The account must have the database administrator (DBA) permission.
	Database Password	Enter the password of the source database account.
Consumer Network Type	Network Type	Select the network type of the change tracking instance. ○ **Classic** If you select **Classic**, no other configurations are required. ○ **VPC** If you select **VPC**, you must specify the **VPC** and **VSwitch**.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the data change types and objects.



Parameter	Description
Required Data Types	◦ **Data Updates** DTS tracks data updates of the selected objects, including the INSERT, DELETE, and UPDATE operations. ◦ **Schema Updates** DTS tracks the create, delete, and modify operations that are performed on all object schemas of the source instance. You must use the change tracking client to filter the required data. **Note** ◦ If you select a database as the object, DTS tracks data changes of all objects, including new objects in the database. ◦ If you select a table as the object, DTS tracks only data changes of this table. In this case, if you want to track data changes of another table, you must add the table to the required objects. For more information, see .
Required Objects	Select objects from the **Required Objects** section and click the icon to move the objects to the **Selected** section.

6. In the lower-right corner of the page, click **Save and Precheck**.

> **Note** You can start a change tracking task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.
>
> Troubleshoot the issues based on the causes and run a precheck again.

7. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.**
   After the change tracking task is configured, DTS performs initial change tracking, which takes about 1 minute. After the initial change tracking is complete, you can consume the tracked data.

What's next

Use a Kafka client to consume tracked data

4.5.4. Create consumer groups

The change tracking feature allows you to create multiple consumer groups. Consumers in different consumer groups can track data changes from the same data source. Consumer groups help you reduce the cost for tracking data changes and improve the efficiency of data consumption.

Note

- You can create multiple consumer groups (up to 20) in a change tracking instance to repeatedly consume data.
- A consumer group consumes each message only once, and only one consumer can consume data.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Change Tracking**.
3. Find the change tracking task and click the task ID.
4. In the left-side navigation pane, click **Consume Data**.
5. On the **Consume Data** page, click **Add Consumer Group** in the upper-right corner.
6. In the **Create Consumer Group** dialog box that appears, set the parameters for the consumer group.

Parameter	Description
Consumer Group Name	Enter a new name for the consumer group. We recommend that you use an informative name for easy identification.
Username	Enter the username of the consumer group. ○ A username must contain one or more of the following character types: uppercase letters, lowercase letters, digits, and underscores (_). ○ The username must be 1 to 16 characters in length.

Parameter	Description
Password	Enter the password that corresponds to the username of the consumer group. ○ A password must contain two or more of the following character types: uppercase letters, lowercase letters, digits, and special characters. ○ The password must be 8 to 32 characters in length.
Confirm Password	Enter the new password again.

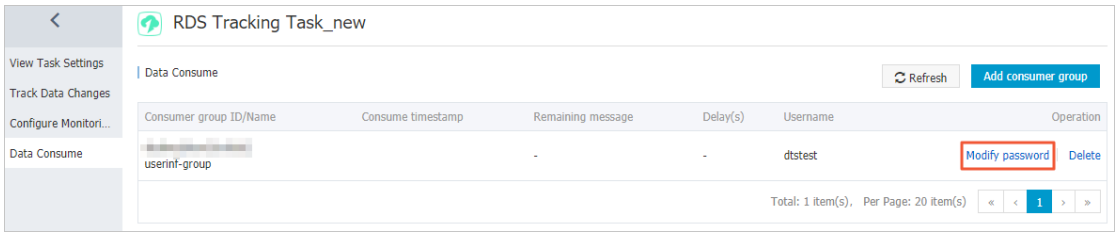
7. Click **Create**.

4.5.5. Manage consumer groups

You can manage consumer groups of a change tracking instance in the DTS console. This topic describes how to modify the password of a consumer group and how to delete a consumer group.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Change Tracking**.
3. Find the change tracking task and click the task ID.
4. In the left-side navigation pane, click **Consume Data**.
5. Modify the password of a consumer group or delete a consumer group. Modify the password of a consumer group
   i. On the **Consume Data** page, find the target consumer group and click **Modify Password** in the **Actions** column.



   ii. In the **Modify Password** dialog box that appears, enter the **old password** and **new password**, and enter the new password again in the **Confirm Password** field.

> **Note**
> - A password must contain two or more of the following character types: uppercase letters, lowercase letters, digits, and special characters.
> - The password must be 8 to 32 characters in length.

   iii. Click **Modify**.

   Delete a consumer group

> Note After a consumer group is deleted, the data in the group will be cleared and cannot be recovered. We recommend that you use caution when performing this operation.

i. On the **Consume Data** page, find the target consumer group and click **Delete** in the **Actions** column.

ii. In the **Delete Consumer Group** message that appears, click **OK**.

4.5.6. Modify objects for change tracking

DTS allows you to add or remove objects for change tracking in the consumption process. This topic describes how to modify objects for change tracking.

Note

- After you add an object, the change tracking task pulls the incremental data of the new object from the time when the modification takes effect.
- After you remove an object, if the change tracking client can tracks data changes of the removed object, you need to filter the object in the change tracking client.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Change Tracking**.
3. Find the change tracking task and click **Modify Required Objects** in the **Actions** column.
4. In the **Select Required Objects** step, you can add and remove objects for change tracking.
   - Add objects for change tracking

     In the **Required Objects** section, select the required objects and click the > button to add the objects to the **Selected** section.
   - Remove objects for change tracking

     In the **Selected** section, select the objects to be removed and click the < button to move the objects to the **Required Objects** section.
5. In the lower-right corner of the page, click **Save and Precheck**.

   > Note
   > - A precheck is performed before you can start the change tracking task. You can start the change tracking task only after the task passes the precheck.
   > - If the task fails the precheck, click the icon next to each failed item to view details.
   >
   >   Fix the issues based on the instructions and run the precheck again.
6. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.**

4.5.7. Use a Kafka client to consume tracked data

This topic describes how to use the demo code of a Kafka client to consume tracked data. The change tracking feature of the new version allows you to consume tracked data by using a Kafka client (V0.11 to V1.1).

Prerequisites

- A change tracking task is created. For more information, see Track data changes from a user-created MySQL database or Track data changes from a user-created Oracle database.
- One or more consumer groups are created. For more information, see Create consumer groups.

Precautions

- If you enable auto commit when you use the change tracking feature, some data may be committed before it is consumed. This results in data loss. We recommend that you manually commit data.

  > **Note** If data fails to be committed due to a fault, you can restart the client to continue consuming data from the last recorded consumer offset. However, duplicate data may be generated during this period. You must manually filter out the duplicate data.

- Data is serialized and stored in the Avro format. For more information, see Record.avsc.

  > **Note** If the client that you use is not a Kafka client, you must parse the tracked data based on the Avro schema.

- Regarding the `offsetFotTimes` interface, the search unit of DTS is seconds, and the search unit of native Kafka is milliseconds.

Download and run the demo code of the Kafka client

Click here to download the demo code of the Kafka client. For more information about how to use the demo code, see Readme.

Download and run the demo code of the Kafka client

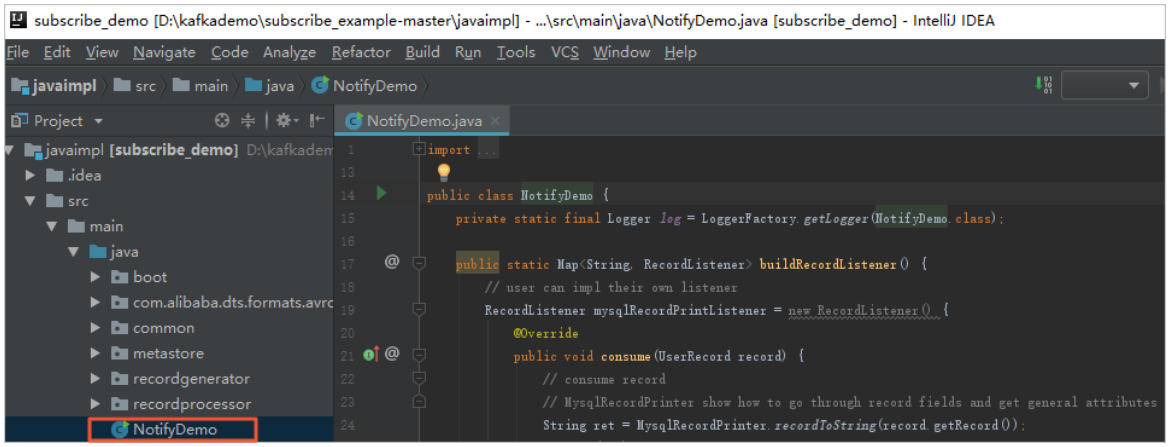
Step	File or directory
1. Use the native Kafka consumer to obtain incremental data from change tracking tasks.	subscribe_example-master/javaimpl/src/main/java/recordgenerator/
2. Deserialize the image of the incremental data, and obtain attributes such as the pre-image and post-image.	subscribe_example-master/javaimpl/src/main/java/boot/MysqlRecordPrinter.java

Step	File or directory
3. Convert the dataTypeNumber values in the deserialized data into MySQL or Oracle data types. **Note** For more information, see Mappings between MySQL data types and dataTypeNumber values and Mappings between Oracle data types and dataTypeNumber values.	subscribe_example-master/javaimpl/src/main/java/recordprocessor/mysql/

Procedure

This procedure uses IntelliJ IDEA (Community Edition 2018.1.4 Windows) as an example.

1. Download the demo code of the Kafka client, and then decompress the package.
2. Open IntelliJ IDEA. In the window that appears, click **Open**.
3. In the dialog box that appears, go to the directory where the downloaded demo code resides. Find the *pom.xml* file.
4. In the dialog box that appears, select **Open as Project**.
5. On the IntelliJ IDEA page, expand folders to find the demo file of the Kafka client, and double-click the file. The file name is *NotifyDemo.java*.



6. Set the parameters in the *NotifyDemo.java* file.

Parameter	Description	Method to obtain

Parameter	Description	Method to obtain
USER_NAME	The username of the consumer group. **Warning** If you are not using the Kafka client that is described in this topic, you must specify the username in the following format: `<Consumer group account>-<Consumer group ID>`, for example, `dtstest-dtsae******bpv`. Otherwise, the connection fails.	In the DTS console, click the instance ID, and then click **Data Consume**. You can obtain the **Consumer Group ID** and the corresponding **Account** information. **Note** The password of the consumer group account is specified when you create a consumer group.
PASSWORD_NAME	The password of the account.	
SID_NAME	The ID of the consumer group.	
GROUP_NAME	The name of the consumer group. Set this parameter to the consumer group ID.	
KAFKA_TOPIC	The topic of the change tracking task.	In the DTS console, click the instance ID. On the **Track Data Changes** page, you can obtain the **tracked topic**, network address, and port number.
KAFKA_BROKER_URL_NAME	The network address and port number of the change tracking task. **Note** If you track data changes over internal networks, the network latency is minimal. This is applicable if the ECS instance where you deploy the Kafka client belongs to the same VPC or classic network as the change tracking instance.	

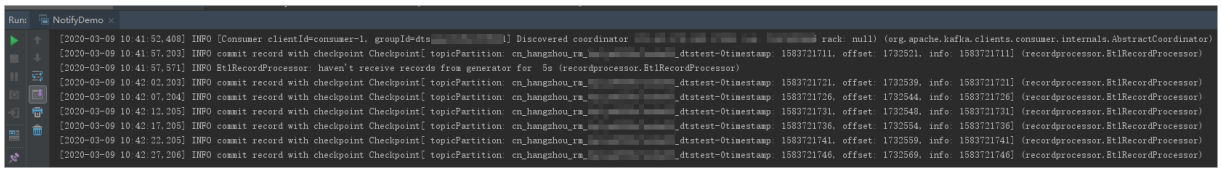
Parameter	Description	Method to obtain
INITIAL_CHECKPOINT_NAME	The consumer offset of consumed data. The value is a UNIX timestamp. **Note** You must save the consumer offset. If the consumption process is interrupted, you can specify the consumer offset on the change tracking client to resume data consumption. This allows you to prevent against data loss. When you start the change tracking client, you can specify the consumer offset to consume data on demand.	When you use the Kafka client to track data changes for the first time, convert the required time point into a UNIX timestamp.
USE_CONFIG_CHECKPOINT_NAME	Default value: *true*. The default value indicates that the client is forced to consume data from the specified consumer offset. This allows you to retain the data that is received but not processed.	None.

7. On the top of the IntelliJ IDEA page, choose **Run** > **Run** to run the client.

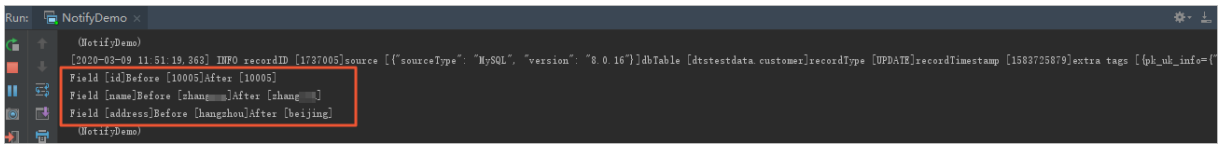
> **Note** When you run IntelliJ IDEA for the first time, it loads and installs the relevant dependency.

Execution result

The following figure shows that the Kafka client can track data changes in the source database.



You can also delete the `//` characters from the `//log.info(ret);` string in line 25 of the *NotifyDemo.java* file. Then, run the client again to view the data change information.



Mappings between MySQL data types and dataTypeNumber values

MySQL data type	Value of dataTypeNumber
MYSQL_TYPE_DECIMAL	0
MYSQL_TYPE_INT8	1
MYSQL_TYPE_INT16	2
MYSQL_TYPE_INT32	3
MYSQL_TYPE_FLOAT	4
MYSQL_TYPE_DOUBLE	5
MYSQL_TYPE_NULL	6
MYSQL_TYPE_TIMESTAMP	7
MYSQL_TYPE_INT64	8
MYSQL_TYPE_INT24	9
MYSQL_TYPE_DATE	10
MYSQL_TYPE_TIME	11
MYSQL_TYPE_DATETIME	12
MYSQL_TYPE_YEAR	13
MYSQL_TYPE_DATE_NEW	14
MYSQL_TYPE_VARCHAR	15
MYSQL_TYPE_BIT	16
MYSQL_TYPE_TIMESTAMP_NEW	17
MYSQL_TYPE_DATETIME_NEW	18
MYSQL_TYPE_TIME_NEW	19
MYSQL_TYPE_JSON	245
MYSQL_TYPE_DECIMAL_NEW	246
MYSQL_TYPE_ENUM	247
MYSQL_TYPE_SET	248
MYSQL_TYPE_TINY_BLOB	249
MYSQL_TYPE_MEDIUM_BLOB	250

MySQL data type	Value of dataTypeNumber
MYSQL_TYPE_LONG_BLOB	251
MYSQL_TYPE_BLOB	252
MYSQL_TYPE_VAR_STRING	253
MYSQL_TYPE_STRING	254
MYSQL_TYPE_GEOMETRY	255

Mappings between Oracle data types and dataTypeNumber values

Oracle data type	Value of dataTypeNumber
VARCHAR2/NVARCHAR2	1
NUMBER/FLOAT	2
LONG	8
DATE	12
RAW	23
LONG_RAW	24
UNDEFINED	29
XMLTYPE	58
ROWID	69
CHAR and NCHAR	96
BINARY_FLOAT	100
BINARY_DOUBLE	101
CLOB/NCLOB	112
BLOB	113
BFILE	114
TIMESTAMP	180
TIMESTAMP_WITH_TIME_ZONE	181
INTERVAL_YEAR_TO_MONTH	182
INTERVAL_DAY_TO_SECOND	183

Oracle data type	Value of dataTypeNumber
UROWID	208
TIMESTAMP_WITH_LOCAL_TIME_ZONE	231

5.Cloud Native Distributed Database PolarDB-X

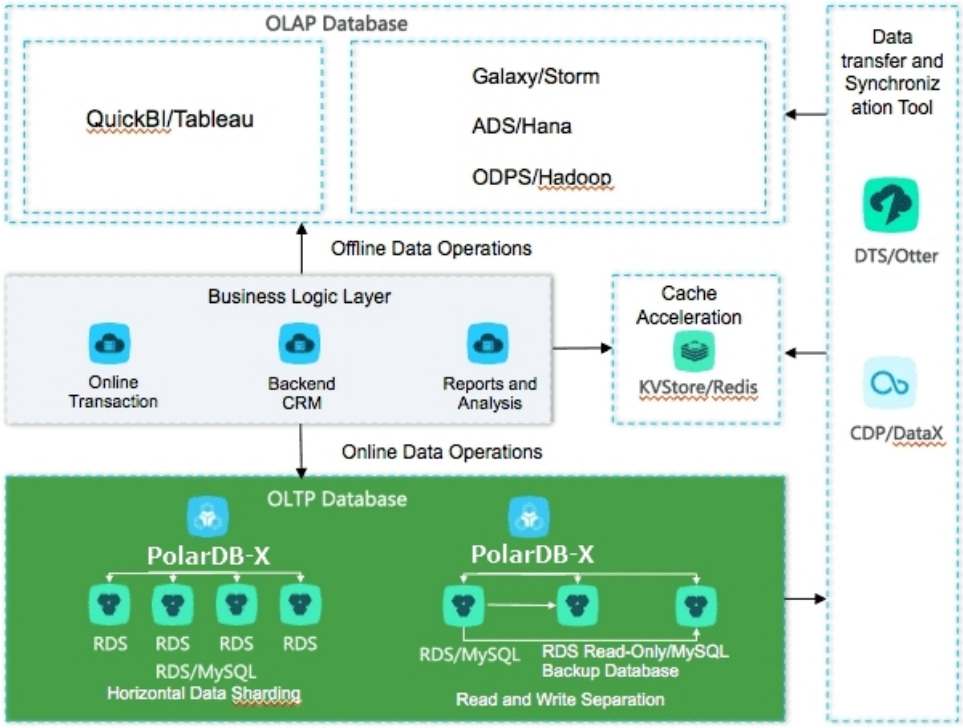
5.1. What is PolarDB-X?

Cloud Native Distributed Database PolarDB-X is a middleware service developed by Alibaba Group for the scale-out of standalone relational databases. It is compatible with former Distributed Relational Database Service (DRDS).

Compatible with the MySQL protocol, PolarDB-X supports most MySQL data manipulation language (DML) and data definition language (DDL) syntax. It provides the core capabilities of distributed databases, such as database sharding, table sharding, smooth scale-out, configuration changing, and transparent read/write splitting. PolarDB-X is lightweight (stateless), flexible, stable, and efficient, and provides you with O&M capabilities throughout the lifecycle of distributed databases.

PolarDB-X is used for operations on large-scale online data. PolarDB-X maximizes the operation efficiency by partitioning data in specific business scenarios, and therefore meets the requirements of online business on relational databases.

PolarDB-X architecture



Benefits

- Capacity bottleneck of standalone databases: As the data volume and access volume increase, traditional standalone databases encounter great challenges that cannot be completely solved by hardware upgrades. Distributed solutions use multiple instances to work jointly, and therefore can effectively resolve the bottlenecks of data storage capacity and access traffic.
- Difficult scale-out of relational databases: Due to the inherent attributes of distributed databases, PolarDB-X allows you to store data to different shards by using smooth data migration. This way, the

relational databases can be dynamically scaled out.

5.2. Quick start

This topic describes how to get started with Cloud Native Distributed Database PolarDB-X.

A PolarDB-X instance is physically a distributed cluster that consists of multiple PolarDB-X server nodes and underlying storage instances. A PolarDB-X database is a logical concept and contains only metadata. Specific data is stored in the physical databases of the underlying storage instances. To get started with PolarDB-X, perform the following steps:

1. Create a PolarDB-X instance.
2. Create a database.

   To create a database in a PolarDB-X instance, you must select one or more ApsaraDB RDS for MySQL instances as the data storage nodes. If no ApsaraDB RDS for MySQL instance exists, create one first. For more information about how to create an ApsaraDB RDS for MySQL instance, see **User Guide > Instance management > Create an instance** in the *ApsaraDB RDS documentation*.
3. After a PolarDB-X database is created, you must create tables in the PolarDB-X database. This is similar to creating tables in a common standalone database. However, the syntax is different in the expression of data partitioning information in the table creation statement of PolarDB-X. For more information about how to create a table, see Table creation syntax.

5.3. Log on to the PolarDB-X console

This topic describes how to log on to the Cloud Native Distributed Database PolarDB-X console by using Google Chrome.

Prerequisites

- The domain name of the Apsara Uni-manager console is obtained from the deployment personnel before you log on to the Apsara Uni-manager console.
- A browser is available. We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to log on to the Apsara Uni-manager console. 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 console for the first time, you must change the password of your username. For security reasons, your password must meet the minimum complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:
>
> - Uppercase or lowercase letters.
> - Digits.
> - Special characters. Special characters include exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to go to the Apsara Uni-manager console homepage.
4. In the top navigation bar, choose **Products > Database Services > Distributed Relational Database Service**.

5.4. Instance management

5.4.1. Create a PolarDB-X instance

To use PolarDB-X, you must first create an instance. This topic describes how to create a PolarDB-X instance.

1. Log on to the PolarDB-X console.
2. On the page that appears, click **Create Instance** in the upper-right corner.
3. On the **Create DRDS Instance** page, set parameters as required.

   Parameters for creating a PolarDB-X instance describes the parameters.

Parameters for creating a PolarDB-X instance

Type	Parameter	Description
Region	**Organization**	The organization to which the instance belongs.
	Resource Set	The resource set to which the instance belongs.
	Region	The region where the instance resides. Services in different regions are not interconnected over the internal network. After the instance is created, the region cannot be changed.
	Zone	The zone where the instance resides.
	Instance Type	The type of the instance. Select an instance type from the options available on the page.

Type	Parameter	Description
Basic Settings	Instance Edition	The edition of the instance. Valid values: ○ Standard ○ Enterprise ○ Starter
	Instance Specifications	The specifications of the instance. The rules vary with instance editions. Select the instance specifications from the options available on the page.
Network Type	Network Type	Only **classic network** is supported. **Note** **Classic Network**: Cloud services on a classic network are not isolated from each other. Unauthorized access to a cloud service is blocked only by the security group or whitelist policy of the service.

4. Click **Submit**.

   After the instance is created, it appears in the instance list and its status changes to **Running**. An instance name uniquely identifies a PolarDB-X instance.

5.4.2. Change instance specifications

When you use PolarDB-X, you can change the specifications of a PolarDB-X instance as needed.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the **Common Operations** section or the **Configuration Information** section, click **Upgrade** or **Downgrade** to go to the Change Specifications page.

   Note Alternatively, on the **DRDS Instance Management** page, choose **More > Downgrade** in the **Actions** column of the instance.

5. On the Change Specifications page, set **Instance Edition** and **Instance Specifications**, and click **Submit**. After a few minutes, you can view the new specifications of the PolarDB-X instance in the instance list.

   Note Specifications downgrade leads to transient disconnections between applications and PolarDB-X within a short period of time. Make sure that your applications can be automatically reconnected.

5.4.3. Read-only PolarDB-X instances

5.4.3.1. Overview

Read-only PolarDB-X instances are an extension and supplement to primary PolarDB-X instances and are compatible with SQL query syntax of primary PolarDB-X instances.

Features

Read-only and primary PolarDB-X instances can share the same replica of data. You can perform complex data query and analytics on read-only or primary ApsaraDB RDS for MySQL instances. Multiple instance types are provided to handle highly concurrent access requests and reduce the RT for complex queries. Read-only PolarDB-X instances provide resource isolation, which alleviates the load pressure on the primary PolarDB-X instances and reduces the link complexity of the business architecture. No additional data synchronization operations are required, and the O&M and budget costs are reduced.

Instance types

Concurrent read-only instances: For high-concurrency and high-traffic simple queries or offline data extraction, resource isolation helps you handle highly concurrent queries. This way, the stability of online business links is ensured.

> **Note** For the business for which primary PolarDB-X instances are used, concurrent read-only instances can be used in the following scenarios:
>
> - High-concurrency and high-traffic simple queries are performed.
> - Data is extracted offline.

Limits

- Primary and read-only PolarDB-X instances must be in the same region, but they can be in different zones.
- A read-only PolarDB-X instance must belong to a primary PolarDB-X instance. Before you create a read-only PolarDB-X instance, you must create a primary PolarDB-X instance. After you create a database on the primary instance, the database is replicated to the read-only instance. If you delete the database from the primary instance, the corresponding database on the read-only instance is also deleted.
- You are not allowed to migrate data to read-only PolarDB-X instances.
- You are not allowed to create databases in or delete databases from read-only PolarDB-X instances.
- Read-only PolarDB-X instances cannot be cloned.
- Read-only PolarDB-X instances support DDL statements but do not support DML statements for data modification.

5.4.3.2. Create a read-only PolarDB-X instance

This topic describes how to create a read-only PolarDB-X instance.

Procedure

1. Log on to the PolarDB-X console.

2. In the instance list, find the primary PolarDB-X instance for which you want to create a read-only instance.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to access the **Basic Information** page.
4. On the page that appears, click **Create DRDS Read-only Instance** in the Related Instances section.
5. Set Region, Basic Settings, and Network Type. Then, click **Submit**.

Parameters for creating a read-only PolarDB-X instance

Section	Parameter	Description
Region	**Region**	The region where the read-only PolarDB-X instance resides. Read-only PolarDB-X instances in different regions cannot communicate with each other. After a read-only PolarDB-X instance is created, the region cannot be changed.
	Zone	The zone where the read-only PolarDB-X instance resides.
Basic Settings	**Instance Type**	The type of the read-only PolarDB-X instance. Select an instance type from the options available on the page.
	Instance Edition	The edition of the read-only PolarDB-X instance. Valid values: ○ Starter ○ Standard ○ Enterprise
	Instance Specifications	The specifications of the read-only PolarDB-X instance. The specifications vary with instance editions. Select the instance specifications from the options available on the page.
	Describe	The description of the read-only PolarDB-X instance. We recommend that you provide an informative description to simplify future management operations.
Network Type	**Network Type**	Only **Classic Network** is supported. **Note** **Classic Network**: Cloud services in the classic network are not isolated from each other. Unauthorized access to a cloud service can be blocked only by the security group or whitelist policy of the service.

6. It takes several minutes to create the instance. Please wait. After the instance is created, it appears in the instance list in the PolarDB-X console.

5.4.3.3. Manage a read-only PolarDB-X instance

Read-only PolarDB-X instances are managed in a similar way as primary instances. However, databases cannot be created or deleted on the read-only instance management page. Databases on read-only instances are created or deleted with databases on primary instances. In the PolarDB-X console, you can go to the read-only instance management page in two ways.

Manage a read-only PolarDB-X instance by its ID

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target read-only instance.
3. Click the target instance ID or choose **More** > **Manage** from the Actions column of the target read-only instance to access the **Basic Information** page.

Manage a read-only PolarDB-X instance by the ID of its primary instance

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target primary instance.
3. Click the target instance ID or choose **More** > **Manage** from the Actions column of the target primary instance to access the **Basic Information** page.
4. On the **Basic Information** page, move the pointer over the number of read-only instances in the **Related Instances** section to view the ID of the read-only PolarDB-X instance.
5. Click the ID of the target read-only PolarDB-X instance. The **Basic Information** page of the read-only instance appears.

5.4.3.4. Release a read-only PolarDB-X instance

If you no longer need a read-only PolarDB-X instance, you can release it.

Prerequisites

The read-only instance must be in the **Running** state.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. In the PolarDB-X instance list, find the target instance, and choose **More** > **Release** from the Actions column.

   > **Notice** You cannot recover the PolarDB-X instances that have been released. Exercise caution when you perform this operation.

4. In the **Release DRDS Instance** dialog box, click **OK**.

5.4.4. Restart a PolarDB-X instance

This topic describes how to restart a PolarDB-X instance.

Prerequisites

The PolarDB-X instance must be in the **Running** state.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More** > **Manage** from the Actions column of the instance to access the **Basic Information** page.
4. Click **Restart Instance** in the upper-right corner.
5. In the **Restart Instance** dialog box, click **OK**.

> **Notice** Restarting a PolarDB-X instance terminates all its connections. Make appropriate service arrangements before you restart a PolarDB-X instance. Exercise caution when you perform this operation.

5.4.5. Release a PolarDB-X instance

This topic describes how to release a running PolarDB-X instance in the PolarDB-X console.

Prerequisites

- All databases on the PolarDB-X instance have been deleted.
- The PolarDB-X instance must be in the Running state.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. In the PolarDB-X instance list, find the target instance, and choose **More** > **Release** from the Actions column.
4. In the **Release DRDS Instance** dialog box, click **OK**.

> **Warning** After the PolarDB-X instance is released, data is not deleted from its attached ApsaraDB RDS for MySQL instances. However, a released PolarDB-X instance cannot be restored. Exercise caution when you perform this operation.

5.4.6. Recover data

5.4.6.1. Backup and restoration

PolarDB-X allows you to back up data of instances and databases and restore them by using the backup data. Instances can be automatically and manually backed up. PolarDB-X provides fast backup and consistent backup. Existing backup sets are used to restore data in instances. Data is restored to the new PolarDB-X and ApsaraDB RDS for MySQL instances based on the existing backup sets.

Considerations

- By default, the automatic backup policy of PolarDB-X is disabled. You must manually enable it.

- The log backup capability of PolarDB-X relies on underlying ApsaraDB RDS for MySQL instances. Therefore, the log backup policy configured in the PolarDB-X console is automatically synchronized to all underlying ApsaraDB RDS for MySQL instances. After the policy is configured, do not modify it in the ApsaraDB for RDS console. Otherwise, relevant data backup sets may be invalid.
- The backup and restoration feature of PolarDB-X relies on log backup. We recommend that you enable the log backup policy by default to prevent backup sets from becoming invalid.
- Data definition language (DDL) operations cannot be performed during the backup process. Otherwise, instance backup and restoration will fail.
- During the backup, make sure that the underlying ApsaraDB RDS for MySQL instances for the PolarDB-X instance are normal to avoid a backup failure.
- Consistent backup and restoration is supported only in PolarDB-X 5.3.8 and later.
- All tables must have primary keys to keep data accuracy during consistent backup and restoration.
- During consistent backup, distributed transactions on PolarDB-X instances are locked for seconds. During the locking period, the execution of non-transactional SQL statements and non-distributed transactions is not affected. However, the commitment of distributed transactions is blocked and the response time (RT) for executing SQL statements may have millisecond-level jitters. We recommend that you perform consistent backup during off-peak hours.
- Due to changes in the inventory of PolarDB-X and ApsaraDB RDS for MySQL, PolarDB-X automatically adjusts the instance type and zone during instance restoration. We recommend that you confirm and adjust the instance type and zone after the instance restoration to avoid business disruption.

Backup methods

For different scenarios, PolarDB-X provides fast backup and consistent backup, and corresponding restoration capabilities. The following table compares the two backup methods.

Backup method	Scenario	Advantage	Disadvantage
Fast backup	It applies to routine backup and restoration scenarios.	• It provides fast data backup and restoration. • It supports data restoration methods: by backup set and by time. • It supports all PolarDB-X instance versions.	It ensures data consistency only within a single ApsaraDB RDS for MySQL instance in sharding scenarios, but does not ensure global data consistency.

Backup method	Scenario	Advantage	Disadvantage
Consistent backup	It applies to backup and restoration for the financial industry and online core transactions that require high data consistency.	It ensures global data consistency in sharding scenarios.	• It features slow backup and restoration. • It supports data restoration by backup set, but does not support restoration by time. • It is supported only in PolarDB-X 5.3.8 and later. • During the backup, distributed transactions on PolarDB-X instances are locked for seconds. During the locking period, the RT for executing SQL statements may have millisecond-level jitters. Therefore, we recommend that you perform consistent backup during off-peak hours.

5.4.6.2. Configure an automatic backup policy

PolarDB-X provides the automatic backup feature. This topic describes how to configure an automatic backup policy.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More** > **Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the left-side navigation pane, choose **Data Recovery** > **Backup and Recovery**.
5. On the page that appears, choose **Backup Policy** > **Edit**.
6. In the **Backup Policy** dialog box, set parameters as needed, and click **OK**.

5.4.6.3. Local log settings

You can use local logs and the backup and restoration feature or the SQL flashback feature of PolarDB-X to accurately restore an instance or a database to the desired time point. This topic describes how to configure local logs.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the left-side navigation pane, choose **Data Recovery > Backup and Recovery**.
5. In the **Local Binlog Settings** dialog box, set parameters as needed, and click **OK**.

5.4.6.4. Manual backup

In addition to automatic backup, PolarDB-X also provides manual backup, which allows you to back up data at any time. This topic describes how to manually back up instances and databases.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the left-side navigation pane, choose **Data Recovery > Backup and Recovery**.
5. On the page that appears, click **Data Backup** on the right.
6. In the dialog box that appears, set Backup Method and Backup Level.
   - Backup Method can be set to **Fast Backup** or **Consistent Backup**. For more information about the differences between the two methods, see Backup methods.

     > **Notice** If you select Consistent Backup, distributed transactions are locked for seconds and the RT may have sub-second-level jitters. Therefore, we recommend that you perform consistent backup during off-peak hours.

   - Backup Level can be set to **Instance Backup** or **Database Backup**. You can select **Instance Backup** to back up the entire instance, or select **Database Backup** to back up a database as needed.
7. Click **OK**.

5.4.6.5. Restore data

You can use the data restoration feature of PolarDB-X to restore an instance or a database to the time when the backup is created. You can perform this operation at any time. This topic describes how to restore the data of an instance or a database to a specific point in time.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to access the **Basic Information** page.

4. In the left-side navigation pane, choose **Data Recovery > Backup and Recovery**.
5. On the page that appears, click **Data Recovery (Original Clone Instance)** on the right.
6. Select a restoration method.
   - **By Time**: Restore data to the selected point in time. You must set **Restoration Time** and **Recovery Level**.
   - **By Backup Set**: Restore data from the selected backup file.

   > **Note** You can also click **Recover** in the Actions column of the backup set to restore data **by backup set**, as shown in the following figure.
7. Click **Precheck** to check whether a valid backup set is available for data restoration. If the precheck fails, the data cannot be restored.
8. Click **Enable** to go to the **Confirm Order** page.
9. Confirm the order details and click **Enable** to restore the data. You can view the data restoration progress in **Task Progress** in the upper-right corner of the page.

5.4.6.6. SQL flashback

5.4.6.6.1. Overview

PolarDB-X provides the SQL flashback feature to recover data of particular rows.

When you mistakenly run an SQL statement such as INSERT, UPDATE, or DELETE on PolarDB-X, provide the relevant SQL information to match the event in the binary log file and generate the corresponding recovery file. You can download the file and recover data as needed. SQL flashback automatically chooses **fuzzy match** or **exact match** to locate lost data caused by the error. For more information, see Exact match and fuzzy match and Rollback SQL statements and original SQL statements.

Features

- Easy-to-use: SQL flashback allows you to retrieve the lost data by entering required information about the corresponding SQL statement.
- Fast and lightweight: Regardless of the backup policy of ApsaraDB RDS for MySQL instances, you only need to enable log backup before an SQL statement error occurs.
- Flexible recovery: Rollback SQL statements and original SQL statements are available for different scenarios.
- Exact match: SQL flashback supports exact match of data about the corresponding SQL statement, which improves precision of data recovery.

Limits

- SQL flashback depends on the binary log retention time and the log backup feature of ApsaraDB RDS for MySQL must be enabled. Binary log files can be retained only for a certain period. Use SQL flashback to generate files for recovery as soon as possible when an error occurs.
- The recovery files generated by SQL flashback are retained for seven days by default, and you need to download these files as soon as possible.
- The following conditions must be met for SQL flashback exact match:
  - The PolarDB-X instance version is 5.3.4-15378085 or later.

  - The version of the ApsaraDB RDS for MySQL instance used by the PolarDB-X database is 5.6 or later.
  - SQL flashback exact match is enabled before the error SQL statement is executed.
  - The TRACE_ID information for the error SQL statement is provided.

- To ensure the precision of data recovery, the exact match feature is enabled by default for the database created in a PolarDB-X instance of 5.3.4-15378085 or later. After this feature is enabled, SQL execution information is included in the binary log file by default, which requires more storage space for ApsaraDB RDS for MySQL instances. If you need to use the exact match feature, we recommend that you upgrade PolarDB-X before enabling the feature. For more information, see Enable exact match.

5.4.6.6.2. Generate a restoration file

This topic describes how to generate a restoration file in the PolarDB-X console.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the left-side navigation pane, choose **Data Recovery > SQL Flashback**. The **SQL Flashback** page appears.
5. On the **SQL Flashback** page, enter the basic information about a mistaken SQL statement, including Database, Time Range, Table Name, TRACE_ID, and SQL Statement Type. The following table describes the parameters.

Parameter	Description
Database	The database where the mistaken SQL statement was executed.
Time Range	The time range during which the mistaken SQL statement was executed. The beginning of the time range is earlier than the time when the execution of the mistaken SQL statement starts. The end of the time range is later than the time when the execution of the mistaken SQL statement ends. To ensure efficient restoration, we recommend that you limit the time range to 5 minutes.
Table Name	The name of the table on which the mistaken SQL statement was executed. This parameter is optional.
TRACE_ID	The unique TRACE_ID that PolarDB-X allocates for each executed SQL statement. You can obtain the TRACE_ID of the mistaken SQL statement by using the SQL audit feature of PolarDB-X.
SQL Statement Type	The type of the mistaken SQL statement. Valid values: ○ INSERT ○ UPDATE ○ DELETE

6. Click **Precheck**. The system checks whether a binary log file exists within the specified time range.

For more information about binary log files, see Configure local logs.

> **Note**
>
> - If no binary log file exists within the specified time range, the precheck fails and the system cannot restore the data for you.
> - If a binary log file exists within the specified time range, the precheck is successful and you can proceed to the next step.

7. Set SQL Statement Type for Recovery to **Rollback SQL** or **Original SQL Statement**. For more information about the differences between the two types, see Rollback SQL statements and original SQL statements.
8. Click **Generate SQL** to generate an SQL flashback task. The status of the SQL flashback tasks that are running on the current instance appears at the lower part of the page.

What's next

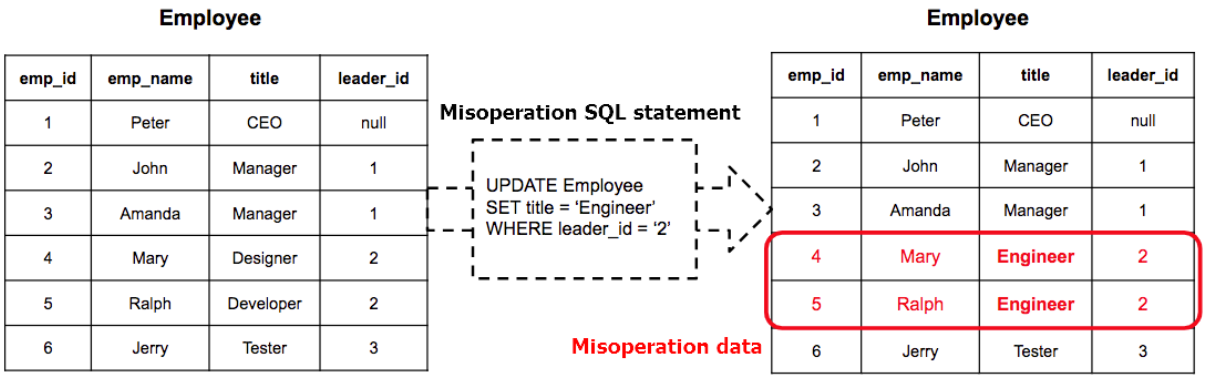
After an SQL flashback task is completed, the task information such as the exact match status and the number of recovered rows appears. You can click **Download** in the Actions column of the SQL flashback task to download the corresponding restoration file.

> **Notice** By default, the restoration file is retained for seven days. Download it as soon as possible.

5.4.6.6.3. Rollback SQL statements and original SQL statements

To support different business scenarios, PolarDB-X SQL flashback provides rollback SQL statements and original SQL statements. Before generating an SQL statement for recovering data, you must select a corresponding recovery method based on your scenario.

Recovery methods



Recovery method	Description	Example

Recovery method	Description	Example
Rollback SQL statement	Traverses the events in the binary log file in reverse order to reverse the INSERT, UPDATE, and DELETE events. • The reverse of INSERT is equivalent to DELETE. • The reverse of DELETE is equivalent to INSERT. • The reverse of UPDATE is equivalent to the value before UPDATE.	UPDATE Employee SET title = 'Developer' WHERE emp_id = '5' UPDATE Employee SET title = 'Designer' WHERE emp_id = '4'
Original SQL statement	Traverses the events in the binary log file in order to mirror all records of the INSERT, UPDATE, and DELETE events. • An INSERT mirror is equivalent to INSERT. • A DELETE mirror is equivalent to INSERT. • An UPDATE mirror is equivalent to the value before INSERT.	INSERT INTO Employee(emp_id,emp_name,title,leader_id) values('4','Mary','Designer','2') INSERT INTO Employee(emp_id,emp_name,title,leader_id) values('5','Ralph','Developer','2')

5.4.6.6.4. Exact match and fuzzy match

SQL flashback supports **exact match** and **fuzzy match** for binary log events. You do not need to select a match policy. SQL flashback automatically detects and selects the optimal match policy, and notifies you when the flashback task is completed.

Match policy	Description	Advantage	Disadvantage

Match policy	Description	Advantage	Disadvantage
Exact match	The system performs exact match on the event of a mistaken SQL statement in the binary log file and generates a restoration file.	The restoration file contains only data that is deleted or modified due to the execution of the mistaken SQL statement. The file can be directly used to ensure the precision and efficiency of data restoration.	The following requirements must be met: • The PolarDB-X instance version is 5.3.4-15378085 or later. • The version of the ApsaraDB RDS for MySQL instance used by PolarDB-X databases is 5.6 or later. • Exact match of SQL flashback is enabled before mistaken SQL statements are executed. • The TRACE_ID of the mistaken SQL statement is provided.
Fuzzy match	The system matches the events in the binary log file based on the information about the mistaken SQL statement, including the time range, table name, and SQL statement type. Then, the system generates a restoration file.	Fuzzy match is supported on all instances, regardless of the instance version or parameter settings.	Data that is deleted or modified by the mistaken SQL statement cannot be exactly matched. The restoration file contains data changes that are made by other business SQL operations. You must filter the required data.

Enable exact match

> **Note** By default, fuzzy match is enabled.

1. Log on to the PolarDB-Xconsole, and go to the Parameter Settings page of the specified instance. For more information, see Set parameters.
2. Change the value of **ENABLE_SQL_FLASHBACK_EXACT_MATCH** to **ON**.

5.4.6.7. Table recycle bin

5.4.6.7.1. Overview

The table recycle bin of PolarDB-X allows you to recover mistakenly deleted tables.

After the table recycle bin is enabled for your PolarDB-X database, the tables that are deleted by using the DROP TABLE statement are moved to the recycle bin and are no longer visible to you. After the tables are moved to the recycle bin for two hours, they are automatically cleared and cannot be recovered. You can view, recover, and clear the deleted tables in the recycle bin.

Limits and notes

- The table recycle bin feature is only supported by PolarDB-X 5.3.3-1670435 and later. For more information, see View the instance version.
- The table recycle bin is disabled for your PolarDB-X database by default. For more information about how to enable it, see Enable the table recycle bin.
- The table recycle bin of PolarDB-X does not support the recovery of tables deleted by the TRUNCATE TABLE command.
- Tables in the recycle bin still occupy the storage space of ApsaraDB RDS for MySQL before they are automatically cleared. To release the storage space as soon as possible, you can access the recycle bin to manually delete them.

5.4.6.7.2. Enable the table recycle bin

This topic describes how to enable the table recycle bin.

Prerequisites

An ApsaraDB RDS for MySQL database has been created in the PolarDB-X instance. For more information, see Create a database.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the left-side navigation pane, choose **Data Recovery > Table Recycle Bin**. The **Table Recycle Bin** page appears.
5. At the top of the **Table Recycle Bin** page, click the tab of the database for which the table recycle bin needs to be enabled.
6. Click **Enabled**.
7. In the dialog box that appears, click **OK**.

5.4.6.7.3. Restore tables

This topic describes how to restore your tables from the table recycle bin.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to access the **Basic Information** page.

4. In the left-side navigation pane, choose **Data Recovery > Table Recycle Bin**. The **Table Recycle Bin** page appears.
5. In the upper part of the **Table Recycle Bin** page, click the tab of the database for which you want to restore a table.
6. Click **Recover** in the **Actions** column of the table that you want to restore.

5.4.6.7.4. Delete tables

This topic describes how to delete unnecessary tables from the table recycle bin.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the left-side navigation pane, choose **Data Recovery > Table Recycle Bin**. The **Table Recycle Bin** page appears.
5. In the upper part of the **Table Recycle Bin** page, click the tab of the database in which you want to delete a table.
6. Click **Delete** in the **Actions** column of the table that you want to delete.

> **Note** To clear all tables from the table recycle bin, click **Empty Recycle Bin** on the tab of the corresponding database.

5.4.6.7.5. Disable the table recycle bin feature

If you no longer need the table recycle bin feature, you can disable it. This topic describes how to disable the table recycle bin feature.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the left-side navigation pane, choose **Data Recovery > Table Recycle Bin**. The **Table Recycle Bin** page appears.
5. In the upper part of the **Table Recycle Bin** page, click the tab of the database for which you want to disable the table recycle bin feature.
6. Click **Disable** to disable the table recycle bin feature for the database.

5.4.7. Set parameters

PolarDB-X allows you to set parameters for instances and databases. You can view and modify parameter values in the PolarDB-X console as needed.

> **Note** You cannot set parameters for read-only PolarDB-X instances.

Procedure

1. Log on to the PolarDB-X console.
2. In the instance list, find the PolarDB-X instance for which you want to set parameters.
3. Click the instance ID or choose **More > Manage** from the Actions column of the instance to go to the **Basic Information** page.
4. In the left-side navigation pane, choose **Diagnostics and Optimization > Parameter Settings**. Click the Instance or Database tab to view parameters that you can modify for instances and databases. For more information about the parameters, see Parameters.
5. Click the icon next to the parameter that you want to modify, enter the expected value, and then click **OK**.
6. Click **Submit** in the upper-right corner to submit the new value.

> **Note** To undo parameter modification, click **Cancel** in the upper-right corner.

Parameters

Parameter	Level	Description
Slow query threshold	Instance	The threshold for slow SQL statements. SQL statements whose execution duration exceed this threshold are recorded in logical slow SQL logs.
Logical idle connection timeout	Instance	The logical timeout period of the idle connection between user applications and PolarDB-X. Unit: milliseconds.
Maximum packet size	Instance	The maximum size of a packet between user applications and PolarDB-X. Unit: bytes.
Limit on instance memory pool size	Instance	The maximum size of the memory pool for an instance. If the memory usage on an instance exceeds the value, an error is reported and the query ends.
Whether to prohibit all table deletion/update	Database	Specifies whether to disable all table deletion and update.
Enable the recycle bin	Database	Specifies whether to enable the table recycle bin feature, which moves deleted logical tables of PolarDB-X to the table recycle bin.
Temporary table size	Database	The size of the temporary table that is used during distributed queries in PolarDB-X. Unit: rows.

Parameter	Level	Description
Number of unioned tables	Database	The maximum number of table shards that can be combined by the JOIN statement when you query multiple table shards in a database.
SQL query timeout period	Database	The timeout period of SQL statements for interaction between PolarDB-X and ApsaraDB RDS for MySQL. Unit: milliseconds. The value 0 indicates the timeout period is not limited.
SQL exact flashback switch	Database	Specifies whether to support exact match of SQL flashback. By default, exact match is disabled. After it is enabled, information about query execution is added to the binary log file that is used by the PolarDB-X database.
Enable logical INFORMATION_SCHEMA query	Database	Specifies whether to enable the logical INFORMATION_SCHEMA query. When it is enabled, it does not rely on the shadow database, and the aggregation results of logical databases and tables are returned. When it is disabled, it relies on the shadow database, and the information of physical databases and tables are returned.
Time period to start transaction log cleanup	Database	The period during which transaction log cleanup starts at a random time.
Limit on database-level memory pool size	Database	The maximum size of the database-level memory pool. When the memory usage of a PolarDB-X database exceeds this value, an error is reported and the query terminates. The value -1 indicates that the maximum size of the database-level memory pool is not limited.
Limit on query-level memory pool size	Database	The maximum size of the query-level memory pool. When the memory usage of a query exceeds this value, an error is reported and the query terminates. The value -1 indicates that the maximum size of the query-level memory pool is not limited.
Enable CBO	Database	Specifies whether to enable the cost-based optimizer (CBO), including features such as Join Reorder and Hash Join.
Parallelism	Database	Specifies whether to enable parallel query, and the degree of parallelism after parallel query is enabled. This parameter takes effect only when CBO is enabled. The value -1 indicates that the degree of parallelism is automatically selected. The value 0 indicates that parallel query is disabled.
Whether to enable the asynchronous DDL engine	Database	Specifies whether to enable the asynchronous data definition language (DDL) engine. If you disable it, the original execution logic of the DDL engine is still used.

Parameter	Level	Description
Enable pure asynchronous mode after the asynchronous DDL engine is enabled	Database	Specifies whether to enable the pure asynchronous mode when the asynchronous DDL engine is enabled. • Enabled: The submitted DDL statement is executed immediately after the client connects to the PolarDB-X instance. You can use only asynchronous DDL management statements to view the execution status. • Disabled: The client still interacts with the instance in synchronous mode. The instance does not respond to the client until the execution of the submitted DDL statement is complete.
Maximum number of physical tables allowed to be created in a single physical database	Database	The maximum number of table shards that can be created in a database shard. If you create a logical table whose data is partitioned into table shards in a database shard, the number of table shards cannot exceed this value.
Aggregate statistics for INFORMATION_SCHEMA.TABLES	Database	Specifies whether to aggregate statistics for INFORMATION_SCHEMA.TABLES queries. By default, the statistics are not aggregated. This ensures high performance.
Maximum number of physical database shard connections	Database	The maximum number of connections between a PolarDB-X database and a single ApsaraDB RDS for MySQL database shard.
Minimum number of physical database shard connections	Database	The minimum number of connections between a PolarDB-X database and a single ApsaraDB RDS for MySQL database shard.
Physical idle connection timeout	Database	The timeout period for an idle connection between a PolarDB-X database and an ApsaraDB RDS for MySQL database shard. Unit: minutes.

5.4.8. Monitor PolarDB-X instances

5.4.8.1. View monitoring information

PolarDB-X provides multi-dimensional monitoring. This topic describes how to view monitoring information in the PolarDB-X console.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More** > **Manage** from the Actions column of the instance to access the **Basic Information** page.

4. On the **Basic Information** page, choose **Monitoring and Alerts > Instance Monitoring** in the left-side navigation pane.
5. On the **Instance Monitoring** page, select the metrics and specify a time range to view detailed information. For more information about metrics, see Monitoring metrics.

5.4.8.2. Metrics

Instance monitoring is divided into resource monitoring and engine monitoring. Engine metrics are classified into metrics at the PolarDB-X instance level and metrics at the PolarDB-X database level. When some engine metrics are abnormal, you can check the metrics of each database to locate the database that has performance problems. The following table describes the metrics of these two types in details.

Metric	Category	Description	Data collection cycle	Data retention period	Description
CPU Utilization (%)	Resource monitoring	The average CPU utilization of PolarDB-X server nodes.	1 minute	3 days	None
Memory Usage (%)	Resource monitoring	The memory usage of the JVM old generation on PolarDB-X server nodes.	1 minute	3 days	Memory usage fluctuations are normal.
Inbound Traffic (Kbps)	Resource monitoring	The total inbound network traffic of PolarDB-X server nodes.	1 minute	3 days	Inbound network traffic is generated when ApsaraDB RDS for MySQL returns data to PolarDB-X.

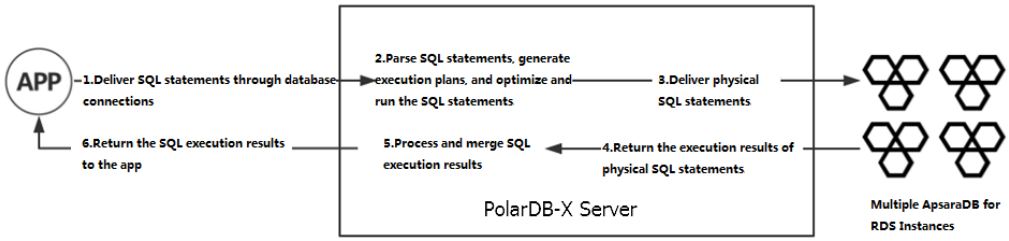
Metric	Category	Description	Data collection cycle	Data retention period	Description
Outbound Traffic (Kbps)	Resource monitoring	The total outbound network traffic of PolarDB-X server nodes.	1 minute	3 days	Outbound network traffic is generated when a PolarDB-X instance routes a physical SQL statement to an AparaDB RDS for MySQL instance or a PolarDB-X instance returns data to an application.
Logical QPS	Engine Monitoring	The total number of SQL statements processed per second on PolarDB-X server nodes.	5 seconds	7 days	None
Physical QPS	Engine Monitoring	The total number of SQL statements sent from PolarDB-X server nodes to ApsaraDB RDS for MySQL per second.	5 seconds	7 days	One logical SQL statement can be partitioned into multiple physical SQL statements.
Logical RT	Engine Monitoring	The average RT for processing each SQL statement by PolarDB-X.	5 seconds	7 days	If a logical SQL statement is partitioned into physical SQL statements, the logical RT of the SQL statement contains the RT of the physical SQL statements.

Metric	Category	Description	Data collection cycle	Data retention period	Description
Physical RT	Engine Monitoring	The average RT for processing SQL statements that are sent from PolarDB-X to ApsaraDB RDS for MySQL.	5 seconds	7 days	None
Connections	Engine Monitoring	The total number of connections established between an application and PolarDB-X.	5 seconds	7 days	The connections from PolarDB-X to ApsaraDB RDS for MySQL are not included.
Active Threads	Engine Monitoring	The number of threads that are used by PolarDB-X to execute SQL statements.	5 seconds	7 days	None

5.4.8.3. How metrics work

Before you analyze metrics, you must understand the execution process of SQL statements on PolarDB-X.

PolarDB-X SQL execution flowchart



In the entire SQL execution process, the execution status of Step 2 to Step 4 is reflected by various metrics of PolarDB-X.

- In Step 2, SQL parsing, optimization, and execution consume CPU resources. A complex SQL statement may have a complex structure or ultra-long length. A more complex SQL statement consumes more CPU resources. You can run the **TRACE** command to trace the SQL execution process. You can view the time consumed for the optimization of an SQL statement. The longer time consumed indicates a

higher CPU utilization.

- In Step 3, the routing and execution of physical SQL statements consume I/O resources. You can analyze the execution status of physical SQL statements based on metrics such as logical queries per second (QPS), physical QPS, logical RT, and physical RT. For example, if the physical QPS is low and the physical RT is high, the current ApsaraDB RDS for MySQL instance is slowly processing SQL statements. You must check the performance of the ApsaraDB RDS for MySQL instance.
- In Step 5, the SQL execution results are processed and integrated. The execution results of physical SQL statements are converted. In most cases, only SQL metadata is converted, which consumes few resources. However, the CPU utilization is high for steps such as `heap sort`. For more information about how to determine the consumption of SQL statements in this step, see Details about a low SQL statement.

5.4.8.4. Prevent performance problems

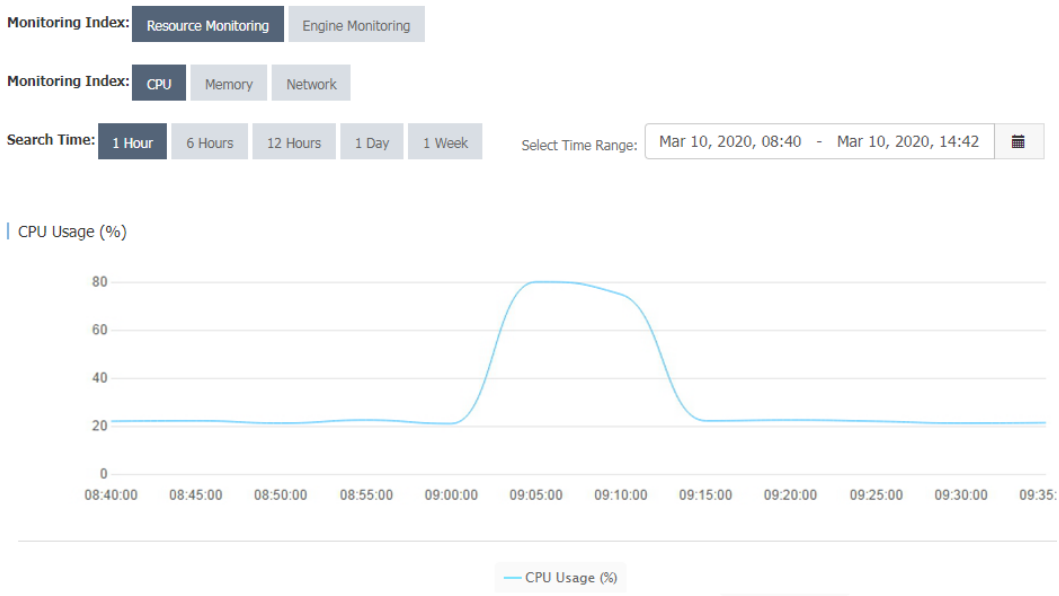
5.4.8.4.1. PolarDB-X CPU utilization

This topic describes the CPU utilization metric.

Performance metrics often fluctuate with system traffic, as shown in the following two common scenarios:

- An application has a shopping spree activity at 09:00 every morning. Therefore, the traffic of the system increases significantly at this time point. As shown by the monitoring data, the CPU utilization of the PolarDB-X instance increased from 20% to about 80% from 09:00. The peak lasts about 10 minutes.

  CPU utilization-1



- An application has increasing traffic. Therefore, the system traffic keeps increasing until it reaches a stable level. The monitored CPU utilization of the PolarDB-X instance also reflects this change.

  CPU utilization-2

When the load on the PolarDB-X instance changes with the business, you must take note of the changes in metrics. If the CPU utilization exceeds the threshold, you must upgrade the PolarDB-X instance specifications to alleviate the performance pressure.

You can set alert rules for instances in the PolarDB-X console. When the average CPU utilization exceeds the preset threshold, the system sends text messages to the corresponding contacts. You can set the CPU utilization threshold as needed. We recommend that you set it to 80%.

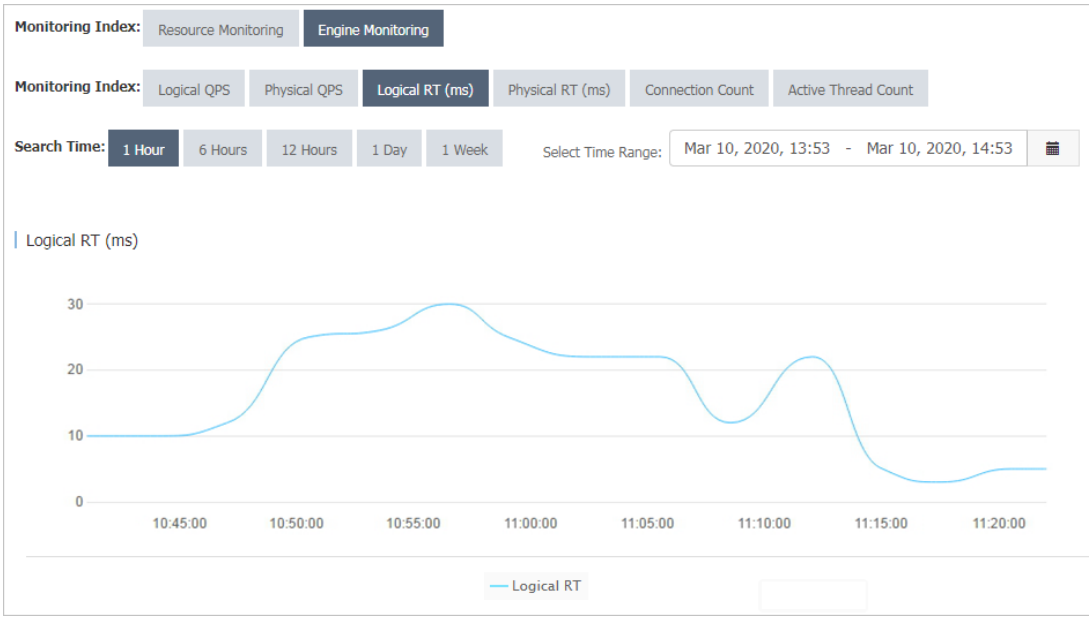
5.4.8.4.2. Logical RT and physical RT

This topic describes two metrics: logical RT and physical RT.

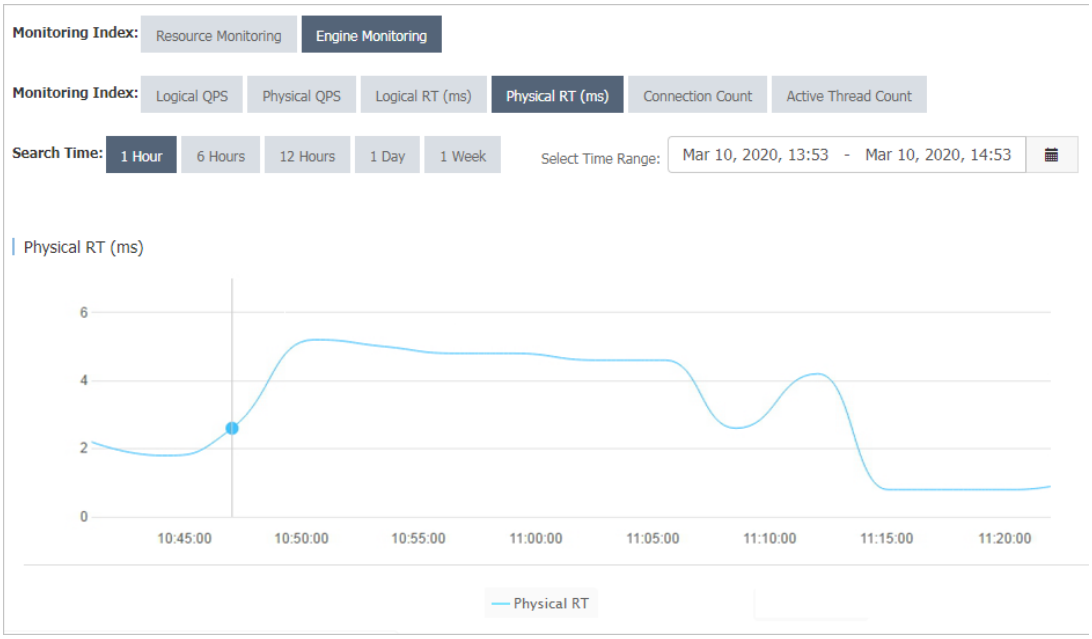
Logical RT is the time from when a PolarDB-X instance receives a logical SQL statement to when it returns data to an application. Physical RT is the time from when a PolarDB-X instance routes a physical SQL statement to an ApsaraDB RDS for MySQL instance to when it receives the data returned by the ApsaraDB RDS for MySQL instance.

If a logical SQL statement is partitioned into one or more physical SQL statements, the logical RT is greater than or equal to the physical RT. Ideally, PolarDB-X performs only a few operations on the data returned by ApsaraDB RDS for MySQL. Therefore, logical RT is slightly longer than physical RT. Under special circumstances, physical SQL queries are fast executed, whereas logical SQL queries are slowly executed. The following figure shows the trends of the logical RT and physical RT.

Logical RT



Physical RT



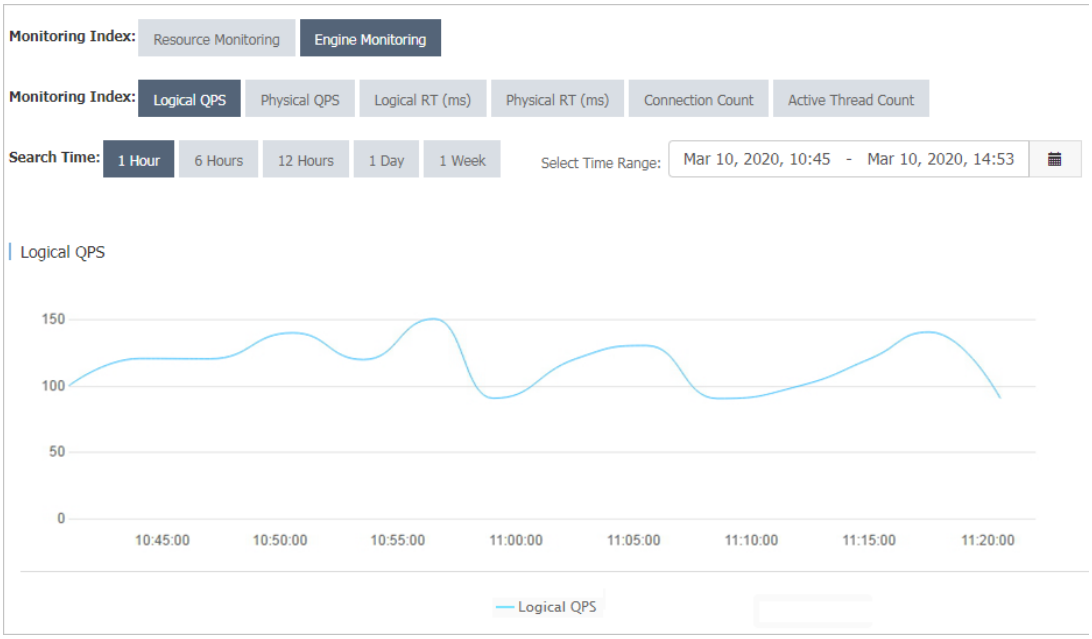
As shown in the preceding figures, the change trends of logical RT and physical RT in the two monitoring charts are basically the same. The logical RT fluctuates between 10 ms and 20 ms, and the physical RT fluctuates between 2 ms and 5 ms. This means that the PolarDB-X instance has a heavy load. To alleviate the load pressure, you can upgrade the PolarDB-X instance configuration. If both the logical RT and physical RT are high, you can upgrade the ApsaraDB RDS for MySQL configuration or optimize SQL statements on the ApsaraDB RDS for MySQL instance.

5.4.8.4.3. Logical QPS and physical QPS

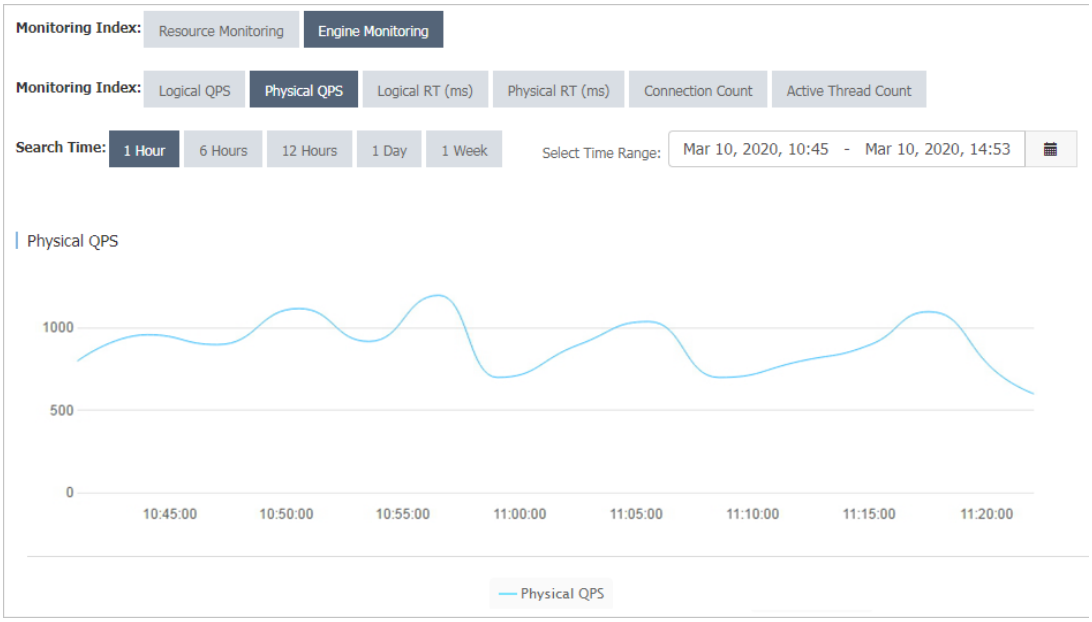
This topic describes two metrics: logical QPS and physical QPS.

As shown by the monitoring data in the following figures, the logical QPS and physical QPS have the same trends. However, the logical QPS and physical QPS have a large difference in value and are in a specific proportion.

Logical QPS



Physical QPS



As shown in the preceding figures, logical QPS fluctuates between 80 and 150, and physical QPS fluctuates between 700 and 1,200.

The reason is that PolarDB-X generates physical SQL statements based on logical SQL statements. The ratio of logical SQL statements to physical SQL statements is not necessarily 1:1. For example, a PolarDB-X logical table is created with the following statement:

```
CREATE TABLE drds_user
(id int,
name varchar(30))
dbpartition by hash(id);
```

When the query condition contains the database shard key, PolarDB-X routes the logical SQL statement to the ApsaraDB RDS for MySQL instance for execution. The following execution plan shows that the number of physical SQL statements is 1:

```
mysql> explain select name from drds_user where id = 1;
+----------------------------+--------------------------------------------------------------------------------+--------+
| GROUP_NAME                 | SQL                                                                            | PARAMS |
+----------------------------+--------------------------------------------------------------------------------+--------+
|   SANGUAN_BSQT_0001_RDS | select `drds_user`.`name` from `drds_user` where (`drds_user`.`id` = 1
) | {}   |
+----------------------------+--------------------------------------------------------------------------------+--------+
```

When the query does not contain the database shard key, PolarDB-X partitions the logical SQL statement into multiple physical SQL statements. The following execution plan shows that the number of physical SQL statements is 8:

```
mysql> explain select name from drds_user where name = 'LiLei';
+----------- -------------+--------------------------------------------------------------------------------+--------+
| GROUP_NAME           | SQL                                                      | PARAMS |
+-------------------------+--------------------------------------------------------------------------------+--------+
| SANGUAN_BSQT_0001_RDS    | select `drds_user`.`name` from `drds_user` where (`drds_user`.`name
` = 'LiLei') | {}    |
| SANGUAN_BSQT_0001_RDS    | select `drds_user`.`name` from `drds_user` where (`drds_user`.`name
` = 'LiLei') | {}    |
| SANGUAN_BSQT_0001_RDS    | select `drds_user`.`name` from `drds_user` where (`drds_user`.`name
` = 'LiLei') | {}    |
| SANGUAN_BSQT_0001_RDS    | select `drds_user`.`name` from `drds_user` where (`drds_user`.`name
` = 'LiLei') | {}    |
| SANGUAN_BSQT_0001_RDS    | select `drds_user`.`name` from `drds_user` where (`drds_user`.`name
` = 'LiLei') | {}    |
| SANGUAN_BSQT_0001_RDS    | select `drds_user`.`name` from `drds_user` where (`drds_user`.`name
` = 'LiLei') | {}    |
| SANGUAN_BSQT_0001_RDS    | select `drds_user`.`name` from `drds_user` where (`drds_user`.`name
` = 'LiLei') | {}    |
| SANGUAN_BSQT_0001_RDS    | select `drds_user`.`name` from `drds_user` where (`drds_user`.`name
` = 'LiLei') | {}    |
+-------------------------+--------------------------------------------------------------------------------+--------+
8 rows in set (0.06 sec)
```

Logical or physical QPS indicates the total number of logical or physical SQL statements processed per unit of time. When most SQL statements in the system contain the shard key, the ratio of logical QPS to physical QPS is close to 1:1. If the difference between the logical and physical QPS is too large, many SQL statements of the current application do not contain the shard key. In this case, check the SQL statements of the application to improve performance.

5.4.8.4.4. High memory usage

This topic describes the memory usage metric.

The overly high memory usage of a PolarDB-X instance is mostly caused by the large number of SQL queries in your application and the overlarge result set that is returned. If the memory usage of your PolarDB-X instance remains at about 100%, perform the Restart a PolarDB-X instance operations to locate and optimize the slow SQL queries of your application.

5.4.9. View the instance version

This topic describes two methods that you can use to view the version of a PolarDB-X instance.

View the instance version in the console

1. Log on to the PolarDB-X console.

2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the **Configuration Information** section, view the value of **Current Version**.

View the instance version by using the *version()* function

Use the MySQL command line to connect to PolarDB-X and execute the `SELECT version()` statement to view the version of the PolarDB-X instance. For example, the following result is returned after you execute the `selectversion();` statement:

```
+----------------------------+
| VERSION()                  |
+----------------------------+
| 5.6.29-TDDL-5.1.28-1320920 |
+----------------------------+
1 row in set (0.00 sec)
```

In the preceding example, 5.1.28-1320920 is the version of the PolarDB-X instance.

5.5. Account management

5.5.1. Basic concepts

This topic introduces the basic concepts of the PolarDB-X account and permission system.

The usage of the account and permission system in PolarDB-X is the same as in MySQL. PolarDB-X supports statements such as `GRANT`, `REVOKE`, `SHOW GRANTS`, `CREATE USER`, `DROP USER`, and `SET PASSWORD`. PolarDB-X allows you to grant permissions at the database and table levels, but does not allow you to grant permissions at the global or column level.

For more information about the MySQL account and permission system, see MySQL documentation.

> **Notice** Accounts that you create by using the `CREATE USER` statement in a PolarDB-X instance exist only in the PolarDB-X instance. The accounts will not be synchronized to the backend ApsaraDB RDS for MySQL instances.

Accounts

An account is a combination of a username and a hostname in the username@'host' format. Accounts that have the same username but different hostnames are different accounts. For example, lily@30.9.73.96 and lily@30.9.73.100 are two different accounts, and their passwords and permissions may be different.

After a database is created in the PolarDB-X console, the system automatically creates two system accounts for the database: the administrator account and the read-only account. These two accounts are built-in accounts. You cannot delete them or modify their permissions.

- The administrator account name is the same as the database name. For example, if the database

name is easydb, the administrator account name is also easydb.

- The read-only account name is the database name suffixed with _RO. For example, if the database name is easydb, the read-only account name is easydb_RO.

Assume that the dreamdb and andordb databases are available. Based on the preceding rules, the dreamdb database contains the administrator account named dreamdb and the read-only account named dreamdb_RO. The andordb database contains the administrator account named andordb and the read-only account named andordb_RO.

Account rules

- An administrator account has all permissions.
- You can use only an administrator account to create accounts and grant permissions. Other accounts can only be created and granted permissions by the administrator account.
- An administrator account is bound to a database and does not have permissions on other databases. The administrator account can access only the bound database, but cannot grant permissions on other databases to an account. For example, the easydb administrator account can connect only to the easydb database, and can grant only the permissions on the easydb database or tables in the easydb database to an account.
- A read-only account has only the SELECT permission.

Username rules

- Usernames are not case-sensitive.
- A username must be 4 to 20 characters in length.
- A username must start with a letter.
- A username can contain uppercase letters, lowercase letters, and digits.

Password rules

- A password must be 6 to 20 characters in length.
- A password can contain uppercase letters, lowercase letters, digits, and the following special characters:

  @#$%^&+=

Hostname matching rules

- A hostname must be an IP address. It can contain underscores (_) and percent signs (%). An underscore (_) represents a single character and a percent sign (%) represents zero or more characters. Hostnames that contain wildcards must be quoted with single quotation marks ('), for example, `lily@'30.9. %.%'` and `david@'%'`.
- If two usernames in the system can be used to log on to the database, the username with the longest prefix (the longest IP segment excluding wildcards) prevails. For example, the `david@'30.9.12_.234'` and `david@'30.9.1%.234'` accounts are available in the system. If you use the david username to log on to a database from the 30.9.127.234 host, the `david@'30.9.12_.234'` account is used.
- When you activate the VPC service, the IP address of the host changes. To avoid invalid configurations in the account and permission system, set the hostname to `'%'` to match all IP addresses.

Permission levels

The following list describes the support for permissions of different levels:

- Global permissions (not supported)
- Database-level permissions (supported)
- Table-level permissions (supported)
- Column-level permissions (not supported)
- Subprogram-level permissions (not supported)

Permissions

Eight table-associated basic permissions are supported: `CREATE`, `DROP`, `ALTER`, `INDEX`, `INSERT`, `DELETE`, `UPDATE`, and `SELECT`.

- The `TRUNCATE` statement requires the table-level DROP permission.
- The `REPLACE` statement requires the table-level INSERT and DELETE permissions.
- The `CREATE INDEX` and `DROP INDEX` statements require the table-level INDEX permission.
- The `CREATE SEQUENCE` statement requires the database-level CREATE permission.
- The `DROP SEQUENCE` statement requires the database-level DROP permission.
- The `ALTER SEQUENCE` statement requires the database-level ALTER permission.
- The `INSERT ON DUPLICATE UPDATE` statement requires the table-level INSERT and UPDATE permissions.

Permission rules

- Permissions are bound to an account (`username@'host'`) instead of a username (`username`).
- An error occurs if the table does not exist during authorization.
- The database permissions are listed by level in descending order: global permissions (not supported), database-level permissions, table-level permissions, and column-level permissions (not supported). A granted higher-level permission overwrites a lower-level permission. If you remove the higher-level permission, the lower-level permission is also removed.
- USAGE authorization is not supported.

5.5.2. Create an account

This topic describes how to create a PolarDB-X account by using the PolarDB-X console and SQL statements.

Prerequisites

You have created or added a database. For more information, see Create a database.

Create an account by using the console

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to

access the **Basic Information** page.

4. In the left-side navigation pane, choose **Configuration and Management > Account Management**.
5. On the **Accounts** page, click **Create Account** in the upper-right corner.
6. Set the following parameters.

Parameter	Description
Database Account	Enter a name for the account. The name of the account must meet the following requirements: ○ The name must be 4 to 20 characters in length. ○ It must start with a letter and end with a letter or digit. ○ The name can contain uppercase letters, lowercase letters, digits, and underscores (_).
New Password	Enter a password for the account. The password of the account must meet the following requirements: ○ The password must be 8 to 32 characters in length. ○ The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. ○ The password can contain the following special characters: ! @#$%^&*()_+-=
Confirm New Password	Enter the password again.
Authorize Databases	You can grant permissions on one or more databases to the account. i. From the Databases section, select one or more databases. Then, click **Add** to add them to the Authorized Databases section. ii. In the Authorized Databases section, select **Read/Write**, **Read-only**, **DDL Only**, or **DML Only** for the specified database. **Note** You can also grant permissions on multiple added databases by clicking **Set All to Read-only**, **Set All to DDL Only**, **Set All to DML Only**, or **Set All to Read/Write** in the upper-right corner of the Authorized Databases section. The buttons in the upper-right corner change after you click them. For example, after you click **Set All to Read-only**, this button is changed to **Set All to DDL Only**.

7. Click **OK**.

Create an account by using the command line

Syntax

```
CREATE USER user_specification [, user_specification] ...
 user_specification: user [ auth_option ]
 auth_option: IDENTIFIED BY 'auth_string'
```

Examples

Create an account whose name is lily and password is 123456. The account can be used to log on to the databases only from 30.xx.xx.96.

```
CREATE USER lily@30.xx.xx.96 IDENTIFIED BY '123456';
```

Create an account that is named david and has no password. The account can be used to log on to the databases from all hosts.

```
CREATE USER david@'%';
```

5.5.3. Reset the password

When you use PolarDB-X, you can reset the password of your database account by using the PolarDB-X console or the command line.

> Note
> - Accounts that have root permissions cannot be deleted or modified.
> - For data security, we recommend that you change your password on a regular basis.

Reset the password by using the console

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More** > **Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the left-side navigation pane, choose **Configuration and Management** > **Account Management**.
5. Find the account for which you want to reset the password, and click **Reset Password** in the Actions column.
6. In the **Reset Account Password** dialog box, enter the new password in the **New Password** field and enter the password again in the **Confirm New Password** field.

> **Note** The password must meet the following requirements:
>
> - The password must be 8 to 32 characters in length.
> - The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters.
> - The password can contain the following special characters:
>
> ! @#$%^&*()_+-

7. After you confirm that the password is correct, click **OK**.

Reset the password by using the command line

Syntax

```
SET PASSWORD FOR user = password_option
password_option: {
  PASSWORD('auth_string')
}
```

Examples

Change the password of the lily@30.xx.xx.96 account to 123456.

```
SET PASSWORD FOR lily@30.xx.xx.96 = PASSWORD('123456')
```

5.5.4. Modify the permissions of an account

You can modify the account permissions of your PolarDB-X instances at any time.

Considerations

- The permissions of a privileged account cannot be modified.
- In the PolarDB-X console, you can grant only DML, DDL, read-only, and read and write permissions to standard accounts. If you need more fine-grained authorization, use the command line.

Modify account permissions in the console

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More** > **Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the left-side navigation pane, choose **Configuration and Management** > **Account Management**.
5. Find the account whose permissions you want to modify, and click **Modify Permission** in the Actions column.
6. In the **Modify Permissions** dialog box, **grant** or **remove** the permissions on one or more databases to or from the account.

- Add databases:

  From the Databases section, select one or more databases. Then, click **Add** to add them to the Authorized Databases section on the right.

- Remove databases:

  From the Authorized Databases section, select one or more databases. Then, click **Remove** to move them to the Databases section on the left.

- Modify permissions on added databases:

  In the Authorized Databases section, select **Read/Write**, **Read-only**, **DDL Only** or **DML Only** for the specified database.

  > **Note** You can also grant permissions on multiple added databases by clicking **Set All to Read-only**, **Set All to DDL Only**, **Set All to DML Only**, or **Set All to Read/Write** in the upper-right corner of the Authorized Databases section.
  >
  > The buttons in the upper-right corner change after you click them. For example, after you click **Set All to Read-only**, this button is changed to **Set All to DDL Only**.

7. After the configuration is completed, click **OK**.

GRANT statement

Syntax

```
GRANT
  priv_type[, priv_type] ...
  ON priv_level
  TO user_specification [, user_specification] ...
  [WITH GRANT OPTION]
priv_level: {
 | db_name.*
 | db_name.tbl_name
 | tbl_name
}
user_specification:
  user [ auth_option ]
auth_option: {
  IDENTIFIED BY 'auth_string'
}
```

> Notice
>
> - If the account specified in the GRANT statement does not exist and no IDENTIFIED BY information is provided, an error message is returned. The error message indicates that the account does not exist.
> - If the account specified in the GRANT statement does not exist but the IDENTIFIED BY information is provided, the account is created and granted with the specified permissions.

For example, create an account that has the username david. The account can be used to log on to the easydb database from all hosts and has all the permissions on the easydb database.

Method 1: Create an account. Then, grant permissions to the account.

```
CREATE USER david@'%' IDENTIFIED BY 'your#password';
GRANT ALL PRIVILEGES ON easydb.* to david@'%';
```

Method 2: Create an account and grant permissions to the account by executing only one statement.

```
GRANT ALL PRIVILEGES ON easydb.* to david@'%' IDENTIFIED BY 'your#password';
```

Create an account that has the username hanson. The account can be used to log on to the easydb database from all hosts and has all the permissions on the easydb.employees table.

```
GRANT ALL PRIVILEGES ON easydb.employees to hanson@'%' IDENTIFIED BY 'your#password';
```

Create an account that has the username hanson. The account can be used to log on to the easydb database from only 192.xx.xx.10 and has the INSERT and SELECT permissions on the easydb.emp table.

```
GRANT INSERT,SELECT ON easydb.emp to hanson@'192.xx.xx.10' IDENTIFIED BY 'your#password';
```

Create a read-only account that has the username actro. The account can be used to log on to the easydb database from all hosts.

```
GRANT SELECT ON easydb.* to actro@'%' IDENTIFIED BY 'your#password';
```

REVOKE statement

Syntax

- Delete specific permissions from an account: Delete the permissions at a specific level from an account. The permission level is specified by priv_level.

```
REVOKE
  priv_type
   [, priv_type] ...
  ON priv_level
  FROM user [, user] ...
```

- Delete all permissions from an account: Delete all permissions at the database and table levels from an account.

```
REVOKE ALL PRIVILEGES, GRANT OPTION
  FROM user [, user] ...
```

Examples

Delete the CREATE, DROP, and INDEX permissions on the easydb.emp table from the hanson@'%' account.

```
REVOKE CREATE,DROP,INDEX ON easydb.emp FROM hanson@'%';
```

Delete all permissions from the lily@30.xx.xx.96 account.

```
REVOKE ALL PRIVILEGES,GRANT OPTION FROM lily@30.xx.xx.96;
```

Notice GRANT OPTION must be added to the preceding statement for compatibility with MySQL.

SHOW GRANTS statement

Syntax

```
SHOW GRANTS[FOR user@host];
```

Query all permissions:

```
SHOW GRANTS;
```

Query the permissions of an account:

```
SHOW GRANTS FOR user@host;
```

5.5.5. Delete an account

You can delete an account in the Cloud Native Distributed Database PolarDB-X (PolarDB-X) console or by using the command line.

Delete an account in the PolarDB-X console

Note You can delete only standard accounts that are created in the console.

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to

access the **Basic Information** page.

4. In the left-side navigation pane, choose **Configuration and Management > Account Management**.
5. Find the target account and click **Delete**.
6. In the **Delete Account** dialog box, click **OK**.

Delete an account by using the command line

Use the following syntax rule:

```
DROP USER user [, user] ...
```

For example:

Delete the lily@30.xx.xx.96 account.

```
DROP USER lily@30.xx.xx.96;
```

5.6. Database management

5.6.1. Create a database

After you create a PolarDB-X instance, you must create a database that is based on one or more ApsaraDB RDS for MySQL instances.

Prerequisites

- You have created an ApsaraDB RDS for MySQL instance in the same department of PolarDB-X.
- You have granted the Resource Access Management (RAM) permissions. For more information, see the *RAM* topic in the *Apsara Stack User Guide*.

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to access the **Basic Information** page.
4. On the **Basic Information** page, click **Create Database** in the upper-right corner.
5. In the **Enter Basic Information** step, enter the following information.

Parameter	Description
Partition Mode	Select **Horizontal Partitioning**.

Parameter	Description
Database Name	Enter a PolarDB-X database name. The name must meet the following requirements: ○ It must be 2 to 24 characters in length. ○ It must start with a letter and end with a letter or digit. ○ It can contain lowercase letters, digits, and underscores (_). ○ It must be unique on the PolarDB-X instance.
Character Set	Select utf8, gbk, latin1, or utf8mb4.
DRDS Link Password	Set the connection password for the PolarDB-X database. The password must meet the following requirements: ○ It must be 8 to 30 characters in length. ○ It must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and underscores (_).
Confirm Password	Enter the password again.

6. Click **Next**.
7. In the **Select RDS Instance** step, click the **Buy New RDS Instance** or **Use Existing RDS Instance** tab.
   i. **Buy New RDS Instance**: Click the **Buy New RDS Instance** tab.
   ii. Set **Storage Type**, **Series**, **Instance Specifications**, **Storage Capacity**, **Availability Zone**, and **Quantity**.
   iii. Click **Next**.
   i. **Use Existing RDS Instance**: Click the **Use Existing RDS Instance** tab.
   ii. In the left-side section, select the ApsaraDB RDS for MySQL instances to be added.
   iii. Click > to move the selected instances to the **Selected RDS Instances** section on the right.
   iv. Click **Next**.
8. After the precheck succeeds in the **Precheck** step, click **Next**.

   Note If the precheck fails, fix the issue as prompted.
9. In the **Preview** step, click **Next**. Wait until the database is created.

5.6.2. View a database

After the database is created, you can view the basic information of the database on the Database Management page in the console.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.

3. Click the target instance ID or choose **More** > **Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the left-side navigation pane, choose **Configuration and Management** > **Database Management**.
5. Find the target database, and click **Manage** in the Actions column. The **Basic Information** page of the database appears.

> **Note** On the **Basic Information** page, you can delete the database or reset the password.

What's next

PolarDB-X is fully compatible with the MySQL protocol. You can use **Command Line URL** on the MySQL client to connect to the PolarDB-X instance and enter the user name and password to log on to the PolarDB-X database. When you use the MySQL client, note the following points:

> **Note**
>
> - MySQL clients of some earlier versions have limits on the user name length, which cannot exceed 16 characters. The PolarDB-X database name and user name are the same. If the database name exceeds 16 characters in length, an error is reported.
> - When you use the MySQL client, you must add the `-c` parameter to the HINT command. In PolarDB-X, an annotation is used to implement HINT. If the `-c` parameter is not added, the annotation is lost and the HINT of PolarDB-X is lost.

5.6.3. Perform smooth scale-out

The underlying storage of the logical database may have physical bottlenecks. For example, the remaining disk space is about 30%. To solve the bottlenecks, you can smoothly scale out the database to improve the performance. The smooth scale-out process is divided into four steps: configuration, migration, switchover, and cleanup.

Configuration

> **Note** In smooth scale-out, ApsaraDB RDS for MySQL instances are added, and some source database shards are migrated to the new ApsaraDB RDS for MySQL instances. This way, the overall data storage capacity is increased, and the number of requests that a single ApsaraDB RDS for MySQL instance needs to process is reduced.

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More** > **Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the left-side navigation pane, choose **Configuration and Management** > **Database Management**.
5. Find the database that you want to smoothly scale out, and click **Manage** in the Actions column. The **Basic Information** page of the database appears.

6. In the left-side navigation pane, choose **Configuration and Management > Scale-out Management**.
7. In the upper-right corner of the **Scale-out Management** page, click **Scale Out**.
8. Select **Smooth Scale-out** and click **Next**.
9. After the precheck succeeds in the **Precheck** step, click **Next**.

   > **Note** If the precheck fails, fix the issue as prompted.

10. In the **Select RDS Instance** step, click the **Buy New RDS Instance** or **Use Existing RDS Instance** tab. **Buy New RDS Instance**:
    i. Click the **Buy New RDS Instance** tab.
    ii. Set **Storage Type**, **Series**, **Instance Specifications**, **Storage Capacity**, **Availability Zone**, and **Quantity**.
    iii. Click **Next**.

    Use Existing RDS Instance:
    i. Click the **Use Existing RDS Instance** tab.
    ii. In the left-side section, select the ApsaraDB RDS for MySQL instances to be added.
    iii. Click > to move the selected instances to the **Selected RDS Instances** section on the right.
    iv. Click **Next**.
11. In the **Preview** step, click **Start Scale-out**.

    > **Note** By default, the console evenly distributes the physical database shards to the ApsaraDB RDS for MySQL instances that you added. You can also manually add or delete physical database shards to or from the new ApsaraDB RDS for MySQL instances.

12. Click the icon in the upper-right corner to view the progress of the scale-out task.

Migration

Some physical database shards are migrated during smooth scale-out.

The migration does not modify the data in the database or affect online services. Before the switchover, you can cancel the smooth scale-out operation by using a rollback.

> **Note**
>
> - This is because before the switchover, the current scale-out operation does not have a real impact on the existing data in the database.
> - During scale-out, the binary log files of the source RDS instance are not cleaned, which may result in insufficient disk space. Therefore, you must reserve sufficient disk space on the source ApsaraDB RDS for MySQL instance. We recommend that you reserve more than 30% of the disk space. If the disk space is not sufficient, you can submit a ticket to expand the storage space of the ApsaraDB RDS for MySQL instance.
> - To reduce the pressure of read operations on the source ApsaraDB RDS for MySQL instance, perform scale-out when the load on the source ApsaraDB RDS for MySQL instance is low.
> - During the scale-out, do not submit DDL tasks in the console or connect to the PolarDB-X instance to execute DDL statements. Otherwise, the scale-out task may fail.
> - Before the scale-out, make sure that all tables in the source database have primary keys.

After historical data and incremental data are migrated, the migration progress reaches 100%. Then, you can **switch** the read and write traffic to the new ApsaraDB RDS for MySQL instance or **roll back** to cancel the scale-out.

Switchover

The switchover task switches the read and write traffic to the new ApsaraDB RDS for MySQL instance. The whole process takes 3 to 5 minutes. During the switchover, the service is not affected except for one or two transient disconnections. Perform the switchover during off-peak hours.

1. In the upper-right corner of the **Basic Information** page, click the  icon. The **Task Progress** dialog box appears.
2. In the **Task Progress** dialog box, click **Switch Over** and click **OK**.

   During the switchover, a switchover task is generated and appears in the task progress.
3. After the switchover is completed, the **Clean Up** button appears in the **Task Progress** dialog box.

Cleanup

In this step, the migrated database shards are deleted from the source ApsaraDB RDS for MySQL instance.

1. After switchover is completed, click **Clean Up** next to the task.
2. Click **OK**. A cleanup task appears in the Task Progress dialog box.

> Note
>
> - The cleanup task is an asynchronous task. You can view the execution status in the Task Progress dialog box.
> - After the cleanup task is completed, the smooth scale-out process ends. The new ApsaraDB RDS for MySQL instance becomes the storage node of the PolarDB-X logical database.
> - You can implement smooth scale-out by migrating physical database shards. If the number of database shards exceeds the capacity of a single ApsaraDB RDS for MySQL instance, no further scale-out is allowed. In this case, you can submit a ticket to apply for increasing the number of database shards and scaling out the database. Then, data will be reallocated based on the hash algorithm.
> - The cleanup task deletes database shards that are no longer used after the current scale-out. You can back up the database shards before you run the cleanup task.
> - The cleanup operation brings pressure to databases. We recommend that you perform this operation during off-peak hours.

5.6.4. View database monitoring information

PolarDB-X displays the historical monitoring information of a PolarDB-X database in two dimensions: data metrics and query time.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More** > **Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the left-side navigation pane, choose **Monitoring and Alerts** > **Database Monitoring**.
5. Select the target **database**, and then set **Data Indexes** and **Query Time**. You can see the corresponding monitoring information.

> Note For more information about instance-level monitoring, see View monitoring information.

5.6.5. Set the IP address whitelist

PolarDB-X allows you to configure the IP address whitelist to block unauthorized access requests. This topic describes how to configure the IP address whitelist.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More** > **Manage** from the Actions column of the instance to access the **Basic Information** page.

4. In the left-side navigation pane, choose **Configuration and Management > Database Management**.
5. Find the target database, and click **Manage** in the Actions column. The **Basic Information** page of the database appears.
6. On the **Basic Information** page of the database, choose **Data Security > Whitelist Settings** in the left-side navigation pane.
7. On the Whitelist Settings page, click **Manually Modify**.
8. Enter the IP addresses that are allowed to access the database, and click **OK**.

> Note
> - The following formats are supported in the whitelist:
>   - Single IP addresses, such as 192.168.1.1.
>   - IP addresses in CIDR format, such as 192.168.1.1/24.
>   - IP addresses that include an asterisk (*) as a wildcard, such as 192.168.1. *. This example indicates that hosts with an IP address in the range from 192.168.1.1 to 192.168.1.254 are allowed to access the database.
>   - IP range, such as 192.168.1.1-192.168.1.254.
> - If you need to add multiple IP addresses or IP ranges, separate them with commas (,). Do not use spaces before and after the commas, for example, 192.168.0.1,172.16.213.9.

5.6.6. Delete a database

This topic describes how to delete a database in the Cloud Native Distributed Database PolarDB-X (PolarDB-X) console.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the left-side navigation pane, choose **Configuration and Management > Database Management**.
5. Find the target database and click **Delete**.

> **Warning** You cannot recover databases that have been deleted. Exercise caution when you perform this operation.

6. In the **Delete Database** dialog box, click **OK**.

5.6.7. Fix database shard connections

This topic describes how to manually fix database shard connections when a PolarDB-X instance cannot access an ApsaraDB RDS for MySQL instance.

Context

When you use a PolarDB-X instance, you must access the mounted ApsaraDB RDS for MySQL instances. If the network configuration of a connected ApsaraDB RDS for MySQL instance changes, the network connection between the PolarDB-X instance and the ApsaraDB RDS for MySQL instance is broken. For example, the network configuration changes if the zone is switched or the network type is changed from the classic network to VPC. Therefore, the PolarDB-X instance cannot access the ApsaraDB RDS for MySQL instance. In this case, you must manually fix the database shard connection in the PolarDB-X console. This way, you can recover the network connection between the PolarDB-X instance and the ApsaraDB RDS for MySQL instance.

Procedure

1. Log on to the PolarDB-X console.
2. Find the target instance in the instance list.
3. Click the target instance ID or choose **More > Manage** from the Actions column of the instance to access the **Basic Information** page.
4. In the left-side navigation pane, choose **Configuration and Management > Database Management**.
5. Find the target database, and click **Manage** in the Actions column. The **Basic Information** page of the database appears.
6. In the **Shortcuts** section, click **Fix Database Shard Connections**.
7. In the message that appears, click **OK**.

5.7. Custom control commands

PolarDB-X provides a series of auxiliary SQL commands to help you conveniently use PolarDB-X.

5.7.1. Overview

PolarDB-X provides unique auxiliary statements for you to use and maintain PolarDB-X.

Syntax description: The identifier provided by the user is in [] and optional content is in (). In addition, this document applies to the current version. If some statements are unavailable, the version is earlier than required.

5.7.2. SHOW HELP statement

This topic describes how to use the SHOW HELP statement to view all the auxiliary SQL statements of PolarDB-X and their descriptions.

Execute the `SHOW HELP;` statement. The following response returned:

```
+---------------------------------------+-------------------------------------------------------+-----------------------------------------
--+
| STATEMENT                   | DESCRIPTION                          | EXAMPLE                      |
+---------------------------------------+-------------------------------------------------------+-----------------------------------------
--+
| show rule                   | Report all table rule                |                              |
| show rule from TABLE        | Report table rule                    | show rule from user          |
| show full rule from TABLE   | Report table full rule               | show full rule from user     |
| show topology from TABLE    | Report table physical topology       | show topology from user
|
| show partitions from TABLE  | Report table dbPartition or tbPartition columns  | show partitions fr
om user          |
| show broadcasts             | Report all broadcast tables          |                              |
| show datasources            | Report all partition db threadPool info  |                          |
| show node                   | Report master/slave read status      |                              |
| show slow                   | Report top 100 slow sql              |                              |
| show physical_slow          | Report top 100 physical slow sql     |                              |
| clear slow                  | Clear slow data                      |                              |
| trace SQL                   | Start trace sql, use show trace to print profiling data | trace select count(*) from us
er; show trace |
| show trace                  | Report sql execute profiling info    |                              |
| explain SQL                 | Report sql plan info                 | explain select count(*) from user  |
| explain detail SQL          | Report sql detail plan info          | explain detail select count(*) from use
r  |
| explain execute SQL         | Report sql on physical db plan info  | explain execute select count(*) f
rom user  |
| show sequences              | Report all sequences status          |                              |
| create sequence NAME [start with COUNT] | Create sequence                  | create sequence test start
with 0      |
| alter sequence NAME [start with COUNT]  | Alter sequence                   | alter sequence test start with
100000    |
| drop sequence NAME          | Drop sequence                        | drop sequence test           |
+---------------------------------------+-------------------------------------------------------+-----------------------------------------
--+
20 rows in set (0.00 sec)
```

5.7.3. Statements for viewing rules and node topologies

This topic describes the statements that are used to view rules and node topologies, and provides examples of the statements.

SHOW RULE [FROM tablename]

You must take note of the following usage notes:

- `SHOW RULE` : You can execute this statement to view the partitioning information of each logical table in a database.
- `SHOW RULE FROM tablename` : You can execute this statement to view the partitioning information of a specified logical table in a database.

The following section describes the important columns:

- BROADCAST: indicates whether the table is a broadcast table. 0 indicates No and 1 indicates Yes.
- DB_PARTITION_KEY: indicates the database shard key. If no database shards exist, the parameter value is empty.
- DB_PARTITION_POLICY: indicates the database sharding policy. Valid values are hash and date policies such as YYYYMM, YYYYDD, and YYYYWEEK.
- DB_PARTITION_COUNT: indicates the number of database shards.
- TB_PARTITION_KEY: indicates the table shard key. If no table shards exist, the parameter value is empty.
- TB_PARTITION_POLICY: indicates the table sharding policy. Valid values are hash and date policies such as MM, DD, MMDD, and WEEK.
- TB_PARTITION_COUNT: indicates the number of table shards.

Execute the `SHOW RULE;` statement. The following response is returned:

```
+------+--------------+-----------+------------------+---------------------+--------------------+------------------+---------------------+---
-----------------+
| ID   | TABLE_NAME   | BROADCAST | DB_PARTITION_KEY | DB_PARTITION_POLICY | DB_PARTITION_COUNT | T
B_PARTITION_KEY | TB_PARTITION_POLICY | TB_PARTITION_COUNT |
+------+--------------+-----------+------------------+---------------------+--------------------+------------------+---------------------+---
-----------------+
|  0 | dept_manager |     0 |          | NULL         | 1          |          | NULL         | 1          |
|  1 | emp          |     0 | emp_no   | hash         | 8          | id       | hash         | 2          |
|  2 | example      |     0 | shard_key | hash        | 8          |          | NULL         | 1          |
+------+--------------+-----------+------------------+---------------------+--------------------+------------------+---------------------+---
-----------------+
3 rows in set (0.01 sec)
```

SHOW FULL RULE [FROM tablename]

You can execute this SQL statement to view the sharding rules of logical tables in a database. This statement shows more detailed information than the SHOW RULE statement.

The following section describes the important columns:

- BROADCAST: indicates whether the table is a broadcast table. 0 indicates No and 1 indicates Yes.

- JOIN_GROUP: a reserved field. Currently, it is meaningless.
- ALLOW_FULL_TABLE_SCAN: indicates whether to allow data query when no table shard key is specified for database or table sharding. If this parameter is set to True, each physical table is scanned to find data that meets the condition. This is a full table scan.
- DB_NAME_PATTERN: The digit 0 inside a pair of braces ({}) in DB_NAME_PATTERN is a placeholder. When the SQL statement is executed, this value is replaced by the value of DB_RULES_STR. The number of digits remains unchanged. For example, if the value of DB_NAME_PATTERN is SEQ_{0000}_RDS and the value of DB_RULES_STR is [1,2,3,4], four DB_NAME values are generated: SEQ_0001_RDS, SEQ_0002_RDS, SEQ_0003_RDS, and SEQ_0004_RDS.
- DB_RULES_STR: indicates the database sharding rule.
- TB_NAME_PATTERN: The digit 0 inside a pair of braces ({}) in TB_NAME_PATTERN is a placeholder. When the SQL statement is executed, this value is replaced by the value of TB_RULES_STR. The number of digits remains unchanged. For example, if the value of TB_NAME_PATTERN is table_{00} and the value of TB_RULES_STR is [1,2,3,4,5,6,7,8], eight tables are generated: table_01, table_02, table_03, table_04, table_05, table_06, table_07, and table_08.
- TB_RULES_STR: indicates the table sharding rule.
- PARTITION_KEYS: indicates the database and table shard keys. When both database sharding and table sharding are performed, the database shard key is placed before the table shard key.
- DEFAULT_DB_INDEX: indicates the database shard in which a single-database non-partitioned table is stored.

Execute the `SHOW FULL RULE;` statement. The following response is returned:

```
+------+-------------+-----------+------------+---------------------+-----------------------------------------------+----------------
--------------------------------+----------------+---------------------------------+---------------+--------------------------------
--------------+
| ID   | TABLE_NAME  | BROADCAST | JOIN_GROUP | ALLOW_FULL_TABLE_SCAN | DB_NAME_PATTERN
| DB_RULES_STR                  | TB_NAME_PATTERN | TB_RULES_STR             | PARTITION_KEYS | DEFA
ULT_DB_INDEX                  |
+------+-------------+-----------+------------+---------------------+-----------------------------------------------+----------------
--------------------------------+----------------+---------------------------------+---------------+--------------------------------
--------------+
|  0 | dept_manager |     0 | NULL    |           0 | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0000_R
DS  | NULL                          | dept_manager  | NULL                | NULL      | SEQ_TEST_148776778081
4RGKKSEQ_TEST_WNJG_0000_RDS |
|  1 | emp      |     0 | NULL    |           1 | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_{0000}_RDS | ((
#emp_no,1,8#).longValue().abs() % 8)          | emp_{0}     | ((#id,1,2#).longValue().abs() % 2) | emp_no    id | S
EQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0000_RDS |
|  2 | example  |     0 | NULL    |           1 | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_{0000}_RDS |
((#shard_key,1,8#).longValue().abs() % 8).intdiv(1) | example     | NULL                | shard_key    | SEQ_TE
ST_1487767780814RGKKSEQ_TEST_WNJG_0000_RDS |
+------+-------------+-----------+------------+---------------------+-----------------------------------------------+----------------
--------------------------------+----------------+---------------------------------+---------------+--------------------------------
--------------+
3 rows in set (0.01 sec)
```

SHOW TOPOLOGY FROM tablename

You can execute this SQL statement to view the topology of a specified logical table. The information contains the database shards to which data in the logical table is partitioned and the table shards in each database shard.

Execute the `SHOW TOPOLOGY FROM emp;` statement. The following response is returned:

```
+------+-------------------------------------------------+------------+
|ID  |GROUP_NAME                      |TABLE_NAME|
+------+-------------------------------------------------+------------+
|  0|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0000_RDS|emp_0   |
|  1|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0000_RDS|emp_1   |
|  2|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0001_RDS|emp_0   |
|  3|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0001_RDS|emp_1   |
|  4|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0002_RDS|emp_0   |
|  5|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0002_RDS|emp_1   |
|  6|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0003_RDS|emp_0   |
|  7|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0003_RDS|emp_1   |
|  8|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0004_RDS|emp_0   |
|  9|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0004_RDS|emp_1   |
| 10|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0005_RDS|emp_0   |
| 11|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0005_RDS|emp_1   |
| 12|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0006_RDS|emp_0   |
| 13|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0006_RDS|emp_1   |
| 14|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0007_RDS|emp_0   |
| 15|SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0007_RDS|emp_1   |
+------+-------------------------------------------------+------------+
16 rows in set (0.01 sec)
```

SHOW PARTITIONS FROM tablename

You can execute this SQL statement to view the database shard key and table shard key, which are separated with commas (,). If the returned results contain two values, both database sharding and table sharding are performed. The first value is the database shard key and the second value is the table shard key. If only one value is returned, only database sharding is performed. This value is the database shard key.

Execute the `SHOW PARTITIONS FROM emp;` statement. The following response is returned:

```
+-----------+
|KEYS    |
+-----------+
|emp_no,id|
+-----------+
1 row in set (0.00 sec)
```

SHOW BROADCASTS

You can execute this SQL statement to view broadcast tables.

Execute the `SHOW BROADCASTS;` statement. The following response is returned:

```
+------+------------+
| ID   | TABLE_NAME |
+------+------------+
|    0 | brd2       |
|    1 | brd_tbl    |
+------+------------+
2 rows in set (0.01 sec)
```

SHOW DATASOURCES

You can execute this SQL statement to view the information about the underlying storage, including the database name, database group name, connection URL, username, storage type, read and write weights, and connection pool information.

The following section describes the important columns:

- SCHEMA: indicates the database name.
- GROUP: indicates the database group name. Grouping aims to manage multiple groups of databases that have identical data. For example, the databases can be the primary and secondary databases after data replication that is implemented by ApsaraDB RDS for MySQL. It is used for read/write splitting and primary/secondary switchovers.
- URL: indicates the connection information of the underlying ApsaraDB RDS for MySQL instances.
- TYPE: indicates the type of the underlying storage. Currently, only ApsaraDB RDS for MySQL instances are supported.
- READ_WEIGHT: indicates the read weight of the database. When the primary ApsaraDB RDS for MySQL instance is under a heavy load of read requests, you can use the read/write splitting feature of PolarDB-X to distribute the read traffic. This way, the read pressure on the primary instance can be reduced. PolarDB-X automatically identifies the read and write traffic. It directs the write traffic to the primary ApsaraDB RDS for MySQL instance and the read traffic to all ApsaraDB RDS for MySQL instances based on the specified weight.
- WRITE_WEIGHT: indicates the write weight. For more information, see READ_WEIGHT.

Execute the `SHOW DATASOURCES;` statement. The following response is returned:

```
+------+----------------------------+----------------------------------------------+------------------------------------------------+------
----------------------------------------------------------------------+-----------+-------+------+------+------+-------------+---------
-+-------------+--------------+-------------------------------------------+------------+-------------+
| ID   | SCHEMA              | NAME                              | GROUP                              | URL
| USER     | TYPE | INIT | MIN  | MAX  | IDLE_TIMEOUT | MAX_WAIT | ACTIVE_COUNT | POOLING_COUNT | ATOM
| READ_WEIGHT | WRITE_WEIGHT |
+------+----------------------------+----------------------------------------------+------------------------------------------------+------
----------------------------------------------------------------------+-----------+-------+------+------+------+-------------+---------
-+-------------+--------------+-------------------------------------------+------------+-------------+
|    0 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0000_iiab_1 | SEQ_TEST_148776
7780814RGKKSEQ_TEST_WNJG_0000_RDS | jdbc:mysql://rds1ur80kcv8g3t6p3ol.mysql.rds.aliyuncs.com:330
6/seq_test_wnjg_0000 | jnkinsea0 | mysql | 0    | 24   | 72   | 15          | 5000     | 0            | 1             | rds1ur80kcv8g3t6p
```

```
3ol_seq_test_wnjg_0000_iiab | 10      | 10      |
|  1 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0001_iiab_2 | SEQ_TEST_148776
7780814RGKKSEQ_TEST_WNJG_0001_RDS | jdbc:mysql://rds1ur80kcv8g3t6p3ol.mysql.rds.aliyuncs.com:330
6/seq_test_wnjg_0001 | jnkinsea0 | mysql | 0  | 24  | 72  | 15      | 5000  | 0      | 1       | rds1ur80kcv8g3t6p
3ol_seq_test_wnjg_0001_iiab | 10      | 10      |
|  2 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0002_iiab_3 | SEQ_TEST_148776
7780814RGKKSEQ_TEST_WNJG_0002_RDS | jdbc:mysql://rds1ur80kcv8g3t6p3ol.mysql.rds.aliyuncs.com:330
6/seq_test_wnjg_0002 | jnkinsea0 | mysql | 0  | 24  | 72  | 15      | 5000  | 0      | 1       | rds1ur80kcv8g3t6p
3ol_seq_test_wnjg_0002_iiab | 10      | 10      |
|  3 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0003_iiab_4 | SEQ_TEST_148776
7780814RGKKSEQ_TEST_WNJG_0003_RDS | jdbc:mysql://rds1ur80kcv8g3t6p3ol.mysql.rds.aliyuncs.com:330
6/seq_test_wnjg_0003 | jnkinsea0 | mysql | 0  | 24  | 72  | 15      | 5000  | 0      | 1       | rds1ur80kcv8g3t6p
3ol_seq_test_wnjg_0003_iiab | 10      | 10      |
|  4 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0004_iiab_5 | SEQ_TEST_148776
7780814RGKKSEQ_TEST_WNJG_0004_RDS | jdbc:mysql://rds1ur80kcv8g3t6p3ol.mysql.rds.aliyuncs.com:330
6/seq_test_wnjg_0004 | jnkinsea0 | mysql | 0  | 24  | 72  | 15      | 5000  | 0      | 1       | rds1ur80kcv8g3t6p
3ol_seq_test_wnjg_0004_iiab | 10      | 10      |
|  5 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0005_iiab_6 | SEQ_TEST_148776
7780814RGKKSEQ_TEST_WNJG_0005_RDS | jdbc:mysql://rds1ur80kcv8g3t6p3ol.mysql.rds.aliyuncs.com:330
6/seq_test_wnjg_0005 | jnkinsea0 | mysql | 0  | 24  | 72  | 15      | 5000  | 0      | 1       | rds1ur80kcv8g3t6p
3ol_seq_test_wnjg_0005_iiab | 10      | 10      |
|  6 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0006_iiab_7 | SEQ_TEST_148776
7780814RGKKSEQ_TEST_WNJG_0006_RDS | jdbc:mysql://rds1ur80kcv8g3t6p3ol.mysql.rds.aliyuncs.com:330
6/seq_test_wnjg_0006 | jnkinsea0 | mysql | 0  | 24  | 72  | 15      | 5000  | 0      | 1       | rds1ur80kcv8g3t6p
3ol_seq_test_wnjg_0006_iiab | 10      | 10      |
|  7 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0007_iiab_8 | SEQ_TEST_148776
7780814RGKKSEQ_TEST_WNJG_0007_RDS | jdbc:mysql://rds1ur80kcv8g3t6p3ol.mysql.rds.aliyuncs.com:330
6/seq_test_wnjg_0007 | jnkinsea0 | mysql | 0  | 24  | 72  | 15      | 5000  | 0      | 1       | rds1ur80kcv8g3t6p
3ol_seq_test_wnjg_0007_iiab | 10      | 10      |
+------+---------------------------+-------------------------------------------------+-----------------------------------------------+------
-------------------------------------------------------------------------------+-----------+-------+------+------+------+--------------+---------
-+--------------+---------------+-----------------------------------------------+-------------+--------------+
8 rows in set (0.01 sec)
```

SHOW NODE

You can execute this SQL statement to view the accumulative number of read and write operations and accumulative read and write weights of a physical database.

The following section describes the important columns:

- MASTER_READ_COUNT: indicates the accumulative number of read-only queries processed by the primary ApsaraDB RDS for MySQL instance.

- SLAVE_READ_COUNT: indicates the accumulative number of read-only queries processed by the secondary ApsaraDB RDS for MySQL instances.
- MASTER_READ_PERCENT: indicates the actual percentage of read-only queries processed by the primary ApsaraDB RDS for MySQL instance, instead of the specified percentage.
- SLAVE_READ_PERCENT: indicates the actual percentage of read-only queries processed by the secondary ApsaraDB RDS for MySQL instances, instead of the specified percentage.

> **Note**
>
> - Read-only queries in transactions are sent to the primary ApsaraDB RDS for MySQL instance.
> - The `MASTER_READ_PERCENT` and `SLAVE_READ_PERCENT` fields indicate the accumulative historical data. After the ratio between the read weight and the write weight is changed, these values do not immediately reflect the latest ratio, which appears after a long period of time.

Execute the `SHOW NODE;` statement. The following response is returned:

```
+------+-----------------------------------------------+------------------+-----------------+--------------------+--------------------+
| ID   | NAME                                          | MASTER_READ_COUNT | SLAVE_READ_COUNT | MASTER_READ_PERCENT | SLAVE_READ_PERCENT |
+------+-----------------------------------------------+------------------+-----------------+--------------------+--------------------+
| 0    | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0000_RDS |               12 |               0 | 100%               | 0%                 |
| 1    | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0001_RDS |                0 |               0 | 0%                 | 0%                 |
| 2    | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0002_RDS |                0 |               0 | 0%                 | 0%                 |
| 3    | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0003_RDS |                0 |               0 | 0%                 | 0%                 |
| 4    | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0004_RDS |                0 |               0 | 0%                 | 0%                 |
| 5    | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0005_RDS |                0 |               0 | 0%                 | 0%                 |
| 6    | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0006_RDS |                0 |               0 | 0%                 | 0%                 |
| 7    | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0007_RDS |                0 |               0 | 0%                 | 0%                 |
+------+-----------------------------------------------+------------------+-----------------+--------------------+--------------------+
8 rows in set (0.01 sec)
```

5.7.4. Statements for SQL optimization

This topic describes statements for SQL optimization and provides examples of the statements.

SHOW [FULL] SLOW [WHERE expr] [limit expr]

SQL statements that take more than one second to execute are slow SQL statements. Logical slow SQL statements are slow SQL statements sent from an application to a PolarDB-X instance.

- `SHOW SLOW` : You can execute this statement to view the top 100 logical slow SQL statements. These SQL statements are recorded since the PolarDB-X instance is started or the last time when the `CLEAR SLOW` statement is executed.

  > **Note** The recorded top 100 slow SQL statements are cached in the PolarDB-X system. When the PolarDB-X instance is restarted or the `CLEAR SLOW` statement is executed, these statements will be discarded.

- `SHOW FULL SLOW` : You can execute this SQL statement to view all the logical slow SQL statements since the PolarDB-X instance is started. These SQL statements are recorded and persisted to the built-in database of the PolarDB-X instance. The upper limit for the number of records is specified in the specifications of the PolarDB-X instance. The PolarDB-X instance deletes earlier slow SQL statements when the disk space is insufficient. If the specifications of the PolarDB-X instance include 4 cores and 4 GB of memory, a maximum of 10,000 slow SQL statements can be recorded, including logical slow and physical slow SQL statements. If the specifications of the PolarDB-X instance include 8 cores and 8 GB of memory, a maximum of 20,000 slow SQL statements can be recorded, including logical slow and physical slow SQL statements. The same rule applies to other instance specifications.

The following section describes the important columns:

- HOST: the IP address of the host from which the SQL statement is sent.
- START_TIME: the time when the SQL statement starts to be executed.
- EXECUTE_TIME: the execution duration of the SQL statement.
- AFFECT_ROW: For DML statements, this parameter indicates the number of affected rows. For query statements, this parameter indicates the number of returned records.

Execute the `SHOW SLOW WHERE execute_time > 1000 limit 1;` statement. The following response is returned:

```
+-----------+---------------------+--------------+------------+-----------+
| HOST      | START_TIME          | EXECUTE_TIME | AFFECT_ROW | SQL       |
+-----------+---------------------+--------------+------------+-----------+
| 127.0.0.1 | 2016-03-16 13:02:57 |         2785 |          7 | show rule |
+-----------+---------------------+--------------+------------+-----------+
1 row in set (0.02 sec)
```

SHOW [FULL] PHYSICAL_SLOW [WHERE expr] [limit expr]

SQL statements that take more than one second to execute are slow SQL statements. Logical slow SQL statements are slow SQL statements sent from an application to a PolarDB-X instance.

- `SHOW SLOW` : You can execute this statement to view the top 100 logical slow SQL statements. These SQL statements are recorded since the PolarDB-X instance is started or the last time when the `CLEAR SLOW` statement is executed.

> **Note** The recorded top 100 slow SQL statements are cached in the PolarDB-X system. When the PolarDB-X instance is restarted or the `CLEAR SLOW` statement is executed, these statements will be discarded.

- `SHOW FULL SLOW` : You can execute this SQL statement to view all the logical slow SQL statements since the PolarDB-X instance is started. These SQL statements are recorded and persisted to the built-in database of the PolarDB-X instance. The upper limit for the number of records is specified in the specifications of the PolarDB-X instance. The PolarDB-X instance deletes earlier slow SQL statements when the disk space is insufficient. If the specifications of the PolarDB-X instance include 4 cores and 4 GB of memory, a maximum of 10,000 slow SQL statements can be recorded, including logical slow and physical slow SQL statements. If the specifications of the PolarDB-X instance include 8 cores and 8 GB of memory, a maximum of 20,000 slow SQL statements can be recorded, including logical slow and physical slow SQL statements. The same rule applies to other instance specifications.
-

The following section describes the important columns:

- GROUP_NAME: the name of the group to which the database that executes the SQL statement belongs.
- START_TIME: the time when the SQL statement starts to be executed.
- EXECUTE_TIME: the execution duration of the SQL statement.
- AFFECT_ROW: For DML statements, this parameter indicates the number of affected rows. For query statements, this parameter indicates the number of returned records.

Execute the `SHOW PHYSICAL_SLOW;` statement. The following response is returned:

```
+----------------+----------------------------------+--------------------+-------------+-----------------+------------------------+------
------------------+------------+-----------------+
| GROUP_NAME    | DBKEY_NAME                | START_TIME      | EXECUTE_TIME | SQL_EXECUTE_TIME | GETLOC
K_CONNECTION_TIME | CREATE_CONNECTION_TIME | AFFECT_ROW | SQL        |
+----------------+----------------------------------+--------------------+-------------+-----------------+------------------------+------
------------------+------------+-----------------+
| TDDL5_00_GROUP | db218249098_sqa_zmf_tddl5_00_3309 | 2016-03-16 13:05:38 |      1057 |        1011 |
0 |              0 |      1 | select sleep(1) |
+----------------+----------------------------------+--------------------+-------------+-----------------+------------------------+------
------------------+------------+-----------------+
1 row in set (0.01 sec)
```

CLEAR SLOW

You can execute this SQL statement to clear the top 100 logical slow SQL statements and the top 100 physical slow SQL statements. These statements are recorded since the PolarDB-X instance is started or the last time when the `CLEAR SLOW` statement is executed.

> **Note** Both `SHOW SLOW` and `SHOW PHYSICAL_SLOW` can be executed to show the top 100 slow SQL statements. If the `CLEAR SLOW` statement has not been executed for a long time, the SQL statements might be recorded a long time ago. Therefore, we recommend that you execute the `CLEAR SLOW` statement after statements for SQL optimization are executed. After the system runs for a while, check the optimized results of slow SQL statements.

Execute the `CLEAR SLOW;` statement. The following response is returned:

```
Query OK, 0 rows affected (0.00 sec)
```

EXPLAIN SQL

You can execute this SQL statement to view the execution plan of a specified SQL statement in PolarDB-X. Note that this SQL statement is not actually executed.

Examples

You can execute this SQL statement to view the execution plan of the `SELECT * FROM doctest` statement. The data of the doctest table is partitioned into database shards based on the id column. Based on the execution plan, the SQL statement will be routed to each database shard for execution, and the execution results will be aggregated.

Execute the `EXPLAIN SELECT * FROM doctest;` statement. The following response is returned:

```
+------------------------------------------------+-------------------------------------+--------+
| GROUP_NAME                  | SQL                | PARAMS |
+------------------------------------------------+-------------------------------------+--------+
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0000_RDS | select `doctest`.`id` from `doctest` | {}     |
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0001_RDS | select `doctest`.`id` from `doctest` | {}     |
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0002_RDS | select `doctest`.`id` from `doctest` | {}     |
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0003_RDS | select `doctest`.`id` from `doctest` | {}     |
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0004_RDS | select `doctest`.`id` from `doctest` | {}     |
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0005_RDS | select `doctest`.`id` from `doctest` | {}     |
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0006_RDS | select `doctest`.`id` from `doctest` | {}     |
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0007_RDS | select `doctest`.`id` from `doctest` | {}     |
+------------------------------------------------+-------------------------------------+--------+
8 rows in set (0.00 sec)
```

You can execute this SQL statement to view the execution plan of the `SELECT * FROM doctest WHERE id = 1` statement. The data of the doctest table is partitioned into database shards based on the id column. Based on the execution plan, the PolarDB-X instance will calculate a specified database shard based on the shard key, which is id. Then, the PolarDB-X instance will directly route the SQL statement to the database shard and aggregate the execution results.

Execute the `EXPLAIN SELECT * FROM doctest WHERE id = 1;` statement. The following response is returned:

```
+-----------------------------------------------+-----------------------------------------------------------------------+--------+
| GROUP_NAME                    | SQL                                   | PARAMS |
+-----------------------------------------------+-----------------------------------------------------------------------+--------+
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0001_RDS | select `doctest`.`id` from `doctest` where (
`doctest`.`id` = 1) | {}    |
+-----------------------------------------------+-----------------------------------------------------------------------+--------+
1 row in set (0.01 sec)
```

EXPLAIN DETAIL SQL

You can execute this SQL statement to view the execution plan of a specified SQL statement in PolarDB-X. Note that this SQL statement is not actually executed.

Execute the `EXPLAIN DETAIL SELECT * FROM doctest WHERE id = 1;` statement. The following response is returned:

```
+-----------------------------------------------+-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------+--------+
| GROUP_NAME                    | SQL
| PARAMS |
+-----------------------------------------------+-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------+--------+
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0001_RDS | Query from doctest as doctest
  keyFilter:doctest.id = 1
  queryConcurrency:SEQUENTIAL
  columns:[doctest.id]
  tableName:doctest
  executeOn:DOCTEST_1488704345426RCUPDOCTEST_CAET_0001_RDS
 | NULL  |
+-----------------------------------------------+-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------+--------+
1 row in set (0.02 sec)
```

EXPLAIN EXECUTE SQL

You can execute this SQL statement to view the execution plan of a specified SQL statement on an underlying ApsaraDB RDS for MySQL instance. This statement is equivalent to the EXPLAIN statement in MySQL.

Execute the `EXPLAIN EXECUTE SELECT * FROM tddl_mgr_log limit 1;` statement. The following response is returned:

```
+----+-------------+--------------+------+---------------+------+---------+------+------+-------+
| id | select_type | table        | type | possible_keys | key  | key_len | ref  | rows | Extra |
+----+-------------+--------------+------+---------------+------+---------+------+------+-------+
|  1 | SIMPLE      | tddl_mgr_log | ALL  | NULL          | NULL | NULL    | NULL |    1 | NULL  |
+----+-------------+--------------+------+---------------+------+---------+------+------+-------+
1 row in set (0.07 sec)
```

TRACE SQL and SHOW TRACE

You can execute these SQL statements to view the execution results of an SQL statement. Note that you must use the TRACE SQL statement and the SHOW TRACE statement together. The difference between the TRACE SQL statement and the EXPLAIN SQL statement is that the TRACE SQL statement is actually executed.

For example, you can execute these statements to view the execution results of the SELECT 1 statement.

Execute the `TRACE SELECT 1;` statement. The following response is returned:

```
+---+
| 1 |
+---+
| 1 |
+---+
1 row in set (0.03 sec)
```

Execute the `SHOW TRACE;` statement. The following response is returned:

```
+------+----------+----------------+----------------------------------+--------------+--------------------------+------+-----------+-------
-+
| ID   | TYPE     | GROUP_NAME     | DBKEY_NAME                       | TIME_COST(MS) | CONNECTION_TIME_COST(MS) | R
OWS | STATEMENT | PARAMS |
+------+----------+----------------+----------------------------------+--------------+--------------------------+------+-----------+-------
-+
|    0 | Optimize | DRDS           | DRDS                             |            3 |                     0.00 |    0 | select 1  | NULL   |
|    1 | Query    | TDDL5_00_GROUP | db218249098_sqa_zmf_tddl5_00_3309 |            7 |                     0.15 |    1 | select
1 | NULL   |
+------+----------+----------------+----------------------------------+--------------+--------------------------+------+-----------+-------
-+
2 rows in set (0.01 sec)
```

CHECK TABLE tablename

You can execute this SQL statement to check a data table. This SQL statement can be used when you fail to create a table by using a DDL statement.

- If the data table is a partitioned table, this SQL statement allows you to check whether an underlying physical table shard is missing and whether the columns and indexes of the underlying physical table shard are consistent.
- If the data table is a single-database non-partitioned table, this SQL statement allows you to check whether this table exists.

Execute the `CHECK TABLE tddl_mgr_log;` statement. The following response is returned:

```
+-----------------------+-------+----------+----------+
| TABLE                 | OP    | MSG_TYPE | MSG_TEXT |
+-----------------------+-------+----------+----------+
| TDDL5_APP.tddl_mgr_log | check | status   | OK       |
+-----------------------+-------+----------+----------+
1 row in set (0.56 sec)
```

Execute the `CHECK TABLE tddl_mg;` statement. The following response is returned:

```
+-------------------+-------+----------+-----------------------------------------+
| TABLE             | OP    | MSG_TYPE | MSG_TEXT                                |
+-------------------+-------+----------+-----------------------------------------+
| TDDL5_APP.tddl_mg | check | Error    | Table 'tddl5_00.tddl_mg' doesn't exist |
+-------------------+-------+----------+-----------------------------------------+
1 row in set (0.02 sec)
```

SHOW TABLE STATUS [LIKE 'pattern' | WHERE expr]

You can execute this SQL statement to query the information about a table. This statement aggregates the data of all underlying physical table shards.

The following section describes the important columns:

- NAME: indicates the name of the table.
- ENGINE: indicates the storage engine of the table.
- VERSION: indicates the version of the storage engine of the table.
- ROW_FORMAT: indicates the format of the rows in the table. Valid values include Dynamic, Fixed, and Compressed. The value Dynamic indicates that the row length is variable, for example, a VARCHAR or BLOB field. The value Fixed indicates that the row length is constant, for example, a CHAR or INTEGER field.
- ROWS: indicates the number of rows in the table.
- AVG_ROW_LENGTH: indicates the average number of bytes in each row.
- DATA_LENGTH: indicates the data volume of the entire table. Unit: bytes.
- MAX_DATA_LENGTH: indicates the maximum volume of data that can be stored in the table.

- INDEX_LENGTH: indicates the size of the disk space occupied by indexes.
- CREATE_TIME: indicates the time when the table was created.
- UPDATE_TIME: indicates the time when the table was last updated.
- COLLATION: indicates the default character set and character sorting rule of the table.
- CREATE_OPTIONS: indicates all the other options specified when the table was created.

Execute the `SHOW TABLE STATUS LIKE 'multi_db_multi_tbl';` statement. The following response is returned:

```
+--------------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+-------------+------------+-----------------+----------+----------------+---------+
| NAME               | ENGINE | VERSION | ROW_FORMAT | ROWS | AVG_ROW_LENGTH | DATA_LENGTH | MAX_DATA_LENGTH | INDEX_LENGTH | DATA_FREE | AUTO_INCREMENT | CREATE_TIME         | UPDATE_TIME | CHECK_TIME | COLLATION       | CHECKSUM | CREATE_OPTIONS | COMMENT |
+--------------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+-------------+------------+-----------------+----------+----------------+---------+
| multi_db_multi_tbl | InnoDB |      10 | Compact    |    2 |          16384 |       16384 |               0 |        16384 |         0 |         100000 | 2017-03-27 17:43:57.0 | NULL        | NULL       | utf8_general_ci | NULL     |                |         |
+--------------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+-------------+------------+-----------------+----------+----------------+---------+
1 row in set (0.03 sec)
```

You can use the SHOW TABLE STATUS statement and the PolarDB-X SCAN hint to view the data volume of each physical table shard.

Execute the `/! TDDL:SCAN='multi_db_multi_tbl'*/SHOW TABLE STATUS LIKE 'multi_db_multi_tbl';` statement. The following response is returned:

```
+----------------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+-------------+------------+-----------------+----------+----------------+---------+--------------+
| Name                 | Engine | Version | Row_format | Rows | Avg_row_length | Data_length | Max_data_length | Index_length | Data_free | Auto_increment | Create_time         | Update_time | Check_time | Collation       | Checksum | Create_options | Comment | Block_format |
+----------------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+-------------+------------+-----------------+----------+----------------+---------+--------------+
| multi_db_multi_tbl_1 | InnoDB |      10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 | 2017-03-27 17:43:57 | NULL        | NULL       | utf8_general_ci | NULL     |                |         | Original     |
| multi_db_multi_tbl_0 | InnoDB |      10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 | 2017-03-27 17:43:57 | NULL        | NULL       | utf8_general_ci | NULL     |                |         | Original     |
| multi_db_multi_tbl_1 | InnoDB |      10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 | 2017-03-27 17:43:57 | NULL        | NULL       | utf8_general_ci | NULL     |                |         | Original     |
| multi_db_multi_tbl_0 | InnoDB |      10 | Compact    |    1 |          16384 |       16384 |               0 |        16384 |         0 |              2 | 2017-03-27 17:43:57 | NULL        | NULL       | utf8_general_ci | NULL     |                |         | Original     |
| multi_db_multi_tbl_1 | InnoDB |      10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 |
```

```
| multi_db_multi_tbl_1 | InnoDB |     10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 |
2017-03-27 17:43:57 | NULL        | NULL        | utf8_general_ci |     NULL |                |         | Original      |
| multi_db_multi_tbl_0 | InnoDB |     10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 |
2017-03-27 17:43:57 | NULL        | NULL        | utf8_general_ci |     NULL |                |         | Original      |
| multi_db_multi_tbl_1 | InnoDB |     10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 |
2017-03-27 17:43:57 | NULL        | NULL        | utf8_general_ci |     NULL |                |         | Original      |
| multi_db_multi_tbl_0 | InnoDB |     10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 |
2017-03-27 17:43:57 | NULL        | NULL        | utf8_general_ci |     NULL |                |         | Original      |
| multi_db_multi_tbl_1 | InnoDB |     10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 |
2017-03-27 17:43:57 | NULL        | NULL        | utf8_general_ci |     NULL |                |         | Original      |
| multi_db_multi_tbl_0 | InnoDB |     10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 |
2017-03-27 17:43:57 | NULL        | NULL        | utf8_general_ci |     NULL |                |         | Original      |
| multi_db_multi_tbl_1 | InnoDB |     10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 |
2017-03-27 17:43:57 | NULL        | NULL        | utf8_general_ci |     NULL |                |         | Original      |
| multi_db_multi_tbl_0 | InnoDB |     10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 |
2017-03-27 17:43:57 | NULL        | NULL        | utf8_general_ci |     NULL |                |         | Original      |
| multi_db_multi_tbl_1 | InnoDB |     10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 |
2017-03-27 17:43:57 | NULL        | NULL        | utf8_general_ci |     NULL |                |         | Original      |
| multi_db_multi_tbl_0 | InnoDB |     10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 |
2017-03-27 17:43:57 | NULL        | NULL        | utf8_general_ci |     NULL |                |         | Original      |
| multi_db_multi_tbl_1 | InnoDB |     10 | Compact    |    0 |              0 |       16384 |               0 |        16384 |         0 |              1 |
2017-03-27 17:43:57 | NULL        | NULL        | utf8_general_ci |     NULL |                |         | Original      |
| multi_db_multi_tbl_0 | InnoDB |     10 | Compact    |    1 |          16384 |       16384 |               0 |        16384 |         0 |
3 | 2017-03-27 17:43:57 | NULL        | NULL        | utf8_general_ci |     NULL |                |         | Original      |
+----------------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+-
---------------+---------------------+-------------+------------+-----------------+----------+----------------+---------+--------------+
16 rows in set (0.04 sec)
```

5.7.5. Statistics query statements

This topic describes statements that you can use to query statistics.

SHOW [FULL] STATS

You can execute this SQL statement to view the overall statistics of a PolarDB-X instance. The statistics are instantaneous values. Note that the results of the `DRDS SHOW FULL STATS` statement vary with the version of the PolarDB-X instance.

The following section describes the important columns:

- QPS: the number of queries per second (QPS) sent from an application to the PolarDB-X instance. This QPS is usually named logical QPS.
- RDS_QPS: the QPS sent from the PolarDB-X instance to an ApsaraDB RDS for MySQL instance. This QPS is usually named physical QPS.

- ERROR_PER_SECOND: the total number of errors that occur on the PolarDB-X instance per second. These errors include SQL syntax errors, primary key conflicts, system errors, and connectivity errors.
- VIOLATION_PER_SECOND: the number of primary key conflicts or unique key conflicts per second.
- MERGE_QUERY_PER_SECCOND: the number of queries processed on multiple tables per second based on database sharding and table sharding.
- ACTIVE_CONNECTIONS: the number of active connections to the PolarDB-X instance.
- CONNECTION_CREATE_PER_SECCOND: the number of connections that are created for the PolarDB-X instance per second.
- RT(MS): the time to respond to an SQL query sent from an application to the PolarDB-X instance. This RT is usually named logical RT.
- RDS_RT(MS): the time to respond to an SQL query sent from the PolarDB-X instance to an ApsaraDB RDS for MySQL instance. This RT is usually named physical RT.
- NET_IN(KB/S): the amount of inbound traffic of the PolarDB-X instance per second.
- NET_OUT(KB/S): the amount of outbound traffic of the PolarDB-X instance per second.
- THREAD_RUNNING: the number of threads that are running in the PolarDB-X instance.
- HINT_USED_PER_SECOND: the number of SQL queries that contain hints and are processed by the PolarDB-X instance per second.
- HINT_USED_COUNT: the total number of SQL queries that contain hints and are processed by the PolarDB-X instance since the instance is started.
- AGGREGATE_QUERY_PER_SECCOND: the number of aggregate SQL queries that are processed by the PolarDB-X instance per second.
- AGGREGATE_QUERY_COUNT: the total number of aggregate SQL queries that are processed by the PolarDB-X instance.
- TEMP_TABLE_CREATE_PER_SECCOND: the number of temporary tables that are created in the PolarDB-X instance per second.
- TEMP_TABLE_CREATE_COUNT: the total number of temporary tables that are created in the PolarDB-X instance since the instance is started.
- MULTI_DB_JOIN_PER_SECCOND: the number of multi-database joins that are processed by the PolarDB-X instance per second.
- MULTI_DB_JOIN_COUNT: the total number of multi-database joins that are processed by the PolarDB-X instance since the instance is started.

Execute the `SHOW STATS;` statement. The following response is returned:

```
+------+---------+----------+------------------+-----------------+----------------------+-------------------+--------+------------+-----
----------+---------------+----------------+
| QPS  | RDS_QPS | SLOW_QPS | PHYSICAL_SLOW_QPS | ERROR_PER_SECOND | MERGE_QUERY_PER_SECOND
| ACTIVE_CONNECTIONS | RT(MS) | RDS_RT(MS) | NET_IN(KB/S) | NET_OUT(KB/S) | THREAD_RUNNING |
+------+---------+----------+------------------+-----------------+----------------------+-------------------+--------+------------+-----
----------+---------------+----------------+
| 1.77 |   1.68 |    0.03 |         0.03 |         0.02 |            0.00 |           7 | 157.13 |    51.14 |     134.49 |       1.48 |
1 |
+------+---------+----------+------------------+-----------------+----------------------+-------------------+--------+------------+-----
----------+---------------+----------------+
1 row in set (0.01 sec)
```

Execute the `SHOW [FULL] STATS;` statement. The following response is returned:

```
+------+---------+----------+------------------+-----------------+----------------------+-----------------------+--------------------+-----
-------------------------+--------+------------+-------------+--------------+---------------+---------------------+----------------+---
-----------------------+---------------------+----------------------------+----------------------+-------------------------+---------
------------+-------+---------+-------------+------------+
| QPS  | RDS_QPS | SLOW_QPS | PHYSICAL_SLOW_QPS | ERROR_PER_SECOND | VIOLATION_PER_SECOND | ME
RGE_QUERY_PER_SECOND | ACTIVE_CONNECTIONS | CONNECTION_CREATE_PER_SECOND | RT(MS) | RDS_RT(
MS) | NET_IN(KB/S) | NET_OUT(KB/S) | THREAD_RUNNING | HINT_USED_PER_SECOND | HINT_USED_COUNT | A
GGREGATE_QUERY_PER_SECOND | AGGREGATE_QUERY_COUNT | TEMP_TABLE_CREATE_PER_SECOND | TEM
P_TABLE_CREATE_COUNT | MULTI_DB_JOIN_PER_SECOND | MULTI_DB_JOIN_COUNT | CPU  | FREEMEM | FUL
LGCCOUNT | FULLGCTIME |
+------+---------+----------+------------------+-----------------+----------------------+-----------------------+--------------------+-----
-------------------------+--------+------------+-------------+--------------+---------------+---------------------+----------------+---
-----------------------+---------------------+----------------------------+----------------------+-------------------------+---------
------------+-------+---------+-------------+------------+
| 1.63 |   1.68 |    0.03 |         0.03 |         0.02 |            0.00 |             0.00 |           6 |                0.01 | 157.13 |    5
1.14 |    134.33 |       1.21 |         1 |           0.00 |        54 |               0.00 |           663 |               0.00 |
512 |              0.00 |          516 | 0.09% |  6.96% |     76446 |  21326906 |
+------+---------+----------+------------------+-----------------+----------------------+-----------------------+--------------------+-----
-------------------------+--------+------------+-------------+--------------+---------------+---------------------+----------------+---
-----------------------+---------------------+----------------------------+----------------------+-------------------------+---------
------------+-------+---------+-------------+------------+
1 row in set (0.01 sec)
```

SHOW DB STATUS

You can execute this SQL statement to view the capacity and performance information of a physical database, which is also named a database shard. All the returned values indicate the real-time information. The capacity information is obtained from the ApsaraDB RDS for MySQL system table, and therefore may be different from the actual capacity information.

The following section describes the important columns:

- NAME: the PolarDB-X internal tag that represents a PolarDB-X logical database corresponding to the database shard. The value is different from the name of the PolarDB-X logical database.
- CONNECTION_STRING: the information about a connection from the PolarDB-X instance to the database shard.
- PHYSICAL_DB: the name of the database shard. The `TOTAL` row indicates the total amount of capacity of all the database shards corresponding to the PolarDB-X logical database.
- SIZE_IN_MB: the size of the space occupied by the data in the database shard. Unit: MB.
- RATIO: the ratio of the data volume of the database shard to the total data volume of the current PolarDB-X logical database.
- THREAD_RUNNING: the number of threads that are running in the ApsaraDB RDS for MySQL instance to which the physical database belongs. The meaning of this parameter is the same as that of the THREAD_RUNNING parameter in the returned results of the `SHOW GLOBAL STATUS` statement in MySQL. For more information, see MySQL documentation.

Execute the `SHOW DB STATUS;` statement. The following response is returned:

```
+------+--------------------------+-------------------+-------------------+------------+--------+----------------+
| ID   | NAME                     | CONNECTION_STRING | PHYSICAL_DB       | SIZE_IN_MB | RATIO  | THREAD_RUNNING |
+------+--------------------------+-------------------+-------------------+------------+--------+----------------+
|    1 | drds_db_1516187088365daui | 100.100.64.1:59077 | TOTAL             | 13.109375 | 100%   | 3              |
|    2 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0000 |  1.578125 | 12.04% |                |
|    3 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0001 |    1.4375 | 10.97% |                |
|    4 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0002 |    1.4375 | 10.97% |                |
|    5 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0003 |    1.4375 | 10.97% |                |
|    6 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0004 |  1.734375 | 13.23% |                |
|    7 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0005 |  1.734375 | 13.23% |                |
|    8 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0006 |  2.015625 | 15.38% |                |
|    9 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0007 |  1.734375 | 13.23% |                |
+------+--------------------------+-------------------+-------------------+------------+--------+----------------+
```

SHOW FULL DB STATUS [LIKE {tablename}]

You can execute this SQL statement to view the capacity and performance information of a table shard in a physical database, which is also named a database shard. All the returned values indicate the real-time information. The capacity information is obtained from the ApsaraDB RDS for MySQL system table, and therefore may be different from the actual capacity information.

The following section describes the important columns:

- NAME: the PolarDB-X internal tag that represents a PolarDB-X logical database corresponding to the database shard. The value is different from the name of the PolarDB-X logical database.

- CONNECTION_STRING: the information about a connection from the PolarDB-X instance to the database shard.
- PHYSICAL_DB: the name of the database shard. If the LIKE keyword is specified in the statement, the TOTAL row indicates the total amount of capacity of the database shard. If the LIKE keyword is not specified in the statement, the TOTAL row indicates the total amount of capacity of all database shards.
- PHYSICAL_TABLE: the name of the table shard in the database shard. If the LIKE keyword is specified in the statement, the TOTAL row indicates the total amount of capacity of the table shard. If the LIKE keyword is not specified in the statement, the TOTAL row indicates the total amount of capacity of all table shards in the database shard.
- SIZE_IN_MB: the size of the space occupied by the data in the database shard. Unit: MB.
- RATIO: the ratio of the data volume of the table shard to the total data volume of all the filtered table shards.
- THREAD_RUNNING: the number of threads that are running in the ApsaraDB RDS for MySQL instance to which the physical database belongs. The meaning of this parameter is the same as that of the THREAD_RUNNING parameter in the returned results of the `SHOW GLOBAL STATUS` statement in MySQL. For more information, see MySQL documentation.

Execute the `SHOW FULL DB STATUS LIKE hash_tb;` statement. The following response is returned:

```
+------+--------------------------+-------------------+------------------+---------------+------------+--------+----------------+
| ID   | NAME                     | CONNECTION_STRING | PHYSICAL_DB      | PHYSICAL_TABLE | SIZE_IN_MB | RATIO  | THREAD_RUNNING |
+------+--------------------------+-------------------+------------------+---------------+------------+--------+----------------+
|  1 | drds_db_1516187088365daui | 100.100.64.1:59077 | TOTAL           |               |   19.875 | 100%  | 3              |
|  2 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0000 | TOTAL       |    3.03125 | 15.25% |
|
|  3 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0000 | hash_tb_00  |  1.515625 | 7.63
% |              |
|  4 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0000 | hash_tb_01  |  1.515625 | 7.63
% |              |
|  5 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0001 | TOTAL       |     2.0 | 10.06% |
|
|  6 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0001 | hash_tb_02  |  1.515625 | 7.63
% |              |
|  7 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0001 | hash_tb_03  |  0.484375 | 2.44
% |              |
|  8 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0002 | TOTAL       |    3.03125 | 15.25% |
|
|  9 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0002 | hash_tb_04  |  1.515625 | 7.63
% |              |
| 10 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0002 | hash_tb_05  |  1.515625 | 7.6
3% |              |
| 11 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0003 | TOTAL       |  1.953125 | 9.83%
```

```
|        |
| 12 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0003 | hash_tb_06   | 1.515625 | 7.6
3% |        |
| 13 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0003 | hash_tb_07   |  0.4375 | 2.2%
|        |
| 14 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0004 | TOTAL      |  3.03125 | 15.25%
|        |
| 15 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0004 | hash_tb_08   | 1.515625 | 7.6
3% |        |
| 16 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0004 | hash_tb_09   | 1.515625 | 7.6
3% |        |
| 17 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0005 | TOTAL      | 1.921875 | 9.67%
|        |
| 18 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0005 | hash_tb_11   | 1.515625 | 7.6
3% |        |
| 19 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0005 | hash_tb_10   |  0.40625 | 2.04
% |        |
| 20 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0006 | TOTAL      |  3.03125 | 15.25%
|        |
| 21 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0006 | hash_tb_12   | 1.515625 | 7.6
3% |        |
| 22 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0006 | hash_tb_13   | 1.515625 | 7.6
3% |        |
| 23 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0007 | TOTAL      |  1.875 | 9.43% |
|
| 24 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0007 | hash_tb_14   | 1.515625 | 7.6
3% |        |
| 25 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0007 | hash_tb_15   | 0.359375 | 1.8
1% |        |
+------+--------------------------+-------------------+-------------------+---------------+------------+--------+---------------+
```

5.7.6. SHOW PROCESSLIST and KILL

This topic describes the SHOW PROCESSLIST and KILL statements.

> Note
>
> - If the version of PolarDB-X is 5.1.28-1408022 or later, PolarDB-X supports the SHOW PROCESSLIST and KILL statements for logical and physical connections. For more information, see this topic.
> - If the version of PolarDB-X is earlier than 5.1.28-1408022, PolarDB-X supports the SHOW PROCESSLIST and KILL statements for only physical connections. For more information, see SHOW PROCESSLIST and KILL commands in earlier versions.

SHOW PROCESSLIST

In a PolarDB-X instance, you can execute the `SHOW PROCESSLIST` statement to view information including connections to the PolarDB-X instance and SQL statements that are being executed in the instance.

Syntax

```
SHOW [FULL] PROCESSLIST
```

Examples

Execute the `SHOW PROCESSLIST;` statement. The following response is returned:

```
   ID: 1971050
 USER: admin
 HOST: 111.111.111.111:4303
   DB: drds_test
COMMAND: Query
 TIME: 0
 STATE:
 INFO: show processlist
1 row in set (0.01 sec)
```

The meanings of the following fields in the result set are described in detail:

- ID: the ID of the connection. The value is a number of the Long type.
- USER: the name of the user who establishes the connection.
- HOST: the IP address and port number of the host that establishes the connection.
- DB: the name of the database to which the connection is established.
- COMMAND: the usage state of the connection. This field can be set to the following values:
  - Query: indicates that the current connection is executing an SQL statement.
  - Sleep: indicates that the current connection is idle.
- TIME: the duration when the connection is in the current state:
  - When the value of COMMAND is Query, this field indicates how long the SQL statement has been being executed over the connection.

  - When the value of COMMAND is Sleep, this field indicates how long the connection has been in the idle state.

- STATE: Currently, this field is meaningless and is constantly empty.
- INFO:
  - When the value of COMMAND is Query, this field indicates the content of the SQL statement that is being executed over the connection. If the FULL parameter is not specified, a maximum of the first 30 characters of the SQL statement are returned. If the FULL parameter is specified, a maximum of the first 1,000 characters of the SQL statement are returned.
  - When the value of COMMAND is other values, this field is meaningless and left empty.

SHOW PHYSICAL_PROCESSLIST

In a PolarDB-X instance, you can execute the SHOW PHYSICAL_PROCESSLIST statement to view information about all the SQL statements that are being executed on underlying ApsaraDB RDS for MySQL instances.

Syntax

```
SHOW [FULL] PHYSICAL_PROCESSLIST
```

When an SQL statement is excessively long, the responses of the `SHOW PHYSICAL_PROCESSLIST` statement may be truncated. In this case, you can execute the `SHOW FULL PHYSICAL_PROCESSLIST` statement to obtain the complete SQL statement.

The meaning of each column in the responses is equivalent to that in the responses of the `SHOW PROCESSLIST` statement. For more information, see SHOW PROCESSLIST Syntax.

> **Note** Different from MySQL, the PolarDB-X instance returns a string instead of a number in the ID column of a physical connection.

Execute the `SHOW PHYSICAL_PROCESSLIST;` statement. The following response is returned:

```
*************************** 1. row ***************************
      ID: 0-0-521414
    USER: tddl5
      DB: tddl5_00
 COMMAND: Query
    TIME: 0
   STATE: init
    INFO: show processlist
*************************** 2. row ***************************
      ID: 0-0-521570
    USER: tddl5
      DB: tddl5_00
 COMMAND: Query
    TIME: 0
   STATE: User sleep
    INFO: /*DRDS /88.88.88.88/b67a0e4d8800000/ */ select sleep(1000)
2 rows in set (0.01 sec)
```

KILL

You can execute the KILL statement to terminate an SQL statement that is being executed.

The PolarDB-X instance connects to an ApsaraDB RDS for MySQL instance by using the username created by the PolarDB-X instance on the ApsaraDB RDS for MySQL instance. Therefore, if you directly connect to the ApsaraDB RDS for MySQL instance, you are not authorized to execute the KILL statement to terminate a request initiated by the PolarDB-X instance.

To terminate an SQL statement that is being executed on the PolarDB-X instance, you must use tools to connect to the PolarDB-X instance. You can use tools such as the MySQL command line and . Then, execute the KILL statement on the PolarDB-X instance.

Syntax

```
KILL PROCESS_ID | 'PHYSICAL_PROCESS_ID' | 'ALL'
```

The KILL statement can be used in the following three ways:

- Execute the `KILL PROCESS_ID` statement to terminate a specified logical SQL statement.

  The PROCESS_ID parameter is obtained from the ID column in the responses of the `SHOW [FULL] PROCESSLIST` statement.

  If you execute the `KILL PROCESS_ID` statement in the PolarDB-X instance, it will terminate both logical and physical SQL statements that are being executed over this connection. In addition, this connection will be disconnected.

  The PolarDB-X instance does not support the `KILL QUERY` statement.

- Execute the `KILL 'PHYSICAL_PROCESS_ID'` statement to terminate a specified physical SQL statement.

  The PHYSICAL_PROCESS_ID parameter is obtained from the ID column in the responses of the `SHOW PHYSICAL_PROCESS_ID` statement.

  > **Note** The PHYSICAL_PROCESS_ID column is a string instead of a number. Therefore, the PHYSICAL_PROCESS_ID parameter must be enclosed in single quotation marks (') in the KILL statement.

  Examples

  Execute the `KILL '0-0-521570';` statement. The following response is returned:

  ```
  Query OK, 0 rows affected (0.01 sec)
  ```

- Execute the `KILL 'ALL'` statement to terminate all the physical SQL statements that are executed by the PolarDB-X instance in the current logical database.

  When the underlying ApsaraDB RDS for MySQL instance is overloaded due to several SQL statements, you can execute the `KILL 'ALL'` statement to terminate all the physical SQL statements that are being executed in the current logical PolarDB-X database.

  All the physical SQL statements indicated by PROCESS that meet the following conditions can be terminated by the `KILL 'ALL'` statement:

  - The value of the User parameter for the physical SQL statement indicated by PROCESS is a username created by the PolarDB-X instance in the ApsaraDB RDS for MySQL instance.
  - The physical SQL statement indicated by PROCESS is executing a query, which means that the value of COMMAND is Query.

5.7.7. SHOW PROCESSLIST and KILL statements in earlier versions

This topic describes the SHOW PROCESSLIST and KILL statements in earlier versions.

> **Note**
>
> - If the version of PolarDB-X is 5.1.28-1408022 or later, PolarDB-X supports the SHOW PROCESSLIST and KILL statements for logical and physical connections. For more information, see SHOW PROCESSLIST and KILL commands.
> - If the version of PolarDB-X is earlier than 5.1.28-1408022, PolarDB-X supports the SHOW PROCESSLIST and KILL statements for only physical connections. For more information, see this topic.

SHOW PROCESSLIST

In a PolarDB-X instance, you can execute the `SHOW PROCESSLIST` statement to view information about all the SQL statements that are being executed on the underlying ApsaraDB RDS for MySQL instances.

Syntax

```
SHOW [FULL] PROCESSLIST
```

When an SQL statement is excessively long, the responses of the `SHOW PROCESSLIST` statement may be truncated. In this case, you can execute the `SHOW FULL PROCESSLIST` statement to obtain the complete SQL statement.

The meaning of each column in the responses is equivalent to that in the responses of the `SHOW PROCESSLIST` statement. For more information, see SHOW PROCESSLIST Syntax.

```
*************************** 1. row ***************************
     ID: 0-0-521414
   USER: tddl5
     DB: tddl5_00
COMMAND: Query
   TIME: 0
  STATE: init
   INFO: show processlist
ROWS_SENT: NULL
ROWS_EXAMINED: NULL
ROWS_READ: NULL
*************************** 2. row ***************************
     ID: 0-0-521570
   USER: tddl5
     DB: tddl5_00
COMMAND: Query
   TIME: 0
  STATE: User sleep
   INFO: /*DRDS /88.88.88.88/b67a0e4d8800000/ */ select sleep(1000)
ROWS_SENT: NULL
ROWS_EXAMINED: NULL
ROWS_READ: NULL
2 rows in set (0.01 sec)
```

KILL

You can execute the KILL statement to terminate an SQL statement that is being executed.

The PolarDB-X instance connects to an ApsaraDB RDS for MySQL instance by using the username created by the PolarDB-X instance on the ApsaraDB RDS for MySQL instance. Therefore, if you directly connect to the ApsaraDB RDS for MySQL instance, you are not authorized to execute the KILL statement to terminate a request initiated by the PolarDB-X instance.

To terminate an SQL statement that is being executed on the PolarDB-X instance, you must use tools to connect to the PolarDB-X instance. You can use tools such as the MySQL command line and . Then, execute the KILL statement on the PolarDB-X instance.

Syntax

```
KILL 'PROCESS_ID' | 'ALL'
```

The KILL statement can be used in the following two ways:

- Execute the `KILL 'PROCESS_ID'` statement to terminate a specified SQL statement.

  The PROCESS_ID parameter is obtained from the ID column in the responses of the `SHOW PROCESSLIST` statement.

  > **Note** Different from MySQL, the PolarDB-X instance returns a string instead of a number in the ID column. Therefore, the PROCESS_ID parameter must be enclosed in single quotation marks (') in the KILL statement.

  Examples

  Execute the `KILL '0-0-521570';` statement. The following response is returned:

  ```
  Query OK, 0 rows affected (0.01 sec)
  ```

- Execute the `KILL 'ALL'` statement to terminate all the SQL statements executed by the PolarDB-X instance in the current logical database.

  When the underlying ApsaraDB RDS for MySQL instance is overloaded due to several SQL statements, you can execute the `KILL 'ALL'` statement to terminate all the SQL statements that are being executed in the current logical PolarDB-X database.

  All SQL statements indicated by PROCESS that meet the following conditions can be terminated by the `KILL 'ALL'` statement:

  - The value of the User parameter for the physical SQL statement indicated by PROCESS is a username created by the PolarDB-X instance in the ApsaraDB RDS for MySQL instance.
  - The physical SQL statement indicated by PROCESS is executing a query, which means that the value of COMMAND is Query.

  PolarDB-X instances in earlier versions do not support the `KILL 'ALL'` statement. An error will be reported if this statement is being executed in these instances. To resolve this problem, you can upgrade the version of the PolarDB-X instance.

5.8. Custom hints

> **Note** This topic is applicable to PolarDB-X 5.3 and later. For earlier versions, see PolarDB-X 5.2 hints.

5.8.1. Introduction to hints

As a supplement to the SQL syntax, hints play a critical role in relational databases. They allow you to modify execution plans of SQL statements by using the relevant syntax. This way, you can optimize the SQL statements. PolarDB-X also provides special hint syntax.

For example, assume that you know the data is stored in table shards in specific database shards. If you need to route an SQL statement directly to the database shards for execution, you can use custom hints provided by PolarDB-X.

```
SELECT /*+TDDL:node('node_name')*/* FROM table_name;
```

In the preceding SQL statement, the part between `/*` and `*/` is a PolarDB-X hint. This means that `+TDDL:node('node_name')` is a hint. The hint specifies the ApsaraDB RDS for MySQL database shard where the SQL statement will be executed.

> **Note**
>
> - PolarDB-X hints can be in the formats of `/*+TDDL:hint_command*/` and `/!+TDDL:hint_command*/`.
> - In the MySQL command-line client, you may need to execute an SQL statement that contains a PolarDB-X hint in the format of `/*+TDDL:hint_command*/`. In this case, add the `-c` parameter to the logon command. Otherwise, the client deletes the PolarDB-X hint before it sends the SQL statement to the server for execution because the PolarDB-X hint is in the format of a MySQL comment. In this case, the hint fails to take effect. For more information, see MySQL Client Options.

Syntax of PolarDB-X hints

Basic syntax

```
/*+TDDL: hint_command [hint_command ...] */
/!+TDDL: hint_command [hint_command ...] */
```

PolarDB-X hints are based on the MySQL comment syntax. A hint is located between `/*` and `*/` or between `/!` and `*/`, and must begin with `+TDDL:`. The `hint_command` parameter indicates a PolarDB-X hint command related to a specific operation. Multiple values of `hint_command` are separated with spaces.

Examples

```
# Query the names of physical table shards in each database shard.
/*+TDDL:scan()*/SHOW TABLES;
# Route the query to database shard 0000 of a read-only ApsaraDB RDS for MySQL instance.
/*+TDDL:node(0) slave()*/SELECT * FROM t1;
```

In the example, `/*+TDDL:scan()*/` and `/*+TDDL:node(0) slave()*/` are PolarDB-X hints that begin with `+TDDL:`. The `scan()`, `node(0)`, and `slave()` parameters are PolarDB-X hint commands. Hint commands are separated with spaces.

- Use a hint in an SQL statement:

  PolarDB-X allows you to use hints in DML, DDL, and data access language (DAL) statements. The following section describes the syntax.

  - For all statements that support hints, you can specify a hint at the beginning of the statements, as shown in the following example:

    ```
    /*+TDDL: ... */ SELECT ...
    /*+TDDL: ... */ INSERT ...
    /*+TDDL: ... */ REPLACE ...
    /*+TDDL: ... */ UPDATE ...
    /*+TDDL: ... */ DELETE ...
    /*+TDDL: ... */ CREATE TABLE ...
    /*+TDDL: ... */ ALTER TABLE ...
    /*+TDDL: ... */ DROP TABLE ...
    /*+TDDL: ... */ SHOW ...
    ...
    ```

  - For DML statements, you can specify a hint behind the first keyword of the statements, as shown in the following example:

    ```
    SELECT /*+TDDL: ... */ ...
    INSERT /*+TDDL: ... */ ...
    REPLACE /*+TDDL: ... */ ...
    UPDATE /*+TDDL: ... */ ...
    DELETE /*+TDDL: ... */ ...
    ...
    ```

    > **Note** Different hints support different statements. For more information, see the hints in the following topics.

- Use multiple hint commands in an SQL statement:

  PolarDB-X allows you to use multiple hint commands in a hint in an SQL statement.

  ```
  SELECT /*+TDDL:node(0) slave()*/ ... ;
  ```

PolarDB-X has the following limits on the use of multiple hint commands in a hint in an SQL statement:

```
# A single SQL statement cannot contain multiple hints.
SELECT /*+TDDL:node(0)*/ /*+TDDL:slave()*/ ... ;
# A hint cannot contain duplicate hint commands.
SELECT /*+TDDL:node(0) node(1)*/ ... ;
```

Classification of PolarDB-X hints

PolarDB-X hints are classified into the following categories based on operation types:

- Read/write splitting
- Specify a timeout period for an SQL statement
- Specify a database shard to run an SQL statement
- Scan all or some of database shards and table shards

5.8.2. Read/write splitting

This topic describes the read/write splitting feature provided by PolarDB-X.

PolarDB-X provides transparent read/write splitting at the application layer. Data synchronization between primary and read-only ApsaraDB RDS for MySQL instances has a latency of several milliseconds. If you need to read the changed data immediately after data in the primary ApsaraDB RDS for MySQL instance is changed, you must ensure that the SQL statement for reading data is routed to the primary ApsaraDB RDS for MySQL instance. To meet this demand, PolarDB-X provides custom hints for read/write splitting. These custom hints allow you to route SQL statements to a specified primary or read-only ApsaraDB RDS for MySQL instance.

> **Note** This topic is applicable to PolarDB-X 5.3 and later. For more information about custom hints in earlier versions, see Read/write splitting.

Syntax

```
/*+TDDL:
  master()
  |slave()
*/
```

The custom hints allow you to specify whether to execute an SQL statement on a primary or read-only ApsaraDB RDS for MySQL instance. When you use the custom hint `/*+TDDL:slave()*/`, if a primary ApsaraDB RDS for MySQL instance has multiple read-only ApsaraDB RDS for MySQL instances, the PolarDB-X instance randomly selects a read-only ApsaraDB RDS for MySQL instance based on its weight, to execute the SQL statement.

> Note
>
> - PolarDB-X hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/` .
> - In the MySQL command-line client, you may need to execute an SQL statement that contains a PolarDB-X hint in the format of `/*+TDDL:hint_command*/` . In this case, add the `-c` parameter to the logon command. Otherwise, the client deletes the PolarDB-X hint before it sends the SQL statement to the server for execution because the PolarDB-X hint is in the format of a MySQL comment. In this case, the hint fails to take effect. For more information, see MySQL Client Options.

Examples

- Execute an SQL statement on a specified primary ApsaraDB RDS for MySQL instance:

  ```
  SELECT /*+TDDL:master()*/ * FROM table_name;
  ```

  After the custom hint `/*+TDDL:master()*/` is added behind the first keyword in the SQL statement, this SQL statement is routed to the primary ApsaraDB RDS for MySQL instance.

- Execute an SQL statement on a specified read-only ApsaraDB RDS for MySQL instance:

  ```
  SELECT /*+TDDL:slave()*/ * FROM table_name;
  ```

  After the custom hint `/*+TDDL:slave()*/` is added behind the first keyword in the SQL statement, this SQL statement is randomly routed to a read-only ApsaraDB RDS for MySQL instance based on its allocated weight.

Considerations

- The custom hints for read/write splitting are only applicable to read SQL statements for non-transactional data. SQL statements for transactional data and write SQL statements are still routed to the primary ApsaraDB RDS for MySQL instance.
- When you use the `/*+TDDL:slave()*/` hint, the PolarDB-X instance routes the SQL statement randomly to a read-only ApsaraDB RDS for MySQL instance based on the allocated weight. If no read-only ApsaraDB RDS for MySQL instance is available, no error is reported. Instead, the primary ApsaraDB RDS for MySQL instance is selected to execute the SQL statement.

5.8.3. Specify a timeout period for an SQL statement

In PolarDB-X, the SQL statements for PolarDB-X instances and ApsaraDB RDS for MySQL instances are timed out after 900 seconds (which can be adjusted) by default. However, for some slow SQL statements, the execution duration may exceed 900 seconds. For these slow SQL statements, PolarDB-X provides a custom hint to adjust their timeout periods. You can use this custom hint to adjust the SQL execution duration as needed.

> **Note** This topic is applicable to PolarDB-X 5.3 and later. For earlier versions, see Specify a timeout period for an SQL statement.

Syntax

The syntax of the PolarDB-X hint for specifying a timeout period for an SQL statement is as follows:

```
/*+TDDL:SOCKET_TIMEOUT(time)*/
```

The `SOCKET_TIMEOUT` parameter is measured in milliseconds. With this custom hint, you can adjust the timeout period for the SQL statement based on business requirements.

> **Note**
>
> - PolarDB-X hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/` .
> - In the MySQL command-line client, you may need to execute an SQL statement that contains a PolarDB-X hint in the format of `/*+TDDL:hint_command*/` . In this case, add the `-c` parameter to the logon command. Otherwise, the client deletes the PolarDB-X hint before it sends the SQL statement to the server for execution because the PolarDB-X hint is in the format of a MySQL comment. In this case, the hint fails to take effect. For more information, see MySQL Client Options.

Examples

Set the timeout period of an SQL statement to 40 seconds:

```
/*+TDDL:SOCKET_TIMEOUT(40000)*/SELECT * FROM t_item;
```

> **Note** A longer timeout period causes database resources to be occupied for a longer period of time. If excessive SQL statements are executed over a long time within the same period, a large number of database resources may be consumed. This will make users unable to use PolarDB-X properly. In this case, we need to use this custom hint to optimize the SQL statements that take a long time to execute.

5.8.4. Execute an SQL statement on a specified database shard

When you execute SQL statements on a PolarDB-X instance, you may find that some SQL statements are not supported by the PolarDB-X instance. You can use the `NODE` hint provided by PolarDB-X to route the SQL statements to one or more database shards. If you need to query the data in a specified database shard or the data in a specified table shard of a known database shard, you can use the `NODE` hint to directly route the SQL statement to the database shard.

> **Note** This topic is applicable to PolarDB-X 5.3 and later. For more information about custom hints in earlier versions, see Specify a database shard to run an SQL statement.

Syntax

The `NODE` hint allows you to specify a database shard to execute an SQL statement by using its shard name. A shard name uniquely identifies a database shard in a PolarDB-X instance. You can execute the SHOW NODE statement to obtain the shard name.

Execute an SQL statement on a database shard by specifying the shard name

This custom hint allows you to specify one or more database shards to execute an SQL statement.

> **Note** Assume that you execute the INSERT statement in a table that uses a sequence. The sequence will not take effect if you use the NODE hint in the INSERT statement. For more information, see Limits and precautions for sequences.

- Execute an SQL statement on a specified database shard:

  ```
  /*+TDDL:node('node_name')*/
  ```

  `node_name` indicates the shard name. This custom hint provided by PolarDB-X allows you to route the SQL statement to the database shard specified by `node_name`.

- Execute an SQL statement on multiple database shards:

  ```
  /*+TDDL:node('node_name'['node_name1','node_name2'])*/
  ```

  You can specify multiple shard names in the parameters and route the SQL statement to multiple database shards. Separate multiple shard names with commas (,).

> **Note**
>
> - When this custom hint is used, the PolarDB-X instance directly routes the SQL statement to the specified database shards. Therefore, the specified shard names in the SQL statement must correspond to existing database shards.
> - The `NODE` hint can be used in DML, DDL, and DAL statements.
> - PolarDB-X hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/`.
> - In the MySQL command-line client, you may need to execute an SQL statement that contains a PolarDB-X hint in the format of `/*+TDDL:hint_command*/`. In this case, add the `-c` parameter to the logon command. Otherwise, the client deletes the PolarDB-X hint before it sends the SQL statement to the server for execution because the PolarDB-X hint is in the format of a MySQL comment. In this case, the hint fails to take effect. For more information, see MySQL Client Options.

Examples

The following example shows the responses for executing the `SHOW NODE` statement on a PolarDB-X logical database that is named `drds_test`.

```
*************************** 1. row ***************************
          ID: 0
        NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS
 MASTER_READ_COUNT: 212
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
*************************** 2. row ***************************
          ID: 1
        NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0001_RDS
 MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
*************************** 3. row ***************************
          ID: 2
        NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0002_RDS
 MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
*************************** 4. row ***************************
          ID: 3
        NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0003_RDS
 MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
*************************** 5. row ***************************
          ID: 4
        NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0004_RDS
 MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
*************************** 6. row ***************************
          ID: 5
        NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0005_RDS
```

```
 MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
*************************** 7. row ******************
           ID: 6
         NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0006_RDS
 MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
*************************** 8. row ******************
           ID: 7
         NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0007_RDS
 MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
8 rows in set (0.02 sec)
```

Each database shard has the `NAME` attribute, which indicates the shard name corresponding to the database shard. Each shard name uniquely corresponds to one database shard name. For example, the shard name `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0003_RDS` corresponds to the database shard name `drds_test_vtla_0003`. Therefore, after you obtain the shard name, you can use the custom hint provided by PolarDB-X to specify the database shard on which you want to execute an SQL statement.

- Execute an SQL statement on database shard 0:

  ```
  SELECT /*TDDL:node('DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS')*/ * FROM table_name;
  ```

- Execute an SQL statement on multiple database shards:

  ```
  SELECT /*TDDL:node('DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS','DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0006_RDS')*/ * FROM table_name;
  ```

  This SQL statement will be executed on the database shards specified by the shard names `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS` and `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0006_RDS`.

- View the execution plan of an SQL statement on database shard 0:

```
/*TDDL:node('DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS')*/ EXPLAIN SELECT * FROM table_name;
```

5.8.5. Scan some or all of the database shards and table shards

You can route an SQL statement to one or more database shards. You can also route an SQL statement to some or all of the database shards and table shards by using the `SCAN` hint provided by PolarDB-X. You can use the `SCAN` hint to route an SQL statement to all database shards at a time. For example, you can view all the table shards in a specified database shard or view the data volume of each physical table shard that corresponds to a specified logical table.

> **Note** This topic is applicable to PolarDB-X 5.3 and later. For more information about custom hints in earlier versions, see Scan all database shards and table shards.

The SCAN hint allows you to execute an SQL statement by using the following methods:

- Execute an SQL statement on all table shards in all database shards.
- Execute an SQL statement on all table shards in specified database shards.
- Execute an SQL statement on specified table shards in specified database shards. The names of the physical database shards and table shards are calculated based on given conditions.
- Execute an SQL statement on table shards in database shards by explicitly specifying the names of the physical table shards.

The `SCAN` hint can be used in data manipulation language (DML) statements, DDL statements, and some data access language (DAL) statements.

> **Note**
>
> - PolarDB-X hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/` .
> - In the MySQL command-line client, you may need to execute an SQL statement that contains a PolarDB-X hint in the format of `/*+TDDL:hint_command*/` . In this case, add the `-c` parameter to the logon command. Otherwise, the client deletes the PolarDB-X hint before it sends the SQL statement to the server for execution because the PolarDB-X hint is in the format of a MySQL comment. In this case, the hint fails to take effect. For more information, see MySQL Client Options.

Syntax

```
SCAN HINT
# Route an SQL statement to all table shards in all database shards.
SCAN()
# Route an SQL statement to all table shards in specified database shards.
SCAN(NODE="node_list")          # Specify the database shards.
# Route an SQL statement to specified table shards in specified database shards. The names of the physical database shards and table shards are calculated based on given conditions.
SCAN(
 [TABLE=]"table_name_list"       # Specify the name of the logical table.
 , CONDITION="condition_string"    # Calculate the names of the physical database shards and table shards based on the values of the TABLE and CONDITION parameters.
 [, NODE="node_list"] )          # Filter the results obtained based on the value of the CONDITION parameter to retain only the results about the specified physical database shards.
# Route an SQL statement to table shards in database shards by explicitly specifying the names of the physical table shards.
SCAN(
 [TABLE=]"table_name_list"       # Specify the name of the logical table.
 , REAL_TABLE=("table_name_list")  # Specify the name of the physical table shards. The same name of physical table shards is applied to all physical database shards.
 [, NODE="node_list"] )          # Filter the results obtained based on the value of the CONDITION parameter to retain only the results about the specified physical database shards.
# Specify the names of physical table shards or logical tables.
table_name_list:
  table_name [, table_name]...
# Specify physical database shards by using GROUP_KEY and GROUP_INDEX. You can obtain their values by executing the `SHOW NODE` statement.
node_list:
  {group_key | group_index} [, {group_key | group_index}]...
# Execute an SQL WHERE statement. You must specify conditions for each table, such as t1.id = 2 and t2.id = 2.
condition_string:
  where_condition
```

Examples

- Execute the following SQL statement on all table shards in all database shards:

  ```
  SELECT /*+TDDL:scan()*/ COUNT(1) FROM t1
  ```

  After this statement is executed, the SQL statement is routed to all the physical table shards corresponding to logical table `t1`, and the result sets are merged and returned.

- Execute the following SQL statement on all table shards in specified database shards:

```
SELECT /*+TDDL:scan(node='0,1,2')*/ COUNT(1) FROM t1
```

  After this statement is executed, all the physical table shards corresponding to logical table `t1` in database shards 0000, 0001, and 0002 are calculated. Then, the SQL statement is routed to the physical table shards, and the result sets are merged and returned.

- Execute the following SQL statement on specified table shards based on conditions:

```
SELECT /*+TDDL:scan('t1', condition='t1.id = 2')*/ COUNT(1) FROM t1
```

  After this statement is executed, all the physical table shards that correspond to logical table `t1` and meet the conditions are calculated. Then, the SQL statement is routed to the specified physical table shards, and the result sets are merged and returned.

- Execute the following JOIN statement on the specified table shards based on conditions:

```
SELECT /*+TDDL:scan('t1, t2', condition='t1.id = 2 and t2.id = 2')*/ * FROM t1 a JOIN t2 b ON a.id = b.id WHE
RE b.name = "test"
```

  After this statement is executed, all the physical table shards that correspond to logical tables `t1` and `t2` and meet the conditions are calculated. Then, the SQL statement is routed to the specified physical table shards, and the result sets are merged and returned.

  > **Notice** Before you use this custom hint, you must ensure that data from logical tables t1 and t2 is partitioned into the same number of table shards in the same number of database shards. Otherwise, the database shards that are calculated by the PolarDB-X instance based on the conditions are different, and an error is returned.

- Execute the following SQL statement on table shards in database shards by explicitly specifying the names of the physical table shards:

```
SELECT /*+TDDL:scan('t1', real_table=("t1_00", "t1_01"))*/ COUNT(1) FROM t1
```

  After this statement is executed, the SQL statement is routed to table shards `t1_00``t1_01` in all database shards, and the result sets are merged and returned.

- Execute the following JOIN statement on table shards in database shards by explicitly specifying the names of the physical table shards:

```
SELECT /*+TDDL:scan('t1, t2', real_table=("t1_00,t2_00", "t1_01,t2_01"))*/ * FROM t1 a JOIN t2 b ON a.id =
b.id WHERE b.name = "test";
```

  After this statement is executed, the SQL statement is routed to table shards `t1_00`, `t2_00`, `t1_01`, and `t2_01` in all database shards, and the result sets are merged and returned.

5.8.6. INDEX HINT

This topic describes the syntax of the INDEX hint and provides examples.

- PolarDB-X supports global secondary indexes (GSIs). The INDEX hint allows you to obtain query results from a specified GSI.
- The INDEX hint takes effect only for SELECT statements.

> **Note** The version of the MySQL engine in your ApsaraDB RDS for MySQL instance must be 5.7 or later, and the version of your PolarDB-X instance must be 5.4.1 or later.

Syntax

```
FORCE INDEX
tbl_name [[AS] alias] [index_hint]
index_hint:
  FORCE INDEX({index_name})INDEX()
/*+TDDL:
  INDEX({table_name | table_alias}, {index_name})
*/
```

The PolarDB-X INDEX hint has the following two variants:

- `FORCE INDEX()` : This syntax is the same as that of MySQL FORCE INDEX.
- `INDEX()` : In this syntax, a GSI is specified with a table name (or alias) and an index name. This hint does not take effect in the following cases:
  - The query does not contain the specified table name or alias.
  - The specified GSI is not in the specified table.

> **Note**
>
> - PolarDB-X hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/` .
> - In the MySQL command-line client, you may need to execute an SQL statement that contains a PolarDB-X hint in the format of `/*+TDDL:hint_command*/` . In this case, add the `-c` parameter to the logon command. Otherwise, the client deletes the PolarDB-X hint before it sends the SQL statement to the server for execution because the PolarDB-X hint is in the format of a MySQL comment. In this case, the hint fails to take effect. For more information, see MySQL Client Options.

Examples

```
CREATE TABLE t_order (
 `id` bigint(11) NOT NULL AUTO_INCREMENT,
 `order_id` varchar(20) DEFAULT NULL,
 `buyer_id` varchar(20) DEFAULT NULL,
 `seller_id` varchar(20) DEFAULT NULL,
 `order_snapshot` longtext DEFAULT NULL,
 `order_detail` longtext DEFAULT NULL,
 PRIMARY KEY (`id`),
 GLOBAL INDEX `g_i_seller`(`seller_id`) dbpartition by hash(`seller_id`),
 UNIQUE GLOBAL INDEX `g_i_buyer` (`buyer_id`) COVERING(`seller_id`, `order_snapshot`)
  dbpartition by hash(`buyer_id`) tbpartition by hash(`buyer_id`) tbpartitions 3
) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by hash(`order_id`);
```

Specify the `g_i_seller` GSI by using the FORCE INDEX hint in the FROM clause:

```
SELECT a.*, b.order_id
  FROM t_seller a
    JOIN t_order b FORCE INDEX(g_i_seller) ON a.seller_id = b.seller_id
  WHERE a.seller_nick="abc";
```

Specify the `g_i_buyer` GSI by using the INDEX hint and a table alias:

```
/*+TDDL:index(a, g_i_buyer)*/ SELECT * FROM t_order a WHERE a.buyer_id = 123
```

5.9. PolarDB-X 5.2 hints

5.9.1. Introduction to hints

As a supplement to the SQL syntax, hints play a critical role in relational databases. They allow you to modify execution plans of SQL statements by using the relevant syntax. This way, you can optimize the SQL statements.

Classification of PolarDB-X hints

PolarDB-X provides special hint syntax.

For example, assume that you know the data is stored in table shards in specific database shards. If you need to route an SQL statement directly to the database shards for execution, you can use custom hints provided by PolarDB-X.

```
/! TDDL:NODE IN('node_name', ...) */SELECT * FROM table_name;
```

In the preceding SQL statement, the part between `/!` and `*/` is a PolarDB-X hint. This means that `TDDL:node in('node_name', ...)` is a hint. The hint specifies the ApsaraDB RDS for MySQL database shard where the SQL statement will be executed.

> **Note**
>
> - PolarDB-X hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/`.
> - In the MySQL command-line client, you may need to execute an SQL statement that contains a PolarDB-X hint in the format of `/*+TDDL:hint_command*/`. In this case, add the `-c` parameter to the logon command. Otherwise, the client deletes the PolarDB-X hint before it sends the SQL statement to the server for execution because the PolarDB-X hint is in the format of a MySQL comment. In this case, the hint fails to take effect. For more information, see MySQL Client Options.

Syntax of PolarDB-X hints

Basic syntax

```
/! TDDL:hint command*/
```

PolarDB-X hints are based on the MySQL comment syntax. Therefore, a PolarDB-X hint is located between `/!` and `*/`, and must begin with `TDDL:`. The `hint command` parameter indicates a PolarDB-X hint command related to a specific operation. For example, a PolarDB-X hint is added to the following SQL statement to show the name of each database shard.

```
/! TDDL:SCAN*/SHOW TABLES;
```

In this SQL statement, `/! TDDL:SCAN*/` is the PolarDB-X hint that begins with `TDDL:`, and `SCAN` is a PolarDB-X hint command.

5.9.2. Read/write splitting

This topic describes the read/write splitting feature provided by PolarDB-X.

PolarDB-X provides transparent read/write splitting at the application layer. Data synchronization between primary and read-only ApsaraDB RDS for MySQL instances has a latency of several milliseconds. If you need to read the changed data immediately after data in the primary ApsaraDB RDS for MySQL instance is changed, you must ensure that the SQL statement for reading data is routed to the primary ApsaraDB RDS for MySQL instance. To meet this demand, PolarDB-X provides custom hints for read/write splitting. These custom hints allow you to route SQL statements to a specified primary or read-only ApsaraDB RDS for MySQL instance.

Syntax

```
/! TDDL:MASTER|SLAVE*/
```

The custom hints allow you to specify whether to execute an SQL statement on a primary or read-only ApsaraDB RDS for MySQL instance. When you use the custom hint `/!TDDL:SLAVE*/`, if a primary ApsaraDB RDS for MySQL instance has multiple read-only ApsaraDB RDS for MySQL instances, the PolarDB-X instance randomly selects a read-only ApsaraDB RDS for MySQL instance based on its weight, to execute the SQL statement.

> Note
>
> - PolarDB-X hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/`.
> - In the MySQL command-line client, you may need to execute an SQL statement that contains a PolarDB-X hint in the format of `/*+TDDL:hint_command*/`. In this case, add the `-c` parameter to the logon command. Otherwise, the client deletes the PolarDB-X hint before it sends the SQL statement to the server for execution because the PolarDB-X hint is in the format of a MySQL comment. In this case, the hint fails to take effect. For more information, see MySQL Client Options.

Examples

- Execute an SQL statement on a specified primary ApsaraDB RDS for MySQL instance:

  ```
  /! TDDL:MASTER*/SELECT * FROM table_name;
  ```

  After the custom hint `/! TDDL:MASTER*/` is added behind the keyword of the SQL statement, this SQL statement is routed to the primary ApsaraDB RDS for MySQL instance.

- Execute an SQL statement on a specified read-only ApsaraDB RDS for MySQL instance:

  ```
  /! TDDL:SLAVE*/SELECT * FROM table_name;
  ```

  After the custom hint `/! TDDL:SLAVE*/` is added behind the keyword of the SQL statement, this SQL statement is randomly routed to a read-only ApsaraDB RDS for MySQL instance based on the allocated weight.

Considerations

- The custom hints for read/write splitting are only applicable to read SQL statements for non-transactional data. SQL statements for transactional data and write SQL statements are still routed to the primary ApsaraDB RDS for MySQL instance.
- When you use the `/*+TDDL:slave()*/` hint, the PolarDB-X instance routes the SQL statement randomly to a read-only ApsaraDB RDS for MySQL instance based on the allocated weight. If no read-only ApsaraDB RDS for MySQL instance is available, no error is reported. Instead, the primary ApsaraDB RDS for MySQL instance is selected to execute the SQL statement.

5.9.3. Prevent the delay from a read-only ApsaraDB RDS for MySQL instance

Normally, if you have configured a read-only ApsaraDB for RDS instance for the primary ApsaraDB RDS for MySQL instance of a logical database in a PolarDB-X instance and set read traffic for both the primary and read-only ApsaraDB RDS for MySQL instances, PolarDB-X routes SQL statements to the primary and read-only ApsaraDB RDS for MySQL instances based on the read/write ratio. However, if asynchronous data replication between the primary and read-only ApsaraDB RDS for MySQL instances has a high delay, an error is reported or error results are returned when PolarDB-X routes the SQL statements to the read-only ApsaraDB RDS for MySQL instance.

To address this issue, the PolarDB-X instance provides a custom hint to cut off the delay of the read-only instance. Specifically, based on the maximum delay of primary/secondary replication, PolarDB-X determines whether to route the SQL statement to the primary or the read-only ApsaraDB RDS for MySQL instance.

Syntax

```
/! TDDL:SQL_DELAY_CUTOFF=time*/
```

With this custom hint, you can specify the value of `SQL_DELAY_CUTOFF`. When the value of SQL_DELAY (primary/secondary replication delay of ApsaraDB RDS for MySQL) for the read-only ApsaraDB RDS for MySQL instance reaches or exceeds the value of `time` (which is measured in seconds), the SQL statement is routed to the primary ApsaraDB RDS for MySQL instance.

> **Note**
>
> - PolarDB-X hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/`.
> - In the MySQL command-line client, you may need to execute an SQL statement that contains a PolarDB-X hint in the format of `/*+TDDL:hint_command*/`. In this case, add the `-c` parameter to the logon command. Otherwise, the client deletes the PolarDB-X hint before it sends the SQL statement to the server for execution because the PolarDB-X hint is in the format of a MySQL comment. In this case, the hint fails to take effect. For more information, see MySQL Client Options.

Examples

- Set the primary/secondary replication delay to 5 seconds:

  ```
  /! TDDL:SQL_DELAY_CUTOFF=5*/SELECT * FROM table_name;
  ```

  In this SQL statement, the value of `SQL_DELAY_CUTOFF` is set to 5. Therefore, when the value of `SQL_DELAY` for the read-only ApsaraDB RDS for MySQL instance reaches or exceeds 5 seconds, the SQL statement is routed to the primary ApsaraDB RDS for MySQL instance.

- Use the custom hint for delay cutoff with other custom hints:

  ```
  /! TDDL:SLAVE AND SQL_DELAY_CUTOFF=5*/SELECT * FROM table_name;
  ```

The custom hint for cutting off the delay of the read-only ApsaraDB RDS for MySQL instance can be used with other hints. By default, the SQL query request is routed to a read-only ApsaraDB RDS for MySQL instance. However, when the primary/secondary replication delay reaches or exceeds 5 seconds, the SQL query request is routed to the primary ApsaraDB RDS for MySQL instance.

5.9.4. Specify a timeout period for an SQL statement

In PolarDB-X, the SQL statements for PolarDB-X instances and ApsaraDB RDS for MySQL instances are timed out after 900 seconds (which can be adjusted) by default. However, for some slow SQL statements, the execution duration may exceed 900 seconds. For these slow SQL statements, PolarDB-X provides a custom hint to adjust their timeout periods. You can use this custom hint to adjust the SQL execution duration as needed.

Syntax

The syntax of the PolarDB-X hint for specifying a timeout period for an SQL statement is as follows:

```
/! TDDL:SOCKET_TIMEOUT=time*/
```

The `SOCKET_TIMEOUT` parameter is measured in milliseconds. With this custom hint, you can adjust the timeout period for the SQL statement based on business requirements.

> **Note**
>
> - PolarDB-X hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/` .
> - In the MySQL command-line client, you may need to execute an SQL statement that contains a PolarDB-X hint in the format of `/*+TDDL:hint_command*/` . In this case, add the `-c` parameter to the logon command. Otherwise, the client deletes the PolarDB-X hint before it sends the SQL statement to the server for execution because the PolarDB-X hint is in the format of a MySQL comment. In this case, the hint fails to take effect. For more information, see MySQL Client Options.

Examples

Set the timeout period of an SQL statement to 40 seconds:

```
/! TDDL:SOCKET_TIMEOUT=40000*/SELECT * FROM t_item;
```

> **Note** A longer timeout period causes database resources to be occupied for a longer period of time. If excessive SQL statements are executed over a long time within the same period, a large number of database resources may be consumed. This will make users unable to use PolarDB-X properly. In this case, we need to use this custom hint to optimize the SQL statements that take a long time to execute.

5.9.5. Execute an SQL statement on a specified database shard

When you execute SQL statements on a PolarDB-X instance, you may find that some SQL statements are not supported by the PolarDB-X instance. You can use the NODE hint provided by PolarDB-X to route the SQL statements to one or more database shards. If you need to query the data in a specified database shard or the data in a specified table shard, you can use the custom hint to directly route the SQL statement to the database shard.

Syntax

The NODE hint allows you to specify a database shard to execute an SQL statement by using its shard name or the value of the database shard key. A shard name uniquely identifies a database shard in a PolarDB-X instance. You can execute the `SHOW NODE` statement to obtain the shard name.

> **Note** Assume that you execute the INSERT statement in a table that uses a sequence. The sequence will not take effect if you use the NODE hint in the INSERT statement. For more information, see Limits and precautions for sequences.

- Execute an SQL statement on a database shard by specifying the shard name

  This custom hint allows you to specify one or more database shards to execute an SQL statement.

  - Execute an SQL statement on a specified database shard:

    ```
    /! TDDL:NODE='node_name'*/
    ```

    `node_name` indicates the shard name. This custom hint provided by PolarDB-X allows you to route the SQL statement to the database shard specified by `node_name`.

  - Execute an SQL statement on multiple database shards:

    ```
    /! TDDL:NODE IN ('node_name'[,'node_name1','node_name2'])*/
    ```

    The `IN` keyword is used to specify multiple shard names. This custom hint allows you to route the SQL statement to multiple database shards. Separate multiple shard names with commas (,).

    > **Note** When this custom hint is used, the PolarDB-X instance directly routes the SQL statement to the specified database shards. Therefore, the specified shard names in the SQL statement must correspond to existing database shards.

- Execute an SQL statement on a database shard by specifying the value of the database shard key

  ```
  /! TDDL:table_name.partition_key=value [and table_name1.partition_key=value1]*/
  ```

In this PolarDB-X hint, `table_name` is the name of a logical table, and this table is a partitioned table. In addition, `partition_key` is a shard key, and `value` is the value specified for the shard key. In this custom hint, you can use the `and` keyword to specify the shard keys of multiple partitioned tables. When this PolarDB-X hint is used, the PolarDB-X instance calculates to determine the database shards and table shards where the SQL statement will be executed. Then, the instance routes the SQL statement to the corresponding database shards.

> Note
>
> - PolarDB-X hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/`.
> - In the MySQL command-line client, you may need to execute an SQL statement that contains a PolarDB-X hint in the format of `/*+TDDL:hint_command*/`. In this case, add the `-c` parameter to the logon command. Otherwise, the client deletes the PolarDB-X hint before it sends the SQL statement to the server for execution because the PolarDB-X hint is in the format of a MySQL comment. In this case, the hint fails to take effect. For more information, see MySQL Client Options.

Examples

The following example shows the responses for executing the `SHOW NODE` statement on a PolarDB-X logical database that is named `drds_test`.

```
*************************** 1. row ******************
           ID: 0
         NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS
 MASTER_READ_COUNT: 212
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
*************************** 2. row ******************
           ID: 1
         NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0001_RDS
 MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
*************************** 3. row ******************
           ID: 2
         NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0002_RDS
 MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
```

```
*************************** 4. row ******************
          ID: 3
        NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0003_RDS
 MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
*************************** 5. row ******************
          ID: 4
        NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0004_RDS
 MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
*************************** 6. row ******************
          ID: 5
        NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0005_RDS
 MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
*************************** 7. row ******************
          ID: 6
        NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0006_RDS
 MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
*************************** 8. row ******************
          ID: 7
        NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0007_RDS
 MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
 SLAVE_READ_PERCENT: 0%
8 rows in set (0.02 sec)
```

Each database shard has the `NAME` attribute, which indicates the shard name corresponding to the database shard. Each shard name uniquely corresponds to one database shard name. For example, the shard name `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0003_RDS` corresponds to the database shard name `drds_test_vtla_0003`. Therefore, after you obtain the shard name, you can use the custom hint provided by PolarDB-X to specify the database shard on which you want to execute an SQL statement.

- Execute an SQL statement on database shard 0:

  ```
  /! TDDL:NODE='DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS'*/SELECT * FROM table_name;
  ```

- Execute an SQL statement on multiple database shards:

  ```
  /! TDDL:NODE IN('DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS','DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0006_RDS')*/SELECT * FROM table_name;
  ```

  This SQL statement will be executed on the database shards specified by the shard names `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS` and `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0006_RDS`.

- View the execution plan of an SQL statement on a specified database shard:

  ```
  /! TDDL:NODE='DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS'*/EXPLAIN SELECT * FROM table_name;
  ```

  After this SQL statement is executed, the execution plan of the `SELECT` statement on the database shard corresponding to the shard name `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS` will be returned.

- Execute an SQL statement on a database shard by specifying the value of the database shard key:

  PolarDB-X does not support subqueries in the `SET` clause of an `UPDATE` statement, because a shard key must be specified for `UPDATE` statements in PolarDB-X. To address this issue, PolarDB-X provides a custom hint to directly route the statement to a database shard.

  For example, the following CREATE TABLE statements can be used to create two logical tables t1 and t2 that are partitioned into table shards in database shards:

```
CREATE TABLE `t1` (
  `id` bigint(20) NOT NULL,
  `name` varchar(20) NOT NULL,
  `val` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`id`)
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by hash(`id`) tbpartition by hash(`name`) tbpart
itions 3
 CREATE TABLE `t2` (
  `id` bigint(20) NOT NULL,
  `name` varchar(20) NOT NULL,
  `val` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`id`)
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by hash(`id`) tbpartition by hash(`name`) tbpart
itions 3
```

The following SQL statement is to be executed for the two tables:

```
UPDATE t1 SET val=(SELECT val FROM t2 WHERE id=1) WHERE id=1;
```

If this statement is directly executed on a PolarDB-X instance, an error will be returned indicating that this statement is not supported. You can add the PolarDB-X hint to this SQL statement before you submit it to the PolarDB-X instance. The following content shows the example SQL statement:

```
/! TDDL:t1.id=1 and t2.id=1*/UPDATE t1 SET val=(SELECT val FROM t2 WHERE id=1) WHERE id=1;
```

This statement will be routed to database shards whose `id` is 1 in logical table `t1`. You can execute the `EXPLAIN` statement to view the execution plan of this SQL statement:

```
mysql> explain /! TDDL:t1.id=1 and t2.id=1*/UPDATE t1 SET val=(SELECT val FROM t2 WHERE id=1) WHERE id=1\G
*************************** 1. row ***************************
GROUP_NAME: TEST_DRDS_1485327111630IXLWTEST_DRDS_IGHF_0001_RDS
    SQL: UPDATE `t1_2` AS `t1` SET `val` = (SELECT val FROM `t2_2` AS `t2` WHERE `id` = 1) WHERE `id` = 1
   PARAMS: {}
*************************** 2. row ***************************
GROUP_NAME: TEST_DRDS_1485327111630IXLWTEST_DRDS_IGHF_0001_RDS
    SQL: UPDATE `t1_1` AS `t1` SET `val` = (SELECT val FROM `t2_1` AS `t2` WHERE `id` = 1) WHERE `id` = 1
   PARAMS: {}
*************************** 3. row ***************************
GROUP_NAME: TEST_DRDS_1485327111630IXLWTEST_DRDS_IGHF_0001_RDS
    SQL: UPDATE `t1_0` AS `t1` SET `val` = (SELECT val FROM `t2_0` AS `t2` WHERE `id` = 1) WHERE `id` = 1
   PARAMS: {}
3 rows in set (0.00 sec)
```

Based on the result set of the `EXPLAIN` statement, the SQL statement is rewritten into three statements, and these three statements are then routed to the database shards. You can further execute the SQL statement on a table shard by specifying the value of the table shard key.

```
mysql> explain /! TDDL:t1.id=1 and t2.id=1 and t1.name='1'*/UPDATE t1 SET val=(SELECT val FROM t2 WHERE id=1) WHERE id=1\G
*************************** 1. row ***************************
GROUP_NAME: TEST_DRDS_1485327111630IXLWTEST_DRDS_IGHF_0001_RDS
    SQL: UPDATE `t1_1` AS `t1` SET `val` = (SELECT val FROM `t2_1` AS `t2` WHERE `id` = 1) WHERE `id` = 1
   PARAMS: {}
1 row in set (0.00 sec)
```

Note Before you use this custom hint, you must ensure that data from logical tables t1 and t2 is partitioned into the same number of table shards in the same number of database shards. Otherwise, the database shards that are calculated by the PolarDB-X instance based on the conditions are different, and an error is returned.

5.9.6. Scan all database shards and table shards

In addition to routing an SQL statement to one or more database shards for execution, PolarDB-X provides a custom hint to allow you to scan all database shards and table shards. With this custom hint, you can route an SQL statement to each database shard at a time. For example, you can use this custom hint to view all the table shards in a specified database shard. In addition, you can use this custom hint to view the data volume of table shards in each database shard corresponding to a specified logical table.

Syntax

With this PolarDB-X hint, you can route an SQL statement to all database shards for execution and route an SQL statement to all database shards to perform an operation on a specified logical table.

- Route an SQL statement to all database shards for execution:

```
/! TDDL:SCAN*/
```

- Perform an operation on a specified logical table:

```
/! TDDL:SCAN='table_name'*/
```

The `table_name` parameter indicates the name of a logical table in the logical database of a PolarDB-X instance. This custom hint is provided for table shards in database shards. Ensure that the value of `table_name` is the name of a table shard in database shards.

> **Note**
>
> - PolarDB-X hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/`.
> - In the MySQL command-line client, you may need to execute an SQL statement that contains a PolarDB-X hint in the format of `/*+TDDL:hint_command*/`. In this case, add the `-c` parameter to the logon command. Otherwise, the client deletes the PolarDB-X hint before it sends the SQL statement to the server for execution because the PolarDB-X hint is in the format of a MySQL comment. In this case, the hint fails to take effect. For more information, see MySQL Client Options.

Examples

- View the data volume of a specified broadcast table in each database shard:

```
/! TDDL:SCAN*/SELECT COUNT(1) FROM table_name
```

In this SQL statement, `table_name` indicates a broadcast table. This hint causes the PolarDB-X instance to route the SQL statement to each database shard for execution. Therefore, the result sets include the total data volume of the broadcast table `table_name` in all database shards. This statement allows you to conveniently check whether the data of a broadcast table is normal.

- Scan a single-database non-partition logical table:

```
/! TDDL:SCAN*/SELECT COUNT(1) FROM table_name
```

This hint causes the PolarDB-X instance to route the SQL `select count(1) from table_name` statement to each database shard for execution. The `table_name` parameter indicates a logical table in a logical database of a PolarDB-X instance. Before using this hint, ensure that each database shard contains the table shard `table_name`. In other words, the table shard `table_name` is a logical table that is only partitioned into database shards, but not partitioned into table shards. Otherwise, an error that indicates that the table is not found will be returned.

- Scan a partitioned logical table in database shards:

```
/! TDDL:SCAN='table_name'*/SELECT COUNT(1) FROM table_name
```

When executing this statement, the PolarDB-X instance first calculates all the database shards and table shards corresponding to the logical table `table_name`, and then generates a COUNT clause for each table shard in each database shard.

- View the execution plans of all database shards:

```
/! TDDL:SCAN='table_name'*/EXPLAIN SELECT * FROM table_name;
```

5.10. Distributed transactions

5.10.1. Distributed transactions based on MySQL 5.7

This topic describes distributed transactions based on MySQL 5.7.

> **Note**
>
> - When the version of the MySQL engine in your ApsaraDB RDS for MySQL instance is 5.7 or later and that of your PolarDB-X instance is 5.3.4 or later, XA distributed transactions are automatically enabled. The user experience of XA distributed transactions is the same as that in standalone MySQL databases. No special commands are required to enable XA distributed transactions.
> - For more information about MySQL and PolarDB-X in other versions, see Distributed transactions based on MySQL 5.6.

How it works

When the version of the MySQL engine in your ApsaraDB RDS for MySQL instance is 5.7 or later, the PolarDB-X instance processes distributed transactions based on the XA protocol by default.

Usage notes

The user experience of XA distributed transactions on your PolarDB-X instance is the same as that in standalone MySQL databases in terms of the following statements:

- `SET AUTOCOMMIT=0`: starts a transaction.
- `COMMIT`: commits the current transaction.

- `ROLLBACK`: rolls back the current transaction.

If the SQL statement in a transaction involves only a single shard, the PolarDB-X instance routes the transaction directly to the ApsaraDB RDS for MySQL instance as a non-distributed transaction. If the SQL statement in the transaction is to modify the data in multiple shards, the PolarDB-X instance automatically upgrades the current transaction to a distributed transaction.

5.10.2. Distributed transactions based on MySQL 5.6

This topic describes distributed transactions based on MySQL 5.6.

How it works

The XA protocol for MySQL 5.6 is immature. Therefore, PolarDB-X independently implements two-phase commit (2PC) transaction policies for distributed transactions. If the version of the MySQL engine in your ApsaraDB RDS for MySQL instance is 5.7 or later, we recommend that you use XA transaction policies.

> **Note** The distributed transactions described in this topic are intended for users who use MySQL 5.6 or PolarDB-X earlier than 5.3.4. For more information about MySQL 5.7 or later and PolarDB-X 5.3.4 or later, see Distributed transactions based on MySQL 5.7.

Usage notes

If a transaction may involve multiple database shards, you must declare the current transaction as a distributed transaction. If a transaction involves only a single database shard, you can process the transaction as a non-distributed transaction in MySQL instead of enabling distributed transactions. No additional operations are required.

To enable distributed transactions, perform the following operation:

After transactions are enabled, execute the `SET drds_transaction_policy = '...'` statement.

To enable 2PC transactions in the MySQL command-line client, execute the following statements:

```
SET AUTOCOMMIT=0;
SET drds_transaction_policy = '2PC'; -- We recommend that you set this parameter if you use MySQL 5.6.
.... -- Here, you can execute your business SQL statement.
COMMIT; -- You can alternatively write ROLLBACK.
```

To enable 2PC transactions by using the Java database connectivity (JDBC) API, execute the following statements:

```
conn.setAutoCommit(false);
try (Statement stmt = conn.createStatement()) {
  stmt.execute("SET drds_transaction_policy = '2PC'");
}
// ... Here, you can execute your business SQL statement.
conn.commit(); // You can alternatively write rollback().
```

FAQ

Q: How can I use PolarDB-X distributed transactions in the Spring framework?

A: If you enable transactions by using the Spring `@Transactional` annotation, you can enable PolarDB-X distributed transactions by extending the transaction manager.

Example:

```
import org.springframework.jdbc.datasource.DataSourceTransactionManager;
import org.springframework.transaction.TransactionDefinition;
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
public class DrdsTransactionManager extends DataSourceTransactionManager {
  public DrdsTransactionManager(DataSource dataSource) {
    super(dataSource);
  }
  @Override
  protected void prepareTransactionalConnection(Connection con, TransactionDefinition definition) throw
s SQLException {
    try (Statement stmt = con.createStatement()) {
      stmt.executeUpdate("SET drds_transaction_policy = '2PC'"); // A 2PC transaction is used as an example
.
    }
  }
}
```

Then, instantiate the preceding class in the Spring configuration. For example, you can write the following code:

```
<bean id="drdsTransactionManager" class="my.app.DrdsTransactionManager">
  <property name="dataSource" ref="yourDataSource" />
</bean>
```

To enable PolarDB-X distributed transactions for a class, you can add the `@Transactional("drdsTransactionManager")` annotation.

5.11. DDL operations

5.11.1. DDL statements

The data definition language (DDL) statement CREATE TABLE in a PolarDB-X instance is similar to that in a MySQL database, and is extended based on the syntax in a MySQL database. To create a table shard in a PolarDB-X instance, you must specify the table sharding manner and the database sharding manner in the `drds_partition_options` parameter. The valid values include `DBPARTITION BY`, `TBPARTITION BY`, `TBPARTITIONS`, and `BROADCAST`.

Currently, you can run a DDL statement in the following ways:

- Run the DDL statement through the MySQL command-line client, for example, by using MySQL command lines, Navicat, or MySQL Workbench.
- Connect to the specified PolarDB-X instance by using program code and then call the DDL statement for execution.

For the syntax of the CREATE TABLE statement in a MySQL database, see MySQL CREATE TABLE Statement.

5.11.2. CREATE TABLE statement

5.11.2.1. Overview

This topic describes the syntax, clauses, parameters, and basic methods for creating a table by using a data definition language (DDL) statement.

> **Note** PolarDB-X instances do not allow you to directly create a database by using a DDL statement. To create a database, you can Log on to the PolarDB-X console. For the information about how to create a database, see Create a database.

Syntax

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
  (create_definition,...)
  [table_options]
  [drds_partition_options]
  [partition_options]
drds_partition_options:
  DBPARTITION BY
    HASH([column])
    [TBPARTITION BY
      { HASH(column)
      | {MM|DD|WEEK|MMDD}(column)}
     [TBPARTITIONS num]
    ]
```

Clauses and parameters for database and table sharding

- `DBPARTITION BY hash(partition_key)` : This parameter specifies the shard key and the sharding algorithm for database sharding. Database sharding by time is not supported.
- `TBPARTITION BY { HASH(column) | {MM|DD|WEEK|MMDD}(column)` : (Optional) This parameter specifies the method of mapping data to a physical table. The value is the same as that of DBPARTITION BY by default.
- `TBPARTITIONS num` : (Optional) This parameter specifies the number of physical tables to be created in each database shard. The default value is 1. If no table sharding is required, you do not need to specify this parameter.

5.11.2.2. Create a single-database non-partitioned table

This topic describes how to create a single-database non-partitioned table.

Create a single-database non-partitioned table

```
CREATE TABLE single_tbl(
 id int,
 name varchar(30),
 primary key(id)
);
```

Execute the `SHOW TOPOLOGY FROM single_tbl;` statement to view the node topology of the logical table. Based on the output, a single-database non-partitioned logical table is created only on logical database 0.

```
+------+-----------------------------------------------------------------+------------+
| ID   | GROUP_NAME                                      | TABLE_NAME |
+------+-----------------------------------------------------------------+------------+
|    0 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | single_tbl |
+------+-----------------------------------------------------------------+------------+
1 row in set (0.01 sec)
```

Set parameters

You can also set the `select_statement` parameter when you create a single-database non-partitioned table. If you need to create a partitioned table, you cannot set this parameter.

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
  [(create_definition,...)]
  [table_options]
  [partition_options]
  select_statement
```

For example, you can execute the following statement to create a single-database non-partitioned table named single_tbl2 to store the data from the single_tbl table:

```
CREATE TABLE single_tbl2(
 id int,
 name varchar(30),
 primary key(id)
) select * from single_tbl;
```

5.11.2.3. Create a logical table partitioned into database shards

This topic describes how to create a logical table whose data is partitioned into database shards but not further into table shards.

Assume that eight database shards are created. You can execute the following statement to create a logical table whose data is hash-partitioned into the eight database shards based on the ID column.

```
CREATE TABLE multi_db_single_tbl(
 id int,
 name varchar(30),
 primary key(id)
) dbpartition by hash(id);
```

Execute the `show topology from multi_db_single_tbl;` statement to view the node topology of the logical table. Based on the output, one table shard is created on each database shard. This means that only database sharding is performed.

```
+------+-------------------------------------------------------------------+--------------------+
| ID   | GROUP_NAME                        | TABLE_NAME      |
+------+-------------------------------------------------------------------+--------------------+
|  0 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | multi_db_single_tbl |
|  1 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | multi_db_single_tbl |
|  2 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0002_RDS | multi_db_single_tbl |
|  3 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0003_RDS | multi_db_single_tbl |
|  4 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0004_RDS | multi_db_single_tbl |
|  5 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005_RDS | multi_db_single_tbl |
|  6 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0006_RDS | multi_db_single_tbl |
|  7 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | multi_db_single_tbl |
+------+-------------------------------------------------------------------+--------------------+
8 rows in set (0.01 sec)
```

5.11.2.4. Create table shards in database shards

This topic describes how to create table shards in database shards in different sharding manners.

- Use HASH for sharding
- Use RANGE_HASH for sharding
- Use date functions for sharding

In the following examples, it is assumed that eight database shards have been created.

Use HASH for sharding

Create a table that is partitioned into table shards in database shards, with each database shard containing three physical tables. The database sharding process calculates the hash by using id as the shard key, and the table sharding process calculates the hash by using bid as the shard key. Specifically, a hash operation is performed on the data of the table based on the id column, to distribute the data to multiple database shards. Then, a hash operation is performed on the data in each database shard based on the bid column, to distribute the data to the three physical tables.

```
CREATE TABLE multi_db_multi_tbl(
 id int auto_increment,
 bid int,
 name varchar(30),
 primary key(id)
) dbpartition by hash(id) tbpartition by hash(bid) tbpartitions 3;
```

According to the node topology of the logical table, you can see that three table shards are created in each database shard.

```
mysql> show topology from multi_db_multi_tbl;
+------+--------------------------------------------------------------------+----------------------+
| ID   | GROUP_NAME                                                         | TABLE_NAME           |
+------+--------------------------------------------------------------------+----------------------+
|   0 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | multi_db_multi_tbl
_00 |
|   1 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | multi_db_multi_tbl
_01 |
|   2 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | multi_db_multi_tbl
_02 |
|   3 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | multi_db_multi_tbl
_03 |
|   4 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | multi_db_multi_tbl
_04 |
|   5 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | multi_db_multi_tbl
_05 |
|   6 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0002_RDS | multi_db_multi_tbl
_06 |
|   7 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0002_RDS | multi_db_multi_tbl
_07 |
|   8 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0002_RDS | multi_db_multi_tbl
_08 |
|   9 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0003_RDS | multi_db_multi_tbl
_09 |
|  10 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0003_RDS | multi_db_multi_tb
l_10 |
|  11 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0003_RDS | multi_db_multi_tb
l_11 |
|  12 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0004_RDS | multi_db_multi_tb
l_12 |
|  13 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0004_RDS | multi_db_multi_tb
```

```
l_13 |
|  14 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0004_RDS | multi_db_multi_tb
l_14 |
|  15 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005_RDS | multi_db_multi_tb
l_15 |
|  16 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005_RDS | multi_db_multi_tb
l_16 |
|  17 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005_RDS | multi_db_multi_tb
l_17 |
|  18 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0006_RDS | multi_db_multi_tb
l_18 |
|  19 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0006_RDS | multi_db_multi_tb
l_19 |
|  20 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0006_RDS | multi_db_multi_tb
l_20 |
|  21 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | multi_db_multi_tb
l_21 |
|  22 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | multi_db_multi_tb
l_22 |
|  23 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | multi_db_multi_tb
l_23 |
+------+-----------------------------------------------------------------+----------------------+
24 rows in set (0.01 sec)
```

According to the sharding rule of the logical table, you can see that both database sharding and table sharding use the hash function, except that the database shard key is id and the table shard key is bid.

```
mysql> show rule from multi_db_multi_tbl;
+------+--------------------+-----------+-----------------+--------------------+--------------------+-----------------+--------------------
--+--------------------+
| ID   | TABLE_NAME         | BROADCAST | DB_PARTITION_KEY | DB_PARTITION_POLICY | DB_PARTITION_COUNT |
TB_PARTITION_KEY | TB_PARTITION_POLICY | TB_PARTITION_COUNT |
+------+--------------------+-----------+-----------------+--------------------+--------------------+-----------------+--------------------
--+--------------------+
|   0 | multi_db_multi_tbl |      0 | id          | hash          | 8           | bid         | hash          | 3           |
+------+--------------------+-----------+-----------------+--------------------+--------------------+-----------------+--------------------
--+--------------------+
1 row in set (0.01 sec)
```

Use RANGE_HASH for sharding

- Requirements

  The shard key must be a character or a number.

- Routing method

  Calculate a hash value based on the last N digits of any shard key, and then calculate the route by using RANGE_HASH. The number N is the third parameter in the function. For example, during calculation of the RANGE_HASH(COL1, COL2, N) function, COL1 is preferentially selected and then truncated to obtain the last N digits for calculation. If COL1 does not exist, COL2 is selected and truncated for calculation.

- Scenarios

  RANGE_HASH is applicable to scenarios where two shard keys are used for sharding but only the value of one shard is used for SQL query. Assume that a PolarDB-X database is partitioned into eight physical databases. Our customer has the following requirements:

  i. The order table of each service needs to be partitioned into database shards by buyer ID and order ID.
  ii. The query is executed based on either the buyer ID or order ID as the condition.

  In this case, you can run the following DDL statement to create the order table:

```
create table test_order_tb (
        id int,
        seller_id varchar(30) DEFAULT NULL,
        order_id varchar(30) DEFAULT NULL,
        buyer_id varchar(30) DEFAULT NULL,
        create_time datetime DEFAULT NULL,
        primary key(id)
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by RANGE_HASH(buyer_id, order_id, 10) tbpartiti
on by RANGE_HASH(buyer_id, order_id, 10) tbpartitions 3;
```

> **Note**
>
> - Neither of the two shard keys can be modified.
> - Data insertion fails if the two shard keys point to different database shards or table shards.

Use date functions for sharding

In addition to using the hash function as the sharding algorithm, you can also use the date functions MM, DD, WEEK, and MMDD as the table sharding algorithms. For more information, see the following examples:

- Create a table that is partitioned into table shards in database shards. The database sharding process calculates the hash by using userId as the shard key, and the table sharding process calculates DAY_OF_WEEK through the WEEK(actionDate) function and then partitions the table into table shards based on the actionDate column, with one week counted as seven days.

For example, if the value in the actionDate column is 2017-02-27, which is on Monday, the value obtained by calculating the WEEK(actionDate) function is 2. In this case, the record is stored in table shard 2, because 2 % 7 = 2. This table shard is located in a database shard and is named user_log_2. For another example, if the value in the actionDate column is 2017-02-26, which is on Sunday, the value obtained by calculating the WEEK(actionDate) function is 1. In this case, the record is stored in table shard 1, because 1 % 7 = 1. This table shard is located in a database shard and is named user_log_1.

```
CREATE TABLE user_log(
userId int,
name varchar(30),
operation varchar(30),
actionDate DATE
) dbpartition by hash(userId) tbpartition by WEEK(actionDate) tbpartitions 7;
```

According to the node topology of the logical table, you can see that seven table shards are created in each database shard, because one week is counted as seven days in the function. The responses are very long, and therefore are omitted by using an ellipsis (...).

```
mysql> show topology from user_log;
+------+-------------------------------------------------------------------+------------+
| ID   | GROUP_NAME                                                        | TABLE_NAME |
+------+-------------------------------------------------------------------+------------+
|   0 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log_0 |
|   1 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log_1 |
|   2 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log_2 |
|   3 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log_3 |
|   4 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log_4 |
|   5 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log_5 |
|   6 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log_6 |
|   7 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log_0 |
|   8 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log_1 |
|   9 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log_2 |
|  10 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log_3 |
|  11 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log_4 |
|  12 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log_5 |
|  13 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log_6 |
...
|  49 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log_0 |
|  50 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log_1 |
|  51 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log_2 |
|  52 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log_3 |
|  53 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log_4 |
|  54 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log_5 |
|  55 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log_6 |
+------+-------------------------------------------------------------------+------------+
56 rows in set (0.01 sec)
```

According to the sharding rule of the logical table, you can see that the database sharding process calculates the hash by using userId as the shard key, and the table sharding process calculates the WEEK function by using actionDate as the shard key.

```
mysql> show rule from user_log;
+------+------------+-----------+-----------------+--------------------+--------------------+-----------------+--------------------+-
------------------+
| ID  | TABLE_NAME | BROADCAST | DB_PARTITION_KEY | DB_PARTITION_POLICY | DB_PARTITION_COUNT |
TB_PARTITION_KEY | TB_PARTITION_POLICY | TB_PARTITION_COUNT |
+------+------------+-----------+-----------------+--------------------+--------------------+-----------------+--------------------+-
------------------+
|  0 | user_log |     0 | userId       | hash         | 8          | actionDate    | week         | 7          |
+------+------------+-----------+-----------------+--------------------+--------------------+-----------------+--------------------+-
------------------+
1 row in set (0.00 sec)
```

According to the specified database and table shard key parameters, you can see the specific physical table in the specific physical database to which the SQL statement is routed.

View the route of the SQL statement

```
mysql> explain select name from user_log where userId = 1 and actionDate = '2017-02-27'\G
*************************** 1. row ***************************
GROUP_NAME: SANGUAN_1490167540907XWDVSANGUAN_BSQT_0001_RDS
       SQL: select `user_log`.`name` from `user_log_2` `user_log` where ((`user_log`.`userId` = 1) AND (`user_log`.`actionDate` = '2017-02-27'))
    PARAMS: {}
1 row in set (0.01 sec)
```

- Create a table that is partitioned into table shards in database shards. The database sharding process calculates the hash by using userId as the shard key, and the table sharding process calculates MONTH_OF_YEAR through the MM(actionDate) function and then partitions the table into table shards based on the actionDate column, with one year counted as 12 months.

  For example, if the value in the actionDate column is 2017-02-27, the value obtained by calculating the MM(actionDate) function is 02. In this case, the record is stored in table shard 02, because 02 % 12 = 02. This table shard is located in a database shard and is named user_log_02. For another example, if the value in the actionDate column is 2016-12-27, the value obtained by calculating the MM(actionDate) function is 12. In this case, the record is stored in table shard 00, because 12 % 12 = 00. This table shard is located in a database shard and is named user_log_00.

```
CREATE TABLE user_log2(
userId int,
name varchar(30),
operation varchar(30),
actionDate DATE
) dbpartition by hash(userId) tbpartition by MM(actionDate) tbpartitions 12;
```

According to the node topology of the logical table, you can see that 12 table shards are created in each database shard, because one year is counted as 12 months in the function. The responses are very long, and therefore are omitted by using an ellipsis (...).

```
mysql> show topology from user_log2;
+------+-------------------------------------------------------------------+-------------+
| ID  | GROUP_NAME                                                        | TABLE_NAME  |
```

```
+------+-------------------------------------------------------------------+--------------+
|  0 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log2_00 |
|  1 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log2_01 |
|  2 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log2_02 |
|  3 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log2_03 |
|  4 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log2_04 |
|  5 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log2_05 |
|  6 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log2_06 |
|  7 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log2_07 |
|  8 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log2_08 |
|  9 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log2_09 |
| 10 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log2_10 |
| 11 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log2_11 |
| 12 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log2_00 |
| 13 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log2_01 |
| 14 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log2_02 |
| 15 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log2_03 |
| 16 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log2_04 |
| 17 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log2_05 |
| 18 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log2_06 |
| 19 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log2_07 |
| 20 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log2_08 |
| 21 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log2_09 |
| 22 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log2_10 |
| 23 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | user_log2_11 |
...
| 84 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log2_00 |
| 85 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log2_01 |
| 86 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log2_02 |
| 87 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log2_03 |
| 88 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log2_04 |
| 89 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log2_05 |
| 90 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log2_06 |
| 91 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log2_07 |
| 92 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log2_08 |
| 93 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log2_09 |
| 94 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log2_10 |
| 95 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log2_11 |
+------+-------------------------------------------------------------------+--------------+
96 rows in set (0.02 sec)
```

According to the sharding rule of the logical table, you can see that the database sharding process calculates the hash by using userId as the shard key, and the table sharding process calculates the MM function by using actionDate as the shard key.

```
mysql> show rule from user_log2;
+------+------------+-----------+------------------+---------------------+-------------------+------------------+---------------------+-
------------------+
| ID  | TABLE_NAME | BROADCAST | DB_PARTITION_KEY | DB_PARTITION_POLICY | DB_PARTITION_COUNT |
TB_PARTITION_KEY | TB_PARTITION_POLICY | TB_PARTITION_COUNT |
+------+------------+-----------+------------------+---------------------+-------------------+------------------+---------------------+-
------------------+
|  0 | user_log2 |      0 | userId       | hash          | 8           | actionDate     | mm           | 12          |
+------+------------+-----------+------------------+---------------------+-------------------+------------------+---------------------+-
------------------+
1 row in set (0.00 sec)
```

- Create a table that is partitioned into table shards in database shards. The database sharding process calculates the hash by using userId as the shard key, and the table sharding process calculates DAY_OF_MONTH through the DD(actionDate) function and then partitions the table into table shards, with one month counted as 31 days.

  For example, if the value in the actionDate column is 2017-02-27, the value obtained by calculating the DD(actionDate) function is 27. In this case, the record is stored in table shard 27, because 27 % 31 = 27. This table shard is located in a database shard and is named user_log_27.

```
CREATE TABLE user_log3(
userId int,
name varchar(30),
operation varchar(30),
actionDate DATE
) dbpartition by hash(userId) tbpartition by DD(actionDate) tbpartitions 31;
```

According to the node topology of the logical table, you can see that 31 table shards are created in each database shard, because one month is counted as 31 days in the function. The responses are very long, and therefore are omitted by using an ellipsis (...).

```
mysql> show topology from user_log3;
+------+-------------------------------------------------------------------+-------------+
| ID  | GROUP_NAME                                | TABLE_NAME  |
+------+-------------------------------------------------------------------+-------------+
|  0 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_00 |
|  1 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_01 |
|  2 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_02 |
|  3 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_03 |
|  4 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_04 |
```

```
|   5 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_05 |
|   6 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_06 |
|   7 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_07 |
|   8 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_08 |
|   9 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_09 |
|  10 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_10 |
|  11 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_11 |
|  12 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_12 |
|  13 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_13 |
|  14 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_14 |
|  15 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_15 |
|  16 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_16 |
|  17 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_17 |
|  18 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_18 |
|  19 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_19 |
|  20 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_20 |
|  21 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_21 |
|  22 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_22 |
|  23 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_23 |
|  24 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_24 |
|  25 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_25 |
|  26 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_26 |
|  27 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_27 |
|  28 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_28 |
|  29 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_29 |
|  30 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log3_30 |
...
| 237 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log3_20 |
| 238 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log3_21 |
| 239 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log3_22 |
| 240 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log3_23 |
| 241 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log3_24 |
| 242 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log3_25 |
| 243 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log3_26 |
| 244 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log3_27 |
| 245 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log3_28 |
| 246 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log3_29 |
| 247 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log3_30 |
+------+-------------------------------------------------------------------+--------------+
248 rows in set (0.01 sec)
```

According to the sharding rule of the logical table, you can see that the database sharding process calculates the hash by using userId as the shard key, and the table sharding process calculates the DD function by using actionDate as the shard key.

```
mysql> show rule from user_log3;
+------+------------+-----------+-----------------+--------------------+-------------------+-----------------+--------------------+-
------------------+
| ID   | TABLE_NAME | BROADCAST | DB_PARTITION_KEY | DB_PARTITION_POLICY | DB_PARTITION_COUNT |
TB_PARTITION_KEY | TB_PARTITION_POLICY | TB_PARTITION_COUNT |
+------+------------+-----------+-----------------+--------------------+-------------------+-----------------+--------------------+-
------------------+
|   0 | user_log3 |      0 | userId       | hash          | 8           | actionDate     | dd           | 31          |
+------+------------+-----------+-----------------+--------------------+-------------------+-----------------+--------------------+-
------------------+
1 row in set (0.01 sec)
```

- Create a table that is partitioned into table shards in database shards. The database sharding process calculates the hash by using userId as the shard key, and the table sharding process calculates DAY_OF_YEAR % 365 through the MMDD(actionDate) tbpartitions 365 function and then partitions the table into 365 physical tables, with one year counted as 365 days.

  For example, if the value in the actionDate column is 2017-02-27, the value obtained by calculating the MMDD(actionDate) function is 58. In this case, the record is stored in table shard 58. This table shard is located in a database shard and is named user_log_58.

```
CREATE TABLE user_log4(
userId int,
name varchar(30),
operation varchar(30),
actionDate DATE
) dbpartition by hash(userId) tbpartition by MMDD(actionDate) tbpartitions 365;
```

  According to the node topology of the logical table, you can see that 365 table shards are created in each database shard, because one year is counted as 365 days in the function. The responses are very long, and therefore are omitted by using an ellipsis (...).

```
mysql> show topology from user_log4;
+------+--------------------------------------------------------------------+--------------+
| ID   | GROUP_NAME                                                         | TABLE_NAME   |
+------+--------------------------------------------------------------------+--------------+
...
| 2896 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_341
|
| 2897 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_342
|
| 2898 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_343
```

```
|
| 2899 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_344
|
| 2900 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_345
|
| 2901 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_346
|
| 2902 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_347
|
| 2903 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_348
|
| 2904 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_349
|
| 2905 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_350
|
| 2906 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_351
|
| 2907 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_352
|
| 2908 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_353
|
| 2909 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_354
|
| 2910 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_355
|
| 2911 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_356
|
| 2912 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_357
|
| 2913 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_358
|
| 2914 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_359
|
| 2915 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_360
|
| 2916 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_361
|
| 2917 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_362
|
| 2918 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_363
|
```

```
|
| 2919 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log4_364
|
+------+----------------------------------------------------------------+---------------+
2920 rows in set (0.07 sec)
```

According to the sharding rule of the logical table, you can see that the database sharding process calculates the hash by using userId as the shard key, and the table sharding process calculates the MMDD function by using actionDate as the shard key.

```
mysql> show rule from user_log4;
+------+------------+-----------+-----------------+--------------------+-------------------+-----------------+--------------------+-
-------------------+
| ID   | TABLE_NAME | BROADCAST | DB_PARTITION_KEY | DB_PARTITION_POLICY | DB_PARTITION_COUNT |
TB_PARTITION_KEY | TB_PARTITION_POLICY | TB_PARTITION_COUNT |
+------+------------+-----------+-----------------+--------------------+-------------------+-----------------+--------------------+-
-------------------+
|   0 | user_log4 |      0 | userId       | hash          | 8           | actionDate     | mmdd          | 365         |
+------+------------+-----------+-----------------+--------------------+-------------------+-----------------+--------------------+-
-------------------+
1 row in set (0.02 sec)
```

- Create a table that is partitioned into table shards in database shards. The database sharding process calculates the hash by using userId as the shard key, and the table sharding process calculates DAY_OF_YEAR % 10 through the MMDD(actionDate) tbpartitions 10 function and then partitions the table into 10 physical tables, with one year counted as 365 days.

```
CREATE TABLE user_log5(
userId int,
name varchar(30),
operation varchar(30),
actionDate DATE
) dbpartition by hash(userId) tbpartition by MMDD(actionDate) tbpartitions 10;
```

According to the node topology of the logical table, you can see that 10 table shards are created in each database shard, because one year is counted as 365 days in the function and the table data is routed to 10 physical tables. The responses are very long, and therefore are omitted by using an ellipsis (...).

```
mysql> show topology from user_log5;
+------+-------------------------------------------------------------------+-------------+
| ID   | GROUP_NAME                                                        | TABLE_NAME  |
+------+-------------------------------------------------------------------+-------------+
|   0 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log5_00 |
|   1 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log5_01 |
|   2 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log5_02 |
|   3 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log5_03 |
|   4 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log5_04 |
|   5 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log5_05 |
|   6 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log5_06 |
|   7 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log5_07 |
|   8 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log5_08 |
|   9 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | user_log5_09 |
...
|  70 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log5_00 |
|  71 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log5_01 |
|  72 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log5_02 |
|  73 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log5_03 |
|  74 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log5_04 |
|  75 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log5_05 |
|  76 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log5_06 |
|  77 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log5_07 |
|  78 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log5_08 |
|  79 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | user_log5_09 |
+------+-------------------------------------------------------------------+-------------+
80 rows in set (0.02 sec)
```

According to the sharding rule of the logical table, you can see that the database sharding process calculates the hash by using userId as the shard key, and the table sharding process calculates the MMDD function by using actionDate as the shard key, and then routing the table data to 10 physical tables.

```
mysql> show rule from user_log5;
+------+------------+-----------+-----------------+--------------------+--------------------+-----------------+--------------------+-
------------------+
| ID   | TABLE_NAME | BROADCAST | DB_PARTITION_KEY | DB_PARTITION_POLICY | DB_PARTITION_COUNT |
TB_PARTITION_KEY | TB_PARTITION_POLICY | TB_PARTITION_COUNT |
+------+------------+-----------+-----------------+--------------------+--------------------+-----------------+--------------------+-
------------------+
|   0 | user_log5 |      0 | userId       | hash           | 8           | actionDate     | mmdd           | 10          |
+------+------------+-----------+-----------------+--------------------+--------------------+-----------------+--------------------+-
------------------+
1 row in set (0.01 sec)
```

5.11.2.5. Use the primary key as the shard key

When no shard key is specified in the sharding algorithm, the system uses the primary key as the shard key by default. The following examples illustrate how to use the primary key as the database shard key and the table shard key.

Use the primary key as the database shard key

```
CREATE TABLE prmkey_tbl(
id int,
name varchar(30),
primary key(id)
) dbpartition by hash();
```

Use the primary key as the database shard key and the table shard key

```
CREATE TABLE prmkey_multi_tbl(
id int,
name varchar(30),
primary key(id)
) dbpartition by hash() tbpartition by hash() tbpartitions 3;
```

5.11.2.6. Create a broadcast table

This topic describes the statement for creating a broadcast table and provides examples.

The BROADCAST clause is used to specify a broadcast table to be created. A broadcast table is a table that is replicated to each database shard, and a synchronization mechanism is used to ensure data consistency between the database shards. Data synchronization between the database shards has a latency of several seconds. This feature allows you to route a join from a PolarDB-X instance to an underlying ApsaraDB RDS for MySQL instance to avoid joins across multiple databases. For more information about how to optimize SQL statements by using broadcast tables, see Overview.

The following statement is an example of creating a broadcast table:

```
CREATE TABLE brd_tbl(
 id int,
 name varchar(30),
 primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 BROADCAST;
```

5.11.2.7. Other attributes of the MySQL CREATE TABLE statement

This topic describes other attributes of the MySQL CREATE TABLE statement.

When you create a logical table whose data is partitioned into table shards in database shards, you can also specify other attributes in the MySQL CREATE TABLE statement. For example, you can specify the following attributes:

```
CREATE TABLE multi_db_multi_tbl(
 id int,
 name varchar(30),
 primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by hash(id) tbpartition by hash(id) tbpartitions 3;
```

5.11.3. Modify a table

This topic describes the syntax of the DDL statement ALTER TABLE and provides examples.

The following syntax of the ALTER TABLE statement is used to modify a table:

```
ALTER [ONLINE|OFFLINE] [IGNORE] TABLE tbl_name
  [alter_specification [, alter_specification] ...]
  [partition_options]
```

On a PolarDB-X instance, you can use this DDL statement to perform routine DDL operations, such as adding a column, adding an index, and modifying a data definition. For more information about the syntax, see MySQL ALTER TABLE Statement.

> **Note** If you need to modify a partitioned table, you are not allowed to modify the shard key.

- Add a column:

```
ALTER TABLE user_log
 ADD COLUMN idcard varchar(30);
```

- Add an index:

```
ALTER TABLE user_log
 ADD INDEX idcard_idx (idcard);
```

- Delete an index:

```
ALTER TABLE user_log
 DROP INDEX idcard_idx;
```

- Modify a field:

```
ALTER TABLE user_log
 MODIFY COLUMN idcard varchar(40);
```

5.11.4. Delete a table

This topic describes the syntax of the DROP TABLE statement and provides examples.

The following syntax of the DROP TABLE statement is used to delete a table:

```
DROP [TEMPORARY] TABLE [IF EXISTS]
  tbl_name [, tbl_name] ...
  [RESTRICT | CASCADE]
```

The DROP TABLE statement executed on a PolarDB-X logical database is the same as that executed on a MySQL database in terms of syntax. After the statement is executed, the system automatically deletes the corresponding physical table shard. For more information, see MySQL DROP TABLE Statement.

For example, you can execute the following statement to delete the user_log table:

```
DROP TABLE user_log;
```

5.11.5. FAQ about DDL statements

This topic provides answers to some frequently asked questions about DDL statements.

What can I do if an error occurs when I create a table?

PolarDB-X processes DDL statements in a distributed manner. If an error occurs, the schemas of all table shards are inconsistent with one another. Therefore, you need to perform manual cleanup.

Perform the following operations:

1. Check the basic error descriptions provided by PolarDB-X, such as syntax errors. If the error message is too long, the system will prompt you to execute the SHOW WARNINGS statement to view the failure cause of each database shard.
2. Execute the SHOW TOPOLOGY statement to view the topology of physical table shards. For example, after you execute the `SHOW TOPOLOGY FROM multi_db_multi_tbl;` statement, the following output is displayed:

```
+------+-----------------+----------------------+
| ID   | GROUP_NAME      | TABLE_NAME           |
+------+-----------------+----------------------+
|  0 | corona_qatest_0 | multi_db_multi_tbl_00 |
|  1 | corona_qatest_0 | multi_db_multi_tbl_01 |
|  2 | corona_qatest_0 | multi_db_multi_tbl_02 |
|  3 | corona_qatest_1 | multi_db_multi_tbl_03 |
|  4 | corona_qatest_1 | multi_db_multi_tbl_04 |
|  5 | corona_qatest_1 | multi_db_multi_tbl_05 |
|  6 | corona_qatest_2 | multi_db_multi_tbl_06 |
|  7 | corona_qatest_2 | multi_db_multi_tbl_07 |
|  8 | corona_qatest_2 | multi_db_multi_tbl_08 |
|  9 | corona_qatest_3 | multi_db_multi_tbl_09 |
| 10 | corona_qatest_3 | multi_db_multi_tbl_10 |
| 11 | corona_qatest_3 | multi_db_multi_tbl_11 |
+------+-----------------+----------------------+
12 rows in set (0.21 sec)
```

3. Run the `check table multi_db_multi_tbl;` command to check whether the logical table is created. For example, the following output indicates that a physical table shard corresponding to the logical table multi_db_multi_tbl failed to be created.

```
+-----------------------------------------------+-------+----------+-----------------------------------------------------------------------------
--------+
| TABLE                                         | OP    | MSG_TYPE | MSG_TEXT                                                                    |
+-----------------------------------------------+-------+----------+-----------------------------------------------------------------------------
--------+
| andor_mysql_qatest. multi_db_multi_tbl | check | Error    | Table 'corona_qatest_0. multi_db_multi_tb
l_02' doesn't exist |
+-----------------------------------------------+-------+----------+-----------------------------------------------------------------------------
--------+
1 row in set (0.16 sec)
```

4. Continue to create or delete the logical table in idempotent mode. This way, the remaining physical table shards will be created or deleted.

```
CREATE TABLE IF NOT EXISTS table1
(id int, name varchar(30), primary key(id))
dbpartition by hash(id);
DROP TABLE IF EXISTS table1;
```

What can I do if I failed to create an index or add a column?

The method for handling the failure in creating an index or adding a column is similar to that for the failure in creating a table. For more information, see Handle DDL exceptions.

5.11.6. DDL functions for sharding

5.11.6.1. Overview

PolarDB-X is a database service that supports both database sharding and table sharding.

Support for PolarDB-X database sharding and table sharding

The following table lists the support for database sharding and table sharding by PolarDB-X DDL sharding functions.

Sharding function	Description	Support for database sharding	Support for table sharding
HASH	Performs a simple modulo operation.	Yes	Yes
UNI_HASH	Performs a simple modulo operation.	Yes	Yes
RIGHT_SHIFT	Performs a signed right shift on the value of the database shard key.	Yes	Yes
RANGE_HASH	Performs hashing when two sharding keys are required.	Yes	Yes
MM	Performs hashing by month.	No	Yes
DD	Performs hashing by date.	No	Yes
WEEK	Performs hashing by week.	No	Yes
MMDD	Performs hashing by month and date.	No	Yes
YYYYMM	Performs hashing by year and month.	Yes	Yes
YYYYWEEK	Performs hashing by year and week.	Yes	Yes

Sharding function	Description	Support for database sharding	Support for table sharding
YYYYDD	Performs hashing by year and date.	Yes	Yes
YYYYMM_OPT	Performs optimized hashing by year and month.	Yes	Yes
YYYYWEEK_OPT	Performs optimized hashing by year and week.	Yes	Yes
YYYYDD_OPT	Performs optimized hashing by year and date.	Yes	Yes

Note When you use database sharding and table sharding in PolarDB-X, take note of the following points:

- In a PolarDB-X instance, the sharding method of a logical table is jointly defined by a sharding function and a shard key. The sharding function specifies the number of shards to be created and the routing algorithm.
- In a PolarDB-X instance, database sharding and table sharding use the same sharding method only when they use the same sharding function and the same shard key. This allows the PolarDB-X instance to uniquely locate a physical table in a physical database based on the value of the shard key.
- Assume that the database sharding method and the table sharding method of a logical table are different. If the database shard key and table shard key are not specified in an SQL query, the PolarDB-X instance scans all database shards or all table shards when it processes the SQL query.

Support for data types in PolarDB-X DDL sharding functions

Different PolarDB-X DDL sharding functions support different data types. The following table lists the support for data types in PolarDB-X sharding functions. The check mark (√) indicates supported whereas the cross mark (×) indicates not supported.

Support for data types in PolarDB-X DDL sharding functions

Sharding function	BIGINT	INT	MEDIUMINT	SMALLINT	TINYINT	VARCHAR	CHAR	DATE	DATETIME	TIMESTAMP	Other types
HASH	√	√	√	√	√	√	√	×	×	×	×
UNI_HASH	√	√	√	√	√	√	√	×	×	×	×
RANGE_HASH	√	√	√	√	√	√	√	×	×	×	×
RIGHT_SHIFT	√	√	√	√	√	×	×	×	×	×	×
MM	×	×	×	×	×	×	×	√	√	√	×
DD	×	×	×	×	×	×	×	√	√	√	×
WEEK	×	×	×	×	×	×	×	√	√	√	×
MMDD	×	×	×	×	×	×	×	√	√	√	×
YYYYMM	×	×	×	×	×	×	×	√	√	√	×
YYYYWEEK	×	×	×	×	×	×	×	√	√	√	×
YYYYDD	×	×	×	×	×	×	×	√	√	√	×
YYYYMM_OPT	×	×	×	×	×	×	×	√	√	√	×
YYYYWEEK_OPT	×	×	×	×	×	×	×	√	√	√	×
YYYYDD_OPT	×	×	×	×	×	×	×	√	√	√	×

Syntax description for PolarDB-X DDL sharding functions

PolarDB-X is compatible with the DDL statements in MySQL, and provides the `drds_partition_options` keyword to support database sharding and table sharding, as shown in the following example:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
  (create_definition,...)
  [table_options]
  [drds_partition_options]
  [partition_options]
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
  [(create_definition,...)]
  [table_options]
  [drds_partition_options]
  [partition_options]
  select_statement
drds_partition_options:
  DBPARTITION BY
    {{HASH|YYYYMM|YYYYWEEK|YYYYDD|YYYYMM_OPT|YYYYWEEK_OPT|YYYYDD_OPT}([column])}
    [TBPARTITION BY
      {{HASH|MM|DD|WEEK|MMDD|YYYYMM|YYYYWEEK|YYYYDD|YYYYMM_OPT|YYYYWEEK_OPT|YYYYDD_OPT}(
column)}
      [TBPARTITIONS num]
    ]
```

5.11.6.2. HASH

This topic describes how to use the HASH function.

Usage notes

- The data type of shard keys must be integer or string.
- This sharding function has no limits on the version of your PolarDB-X instance. By default, it supports all PolarDB-X instance versions.

Routing method

If different shard keys are used to execute the HASH function for database sharding and table sharding, divide a value of the database shard key by the number of database shards and find the remainder. If the key value is a string, the string is first converted into a hash value and then used for route calculation. For example, HASH('8') is equivalent to 8 % D, where D indicates the number of database shards.

If the same shard key is used to execute the HASH function for database sharding and table sharding, divide a value of the shard key by the total number of table shards and find the remainder. Assume that two database shards are created, each database shard has four table shards, table shards 0 to 3 are stored in database shard 0, and table shards 4 to 7 are stored in database shard 1. Based on this routing method, shard key value 15 is distributed to table shard 7 in database shard 1. The equation is 15 % (2 × 4) = 7.

Scenarios

- The function can be used to partition data from a logical table into database shards by user ID or order ID.
- Values of shard keys are strings.

Examples

Assume that you need to partition data from a logical table into database shards by executing the HASH function based on the ID column. Execute the following DDL statement to create a table:

```
create table test_hash_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by HASH(ID);
```

Considerations

The HASH function is a simple modulus operation. The output of the HASH function can be evenly distributed only when the values in the shard key columns are evenly distributed.

5.11.6.3. UNI_HASH

This topic describes how to use the UNI_HASH function.

Usage notes

- The data type of shard keys must be integer or string.
- The version of your PolarDB-X instance must be 5.1.28-1508068 or later. For more information about the release notes of PolarDB-X, see View the instance version.

Routing method

If you execute the UNI_HASH function for database sharding, divide a value of the database shard key by the total number of database shards and find the remainder. If the key value is a string, the string is first converted into a hash value and then used for route calculation. For example, HASH('8') is equivalent to 8 % D, where D indicates the number of database shards.

If the same shard key is used to execute the UNI_HASH function for database sharding and table sharding, divide a value of the database shard key by the number of database shards first. (This step is different from that in the HASH function.) Then, evenly distribute data to the table shards in the database shards.

Scenarios

- The function can be used to partition data from a logical table into database shards by user ID or order ID.
- The data type of shard keys is integer or string.
- You can use this function when you need to partition data from two logical tables into database shards based on the same shard key. In each database shard, the number of table shards for one logical table is different from that for the other logical table. You need to frequently perform joins on the two logical tables based on the shard key.

Comparison with the HASH function

If you execute the UNI_HASH function to partition data from a logical table into database shards but not further into table shards, the routing method is the same as that used in the HASH function. The route is calculated by dividing the values of database shard keys by the number of database shards.

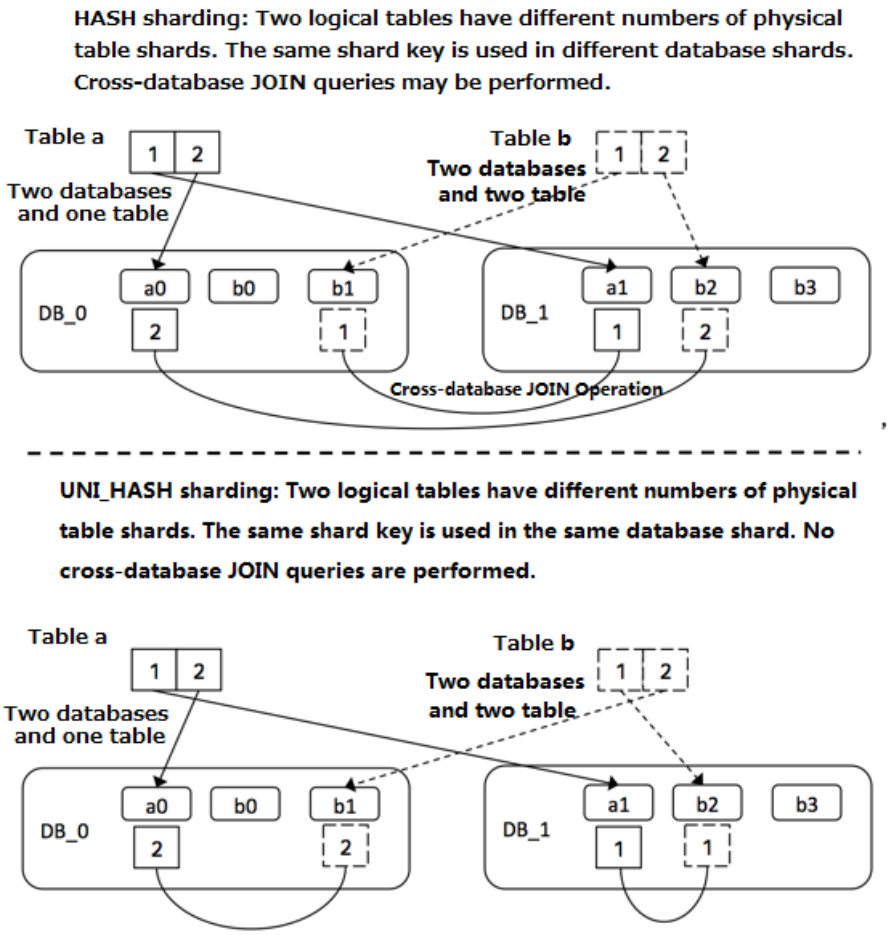
If you use the same shard key to execute the HASH function for database sharding and table sharding, the same key value may be routed to different database shards as the number of table shards changes.

If you use the same shard key to execute the UNI_HASH function for database sharding and table sharding, the same key value is always routed to the same database shard regardless of the number of table shards.

Assume that you execute the HASH or UNI_HASH function to partition data from two logical tables into table shards in database shards based on the same shard key and the total number of table shards for the two logical tables are different. If you perform a join on the two logical tables based on the shard key, the join is performed across database shards when the HASH function is used but in the same database shard when the UNI_HASH function is used.

Assume that you have two logical tables a and b. Each logical table has two database shards. Each database shard corresponding to logical table a stores one table shard, and each database shard corresponding to logical table b stores two table shards. The following figures separately show the results of performing a join on logical tables a and b after the HASH and UNI_HASH functions are used to partition data in the two logical tables.

Comparison between HASH and UNI_HASH



Examples

Assume that you need to partition data from a logical table into four table shards in each database shard by executing the UNI_HASH function based on the ID column. Execute the following CREATE TABLE statement:

```
create table test_hash_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by UNI_HASH(ID)
tbpartition by UNI_HASH(ID) tbpartitions 4;
```

Considerations

The UNI_HASH function is a simple modulus operation. The output of the HASH function can be evenly distributed only when the values in the shard key columns are evenly distributed.

5.11.6.4. RIGHT_SHIFT

This topic describes how to use the RIGHT_SHIFT function.

Usage notes

- The data type of shard keys must be integer.
- The version of your PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Routing method

Shift a value of the database shard key to the right by a specified number of binary digits. Then, divide the obtained integer by the number of database shards or table shards and find the remainder. Make sure that the value of the database shard key is an integer. You can specify the number of shifted digits in the DDL statement.

Scenarios

The RIGHT_SHIFT function can produce more uniform hashing when the lower-digit parts of most shard key values are similar but the higher-digit parts vary greatly.

Assume that four shard key values are available: 12340000, 12350000, 12460000, and 12330000. The four lower digits of the four values are all 0000. If the values of the shard keys are directly hashed, hashing is less effective. However, if you execute the RIGHT_SHIFT(shardKey, 4) statement to shift the values of the shard keys to the right by four digits and obtain 1234, 1235, 1246, and 1233, these new values result in better uniformity of hashing.

Examples

Assume that you need to use the ID column as a shard key and shift the values of the ID column to the right by four binary digits for hashing purposes. Execute the following CREATE TABLE statement:

```
create table test_hash_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by RIGHT_SHIFT(id, 4)
tbpartition by RIGHT_SHIFT(id, 4) tbpartitions 2;
```

Considerations

The number of shifted digits cannot exceed the number of digits occupied by the integer.

5.11.6.5. RANGE_HASH

This topic describes how to use the RANGE_HASH function.

Usage notes

- The data type of shard keys must be character or number.
- The version of your PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Routing method

Calculate the hash value based on the last N digits of the value of a shard key. Then, divide the hash value by the number of database shards and find the remainder. The number N is the third parameter in the function.

For example, during the calculation of the RANGE_HASH(COL1, COL2, N) function, COL1 is preferentially selected and then truncated to obtain the last N digits for calculation. If COL1 does not exist, COL2 is selected and truncated for calculation.

Scenarios

The RANGE_HASH function can be used in scenarios where a table needs to be partitioned by two shard keys but queries are performed based on the values of only one shard key.

Examples

Assume that data in a PolarDB-X logical database is partitioned into eight database shards and that you have the following requirements:

Partition data into database shards by buyer ID and order ID. However, values of only one ID are available for queries.

You can execute the following DDL statement to create an order table:

```
create table test_order_tb (
  id int,
  buyer_id varchar(30) DEFAULT NULL,
  order_id varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8
  dbpartition by RANGE_HASH(buyer_id,order_id, 10)
  tbpartition by RANGE_HASH (buyer_id,order_id, 10) tbpartitions 3;
```

Considerations

- The two shard keys cannot be modified.
- Data insertion fails if data is routed to different database shards or table shards based on the two shard keys.

5.11.6.6. MM

This topic describes how to use the MM function.

Usage notes

- The data type of shard keys must be DATE, DATETIME, or TIMESTAMP.

- The MM function can be used only for table sharding, but cannot be used for database sharding.
- The version of your PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Routing method

Divide the month in a time value of the database shard key by the total number of table shards, and find the remainder. The remainder is the table shard subscript.

Scenarios

The MM function can be used to partition data into table shards by month. The table shard name indicates a specific month.

Examples

Assume that you need to partition data first into database shards based on the ID column and then into table shards based on the create_time column by month. Meanwhile, you need to map each month to a physical table shard. Execute the following DDL statement to create a table:

```
create table test_mm_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by HASH(id)
tbpartition by MM(create_time) tbpartitions 12;
```

Considerations

When you partition data into table shards by executing the MM function, make sure that each database shard has no more than 12 table shards because a year has 12 months.

5.11.6.7. DD

This topic describes how to use the DD function.

Usage notes

- The data type of shard keys must be DATE, DATETIME, or TIMESTAMP.
- The DD function can be used only for table sharding, but cannot be used for database sharding.
- The version of your PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Routing method

Divide the day of the month in a time value of the database shard key by the total number of table shards, and find the remainder. The remainder is the table shard subscript.

Scenarios

The DD function can be used to partition data into table shards based on date, which is the day of a month. The subscript of the table shard name indicates the day of a month. A month has 31 days at most.

Examples

Assume that you need to partition data first into database shards based on the ID column and then into table shards based on the create_time column by day of a month. Meanwhile, you need to map each day of the month to a physical table shard. Execute the following DDL statement to create a table:

```
create table test_dd_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by HASH(id)
tbpartition by DD(create_time) tbpartitions 31;
```

Considerations

When you partition data into table shards by executing the DD function, make sure that each database shard has no more than 31 table shards because a month has 31 days at most.

5.11.6.8. WEEK

This document describes how to use the WEEK function.

Usage notes

- The data type of shard keys must be DATE, DATETIME, or TIMESTAMP.
- The WEEK function can be used only for table sharding, but cannot be used for database sharding.
- The version of your PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Routing method

Divide the day of the week in a time value of the database shard key by the total number of table shards, and find the remainder. The remainder is the table shard subscript.

Scenarios

The WEEK function can be used to partition data into table shards by day of a week. The subscript of the table shard name corresponds to each day of a week, from Monday to Sunday.

Examples

Assume that you need to partition data first into database shards based on the ID column and then into table shards based on the create_time column by day of a week. Meanwhile, you need to map each day of a week (from Monday to Sunday) to a physical table shard. Execute the following DDL statement to create a table:

```
create table test_week_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by HASH(name)
tbpartition by WEEK(create_time) tbpartitions 7;
```

Considerations

When you partition data into table shards by executing the WEEK function, make sure that each database shard has no more than seven table shards because a week has seven days.

5.11.6.9. MMDD

This topic describes how to use the MMDD function.

Usage notes

- The data type of shard keys must be DATE, DATETIME, or TIMESTAMP.
- The MMDD function can be used only for table sharding, but cannot be used for database sharding.
- The version of your PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Routing method

Divide the day of the year in a time value of the database shard key by the total number of table shards, and find the remainder. The remainder is the table shard subscript.

Scenarios

The MMDD function can be used to partition data into table shards by day of a year. The subscript of the table shard name indicates the day of the year. A year has 366 days at most.

Examples

Assume that you need to partition data first into database shards based on the ID column and then into table shards based on the create_time column by day of a year. Meanwhile, you need to map each day of a year to a physical table shard. Execute the following data definition language (DDL) statement to create a table:

```
create table test_mmdd_tb (
  id int,
  name varchar(30) DEFAULT NULL,
  create_time datetime DEFAULT NULL,
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by HASH(name)
tbpartition by MMDD(create_time) tbpartitions 365;
```

Considerations

When you partition data into table shards by executing the MMDD function, make sure that each database shard has no more than 366 table shards because a year has 366 days at most.

5.11.6.10. YYYYMM

This topic describes how to use the YYYYMM function.

Usage notes

- The data type of shard keys must be DATE, DATETIME, or TIMESTAMP.
- The version of your PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Routing method

Calculate the hash value based on the year and month in a time value of the database shard key. Then, divide the hash value by the number of database shards and find the remainder.

For example, the YYYYMM('2012-12-31 12:12:12') function is equivalent to (2012 × 12 + 12) % D, where D indicates the number of database shards.

Scenarios

The YYYYMM function can be used to partition data into database shards by year and month. We recommend that you use the YYYYMM function with the tbpartition YYYYMM(ShardKey) function.

Assume that data in a PolarDB-X logical database is partitioned into eight physical database shards and that you have the following requirements:

- Partition data into the database shards by year and month.
- Distribute data from the same month to the same table shard and ensure that each month within two years corresponds to a separate table shard.
- Directly query data from a physical table shard in a physical database shard when database shard keys and table shard keys are specified in the query.

Then, you can use the YYYYMM function. In the case that each month within two years corresponds to a physical table shard, a total of 24 physical table shards must be created, because a year has 12 months. Data in the PolarDB-X logical database is partitioned into eight database shards. Therefore, three physical table shards must be created in each database shard. Execute the following DDL statement to create a table:

```
create table test_yyyymm_tb (
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8
  dbpartition by YYYYMM(create_time)
  tbpartition by YYYYMM(create_time) tbpartitions 3;
```

Considerations

- You cannot use the YYYYMM function to distribute data from every month in every year to a separate table shard. Instead, the number of table shards must be fixed.
- After a cycle over months, data from the same month may be routed to the same table shard in the same database shard based on the number of table shards. For example, a cycle is from 2012-03 to 2013-03.

5.11.6.11. YYYYWEEK

This topic describes how to use the YYYYWEEK function.

Usage notes

- The data type of shard keys must be DATE, DATETIME, or TIMESTAMP.
- The version of your PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Routing method

Calculate the hash value based on the year and week of the year in a time value of the database shard key. Then, divide the hash value by the number of database shards and find the remainder.

For example, the YYYYWEEK('2012-12-31 12:12:12') function is equivalent to (2013 × 52 + 1) % D, where D indicates the number of database shards. Therefore, the date 2012-12-31 falls on the first week of 2013.

Scenarios

The YYYYWEEK function can be used to partition data into database shards by year and week of the year. We recommend that you use the YYYYWEEK function with the tbpartition YYYYWEEK(ShardKey) function.

Assume that data in a PolarDB-X logical database is partitioned into eight physical database shards and that you have the following requirements:

- Partition data into the database shards by year and week of the year.
- Distribute data from the same week to the same table shard and ensure that each week within two years corresponds to a separate table shard.
- Directly query data from a physical table shard in a physical database shard when database shard keys and table shard keys are specified in the query.

Then, you can use the YYYYWEEK function. In the case that each week within two years corresponds to a physical table shard, at least 106 physical table shards must be created, because a year has up to 53 weeks. Data in the PolarDB-X logical database is partitioned into eight database shards, and therefore 14 physical table shards must be created in each database shard. The total number of physical database shards is calculated in the following formula: 14 × 8 = 112, where 112 is greater than 106. We recommend that the number of physical table shards be an integer multiple of the number of database shards. Execute the following DDL statement to create a table:

```
create table test_yyyymm_tb (
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8
  dbpartition by YYYYWEEK(create_time)
  tbpartition by YYYYWEEK(create_time) tbpartitions 14;
```

Considerations

- You cannot use the YYYYWEEK function to distribute data from every week in every year to a separate table shard. Instead, the number of table shards must be fixed.
- After a cycle over weeks, data from the same week may be routed to the same table shard in the same database shard based on the number of table shards. For example, a cycle is from the first week in 2012 to the first week in 2013.

5.11.6.12. YYYYDD

This topic describes how to use the YYYYDD function.

Usage notes

- The data type of shard keys must be DATE, DATETIME, or TIMESTAMP.
- The version of your PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Routing method

Calculate the hash value based on the year and day of the year in a time value of the database shard key. Then, divide the hash value by the number of database shards and find the remainder.

For example, the YYYYDD('2012-12-31 12:12:12') function is equivalent to (2012 × 366 + 365) % D, where D indicates the number of database shards. Therefore, the value 2012-12-31 is the 365th day of 2012.

Scenarios

The YYYYDD function can be used to partition data into database shards by year and day of the year. We recommend that you use the YYYYDD function with the tbpartition YYYYDD(ShardKey) function.

Assume that data in a PolarDB-X logical database is partitioned into eight physical database shards and that you have the following requirements:

- Partition data into the database shards by year and day of the year.
- Distribute data from the same week to the same table shard and ensure that each day within two years corresponds to a separate table shard.
- Directly query data from a physical table shard in a physical database shard when database shard keys and table shard keys are specified in the query.

Then, you can use the YYYYDD function. In the case that each day within two years corresponds to a physical table shard, a total of 732 physical table shards must be created, because a year has up to 366 days. Data in the PolarDB-X logical database is partitioned into eight database shards, and therefore 92 physical table shards must be created in each database shard. This number is calculated in the following formula: 732/8 = 91.5, which is rounded to 92. We recommend that the number of table shards be an integer multiple of the number of database shards. Execute the following DDL statement to create a table:

```
create table test_yyyydd_tb (
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8
  dbpartition by YYYYDD(create_time)
  tbpartition by YYYYDD(create_time) tbpartitions 92;
```

Considerations

- You cannot use the YYYYDD function to distribute data from every day in every year to a separate table shard. Instead, the number of table shards must be fixed.
- After a cycle over dates, data from the same date may be routed to the same table shard in the same database shard based on the number of table shards. For example, a cycle is from 2012-03-01 to 2013-03-01.

5.11.6.13. YYYYMM_OPT

This topic describes how to use the YYYYMM_OPT function.

Usage notes

- The data type of shard keys must be DATE, DATETIME, or TIMESTAMP.
- The year and month of user data does not randomly increase but naturally increase over time.
- The version of your PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Optimizations

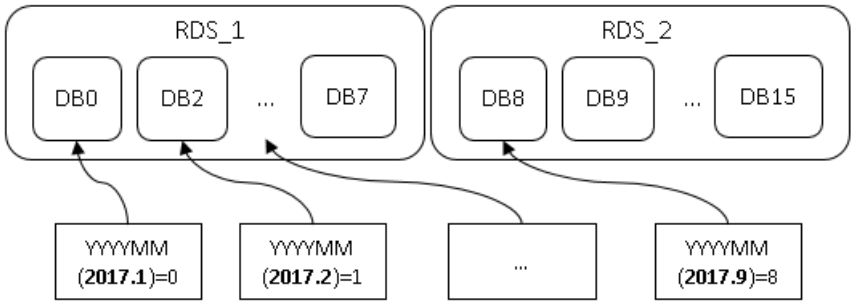
Compared with the YYYYMM function, the YYYYMM_OPT function can ensure that data is evenly distributed across ApsaraDB RDS for MySQL instances as the timeline increases.

Assume that two ApsaraDB RDS for MySQL instances are attached to your PolarDB-X instance and that data in your logical table is partitioned into 16 database shards. DB0 to DB7 shards are located on one ApsaraDB RDS for MySQL instance, and DB8 to DB15 shards are located on the other ApsaraDB RDS for MySQL instance.

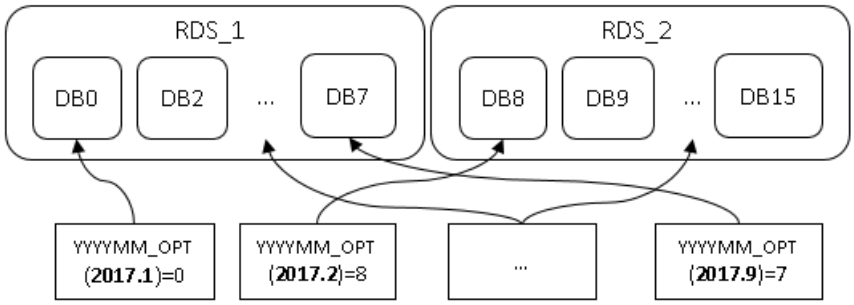
The following figures compare the mappings when the YYYYMM and YYYYMM_OPT functions are used for database sharding.

Comparison between the YYYYMM and YYYYMM_OPT functions

As the time goes on linearly, YYYYMM fills data in ApsaraDB for RDS instances in sequence (data is first distributed to the database shards of RDS_1, then to the database shards of RDS_2, and then to the database shards of RDS_1 again).



YYYYMM_OPT evenly distributes data between ApsaraDB for RDS instances as the time goes on (data is alternately distributed between RDS_1 and RDS_2, so that the data size of the two RDS instances is balanced).



- The YYYYMM_OPT function evenly distributes data across ApsaraDB RDS for MySQL instances. This maximizes the performance of each ApsaraDB RDS for MySQL instance.
- Choose between the YYYYMM and YYYYMM_OPT functions:
  - If the time of business data sequentially increases and the data volume does not differ much between time points, you can use the YYYYMM_OPT function to evenly distribute data across ApsaraDB RDS for MySQL instances.
  - You can use the YYYYMM function if the time of business data randomly increases.

Routing method

- Calculate the hash value based on the year and month of the year in a time value of the database shard key. Then, divide the hash value by the number of database shards and find the remainder.

- The hash calculation based on the database and table shard key considers the evenness of data distribution among the ApsaraDB RDS for MySQL instances attached to your PolarDB-X instance.

Scenarios

- Data needs to be partitioned into table shards in database shards by year and month of the year.
- Data must be evenly distributed across the ApsaraDB RDS for MySQL instances attached to your PolarDB-X instance.
- The time of the shard key sequentially increases, and data is evenly distributed across all ApsaraDB RDS for MySQL instances from month to month. For example, the number of monthly journal logs increases every month, but the logs are evenly distributed across all ApsaraDB RDS for MySQL instances.

Considerations

- The YYYYMM_OPT function does not allow you to distribute data from every month in every year to a separate table shard. Instead, the number of table shards must be fixed.
- After a cycle over months, data from the same month may be routed to the same table shard in the same database shard based on the number of table shards. For example, a cycle is from 2012-03 to 2013-03.

5.11.6.14. YYYYWEEK_OPT

This topic describes how to use the YYYYWEEK_OPT function.

Usage notes

- The data type of shard keys must be DATE, DATETIME, or TIMESTAMP.
- The version of your PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Optimizations

- Compared with the YYYYWEEK function, the YYYYWEEK_OPT function can ensure that data is evenly distributed across ApsaraDB RDS for MySQL instances as the timeline increases. The YYYYWEEK_OPT function has the similar effect to the YYYYMM_OPT function.
- The YYYYWEEK_OPT function evenly distributes data across ApsaraDB RDS for MySQL instances. This maximizes the performance of each ApsaraDB RDS for MySQL instance.
- Choose between the YYYYWEEK and YYYYWEEK_OPT functions:
  - If the time of business data sequentially increases and the data volume does not differ much between time points, you can use the YYYYWEEK_OPT function to evenly distribute data across ApsaraDB RDS for MySQL instances.
  - You can use the YYYYWEEK function if the time of business data randomly increases.

Routing method

- Calculate the hash value based on the year and week of the year in a time value of the database shard key. Then, divide the hash value by the number of database shards and find the remainder.
- The hash calculation based on the database and table shard key considers the evenness of data distribution among the ApsaraDB RDS for MySQL instances attached to your PolarDB-X instance.

Scenarios

- Data needs to be partitioned into table shards in database shards by year and week of the year.
- Data must be evenly distributed across the ApsaraDB RDS for MySQL instances attached to your PolarDB-X instance.
- The time of the shard key sequentially increases, and data is evenly distributed across all ApsaraDB RDS for MySQL instances from week to week. For example, the number of weekly journal logs increases every week, but the logs are evenly distributed across all ApsaraDB RDS for MySQL instances.

Considerations

- The YYYYWEEK_OPT function does not allow you to distribute data from every week in every year to a separate table shard. Instead, the number of table shards must be fixed.
- After a cycle over weeks, data from the same week may be routed to the same table shard in the same database shard based on the number of table shards. For example, a cycle is from the first week in 2012 to the first week in 2013.

5.11.6.15. YYYYDD_OPT

This topic describes how to use the YYYYDD_OPT function.

Usage notes

- The data type of shard keys must be DATE, DATETIME, or TIMESTAMP.
- The version of your PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Optimizations

- Compared with YYYYDD function, the YYYYDD_OPT function can ensure that data is evenly distributed across ApsaraDB RDS for MySQL instances as the timeline increases. The YYYYDD_OPT function has the similar effect to the YYYYMM_OPT function.
- The YYYYDD_OPT function evenly distributes data across ApsaraDB RDS for MySQL instances. This maximizes the performance of each ApsaraDB RDS for MySQL instance.
- Choose between the YYYYDD and YYYYDD_OPT functions:
  - If the time of business data sequentially increases and the data volume does not differ much between time points, you can use the YYYYDD_OPT function to evenly distribute data across ApsaraDB RDS for MySQL instances.
  - You can use the YYYYDD function if the time of business data randomly increases.

Routing method

- Calculate the hash value based on the year and day of the year in a time value of the database shard key. Then, divide the hash value by the number of database shards and find the remainder.
- The hash calculation based on the database and table shard key considers the evenness of data distribution among the ApsaraDB RDS for MySQL instances attached to your PolarDB-X instance.

Scenarios

- Data needs to be partitioned into table shards in database shards by year and day of the year.

- Data must be evenly distributed across the ApsaraDB RDS for MySQL instances attached to your PolarDB-X instance.
- The time of the shard key sequentially increases, and data is evenly distributed across all ApsaraDB RDS for MySQL instances from day to day. For example, the number of daily journal logs increases every day, but the logs are evenly distributed across all ApsaraDB RDS for MySQL instances.

Considerations

- The YYYYDD_OPT function does not allow you to distribute data from every month in every year to a separate table shard. Instead, the number of table shards must be fixed.
- After a cycle over dates, data from the same date may be routed to the same table shard in the same database shard based on the number of table shards. For example, a cycle is from 2012-03-01 to 2013-03-01.

5.12. Automatic protection of high-risk SQL statements

In PolarDB-X, the data manipulation language (DML) statements are the same as MySQL statements.

We recommend that you include the shard key in the **SELECT** and **UPDATE** statements of PolarDB-X. The **INSERT** statement of PolarDB-X must include the shard key and a non-null key value.

By default, PolarDB-X disables full-table deletion and updating to prevent misoperation.

The following statements are prohibited by default:

- A DELETE statement without the WHERE or LIMIT condition
- An UPDATE statement without the WHERE or LIMIT condition

If you need to perform full-table deletion or update, you can temporarily skip this limit by using the following hint:

```
HINT: /! TDDL:FORBID_EXECUTE_DML_ALL=false*/
```

Examples

- Full-table deletion is intercepted by default.

  ```
  mysql> delete from tt;
  ERR-CODE: [TDDL-4620][ERR_FORBID_EXECUTE_DML_ALL] Forbid execute DELETE ALL or UPDATE ALL sql.
  More: [http://middleware.alibaba-inc.com/faq/faqByFaqCode.html?faqCode=TDDL-4620]
  ```

  The operation is successful if the following HINT is added:

  ```
  mysql> /! TDDL:FORBID_EXECUTE_DML_ALL=false*/delete from tt;
  Query OK, 10 row affected (0.21 sec)
  ```

- Full-table update is intercepted by default.

```
mysql> update tt set id = 1;
ERR-CODE: [TDDL-4620][ERR_FORBID_EXECUTE_DML_ALL] Forbid execute DELETE ALL or UPDATE ALL sql.
More: [http://middleware.alibaba-inc.com/faq/faqByFaqCode.html?faqCode=TDDL-4620]
```

The operation is successful if the following HINT is added:

```
mysql> /! TDDL:FORBID_EXECUTE_DML_ALL=false*/update tt set id = 1;
Query OK, 10 row affected (0.21 sec)
```

- This limit does not apply to DELETE or UPDATE statements that contain the WHERE or LIMIT condition.

```
mysql> delete from tt where id = 1;
Query OK, 1 row affected (0.21 sec)
```

5.13. PolarDB-X sequence

5.13.1. Overview

A PolarDB-X sequence (a 64-digit number of the signed BIGINT type in MySQL) is used to create a globally unique and sequentially incremental numeric sequence, such as values of primary key columns and unique key columns.

PolarDB-X sequences are used in the following two ways:

- Explicit sequences are created and maintained by using sequence-specific data definition language (DDL) syntax and can be used independently. The sequence value can be retrieved by using `select seq.nextval;`, where `seq` indicates the sequence name.
- Implicit sequences are used to automatically fill in primary keys with AUTO_INCREMENT defined and are automatically maintained by PolarDB-X.

> **Notice** PolarDB-X creates implicit sequences only after AUTO_INCREMENT is defined for partitioned tables and broadcast tables. This is not the case for non-partition tables. The AUTO_INCREMENT value of a non-partition table is created by ApsaraDB RDS for MySQL.

Types and features of PolarDB-X sequences

Currently, three types of PolarDB-X sequences are supported.

Type (abbreviation)	Globally unique	Consecutive	Monotonically increasing	Monotonically increasing within the same connection	Non-single point	Date types	Readability
Group sequence (GROUP)	Yes	No	No	Yes	Yes	All integer types	High
Time-based sequence (TIME)	Yes	No	Monotonically increasing at the macro level and non-monotonically increasing at the micro level	Yes	Yes	Only BIGINT	Low
Simple sequence (SIMPLE)	Yes	Yes	Yes	Yes	No	All integer types	High

Concepts:

- **Consecutive:** If the current value is n, the next value must be n + 1. If the next value is not n + 1, it is nonconsecutive.
- **Monotonically increasing:** If the current value is n, the next value must be a number greater than n.
- **Single point:** The risk of a single point of failure (SPOF) exists.
- **Monotonically increasing at the macro level and non-monotonically increasing at the micro level:** An example of this is `1, 3, 2, 4, 5, 7, 6, 8, ...` Such a sequence is monotonically increasing at the macro level and non-monotonically increasing at the micro level.

Group sequence (GROUP, used by default)

Features

A group sequence is a globally unique sequence with natural numeric values, which are not necessarily consecutive or monotonically increasing. If the sequence type is not specified, PolarDB-X uses the group sequence type by default.

- **Advantages:** A group sequence is globally unique and provides excellent performance, preventing single points of failure (SPOFs).
- **Disadvantages:** A group sequence may contain nonconsecutive values, which may not necessarily start from the initial value and do not cycle.

Implementation

The values of a group sequence are created by multiple nodes to ensure high availability. The values in a segment are nonconsecutive if the values are not all used, such as in the case of disconnection.

Time-based sequence (TIME)

Features

A time-based sequence consists of **a timestamp, node ID, and serial number**. It is globally unique and automatically increments at the macro level. Value updates are database-independent and not persistently stored in databases. Only names and types are stored in databases. This delivers good performance to time-based sequences, which create values like 776668092129345536, 776668098018148352, 776668111578333184, and 776668114812141568.

> **Notice** **Sequence values must be of the BIGINT type** when used in the auto-increment columns of tables.

- **Advantages:** Time-based sequences are globally unique with good performance.
- **Disadvantages:** The values of a time-based sequence are nonconsecutive. The START WITH, INCREMENT BY, MAXVALUE, and CYCLE or NOCYCLE parameters are invalid for time-based sequences.

Simple sequence (SIMPLE)

Features

Only simple sequences support the START WITH, INCREMENT BY, MAXVALUE, and CYCLE or NOCYCLE parameters.

- **Advantages:** Simple sequences are globally unique and monotonically increasing with consecutive values.
- **Disadvantages:** Simple sequences are prone to SPOFs, poor performance, and bottlenecks. Use them with caution.

Implementation

Each sequence value must be persistently stored.

Scenarios

Group sequences, time-based sequences, and simple sequences are globally unique and can be used in **primary key columns** and **unique index columns**.

- We recommend that you use **group sequences**.
- Use only simple sequences for services that strongly depend on consecutive sequence values. Pay attention to sequence performance.
- We recommend that you use time-based sequences if you have high requirements for sequence performance, the amount of data inserted to tables is small, and large sequence values are acceptable. It is CPU-bound with no requirements on computing lock, database dependence, or persistent storage.

The following example shows how to create a sequence with a start value value of 100000 and a step of 1.

- A **simple sequence** creates globally unique, consecutive, and monotonically increasing values, such as 100000, 100001, 100002, 100003, 100004, ..., 200000, 200001, 200002, 200003... Simple sequences

are persistently stored. Even after services are restarted upon an SPOF, values are still created consecutively from the breakpoint. However, simple sequences have poor performance because each value is persistently stored once it is created.

- A **group sequence** may create values like 200001, 200002, 200003, 200004, 100001, 100002, 100003...

  > Notice
  >
  > - The start value of a group sequence is not necessarily the same as the START WITH value (which is 100000 in this example) but is invariably greater than this value. In this example, the start value is 200001.
  > - A group sequence is globally unique but may contain nonconsecutive values, for example, when a node is faulty or the connection that only uses partial values is closed. The group sequence in this example contains nonconsecutive values because the values between 200004 and 100001 are missing.

- A **time-based sequence** may create values like 776668092129345536, 776668098018148352, 776668111578333184, 776668114812141568...

5.13.2. Explicit sequence usage

This topic describes how to use data definition language (DDL) statements to create, modify, delete, and query sequences and how to retrieve the values of explicit sequences.

Create a sequence

Syntax:

```
CREATE [ GROUP | SIMPLE | TIME ] SEQUENCE <name>
[ START WITH <numeric value> ] [ INCREMENT BY <numeric value> ]
[ MAXVALUE <numeric value> ] [ CYCLE | NOCYCLE ]
```

Parameter description:

Parameter	Description	Applicable To
START WTIH	The initial sequence value. If it is not set, the default value is 1.	Simple sequence and group sequence
INCREMENT BY	The increment (or interval value or step) of each sequence increase. If it is not set, the default value is 1.	Simple Sequence
MAXVALUE	The maximum sequence value. If it is not specified, the default value is the maximum value of the signed BIGINT type.	Simple Sequence

Parameter	Description	Applicable To
CYCLE or NOCYCLE	Indicates whether to repeat the sequence value which starts from the value specified by START WITH after the sequence value reaches the maximum value. If it is not specified, the default value is NOCYCLE.	Simple Sequence

> **Note**
>
> - If the sequence type is not specified, the group sequence type is used by default.
> - INCREMENT BY, MAXVALUE, and CYCLE or NOCYCLE do not take effect for group sequences.
> - START WITH, INCREMENT BY, MAXVALUE, and CYCLE or NOCYCLE do not take effect for time-based sequences.
> - Group sequences are nonconsecutive. The START WITH parameter only provides reference for group sequences. **The start value of a group sequence is not necessarily the same as the START WITH value but is invariably greater than this value.**

Example 1: Create a group sequence.

- Method 1:

```
mysql> CREATE SEQUENCE seq1;
Query OK, 1 row affected (0.27 sec)
```

- Method 2:

```
mysql> CREATE GROUP SEQUENCE seq1;
Query OK, 1 row affected (0.27 sec)
```

Example 2: Create a time-based sequence.

```
mysql> CREATE TIME SEQUENCE seq1;
Query OK, 1 row affected (0.27 sec)
```

Example 3: Create a simple sequence with a start value of 1,000, a step size of 2, and a maximum value of 99999999999, which does not repeat after increasing to the maximum value.

```
mysql> CREATE SIMPLE SEQUENCE seq2 START WITH 1000 INCREMENT BY 2 MAXVALUE 99999999999 NOCYCL
E;
Query OK, 1 row affected (0.03 sec)
```

Modify a sequence

PolarDB-X allows you to modify sequences in the following ways:

- For simple sequences, change the values of START WITH, INCREMENT BY, MAXVALUE, and CYCLE or NOCYCLE.

- For group sequences, change the value of START WITH.
- Convert the sequence type to another.

Syntax:

```
ALTER SEQUENCE <name> [ CHANGE TO GROUP | SIMPLE | TIME ]
START WITH <numeric value> [ INCREMENT BY <numeric value> ]
[ MAXVALUE <numeric value> ] [ CYCLE | NOCYCLE ]
```

Parameter description:

Parameter	Description	Applicable To
START WITH	The initial sequence value. If it is not set, the default value is 1.	Simple sequence and group sequence
INCREMENT BY	The increment (or interval value or step) of each sequence increase. If it is not set, the default value is 1.	Simple Sequence
MAXVALUE	The maximum sequence value. If it is not specified, the default value is the maximum value of the signed BIGINT type.	Simple Sequence
CYCLE or NOCYCLE	Indicates whether to repeat the sequence value which starts from the value specified by START WITH after the sequence value reaches the maximum value. If it is not specified, the default value is NOCYCLE.	Simple Sequence

Note

- Group sequences are nonconsecutive. The START WITH parameter only provides reference for group sequences. **The start value of a group sequence is not necessarily the same as the START WITH value but is invariably greater than this value.**
- If you set START WITH when modifying a simple sequence, the START WITH value takes effect immediately. The next automatically generated sequence value starts from the new START WITH value. For example, if you change the START WITH value to 200 when the sequence value increases to 100, the next automatically generated sequence value starts from 200.
- Before changing the START WITH value, you need to analyze the existing sequence values and the speed of creating sequence values to avoid conflicts. Exercise caution when you modify the START WITH value.

For example, change the start value of the simple sequence seq2 to 3000, the step size to 5, and the maximum value to 1000000. The sequence value repeats after increasing to the maximum value.

```
mysql> ALTER SEQUENCE seq2 START WITH 3000 INCREMENT BY 5 MAXVALUE 1000000 CYCLE;
Query OK, 1 row affected (0.01 sec)
```

Convert the sequence type to another.

- Use the `CHANGE TO <sequence_type>` clause of `ALTER SEQUENCE`.
- If you specify the `CHANGE TO` clause in `ALTER SEQUENCE`, the `START WITH` parameter must be added to avoid forgetting to specify the start value and get duplicate values. If the `CHANGE TO` clause is not specified, it is not required to add the START WITH parameter.

Example: Convert a group sequence to a simple sequence.

```
mysql> ALTER SEQUENCE seq1 CHANGE TO SIMPLE START WITH 1000000;
Query OK, 1 row affected (0.02 sec)
```

Delete a sequence

Syntax:

```
DROP SEQUENCE <name>
```

Example:

```
mysql> DROP SEQUENCE seq3;
Query OK, 1 row affected (0.02 sec)
```

Query sequences

Syntax:

```
SHOW SEQUENCES
```

Example: The TYPE column lists the abbreviations of sequence types.

```
mysql> SHOW SEQUENCES;
+------------+--------------------+--------------+------------+--------------------+-------+--------+
| NAME       | VALUE              | INCREMENT_BY | START_WITH | MAX_VALUE          | CYCLE | TYPE   |
+------------+--------------------+--------------+------------+--------------------+-------+--------+
| AUTO_SEQ_1 | 91820513           | 1            | 91820200   | 9223372036854775807 | N     | SIMPLE |
| AUTO_SEQ_4 | 91820200           | 2            | 1000       | 9223372036854775807 | Y     | SIMPLE |
| seq_test   | N/A                | N/A          | N/A        | N/A                | N/A   | TIME   |
| AUTO_SEQ_2 | 100000             | N/A          | N/A        | N/A                | N/A   | GROUP  |
| AUTO_SEQ_3 | 200000             | N/A          | N/A        | N/A                | N/A   | GROUP  |
+------------+--------------------+--------------+------------+--------------------+-------+--------+
5 rows in set (0.01 sec)
```

Retrieve a sequence value

Syntax:

```
< sequence name >.NEXTVAL
```

Example:

```
SELECT sample_seq.nextVal FROM dual;
+--------------------+
| SAMPLE_SEQ.NEXTVAL |
+--------------------+
|             101001 |
+--------------------+
1 row in set (0.04 sec)
```

You can also write SAMPLE_SEQ.nextVal as a value to the SQL statement:

```
mysql> INSERT INTO some_users (name,address,gmt_create,gmt_modified,intro) VALUES ('sun',SAMPLE_SE
Q.nextVal,now(),now(),'aa');
Query OK, 1 row affected (0.01 sec)
```

> **Note** If you set the AUTO_INCREMENT parameter when creating a table, you do not need to specify an auto-increment column when running the INSERT statement. The auto-increment column is automatically maintained by PolarDB-X.

Retrieve sequence values in batches

Syntax:

```
SELECT < sequence name >.NEXTVAL FROM DUAL WHERE COUNT = < numeric value >
```

Example:

```
SELECT sample_seq.nextVal FROM dual WHERE count = 10;
+--------------------+
| SAMPLE_SEQ.NEXTVAL |
+--------------------+
|             101002 |
|             101003 |
|             101004 |
|             101005 |
|             101006|
|             101007 |
|             101008 |
|             101009 |
|             101010 |
|             101011 |
+--------------------+
10 row in set (0.04 sec)
```

5.13.3. Implicit sequence usage

After AUTO_INCREMENT is set for a primary key, the primary key is automatically filled in by using a sequence which is maintained by PolarDB-X.

CREATE TABLE

The standard CREATE TABLE syntax is extended to add the sequence type for auto-increment columns. If the type keyword is not specified, the default type is GROUP. Sequence names automatically created by PolarDB-X and associated with tables are all prefixed with AUTO_SEQ_ and followed by the table names.

```
CREATE TABLE <name> (
  <column> ... AUTO_INCREMENT [ BY GROUP | SIMPLE | TIME ],
  <column definition>,
  ...
) ... AUTO_INCREMENT=<start value>
```

SHOW CREATE TABLE

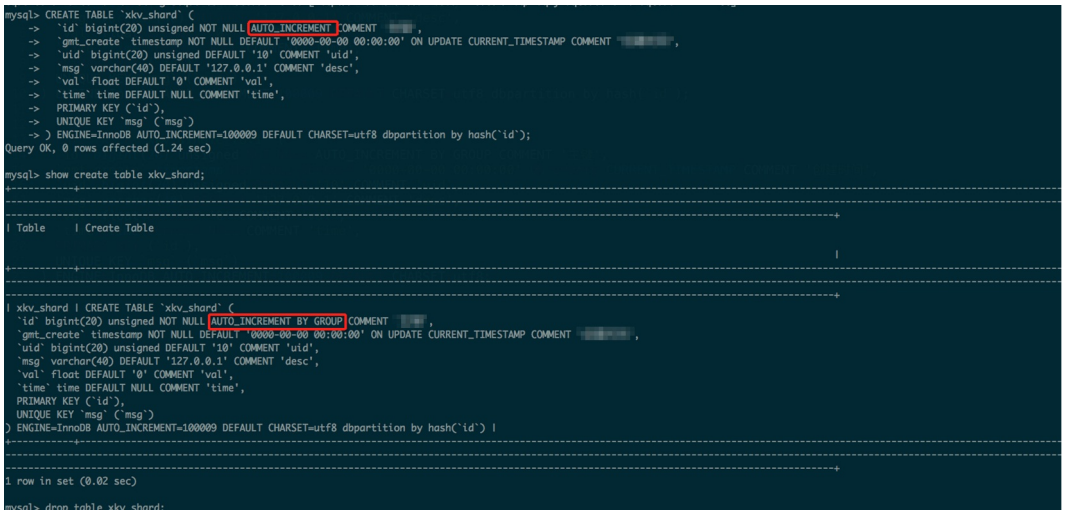
The sequence type is displayed for the auto-increment column of a table shard or broadcast table.

```
SHOW CREATE TABLE <name>
```

Examples

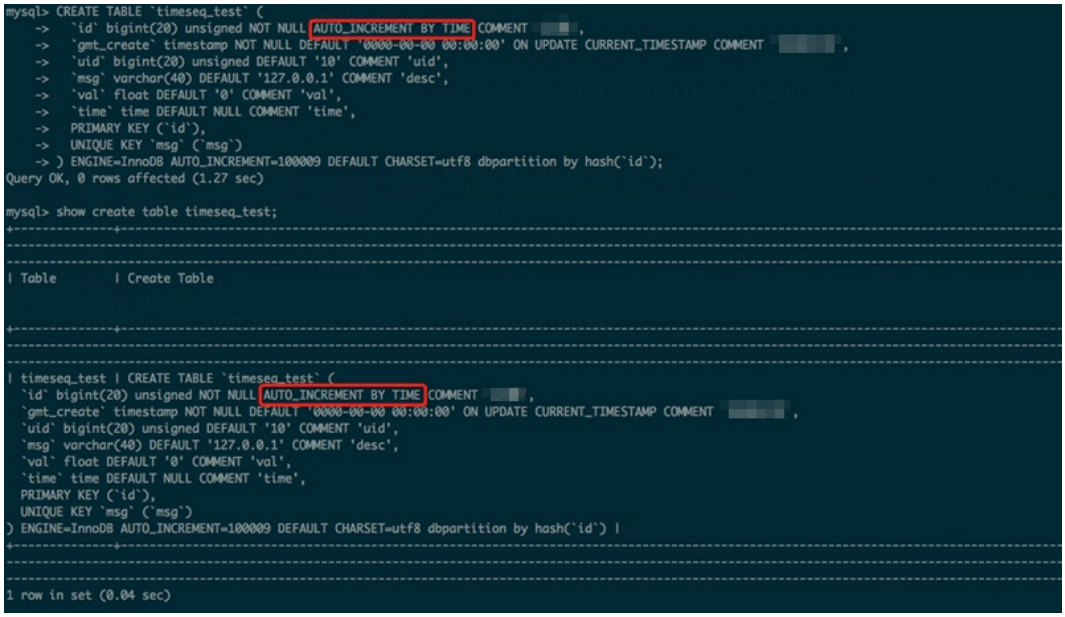
- If AUTO_INCREMENT is set but the sequence type is not specified when a table is created, a group sequence is used by default.

  Example 1



- When creating a table, set AUTO_INCREMENT and specify a time-based sequence as the primary key value.

  Example 2



ALTER TABLE

Currently, **ALTER TABLE** cannot be used to change the sequence type but can be used to change the initial value. To modify the implicit sequence type in a table, use the **SHOW SEQUENCES** command to find the sequence names and types, and then use the **ALTER SEQUENCE** command to modify the sequences.

```
ALTER TABLE <name> ... AUTO_INCREMENT=<start value>
```

Notice If a PolarDB-X sequence is used, exercise caution when modifying the start value of AUTO_INCREMENT. You need to carefully evaluate the already generated sequence values and the speed of generating new sequence values to prevent conflicts.

5.13.4. Limits and precautions

This topic describes the limits and precautions for sequences.

Limits and precautions

- When a time-based sequence is used in an auto-increment column of a table, the data type of the column must be BIGINT.
- The `START WITH` parameter must be set when the sequence is changed to another type.
- Assume that you need to execute the `INSERT` statement on a non-partitioned PolarDB-X database to which only one ApsaraDB RDS for MySQL database is attached. The PolarDB-X database automatically optimizes and routes the statement to the ApsaraDB RDS for MySQL database and bypasses the optimizer that allocates sequence values. The INSERT statement is processed the same way if you need to execute this statement on a partitioned database to which single-database table shards are attached and these table shards are not broadcast table shards. The `INSERT INTO ... VALUES (seq.nextval, ...)` syntax is not supported. We recommend that you use the auto-increment column feature of ApsaraDB RDS for MySQL instead.
- Assume that you execute an `INSERT` statement on a PolarDB-X logical table that uses a sequence. For example, you need to execute the `INSERT INTO ··· VALUES ···` or `INSERT INTO ··· SELECT···` statement. PolarDB-X bypasses the optimizer and directly routes the statement to the underlying ApsaraDB RDS for MySQL tables so that the sequence does not take effect. The PolarDB-X logical table creates IDs by using the auto-increment column feature of the ApsaraDB RDS for MySQL tables.
- To allocate IDs for the same table, you can use the sequence feature of PolarDB-X or the auto-increment feature of ApsaraDB RDS for MySQL. If you use the two allocation methods together for the same table, duplicate IDs may be created. This makes troubleshooting difficult.

Troubleshoot primary key conflicts

Assume that data is directly written to the underlying ApsaraDB RDS for MySQL database and that the related primary key values are not the sequence values created by the PolarDB-X database. The primary key values automatically created by the PolarDB-X database may conflict with the directly written data. To troubleshoot this issue, perform the following operations:

1. View the existing sequences by executing the SHOWSEQUENCES SQL statement of PolarDB-X. The sequence prefixed with `AUTO_SEQ_` is an implicit sequence. This sequence is generated if you set the `AUTO_INCREMENT` parameter to create the table. Execute the `SHOWSEQUENCES;` statement. The following response is returned:

```
+------------------------+-------+-------------+------------+-----------+-------+-------+
| NAME                   | VALUE | INCREMENT_BY | START_WITH | MAX_VALUE | CYCLE | TYPE |
+------------------------+-------+-------------+------------+-----------+-------+-------+
| AUTO_SEQ_timeseq_test  | N/A   | N/A         | N/A        | N/A       | N/A   | TIME |
| AUTO_SEQ_xkv_shard_tbl1 | 0    | N/A         | N/A        | N/A       | N/A   | GROUP |
| AUTO_SEQ_xkv_shard     | 0     | N/A         | N/A        | N/A       | N/A   | GROUP |
+------------------------+-------+-------------+------------+-----------+-------+-------+
3 rows in set (0.04 sec)
```

2. For example, if the t_item table has a conflict and the primary key of this table is ID, execute the `SELECT MAX(id) FROM t_item;` statement to query the largest primary key value of the table from PolarDB-X. The following response is returned:

```
+-----------+
| max(id)   |
+-----------+
| 8231 |
+-----------+
1 row in set (0.01 sec)
```

3. Update the related value in the PolarDB-X sequence table to a value greater than 8231, such as 9000. Then, no error is returned for the primary key values that are automatically created by the sequence you set in the statement.

   Execute the `ALTER SEQUENCE AUTO_SEQ_USERS START WITH 9000;` statement. The following response is returned:

```
Query OK, 1 row affected (0.01 sec)
```

5.14. Best practices

5.14.1. Determine shard keys

A shard key is a field for database sharding or table sharding. The shard key is used to create sharding rules. A PolarDB-X instance horizontally partitions data from a logical table into physical database shards on each ApsaraDB RDS for MySQL instance based on the shard key.

When you perform table sharding, you must comply with the following primary principle: Determine the appropriate entities to which data belongs based on your business logic. You must make sure that most or core SQL operations are performed based on the entities. After you determine the entities, you can use the fields that identify the entities as your shard keys.

In most cases, entities vary based on application scenarios. In the following typical application scenarios, entities are clear and the fields that identify the entities can be used as shard keys:

- User-oriented Internet applications are designed to meet user requirements. Users are the entities and the user ID field can be used as the shard key.
- Seller-oriented e-commerce applications are designed to meet seller requirements. Sellers are the

entities and the seller ID field can be used as the shard key.

- Gaming applications are designed to meet gamer requirements. Gamers are the entities and the gamer ID field can be used as the shard key.
- Internet of Vehicles (IoV) applications are designed based on vehicles. Vehicles are the entities and the vehicle ID field can be used as the shard key.
- Tax applications are designed based on taxpayers. Taxpayers are the entities and the taxpayer ID field can be used as the shard key.

In other scenarios, you can also use the appropriate field that identifies the entities as the shard key.

Assume that you need to horizontally partition data in a single table of a seller-oriented e-commerce application.

```
CREATE TABLE sample_order (
 id INT(11) NOT NULL,
 sellerId INT(11) NOT NULL,
 trade_id INT(11) NOT NULL,
 buyer_id INT(11) NOT NULL,
 buyer_nick VARCHAR(64) DEFAULT NULL,
 PRIMARY KEY (id)
);
```

Sellers are the entities. Therefore, you can use the sellerId field as the shard key to perform only database sharding. You can execute the following data definition language (DDL) statement to create a table:

```
CREATE TABLE sample_order (
 id INT(11) NOT NULL,
 sellerId INT(11) NOT NULL,
 trade_id INT(11) NOT NULL,
 buyer_id INT(11) NOT NULL,
 buyer_nick VARCHAR(64) DEFAULT NULL,
 PRIMARY KEY (id)
) DBPARTITION BY HASH(sellerId);
```

If no entity can be used as the shard key, use the following methods to determine the shard key:

- Determine your shard key based on data distribution and data access requests. Make sure that your data is evenly distributed across table shards in database shards if possible. This method can be used if a large number of analytical queries need to be performed and query concurrency stays at 1 in most cases.
- Combine the fields of the string, date, and time data types and use the combined result as your shard key for database sharding or table sharding. This method is applicable to log retrieval.

Assume that a log system records all user operations and that you need to horizontally partition the following single log table:

```
CREATE TABLE user_log (
 userId INT(11) NOT NULL,
 name VARCHAR(64) NOT NULL,
 operation VARCHAR(128) DEFAULT NULL,
 actionDate DATE DEFAULT NULL
);
```

You can combine the user ID and time fields as the shard key to partition the table by week. You can execute the following DDL statement to create a table:

```
CREATE TABLE user_log (
 userId INT(11) NOT NULL,
 name VARCHAR(64) NOT NULL,
 operation VARCHAR(128) DEFAULT NULL,
 actionDate DATE DEFAULT NULL
) DBPARTITION BY HASH(userId) TBPARTITION BY WEEK(actionDate) TBPARTITIONS 7;
```

For more information about how to determine shard keys and how to perform table sharding, see DDL statements.

Notice Whatever shard key and sharding policy are used, avoid querying data only from one shard if possible.

5.14.2. Select the number of shards

PolarDB-X supports horizontal partitioning of databases and tables. Eight physical database shards are created on each ApsaraDB RDS for MySQL instance by default, and one or more physical table shards can be created on each physical database shard. The number of table shards is also called the number of shards.

Generally, we recommend that each physical table shard contain no more than 5 million rows of data. Generally, you can estimate the data growth in one to two years. Divide the estimated total data size by the total number of physical database shards, and then divide the result by the recommended maximum data size of 5 million, to obtain the number of physical table shards to be created on each physical database shard:

```
Number of physical table shards in each physical database shard = CEILING(Estimated total data size/(Numb
er of ApsaraDB RDS for MySQL instances x 8)/5,000,000)
```

Therefore, when the calculated number of physical table shards is equal to 1, only database sharding needs to be performed, that is, a physical table shard is created in each physical database shard. If the calculation result is greater than 1, we recommend that you perform both database sharding and table sharding, that is, there are multiple physical table shards in each physical database shard.

For example, if a user estimates that the total data size of a table will be about 0.1 billion rows two years later and the user has four ApsaraDB RDS for MySQL instances, then according to the preceding formula:

```
Number of physical table shards in each physical database shard = CEILING(100,000,000/(4 x 8)/5,000,000) = CEILING(0.625) = 1
```

The result is 1, so only database sharding is needed, that is, one physical table shard is created in each physical database shard.

If only one ApsaraDB RDS for MySQL instance is used in the preceding example, the formula is as follows:

```
Number of physical table shards in each physical database shard = CEILING(100,000,000/(1 x 8)/5,000,000) = CEILING(2.5) = 3
```

The result is 3, so we recommend that you create three physical table shards in each physical database shard.

5.14.3. Basic concepts of SQL optimization

PolarDB-X is an efficient and stable distributed relational database service that processes distributed relational computing. PolarDB-X optimizes SQL statements differently from standalone relational databases such as MySQL. PolarDB-X focuses on the network I/O overheads in a distributed environment and routes SQL operations to the underlying ApsaraDB RDS for MySQL database shards. This reduces the network I/O overheads and improves the SQL execution efficiency.

PolarDB-X provides statements for you to obtain SQL execution information and optimize SQL execution. For example, you can execute the EXPLAIN statements to query SQL execution plans and execute the TRACE statements to query SQL execution processes and overheads. This topic introduces the basic concepts and common statements related to SQL optimization in PolarDB-X.

Execution plans

An SQL execution plan (or execution plan) is a set of ordered steps generated to access data. In PolarDB-X, the execution plan is divided into the execution plan at the PolarDB-X layer and the execution plan at the ApsaraDB RDS for MySQL layer. Execution plan analysis is an effective way to optimize SQL statements. Execution plan analysis allows you to know whether PolarDB-X or ApsaraDB RDS for MySQL has generated optimal execution plans for SQL statements and whether further optimization can be made.

When an SQL statement is executed, the PolarDB-X optimizer determines on which database shards the SQL statement is executed, based on the basic information of the SQL statement and related tables. It also determines the specific SQL statement form, execution policy, and data merging and computing policy for each database shard based on this basic information. This process optimizes SQL statement execution and generates execution plans at the PolarDB-X layer. The execution plan at the ApsaraDB RDS for MySQL layer is the native MySQL execution plan.

PolarDB-X provides a set of EXPLAIN statements to show execution plans at different levels or with different levels of details.

The following table briefly describes the EXPLAIN statements in PolarDB-X.

EXPLAIN statements

Statement	Description	Example
EXPLAIN { SQL }	Shows the summary execution plan of an SQL statement at the PolarDB-X layer, including the database shards on which the SQL statement is executed, physical statements, and general parameters.	EXPLAIN SELECT * FROM test
EXPLAIN DETAIL { SQL }	Shows the detailed execution plans of an SQL statement at the PolarDB-X layer, including the statement type, concurrency, returned field information, physical table shards, and database shard groups.	EXPLAIN DETAIL SELECT * FROM test
EXPLAIN EXECUTE { SQL }	Shows the execution plan at the underlying ApsaraDB RDS for MySQL layer, which is equivalent to the EXPLAIN statement of open-source MySQL.	EXPLAIN EXECUTE SELECT * FROM test

Execution plan at the PolarDB-X layer

The following table describes the fields in the response returned for an execution plan at the PolarDB-X layer.

Fields in an execution plan at the PolarDB-X layer

Field	Description
GROUP_NAME	The name of a database shard for a PolarDB-X logical table. The suffix identifies the specific database shard. The value is consistent with the result of the SHOW NODE statement.
SQL	The SQL statement executed on this database shard.
PARAMS	The SQL statement parameters used when PolarDB-X communicates with ApsaraDB RDS for MySQL over the Prepare protocol.

The SQL field can be in two forms:

1. If an SQL statement does not contain the following parts, the execution plan is displayed as an SQL statement:
   - An aggregate function that involves multiple database shards.
   - A distributed join that involves multiple table shards.
   - A complex subquery.

Assume that you execute the `EXPLAIN SELECT * FROM test;` statement. The following response is returned:

```
+-----------------------------------------------+----------------------------------------------+--------+
| GROUP_NAME                                    | SQL                                          | PARAMS |
+-----------------------------------------------+----------------------------------------------+--------+
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0000_RDS | select `test`.`c1`,`test`.`c2` from `test` |
{}  |
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0001_RDS | select `test`.`c1`,`test`.`c2` from `test` |
{}  |
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0002_RDS | select `test`.`c1`,`test`.`c2` from `test` |
{}  |
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0003_RDS | select `test`.`c1`,`test`.`c2` from `test` |
{}  |
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0004_RDS | select `test`.`c1`,`test`.`c2` from `test` |
{}  |
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0005_RDS | select `test`.`c1`,`test`.`c2` from `test` |
{}  |
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0006_RDS | select `test`.`c1`,`test`.`c2` from `test` |
{}  |
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0007_RDS | select `test`.`c1`,`test`.`c2` from `test` |
{}  |
+-----------------------------------------------+----------------------------------------------+--------+
8 rows in set (0.04 sec)
```

You can find the group names in the GROUP_NAME column by executing the `SHOW NODE;` statement.

```
+----+-----------------------------------------+-----------------+-----------------+--------------------+-------------------+
| ID | NAME                                    | MASTER_READ_COUNT | SLAVE_READ_COUNT | MASTER_READ_PERCENT |
SLAVE_READ_PERCENT |
+----+-----------------------------------------+-----------------+-----------------+--------------------+-------------------+
| 0 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0000_RDS |         69 |         0 | 100%          | 0%
|
| 1 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0001_RDS |         45 |         0 | 100%          | 0%
|
| 2 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0002_RDS |         30 |         0 | 100%          | 0%
|
| 3 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0003_RDS |         29 |         0 | 100%          | 0%
|
| 4 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0004_RDS |         11 |         0 | 100%          | 0%
|
| 5 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0005_RDS |          1 |         0 | 100%          | 0%
|
| 6 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0006_RDS |          8 |         0 | 100%          | 0%
|
| 7 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0007_RDS |         50 |         0 | 100%          | 0%
|
+----+-----------------------------------------+-----------------+-----------------+--------------------+-------------------+
8 rows in set (0.10 sec)
```

2. Execution plans that cannot be expressed by SQL statements can be expressed in custom formats of PolarDB-X.

   Assume that you execute the `EXPLAIN DETAIL SELECT COUNT(*) FROM test;` statement The following response is returned:

```
+------------------------------------------------+--------------------------------+--------+
| GROUP_NAME                    | SQL                | PARAMS |
+------------------------------------------------+--------------------------------+--------+
| TEST_DB_1478746391548CDTCTEST_DB_OXGJ_0000_RDS | Merge as test
  queryConcurrency:GROUP_CONCURRENT
  columns:[count(*)]
  executeOn: TEST_DB_1478746391548CDTCTEST_DB_OXGJ_0000_RDS
    Query from test as test
      queryConcurrency:SEQUENTIAL
      columns:[count(*)]
      tableName:test
      executeOn: TEST_DB_1478746391548CDTCTEST_DB_OXGJ_0000_RDS
    Query from test as test
      queryConcurrency:SEQUENTIAL
      columns:[count(*)]
      tableName:test
      executeOn: TEST_DB_1478746391548CDTCTEST_DB_OXGJ_0001_RDS
    ... ...
    Query from test as test
      queryConcurrency:SEQUENTIAL
      columns:[count(*)]
      tableName:test
      executeOn: TEST_DB_1478746391548CDTCTEST_DB_OXGJ_0007_RDS
| NULL   |
+------------------------------------------------+--------------------------------+--------+
1 row in set (0.00 sec)
```

executeOn indicates the database shard on which the SQL statement is executed. PolarDB-X finally merges the results returned by the related database shards.

Execution plans at the ApsaraDB RDS for MySQL layer

Execution plans at the ApsaraDB RDS for MySQL layer are the same as the native MySQL execution plans. For more information, see MySQL documentation.

One PolarDB-X logical table may consist of multiple table shards distributed in different database shards. Therefore, you can view the execution plans at the ApsaraDB RDS for MySQL layer in multiple ways.

1. View the execution plan of an SQL statement on an ApsaraDB RDS for MySQL database shard.

   If the query condition contains a shard key, directly execute the EXPLAIN EXECUTE statement to show the execution plan on the corresponding database shard. Assume that you execute the `EXPLAIN EXECUTE SELECT * FROM test WHERE c1 = 1;` statement. The following response is returned:

```
+----+-------------+-------+-------+---------------+---------+---------+-------+------+-------+
| id | select_type | table | type  | possible_keys | key     | key_len | ref   | rows | Extra |
+----+-------------+-------+-------+---------------+---------+---------+-------+------+-------+
|  1 | SIMPLE      | test  | const | PRIMARY       | PRIMARY | 4       | const |    1 | NULL  |
+----+-------------+-------+-------+---------------+---------+---------+-------+------+-------+
1 row in set (0.04 sec)
```

> **Notice** If an SQL statement is executed on multiple table shards, the EXPLAIN EXECUTE statement returns an execution plan on a random ApsaraDB for RDS table shard. For example, the query condition in the SQL statement does not contain a shard key.

To view the execution plan of an SQL statement on a specified table shard, you can add a NODE hint. Assume that you execute the `/! TDDL:node='TESTDB_1478746391548CDTCTESTDB_OXGJ_0000_RDS'*/EXPLAIN SELECT * FROM test;` statement. The following response is returned:

```
+----+-------------+-------+------+---------------+-------+---------+-------+------+-------+
| id | select_type | table | type | possible_keys | key   | key_len | ref   | rows | Extra |
+----+-------------+-------+------+---------------+-------+---------+-------+------+-------+
|  1 | SIMPLE      | test  | ALL  | NULL          | NULL  | NULL    | NULL  |    2 | NULL  |
+----+-------------+-------+------+---------------+-------+---------+-------+------+-------+
1 row in set (0.04 sec)
```

2. View the execution plans of all ApsaraDB RDS for MySQL table shards.

   You can add the SCAN hint in the SQL statement to view the execution plans of the SQL statement on all table shards: Execute the `/! TDDL:scan='test'*/EXPLAIN SELECT * FROM test;` statement. The following response is returned:

```
+----+-------------+-------+------+---------------+-------+---------+-------+------+-------+
| id | select_type | table | type | possible_keys | key   | key_len | ref   | rows | Extra |
+----+-------------+-------+------+---------------+-------+---------+-------+------+-------+
|  1 | SIMPLE      | test  | ALL  | NULL          | NULL  | NULL    | NULL  |    2 | NULL  |
|  1 | SIMPLE      | test  | ALL  | NULL          | NULL  | NULL    | NULL  |    3 | NULL  |
+----+-------------+-------+------+---------------+-------+---------+-------+------+-------+
2 rows in set (0.08 sec)
```

> **Notice**
>
> i. After you add the hint, PolarDB-X replaces only the table shard names if database sharding and table sharding need to be performed. Then, it directly routes the logical SQL statement to the underlying ApsaraDB RDS for MySQL table shards. It will not process the result.
>
> ii. Execution plans obtained by using an EXPLAIN statement are generated by static analysis and are not actually executed in databases.

TRACE statements

The TRACE statements in PolarDB-X can track the SQL execution process and the overheads at each stage. It can be used together with the execution plan to optimize SQL statements.

PolarDB-X provides the TRACE and SHOW TRACE statements, and these two statements must be used together.

5.14.4. SQL optimization methods

5.14.4.1. Overview

This topic describes the principles of SQL optimization and methods for optimizing different types of SQL statements in PolarDB-X.

Basic principles of SQL optimization

In PolarDB-X, SQL computing that can be performed by the underlying ApsaraDB RDS for MySQL instances is called push-down computing. Push-down computing reduces data transfers, decreases overheads at the network layer and PolarDB-X layer, and improves the execution efficiency of SQL statements.

Therefore, the basic principle of SQL optimization in PolarDB-X is to push down as many computations as possible to the underlying ApsaraDB RDS for MySQL instances.

The following list shows the push-down computations:

- JOIN connections
- Filter conditions, such as the `WHERE` or `HAVING` conditions
- Aggregate computations, such as the `COUNT` and `GROUP BY`
- Sorting, such as `ORDER BY`
- Deduplication, such as `DISTINCT`
- Function computations, such as the `NOW()` function
- Subqueries

> **Notice** The preceding list only describes possible forms of push-down computations. This does not mean that all clauses, conditions, or combinations of clauses or conditions can be pushed down for computing.

SQL statements that have different types and conditions can be optimized in different ways. The following list shows some specific scenarios:

- Single-table SQL optimization
  - Filter condition optimization
  - Optimization of the number of returned rows for a query
  - Grouping and sorting optimization
- Join optimization
  - Optimization of joins that can be pushed down
  - Optimization of distributed joins

- Subquery optimization

5.14.4.2. Single-table SQL optimization

This topic describes how to optimize SQL statements for a single table.

Single-table SQL optimization must follow the following rules:

- Make sure that the SQL statement contains the shard key.
- Use an equivalence condition for the shard key whenever possible.
- If the shard key is an IN condition, the number of values after IN must be as small as possible. This means that this number must be far smaller than the number of table shards and remain unchanged as your business grows.
- If the SQL statement does not contain a shard key, use only one of the DISTINCT, GROUP BY, and ORDER BY clauses in the same SQL statement.

Filter condition optimization

PolarDB-X horizontally partitions data by the shard key. This requires the filter condition to contain the shard key so that PolarDB-X can push queries down to specific database shards based on the shard key values. This way, PolarDB-X does not need to scan all table shards.

For example, the shard key of the test table is c1. If the filter condition does not contain this shard key, full table scan is performed.

Execute the `SELECT * FROM test WHERE c2 = 2` statement. The following response is returned:

```
+----+----+
| c1 | c2 |
+----+----+
|  2 |  2 |
+----+----+
1 row in set (0.05 sec)
```

To view the corresponding execution plan, execute the `EXPLAIN SELECT * FROM test WHERE c2 = 2;` statement. The following response is returned:

```
+------------------------------------------------+-------------------------------------------------------------------------+--------+
| GROUP_NAME                                     | SQL                                                                     | PARAMS |
+------------------------------------------------+-------------------------------------------------------------------------+--------+
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0004_RDS | select `test`.`c1`,`test`.`c2` from `test` where (`test`.`c2` = 2) | {}     |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0007_RDS | select `test`.`c1`,`test`.`c2` from `test` where (`test`.`c2` = 2) | {}     |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0005_RDS | select `test`.`c1`,`test`.`c2` from `test` where (`test`.`c2` = 2) | {}     |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0002_RDS | select `test`.`c1`,`test`.`c2` from `test` where (`test`.`c2` = 2) | {}     |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0003_RDS | select `test`.`c1`,`test`.`c2` from `test` where (`test`.`c2` = 2) | {}     |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0006_RDS | select `test`.`c1`,`test`.`c2` from `test` where (`test`.`c2` = 2) | {}     |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0000_RDS | select `test`.`c1`,`test`.`c2` from `test` where (`test`.`c2` = 2) | {}     |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0001_RDS | select `test`.`c1`,`test`.`c2` from `test` where (`test`.`c2` = 2) | {}     |
+------------------------------------------------+-------------------------------------------------------------------------+--------+
8 rows in set (0.00 sec)
```

The smaller the value range of the filter condition that contains the shard key, the faster the query speed of PolarDB-X.

Assume that you execute the `SELECT * FROM test WHERE c1 > 1 AND c1 < 4;` statement to query data that contains c1 and meets the condition from the test table. The following response is returned:

```
+----+----+
| c1 | c2 |
+----+----+
|  2 |  2 |
|  3 |  3 |
+----+----+
2 rows in set (0.04 sec)
```

To view the corresponding execution plan, execute the `EXPLAIN SELECT * FROM test WHERE c1 > 1 AND c1 < 4;` statement. The following response is returned:

```
+-----------------------------------------------+-----------------------------------------------------------------------------------------------+-
-------+
| GROUP_NAME                    | SQL                                                                                           | PARAMS |
+-----------------------------------------------+-----------------------------------------------------------------------------------------------+-
-------+
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0002_RDS | select `test`.`c1`,`test`.`c2` from `test` wh
ere ((`test`.`c1` > 1) AND (`test`.`c1` < 4)) | {}    |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0003_RDS | select `test`.`c1`,`test`.`c2` from `test` wh
ere ((`test`.`c1` > 1) AND (`test`.`c1` < 4)) | {}    |
+-----------------------------------------------+-----------------------------------------------------------------------------------------------+-
-------+
2 rows in set (0.00 sec)
```

The equivalence condition is executed faster than the range condition. Assume that you execute the `SELECT * FROM test WHERE c1 = 2;` statement. The following response is returned:

```
+----+----+
| c1 | c2 |
+----+----+
|  2 |  2 |
+----+----+
1 row in set (0.03 sec)
```

To view the corresponding execution plan, execute the `EXPLAIN SELECT * FROM test WHERE c1 = 2;` statement. The following response is returned:

```
+-----------------------------------------------+-----------------------------------------------------------------------------+--------+
| GROUP_NAME                    | SQL                                                                         | PARAMS |
+-----------------------------------------------+-----------------------------------------------------------------------------+--------+
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0002_RDS | select `test`.`c1`,`test`.`c2` from `test` wh
ere (`test`.`c1` = 2) | {}    |
+-----------------------------------------------+-----------------------------------------------------------------------------+--------+
1 row in set (0.00 sec)
```

If you need to insert data into a partitioned table, the inserted field must contain a shard key.

Assume that you specify shard key c1 when you insert data to the test table. Execute the `INSERT INTO test(c1,c2) VALUES(8,8);` statement. The following response is returned:

```
Query OK, 1 row affected (0.07 sec)
```

Optimization of the number of returned rows for a query

When PolarDB-X performs a query that contains **LIMIT** [*offset*,] *row_count*, PolarDB-X actually reads records before *offset* in order and directly discards them. When the value of *offset* is large, the query is slow even if the value of *row_count* is small. For example, execute the following SQL statement:

```
SELECT *
FROM sample_order
ORDER BY sample_order.id
LIMIT 10000, 2;
```

Although only the 10,000th and 10,001st records are returned, it takes about 12 seconds to execute the SQL statement because PolarDB-X actually reads 10,002 records.

Execute the `SELECT * FROM sample_order ORDER BY sample_order.id LIMIT 10000,2;` statement. The following response is returned.

```
+--------------+------------+--------------+--------------+------------+
| id           | sellerId   | trade_id     | buyer_id     | buyer_nick |
+--------------+------------+--------------+--------------+------------+
| 242012755468 | 1711939506 | 242012755467 | 244148116334 | zhangsan   |
| 242012759093 | 1711939506 | 242012759092 | 244148138304 | wangwu     |
+--------------+------------+--------------+--------------+------------+
2 rows in set (11.93 sec)
```

To view the corresponding execution plan, execute the `EXPLAIN SELECT * FROM sample_order ORDER BY sample_order.id LIMIT 10000,2;` statement. The following response is returned.

```
+------------------+-------------+------------+
| GROUP_NAME     | SQL      | PARAMS   |
+------------------+-------------+------------+
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0004_RDS | select `sample_order`.`id`,`sample_order`.`
sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `s
ample_order` order by `sample_order`.`id` asc limit 0,10002 | {}    |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0007_RDS | select `sample_order`.`id`,`sample_order`.`
sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `s
ample_order` order by `sample_order`.`id` asc limit 0,10002 | {}    |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0005_RDS | select `sample_order`.`id`,`sample_order`.`
sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `s
ample_order` order by `sample_order`.`id` asc limit 0,10002 | {}    |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0002_RDS | select `sample_order`.`id`,`sample_order`.`
sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `s
ample_order` order by `sample_order`.`id` asc limit 0,10002 | {}    |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0003_RDS | select `sample_order`.`id`,`sample_order`.`
sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `s
ample_order` order by `sample_order`.`id` asc limit 0,10002 | {}    |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0006_RDS | select `sample_order`.`id`,`sample_order`.`
sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `s
ample_order` order by `sample_order`.`id` asc limit 0,10002 | {}    |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0000_RDS | select `sample_order`.`id`,`sample_order`.`
sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `s
ample_order` order by `sample_order`.`id` asc limit 0,10002 | {}    |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0001_RDS | select `sample_order`.`id`,`sample_order`.`
sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `s
ample_order` order by `sample_order`.`id` asc limit 0,10002 | {}    |
+------------------+----------------------------+--------+
8 rows in set (0.01 sec)
```

To optimize the preceding SQL statement, find the ID set, and use the IN query to match the actual records. The following example shows the SQL query after modification:

```
SELECT *
FROM sample_order o
WHERE o.id IN (
    SELECT id
    FROM sample_order
    ORDER BY id
    LIMIT 10000, 2 );
```

This is to cache IDs in the memory first on the premise that the number of IDs is small. If the shard key of the sample_order table is ID, PolarDB-X can also push down such an IN query to different database shards by performing rule-based calculation. This avoids full table scan and unnecessary network I/O operations. View the effect of the SQL query after modification:

Execute the `SELECT * FROM sample_order o WHERE o.id IN ( SELECT id FROM sample_order ORDER BY id LIMIT 10000,2 );` statement. The following response is returned:

```
+--------------+------------+--------------+--------------+------------+
| id         | sellerId  | trade_id    | buyer_id    | buyer_nick |
+--------------+------------+--------------+--------------+------------+
| 242012755468 | 1711939506 | 242012755467 | 244148116334 | zhangsan   |
| 242012759093 | 1711939506 | 242012759092 | 244148138304 | wangwu     |
+--------------+------------+--------------+--------------+------------+
2 rows in set (1.08 sec)
```

The execution time is significantly reduced from 12 seconds to 1.08 seconds.

To view the corresponding execution plan, execute the `EXPLAIN SELECT * FROM sample_order o WHERE o.id IN ( SELECT id FROM sample_order ORDER BY id LIMIT 10000,2 );` statement. The following response is returned:

```
+---------------------------------------------+----------------------------------------------------------------------------------------------
----------------------------------+--------+
| GROUP_NAME                                  | SQL                                                                                | PARAMS |
+---------------------------------------------+----------------------------------------------------------------------------------------------
----------------------------------+--------+
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0002_RDS | select `o`.`id`,`o`.`sellerId`,`o`.`trade_id`
,`o`.`buyer_id`,`o`.`buyer_nick` from `sample_order` `o` where (`o`.`id` IN (10002)) | {}   |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0001_RDS | select `o`.`id`,`o`.`sellerId`,`o`.`trade_id`
,`o`.`buyer_id`,`o`.`buyer_nick` from `sample_order` `o` where (`o`.`id` IN (10001)) | {}   |
+---------------------------------------------+----------------------------------------------------------------------------------------------
----------------------------------+--------+
2 rows in set (0.03 sec)
```

Grouping and sorting optimization

In PolarDB-X, if an SQL query must use all of the DISTINCT, GROUP BY, and ORDER BY clauses, make sure that the fields after these clauses are the same and that the fields are shard keys. This way, only a small amount of data is returned for the SQL query. This minimizes the network bandwidth consumed by distributed queries and removes the need to retrieve a large volume of data and sort the data in a temporary table. This way, the system performance is maximized.

5.14.4.3. Join optimization

Joins in PolarDB-X are classified into joins that can be pushed down to ApsaraDB RDS for MySQL and those that cannot be pushed down to ApsaraDB RDS for MySQL. Joins that cannot be pushed down to ApsaraDB RDS for MySQL are distributed joins. The optimization policies for these two types of joins are different.

Optimize joins that can be pushed down

Joins that can be pushed down are classified into the following types:

- Joins between single tables, which are non-partitioned tables.
- The tables involved in the join contain the shard key in the filter condition and use the same sharding algorithm. This means that the data calculated by the sharding algorithm is distributed to the same table shard.
- The tables involved in the join use the shard key as the join condition and use the same sharding algorithm.
- Joins between broadcast tables and partitioned tables. Broadcast tables are also called small table broadcast.

In PolarDB-X, optimize joins to those that can be pushed down to and executed on database shards.

Take a join between a broadcast table and a partitioned table as an example. The broadcast table is used as the driving table for the join. The left table for the join is called the driving table. The broadcast table of PolarDB-X is replicated to all database shards. When the broadcast table is used as the driving table for the join, the join between this broadcast table and table shards can be pushed down to each database shard. The results are combined for computing. This way, the query performance is improved.

For example, a join is performed on the following three tables, among which the sample_area table is the broadcast table whereas the sample_item and sample_buyer tables are partitioned tables. The query execution time is about 15 seconds.

Execute the `SELECT sample_area.name FROM sample_item i JOIN sample_buyer b ON i.sellerId = b.sellerId JOIN sample_area a ON b.province = a.id WHERE a.id < 110107 LIMIT 0, 10;` statement. The following response is returned:

```
+------+
| name |
+------+
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
+------+
10 rows in set (14.88 sec)
```

If you adjust the join order and move the broadcast table to the leftmost as the driving table for the join, the join is pushed down to each database shard of the PolarDB-X instance.

Execute the `SELECT sample_area.name FROM sample_area a JOIN sample_buyer b ON b.province = a.id JOIN sample_item i ON i.sellerId = b.sellerId WHERE a.id < 110107 LIMIT 0, 10;` statement. The following response is returned:

```
+------+
| name |
+------+
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
+------+
10 rows in set (0.04 sec)
```

The query execution time decreases from 15 seconds to 0.04 seconds. The query performance is significantly improved.

> **Notice** The broadcast table achieves data consistency among database shards by using a synchronization mechanism. Data synchronization has a latency of several seconds.

Optimization of distributed joins

If a join cannot be pushed down, PolarDB-X must complete part of the computing in the query. A join that cannot be pushed down means that the join condition and filter condition do not contain a shard key. Such a join is a distributed join.

Tables in a distributed join are classified into two types based on the data size:

- Small table: A table that contains a small amount of data (less than 100 data records or less data than other tables) that is involved in join computation after filtering.
- Large table: A table that contains a large amount of data (more than 100 data records or more data than other tables) that is involved in join computation after filtering.

In most cases, Nested Loop and its derived algorithms are used in join computation at the PolarDB-X layer. If sorting is required for joins, the Sort Merge algorithm is used. When the Nested Loop algorithm is used, a smaller data size in the left table for a join indicates a smaller number of queries performed by PolarDB-X on the right table. If the right table has indexes or contains a small volume of data, the join is even faster. In PolarDB-X, the left table for a distributed join is called the driving table. To optimize a distributed join, use a small table as the driving table and set as many filter conditions as possible for the driving table.

Take the following distributed join as an example. The query takes about 24 seconds:

Execute the `SELECT t.title, t.price FROM sample_order o, ( SELECT * FROM sample_item i WHERE i.id = 242002396687 ) t WHERE t.source_id = o.source_item_id AND o.sellerId &lt; 1733635660;` statement. The following response is returned:

```
+----------------------------------+--------+
| title                            | price  |
+----------------------------------+--------+
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
+----------------------------------+--------+
10 rows in set (23.79 sec)
```

The preceding join is an INNER JOIN query, and the actual size of the intermediate data involved in the join computation is unknown. Perform COUNT() operations on the o table and the t table to obtain the actual data size.

The o.sellerId < 1733635660 filter in the WHERE condition is only related to the o table. Extract this filter and add it to the COUNT() condition of the o table. Execute the `SELECT COUNT(*) FROM sample_order o WHERE o.sellerId < 1733635660;` statement. The following response is returned:

```
+----------+
| count(*) |
+----------+
|  504018 |
+----------+
1 row in set (0.10 sec)
```

The intermediate result of the o table contains about 500,000 records. Similarly, the t table is a subquery. Directly extract the subquery and add it to the COUNT() query. Then, execute the `SELECT COUNT(*) FROM sample_item i WHERE i.id = 242002396687;` statement. The following response is returned:

```
+----------+
| count(*) |
+----------+
|     1 |
+----------+
1 row in set (0.01 sec)
```

The intermediate result of the t table contains only one record. Therefore, the o table is the large table and the t table is the small table. Use the small table as the driving table for the distributed join. Then, execute the `SELECT t.title, t.price FROM ( SELECT * FROM sample_item i WHERE i.id = 242002396687 ) t, sample_order o WHERE t.source_id = o.source_item_id AND o.sellerId < 1733635660;` statement. The following response is returned:

```
+----------------------------------+--------+
| title                    | price |
+----------------------------------|--------+
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
| Sample Item for Distributed JOIN | 239.00 |
+----------------------------------+--------+
10 rows in set (0.15 sec)
```

The query time was reduced from about 24 seconds to 0.15 seconds. The query performance is significantly improved.

5.14.4.4. Subquery optimization

When you optimize an SQL statement that contains subqueries, push the subqueries down to database shards as many as possible to reduce the computing workload at the PolarDB-X layer.

For this purpose, you can try the following two optimization methods:

- Rewrite subqueries into multi-table joins, and optimize the joins.
- Use the shard key in the join condition or filter condition so that PolarDB-X can push the queries down to specific database shards to avoid full table scan.

Take the following subquery as an example:

```
SELECT o. *
FROM sample_order o
WHERE NOT EXISTS
   (SELECT sellerId FROM sample_seller s WHERE o.sellerId = s.id);
```

Rewrite the subquery into a join:

```
SELECT o. *
FROM sample_order o LEFT JOIN sample_seller s ON o.sellerId = s.id
WHERE s.id IS NULL;
```

5.14.5. Choose a connection pool for an application

A database connection pool is used to manage database connections in a centralized manner. This improves application performance and reduce database loads.

Connection pools offer the following benefits:

- Resource reuse: Connections can be reused to avoid high performance overheads caused by frequent connection establishment and release. Resource reuse can also improve the system stability.
- Improvement of system response efficiency: After the connection initialization is complete, all requests can directly use the existing connections. This avoids the overheads of connection initialization and release and improves the system response efficiency.
- Connection leakage prevention: The connection pool forcibly revokes connections based on the preset revocation policy to avoid connection resource leakage.

We recommend that you use a connection pool to connect applications and databases for business operations. For Java programs, we recommend that you use Druid connection pools.

The following code shows the standard Spring configurations of Druid connection pools:

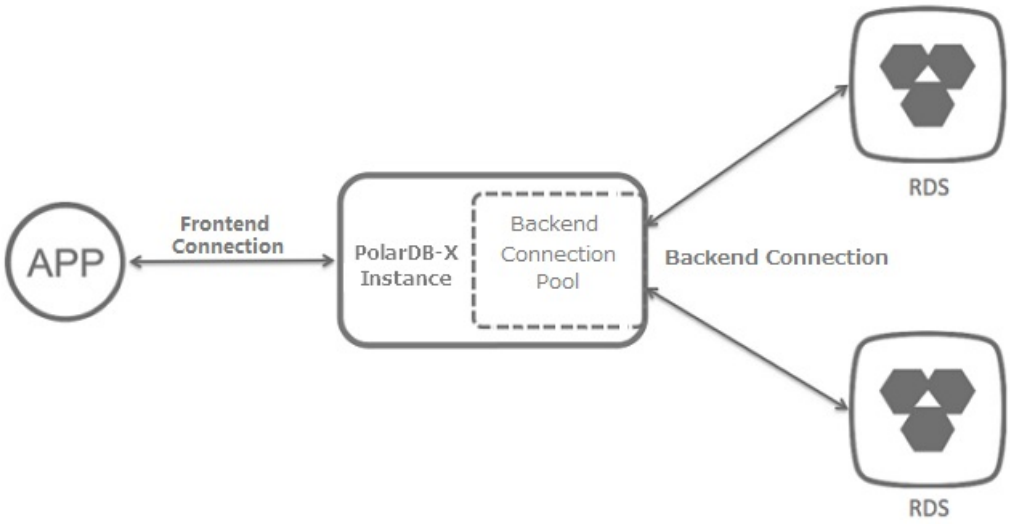
```
  <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close">
      <property name="driverClassName" value="com.mysql.jdbc.Driver" />
      <! -- Set the basic properties, including the URL, username, and password. -->
      <property name="url" value="jdbc:mysql://ip:port/db? autoReconnect=true&amp;rewriteBatchedStatements=true&amp;socketTimeout=30000&amp;connectTimeout=3000" />
      <property name="username" value="root" />
      <property name="password" value="123456" />
      <! -- Configure the initial connection pool size, maximum number of active connections, and minimum number of idle connections. -->
      <property name="maxActive" value="20" />
      <property name="initialSize" value="3" />
      <property name="minIdle" value="3" />
      <! -- maxWait indicates the timeout period for obtaining the connection. -->
      <property name="maxWait" value="60000" />
      <! -- timeBetweenEvictionRunsMillis indicates the interval for detecting idle connections to be closed, in milliseconds. -->
      <property name="timeBetweenEvictionRunsMillis" value="60000" />
      <! -- minEvictableIdleTimeMillis indicates the minimum idle time of a connection in the connection pool, in milliseconds.-->
      <property name="minEvictableIdleTimeMillis" value="300000" />
      <! -- Set the SQL statement used to check whether connections are available. -->
      <property name="validationQuery" value="SELECT 'z'" />
      <! -- Set whether to enable idle connection checks. -->
      <property name="testWhileIdle" value="true" />
      <! -- Set whether to check the connection status before the system obtains a connection. -->
      <property name="testOnBorrow" value="false" />
      <! -- Set whether to check the connection status before the system releases a connection. -->
      <property name="testOnReturn" value="false" />
  </bean>
```

5.14.6. Connections to PolarDB-X instances

When an application connects to a PolarDB-X instance to perform operations, two types of connections are available from the perspective of the PolarDB-X instance:

- Frontend connection: a connection established by an application to the logical database in the PolarDB-X instance.
- Backend connection: a connection established by a node in a PolarDB-X instance to a physical database shard in a backend ApsaraDB RDS for MySQL instance.

PolarDB-X instance connection diagram



Frontend connection

Theoretically, the number of frontend connections is limited only by the available memory size and the number of network connections to the nodes of the PolarDB-X instance. In actual scenarios, when an application connects to a PolarDB-X instance, the nodes of the PolarDB-X instance manage a limited number of connections to perform requested operations. The nodes do not maintain a large number of concurrent persistent connections, such as tens of thousands of concurrent persistent connections. Therefore, the number of frontend connections that a PolarDB-X instance can accept can be considered unlimited.

The number of frontend connections is unlimited and a large number of idle connections are allowed. Therefore, this method applies to scenarios where a large number of servers exist to deploy applications on the business side and need to maintain their connections to the PolarDB-X instance.

> **Note** Although the number of frontend connections is considered unlimited, operation requests obtained from frontend connections are actually executed by internal threads of the PolarDB-X instance over backend connections. Due to the limited number of internal threads and backend connections, the total number of concurrent requests that the PolarDB-X instance can process is limited.

Backend connection

Each node of a PolarDB-X instance creates a backend connection pool to automatically manage and maintain the backend connections to the physical database shards in the ApsaraDB RDS for MySQL instance. Therefore, the maximum number of connections in the backend connection pool of a PolarDB-X instance is directly related to the maximum number of connections that the ApsaraDB RDS for MySQL instance supports. You can use the following formula to calculate the maximum number of connections in the backend connection pool of a PolarDB-X instance:

Maximum number of connections in a backend connection pool of a PolarDB-X instance = FLOOR(Maximum number of connections in an ApsaraDB RDS for MySQL instance/Number of physical database shards in the ApsaraDB RDS for MySQL instance/Number of nodes on the PolarDB-X instance)

Assume that you have an ApsaraDB RDS for MySQL instance and a PolarDB-X instance with the following specifications:

- The ApsaraDB RDS for MySQL instance has 8 physical database shards, 4 cores, and 16 GB of memory, and supports a maximum number of 4,000 connections.
- The PolarDB-X dedicated instance has 32 cores and 32 GB of memory, in which each PolarDB-X node has 2 cores and 2 GB of memory. This means that this instance has 16 PolarDB-X nodes.

You can use the following formula to calculate the maximum number of connections in the backend connection pool of the PolarDB-X instance:

Maximum number of connections in the backend connection pool of the PolarDB-X instance = FLOOR(4000/8/16) = FLOOR(31.25) = 31

> **Note**
>
> - The calculation result of the preceding formula is the maximum number of connections in the backend connection pool of the PolarDB-X instance. To reduce the connection pressure on the ApsaraDB RDS for MySQL instance, the PolarDB-X instance adjusts the maximum number of connections in the backend connection pool to make it lower than the upper limit.
> - We recommend that you create databases in a PolarDB-X instance on a dedicated ApsaraDB RDS for MySQL instance. Do not create databases for other applications or PolarDB-X instances on the dedicated ApsaraDB RDS for MySQL instance.

Relationship between frontend and backend connections

After an application establishes frontend connections to a PolarDB-X instance and sends requests for SQL statement execution, nodes of the PolarDB-X instance asynchronously process the requests. The nodes obtain backend connections from the internal backend connection pool, and then execute optimized SQL statements on one or more physical database shards.

Nodes of the PolarDB-X instance asynchronously process requests, and frontend connections are not bound to backend connections. Therefore, a small number of backend connections can process a large number of requests for short transactions and simple queries from many concurrent frontend connections. This is why you need to focus on the queries per second (QPS) in PolarDB-X, instead of the number of concurrent connections.

Although the number of frontend connections is considered unlimited, the maximum number of connections maintained in the backend connection pool of a PolarDB-X instance is limited. For more information, see Backend connection. Take note of the following points in actual scenarios:

- Avoid long or large transactions in applications. These transactions occupy many or even all backend connections when they are not committed or rolled back for a long time. This reduces the overall concurrent processing capability and increases the response time (RT).
- Monitor and optimize or remove slow SQL queries in the PolarDB-X instance to prevent them from occupying an excessive number of backend connections. Otherwise, connections in the backend connection pool are not sufficient or the maximum number of backend connections is reached. In this case, the PolarDB-X instance or the ApsaraDB RDS for MySQL instance is under greater processing pressure. This may lead to reduced concurrent processing capability, increased RT, or a higher SQL execution failure rate due to execution timeout. For more information about how to troubleshoot and optimize slow queries, see Details about a low SQL statement and .

- When connections are normally used and queries are normally executed, if the maximum number of connections in the backend connection pool of the PolarDB-X instance is reached, contact Customer Services for assistance.

5.14.7. Upgrade instance specifications

Database performance can be measured by the response time (RT) and queries per second (QPS). RT reflects the performance of a single SQL statement. You can optimize SQL statements to solve this type of performance problem. When you upgrade the specifications of your PolarDB-X instance, the capacity is expanded to improve performance. You can upgrade the specifications for database access business with low latency and high QPS.

The performance of a PolarDB-X instance depends on the performance of PolarDB-X and ApsaraDB RDS for MySQL. Insufficient performance of a PolarDB-X or ApsaraDB RDS for MySQL node can create a bottleneck in the overall performance. This topic describes how to monitor the performance metrics of a PolarDB-X instance and upgrade specifications of the PolarDB-X instance to solve performance bottlenecks. For more information about how to determine the performance of an ApsaraDB RDS for MySQL instance and upgrade the ApsaraDB RDS for MySQL instance, see the ApsaraDB RDS for MySQL documentation.

Determine a performance bottleneck of a PolarDB-X instance

The QPS and CPU performance of a PolarDB-X instance are in positive correlation. If a PolarDB-X instance encounters a performance bottleneck, the CPU utilization of the PolarDB-X instance remains high.

Monitor the CPU utilization

1. Open the **Basic Information** page of your instance. In the left-side navigation pane, choose **Monitoring and Alerts > Instance Monitoring**.
2. On the Instance Monitoring page, select a monitoring dimension and the corresponding metrics to view details.

If the CPU utilization **exceeds 90%** or **remains higher than 80%**, the PolarDB-X instance faces a performance bottleneck. If the ApsaraDB RDS for MySQL instance does not encounter any bottlenecks, the current PolarDB-X instance specifications cannot meet the QPS performance requirements of the business. Then, you can upgrade the specifications of the PolarDB-X instance.

For more information about performance-related service monitoring scenarios and methods of configuring a CPU utilization alert for a PolarDB-X instance, see View monitoring information.

Upgrade the specifications of your PolarDB-X instance

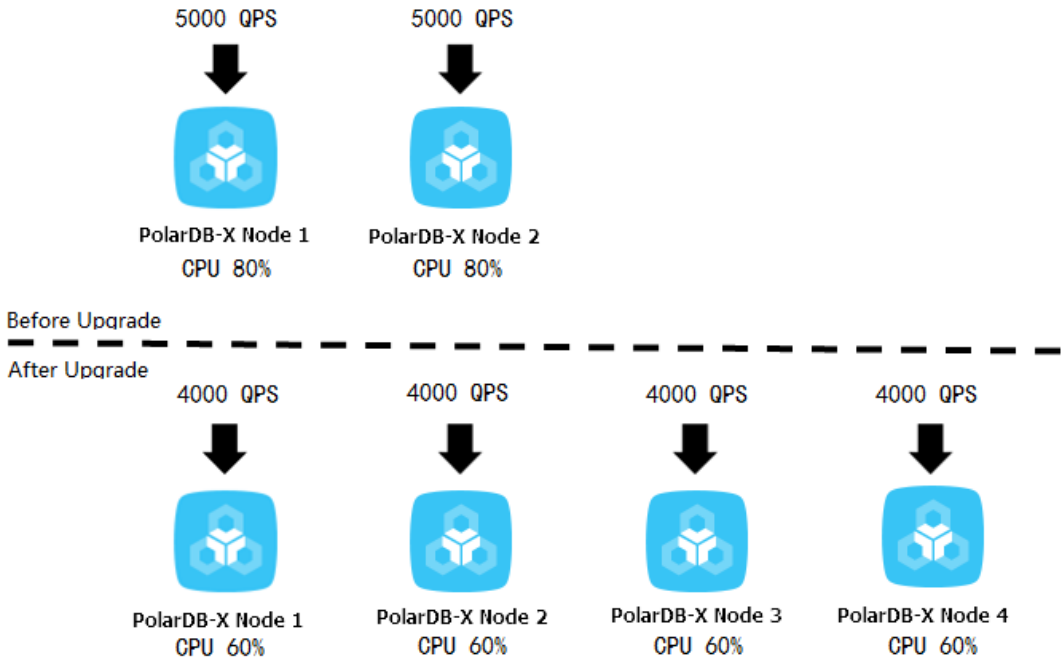
QPS is an important metric to determine whether the PolarDB-X instance specifications can meet the business requirements. Each type of instance specification corresponds to a reference QPS value.

Some special SQL statements require more computing, such as temporary table sorting and aggregate computing, in PolarDB-X. Therefore, the QPS supported by each PolarDB-X instance is lower than the standard value specified in the specifications.

Specifications upgrade of a PolarDB-X instance improves the processing performance of the instance by adding nodes to share the QPS. This upgrade method linearly improves the performance of the PolarDB-X instance, because PolarDB-X nodes are stateless.

For example, Business A requires about 15,000 QPS. The current PolarDB-X instance has 4 vCPUs, 4 GB of memory, and 2 nodes, which support only 10,000 QPS. In addition, the CPU utilization of the PolarDB-X instance remains high. You can upgrade the instance to 8 vCPUs and 8 GB of memory. After the upgrade, each node handles about 4,000 QPS. Then, the performance meets business needs, and the CPU utilization drops to a reasonable level. The following figure shows the procedure.

PolarDB-X specifications upgrade



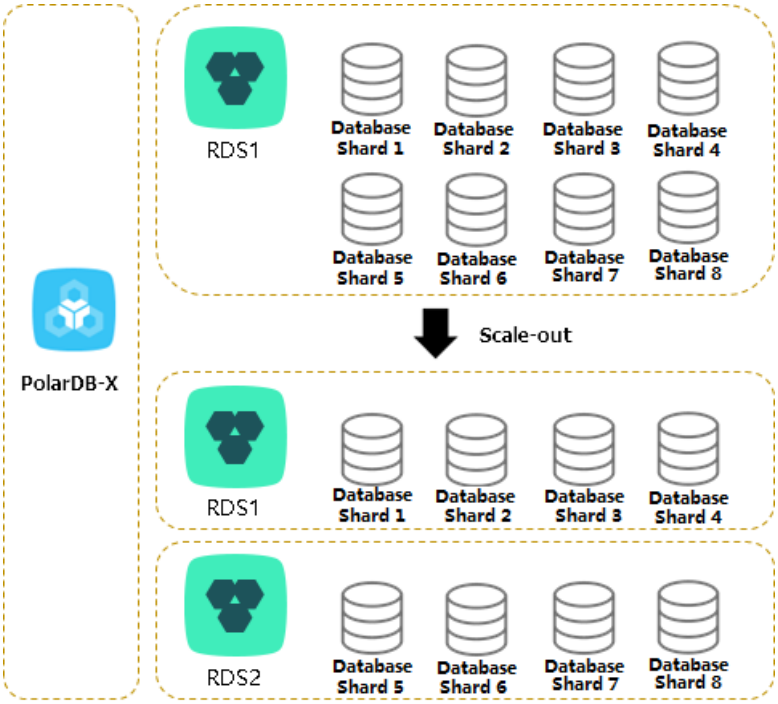
For more information about how to upgrade the specifications of a PolarDB-X instance, see Change specifications.

5.14.8. Perform scale-out

In PolarDB-X, smooth scale-out improves the overall performance by increasing the number of ApsaraDB RDS for MySQL instances. You can increase the number of ApsaraDB RDS for MySQL instances by performing PolarDB-X smooth scale-out to increase the PolarDB-X database capacity when the following conditions are met: 1. The IOPS, CPU utilization, disk space, and other metrics of the ApsaraDB RDS for MySQL instance reach their bottlenecks. 2. You are unable to remove the bottlenecks by optimizing SQL statements or upgrading ApsaraDB RDS for MySQL specifications. For example, the disk has been upgraded to the top configuration.

PolarDB-X smooth scale-out reduces the pressure on the original ApsaraDB RDS for MySQL instance by migrating database shards to the new ApsaraDB RDS for MySQL instance. Assume that all the eight databases are deployed in one ApsaraDB RDS for MySQL instance before scale-out. After scale-out, the eight databases are deployed in two ApsaraDB RDS for MySQL instances. Therefore, the pressure on a single ApsaraDB RDS for MySQL instance is significantly reduced. The following figure shows the procedure.

PolarDB-X scale-out



After multiple scale-out operations, the number of ApsaraDB RDS for MySQL instances may be equal to the number of database shards. In this case, you need to create another PolarDB-X instance and ApsaraDB RDS for MySQL databases with the expected capacity, and migrate data to further increase the data capacity. This procedure is complex. We recommend that you consider the data growth expected in the next two to three years and plan the number of ApsaraDB RDS for MySQL instances when you create a PolarDB-X database.

Determine whether scale-out is required

You can determine whether PolarDB-X smooth scale-out is required based on three ApsaraDB RDS for MySQL metrics: IOPS, CPU utilization, and disk space. You can view these metrics in the ApsaraDB RDS for MySQL console. For more information, see **User Guide > Monitoring and alerting > View monitoring data of system resources and engines** in the *ApsaraDB RDS documentation*.

IOPS and CPU utilization

If you find that the IOPS or CPU utilization remains higher than 80% for a long time or you frequently receive alerts, perform the following steps:

1. Optimize SQL statements. In most cases, you can solve the high CPU utilization problem by using this method. For more information, see Overview.
2. If the problem persists, upgrade the ApsaraDB RDS for MySQL instance. For more information, see **User Guide > Instance management > Change the configurations** in the *ApsaraDB RDS documentation*.
3. When the CPU utilization or IOPS exceeds the threshold, you can set read-only databases to share the load on the primary database. However, read/write splitting affects read consistency. For more information, see the Read/write splitting documentation.
4. If the problem persists, scale out the PolarDB-X instance.

Disk space

ApsaraDB RDS for MySQL has the following types of disk space:

1. Data space: the space occupied by data. The disk usage continues to increase as more data is inserted. We recommend that you keep the remaining disk space higher than 30%.
2. System file space: the space occupied by shared tables and error log files.
3. Binary log file space: the space occupied by binary logs generated during database operation. The more the update transactions, the larger the occupied space.

Whether scale-out is required depends on the data space. When the data space is about to or expected to exceed the disk capacity, you can distribute the data to the database shards on multiple ApsaraDB RDS for MySQL instances by performing scale-out.

Scale-out risks and precautions

PolarDB-X scale-out consists of four steps: **configuration > migration > switchover > cleanup**. For more information, see the Perform smooth scale-out documentation.

Take note of the following points before scale-out:

- To reduce the pressure of read operations on the source ApsaraDB RDS for MySQL instance, perform scale-out when the load on the source ApsaraDB RDS for MySQL instance is low.
- During scale-out, do not submit any data definition language (DDL) tasks in the console or connect to the PolarDB-X instance to directly execute DDL SQL statements. Otherwise, the scale-out task may fail.
- Scale-out requires that the source table have a primary key. If the source table does not have a primary key, add one first.
- During scale-out, the read and write traffic is switched to the new ApsaraDB RDS for MySQL instance. The switchover process takes 3 to 5 minutes. We recommend that you perform a switchover during off-peak hours.
- Scale-out does not affect the PolarDB-X instance before the switchover. Therefore, you can cancel the scale-out by performing a rollback before the switchover.
- The scale-out operation creates pressure on databases. We recommend that you perform this operation during off-peak hours.

5.14.9. Troubleshoot slow SQL statements in PolarDB-X

5.14.9.1. Details about a low SQL statement

PolarDB-X defines an SQL statement that takes more than 1 second to execute as a slow SQL statement. Slow SQL statements in PolarDB-X are classified into logical slow SQL statements and physical slow SQL statements. In PolarDB-X, an SQL statement is executed step by step on PolarDB-X and ApsaraDB RDS for MySQL nodes. Large execution loss on a node will result in slow SQL statements.

- Logical slow SQL statements are slow SQL statements sent from an application to your PolarDB-X instance.
- Physical slow SQL statements are slow SQL statements sent from your PolarDB-X instance to ApsaraDB RDS for MySQL instance.

Syntax

```
SHOW FULL {SLOW | PHYSICAL_SLOW} [WHERE where_condition]
                [ORDER BY col_name [ASC | DESC], ...]
                [LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

Description

The `SHOW FULL SLOW` statement shows logical slow SQL statements, which are SQL statements sent from an application to your PolarDB-X instance.

The result set of the `SHOW FULL SLOW` statement contains the following columns:

- TRACE_ID: the unique identifier of the SQL statement. A logical SQL statement and the physical SQL statements generated by this logical SQL statement have the same TRACE_ID value. The TRACE_ID value is also sent as a comment to your ApsaraDB RDS for MySQL instance.
- HOST: the IP address of the client that sends the SQL statement.

  > **Notice** The client IP address may not be obtained when you use a virtual private cloud (VPC).

- START_TIME: the time when the PolarDB-X instance starts to execute the SQL statement.
- EXECUTE_TIME: the time consumed by the PolarDB-X instance to execute the SQL statement.
- AFFECT_ROW: the number of records returned or the number of rows affected by the SQL statement.
- SQL: the statement that is executed.

The `SHOW FULL PHYSICAL_SLOW` statement shows the physical slow SQL statements, which are SQL statements sent from your PolarDB-X instance to ApsaraDB RDS for MySQL instance.

The result set of the `SHOW FULL PHYSICAL_SLOW` statement contains the following columns:

- TRACE_ID: the unique identifier of the SQL statement. A logical SQL statement and the physical SQL statements generated by this logical SQL statement have the same TRACE_ID value. The TRACE_ID value is also sent as a comment to your ApsaraDB RDS for MySQL instance.
- GROUP_NAME: the name of a database group. Grouping aims to manage multiple groups of databases that have identical data. For example, the databases can be the primary and secondary databases after data replication that is implemented by ApsaraDB RDS for MySQL. It is used for read/write splitting and primary/secondary switchovers.
- DBKEY_NAME: the name of the database shard on which the SQL statement is executed.
- START_TIME: the time when the PolarDB-X instance starts to execute the SQL statement.
- EXECUTE_TIME: the time consumed by the PolarDB-X instance to execute the SQL statement.
- SQL_EXECUTE_TIME: the time consumed by the PolarDB-X instance to call the ApsaraDB RDS for MySQL instance to execute the SQL statement on the database shard.
- GETLOCK_CONNECTION_TIME: the time that the PolarDB-X instance takes to obtain a connection from the connection pool. If the value is large, the ApsaraDB RDS for MySQL connections are exhausted. This is typically due to a large number of slow SQL statements. You can log on to the corresponding ApsaraDB RDS for MySQL instance and execute the SHOW PROCESSLIST statement to locate slow SQL statements.
- CREATE_CONNECTION_TIME: the time that the PolarDB-X instance takes to establish a connection to the ApsaraDB RDS for MySQL instance. If the value is large, it is largely because the ApsaraDB RDS for

MySQL instance is overloaded or faulty.

- AFFECT_ROW: the number of records returned or the number of rows affected by the SQL statement.
- SQL: the statement that is executed.

Example 1

The following example describes how to locate the execution information of a slow SQL statement on a PolarDB-X instance and between PolarDB-X and ApsaraDB RDS for MySQL instances.

1. You can query the specified slow SQL statement by setting conditions such as execution time and SQL string match: Assume that you execute the `SHOW FULL SLOW where `SQL` like '%select sleep(50)%';` statement. The following response is returned:

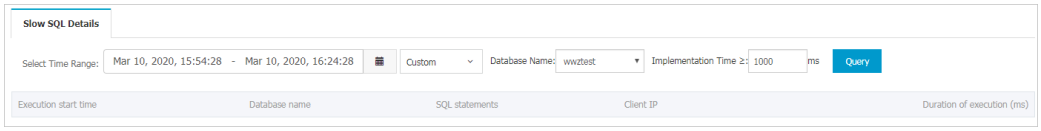
```
+-----------------+-----------+-------------------------+--------------+------------+------------------+
| TRACE_ID        | HOST      | START_TIME              | EXECUTE_TIME | AFFECT_ROW | SQL              |
+-----------------+-----------+-------------------------+--------------+------------+------------------+
| ae0e565b8c00000 | 127.0.0.1 | 2017-03-29 19:28:43.028 |        50009 |          1 | select sleep(50) |
+-----------------+-----------+-------------------------+--------------+------------+------------------+
1 row in set (0.02 sec)
```

2. Based on the value of TRACE_ID that you obtain from the slow SQL statement, execute the `SHOW FULL PHYSICAL_SLOW where trace_id = 'ae0e565b8c00000';` to query the physical execution information of this SQL statement.

```
+-----------------+-------------------------------------------------+-----------------------------------------------+-----------------
-------+--------------+-----------------+-------------------------+-----------------------+------------+------------------+
| TRACE_ID        | GROUP_NAME                                      | DBKEY_NAME                                    | START_TIME              | EXEC
UTE_TIME | SQL_EXECUTE_TIME | GETLOCK_CONNECTION_TIME | CREATE_CONNECTION_TIME | AFFECT_
ROW | SQL              |
+-----------------+-------------------------------------------------+-----------------------------------------------+-----------------
-------+--------------+-----------------+-------------------------+-----------------------+------------+------------------+
| ae0e565b8c00000 | PRIV_TEST_1489167306631PJAFPRIV_TEST_APKK_0000_RDS | rdso6g5b6206sdq83
2ow_priv_test_apkk_0000_nfup | 2017-03-29 19:27:53.02 |      50001 |        50001 |                 0 |
0 |       1 | select sleep(50) |
+-----------------+-------------------------------------------------+-----------------------------------------------+-----------------
-------+--------------+-----------------+-------------------------+-----------------------+------------+------------------+
1 row in set (0.01 sec)
```

3. In the SQL statement details and slow SQL statement records of the ApsaraDB RDS for MySQL instance, you can query the execution information of this SQL statement on the ApsaraDB RDS for MySQL instance based on the value of the TRACE_ID parameter.

   Slow query logs



Example 2

This example describes how to locate the original SQL statement in a PolarDB-X instance based on the slow SQL statement that you locate in the ApsaraDB RDS for MySQL instance.

1. The slow query log generated by the ApsaraDB RDS for MySQL instance shows the TRACE_ID value of the slow SQL query. Assume that the value is ae0e55660c00000.
2. Based on the TRACE_ID value that you obtain in Step 1, execute the `SHOW FULL PHYSICAL_SLOW` statement to obtain the physical execution information of this SQL statement. Execute the `SHOW FULL PHYSICAL_SLOW where trace_id = 'ae0e55660c00000';` statement. The following response is returned:

```
+-----------------+-------------------------------------------------+-----------------------------------------+-----------------
--------+-------------+-----------------+-----------------------+----------------------+------------+-----------------+
| TRACE_ID      | GROUP_NAME                      | DBKEY_NAME                    | START_TIME         | EXE
CUTE_TIME | SQL_EXECUTE_TIME | GETLOCK_CONNECTION_TIME | CREATE_CONNECTION_TIME | AFFECT
_ROW | SQL          |
+-----------------+-------------------------------------------------+-----------------------------------------+-----------------
--------+-------------+-----------------+-----------------------+----------------------+------------+-----------------+
| ae0e55660c00000 | PRIV_TEST_1489167306631PJAFPRIV_TEST_APKK_0000_RDS | rdso6g5b6206sdq83
2ow_priv_test_apkk_0000_nfup | 2017-03-29 19:27:37.308 |      10003 |        10001 |              0 |
0 |       1 | select sleep(10) |
+-----------------+-------------------------------------------------+-----------------------------------------+-----------------
--------+-------------+-----------------+-----------------------+----------------------+------------+-----------------+
1 row in set (0.02 sec)
```

5.14.9.2. Locate slow SQL statements

You can locate a slow SQL statement in two ways. You can obtain historical information about a slow SQL statement from slow SQL statement records. Alternatively, you can execute the `SHOW PROCESSLIST` statement to obtain the real-time execution information about a slow SQL statement.

Perform the following steps to troubleshoot slow SQL statements:

1. Locate slow SQL statements.
2. Locate nodes with performance loss.
3. Troubleshoot the performance loss.

> **Note** During troubleshooting, we recommend that you execute the `mysql -hIP -PPORT -uUSER -pPASSWORD -c` statement in the MySQL command-line client to establish the connection. Be sure to add `-c` to prevent the MySQL command-line client from filtering out the comments (default operation) and therefore affecting the execution of hints.

- View slow SQL statement records

Execute the following statement to query top 10 slow SQL statements. This statement can query logical SQL statements in a PolarDB-X instance. One logical SQL statement corresponds to SQL statements that are executed on one or more databases or tables of the ApsaraDB RDS for MySQL instance. For more information, see Details about a low SQL statement.

Execute the `SHOW SLOW limit 10;` statement. The following response is returned:

```
+-----------------+-------------+------------------------+-------------+------------+-----------------------------------------------------------------
-------------+
| TRACE_ID      | HOST      | START_TIME          | EXECUTE_TIME | AFFECT_ROW | SQL
|
+-----------------+-------------+------------------------+-------------+------------+-----------------------------------------------------------------
-------------+
| ac3133132801001 | xx.xxx.xx.97 | 2017-03-06 15:48:32.330 |     900392 |      -1 | select detail_url, sum(price) f
rom t_item group by detail_url; |
......
+-----------------+-------------+------------------------+-------------+------------+-----------------------------------------------------------------
-------------+
10 rows in set (0.01 sec)
```

- View real-time SQL execution information

  If the execution of an SQL statement is slow in the current server, execute the **SHOW PROCESSLIST** statement to view the real-time SQL execution information in the current PolarDB-X database. The value in the TIME column indicates how long the current SQL statement has been executed.

  Execute the `SHOW PROCESSLIST WHERE COMMAND !='Sleep';` statement. The following response is returned:

```
+---------------+-----------+-------------------+-------------+---------+-----------------------------------------------------------------------------------------------------------------------+-----------+---------------+-----------+
| ID        | USER    | DB          | COMMAND    | TIME   | STATE                                 | INFO                                                                                                                  | ROWS_SENT | ROWS_EXAMINED | ROWS_READ |
+---------------+-----------+-------------------+-------------+---------+-----------------------------------------------------------------------------------------------------------------------+-----------+---------------+-----------+
| 0-0-352724126 | ifisibhk0 | test_123_wvvp_0000 | Query     |   13 | Sending data                          | /*DRDS /42.120.74.88/ac47e5a72801000/ */select `t_item`.`detail_url`,SUM(`t_item`.`price`) from `t_i |      NULL |          NULL |      NULL |
| 0-0-352864311 | cowxhthg0 | NULL          | Binlog Dump |   17 | Master has sent all binlog to slave; waiting for binlog to be updated | NULL                                                                                                                  |      NULL |          NULL |      NULL |
| 0-0-402714795 | ifisibhk0 | test_123_wvvp_0005 | Alter     |  114 | Sending data                          | /*DRDS /42.120.74.88/ac47e5a72801000/ */ALTER TABLE `Persons` ADD `Birthday` date                             |      NULL |          NULL |      NULL |
......
+---------------+-----------+-------------------+-------------+---------+-----------------------------------------------------------------------------------------------------------------------+-----------+---------------+-----------+
12 rows in set (0.03 sec)
```

The following list describes each column:

- ID: the ID of the connection.
- USER: the username of the database shard on which this SQL statement is executed.
- DB: the specified database. If no database is specified, the value is NULL.
- COMMAND: the type of the statement that is being executed. SLEEP: indicates an idle connection. For more information about other details, see MySQL thread information documentation.
- TIME: the elapsed execution time of the SQL statement, in seconds.
- STATE: the current execution status. For more information, see MySQL thread status documentation.
- INFO: the information used to derive the complete SQL statement. The SQL statement that is being executed may be so long that it cannot be completely displayed. You can derive the complete SQL statement based on information such as business parameters.

In this example, the following slow SQL statement is located:

```
ALTER TABLE `Persons` ADD `Birthday` date
```

5.14.9.3. Locate nodes with performance loss

When you locate a slow SQL statement in slow SQL statement records or real-time SQL execution information, you can run the TRACE command to trace the running time of the SQL statement in PolarDB-X and ApsaraDB RDS for MySQL to locate the bottleneck.

The TRACE command actually runs the SQL statement, records the time consumed on all nodes, and returns the execution result. For more information about TRACE and other control commands, see Help statements.

> **Note** The PolarDB-X TRACE command needs to maintain the context information of the connection. Some GUI clients may use connection pools, which results in command exceptions. Therefore, we recommend that you use the MySQL command line to run the TRACE command.

Run the following command for the identified slow SQL statement:

```
mysql> trace select detail_url, sum(distinct price) from t_item group by detail_url;
  +---------------+---------------+
  | detail_url    | sum(price)    |
  +---------------+---------------+
  | www.xxx.com   | 1084326800.00 |
  | www.xx1.com   | 1084326800.00 |
  | www.xx2.com   | 1084326800.00 |
  | www.xx3.com   | 1084326800.00 |
  | www.xx4.com   | 1084326800.00 |
  | www.xx5.com   | 1084326800.00 |
  ......
  +---------------+---------------+
  1 row in set (7 min 2.72 sec)
```

After the TRACE command is run, run SHOW TRACE to view the result. You can identify the bottleneck of the slow SQL statement based on the time consumption of each component.

```
 mysql> SHOW TRACE;
  +------+------------+-------------+-------------------------------------------------------------+-----------------------------------------
+---------------+-------------------------+------+-----------------------------------------------------------------------------------------------
-------------------------------------------------+--------+
  | ID  | TIMESTAMP | TYPE      | GROUP_NAME                      | DBKEY_NAME                  | TIME_COS
T(MS) | CONNECTION_TIME_COST(MS) | ROWS | STATEMENT
| PARAMS |
  +------+------------+-------------+-------------------------------------------------------------+-----------------------------------------
+---------------+-------------------------+------+-----------------------------------------------------------------------------------------------
-------------------------------------------------+--------+
  |  0 |    0.000 | Optimize   | DRDS                            | DRDS                        | 2       | 0.00            |  0
| select detail_url, sum(price) from t_item group by detail_url                                                                                 | NULL   |
  |  1 | 423507.342 | Merge Sorted | DRDS                          | DRDS                        | 411307  | 0.00
|  8 | Using Merge Sorted Order By (`t_item`.`detail_url` asc)                                                                                       | NULL
```

```
| 0 | Using Merge Sorted, Order By (`t_item`.`detail_url` asc )                                    |NULL
|
  | 2 |    2.378 | Query     | TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0003_RDS | rdso6g5b
6206sdq832ow_test_123_wvvp_0003_hbpz | 15       | 1.59             |   1 | select `t_item`.`detail_url`,SUM(d
istinct `t_item`.`price`) from `t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url` asc
| NULL  |
  | 3 |    2.731 | Query     | TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | rdso6g5b
6206sdq832ow_test_123_wvvp_0000_hbpz | 11       | 1.78             |   1 | select `t_item`.`detail_url`,SUM(d
istinct `t_item`.`price`) from `t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url` asc
| NULL  |
  | 4 |    2.933 | Query     | TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0004_RDS | rdso6g5b
6206sdq832ow_test_123_wvvp_0004_hbpz | 15       | 1.48             |   1 | select `t_item`.`detail_url`,SUM(d
istinct `t_item`.`price`) from `t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url` asc
| NULL  |
  | 5 |    3.111 | Query     | TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001_RDS | rdso6g5b
6206sdq832ow_test_123_wvvp_0001_hbpz | 15       | 1.56             |   1 | select `t_item`.`detail_url`,SUM(d
istinct `t_item`.`price`) from `t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url` asc
| NULL  |
  | 6 |    3.323 | Query     | TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS | rdso6g5b
6206sdq832ow_test_123_wvvp_0007_hbpz | 15       | 1.54             |   1 | select `t_item`.`detail_url`,SUM(d
istinct `t_item`.`price`) from `t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url` asc
| NULL  |
  | 7 |    3.496 | Query     | TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0006_RDS | rdso6g5b
6206sdq832ow_test_123_wvvp_0006_hbpz | 18       | 1.30             |   1 | select `t_item`.`detail_url`,SUM(d
istinct `t_item`.`price`) from `t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url` asc
| NULL  |
  | 8 |    3.505 | Query     | TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005_RDS | rdso6g5b
6206sdq832ow_test_123_wvvp_0005_hbpz | 423507   | 1.97             |   1 | select `t_item`.`detail_url`,SU
M(distinct `t_item`.`price`) from `t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url`
asc | NULL  |
  | 9 |    3.686 | Query     | TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0002_RDS | rdso6g5b
6206sdq832ow_test_123_wvvp_0002_hbpz | 14       | 1.47             |   1 | select `t_item`.`detail_url`,SUM(d
istinct `t_item`.`price`) from `t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url` asc
| NULL  |
  | 10 | 423807.906 | Aggregate   | DRDS                                    | DRDS                    | 1413     | 0.00
|   1 | Aggregate Function (SUM(`t_item`.`price`)), Group By (`t_item`.`detail_url` asc )
| NULL  |
  +------+-----------+-------------+--------------------------------------------------------+--------------------------------------------
+--------------+------------------------+------+-------------------------------------------------------------------------------------------
-----------------------------------------------+--------+
  11 rows in set (0.01 sec)
```

In the returned results of SHOW TRACE, you can determine which node has a long execution time based on the values (in milliseconds) in the TIME_COST column. You can also see the corresponding GROUP_NAME (that is, the PolarDB-X or ApsaraDB RDS for MySQL node) and the STATEMENT column information (that is, the SQL statement being executed). By checking whether the value of GROUP_NAME is PolarDB-X, you can determine whether the slow node exists in PolarDB-X or ApsaraDB RDS for MySQL.

According to the preceding results, the Merge Sorted action on the PolarDB-X node and the TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005_RDS node of ApsaraDB RDS for MySQL take a lot of time.

5.14.9.4. Troubleshoot the performance loss

Slow nodes may exist on the PolarDB-X or ApsaraDB RDS for MySQL instance. Troubleshoot the fault accordingly after the cause is determined.

Solution for slow PolarDB-X nodes

When the `GROUP_NAME` of a slow node is in the PolarDB-X instance, check whether time-consuming computing operations such as Merge Sorted, Temp Table Merge, and Aggregate exist during SQL statement execution. If so, rectify it. For more information, see Overview.

Solution for slow ApsaraDB RDS for MySQL nodes

When the slow node is on the ApsaraDB RDS for MySQL instance, check the execution plan of this SQL statement on the ApsaraDB RDS for MySQL instance.

In PolarDB-X, you can run `/! TDDL:node={GROUP_NAME}*/ EXPLAIN` to check the SQL execution plan of an ApsaraDB RDS for MySQL instance. The execution plan displays the SQL execution process information, including inter-table association and index information.

The detailed process is as follows:

1. Based on `GROUP_NAME`, assemble the HINT: `/! TDDL:node='TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005_RDS'*/`.
2. Combine the assembled HINT and the statement prefixed by EXPLAIN to form a new SQL statement and run it. The EXPLAIN command does not actually run. It only displays the execution plan of the SQL statement.

The following example describes how to query the execution plan of the identified slow node.

```
mysql> /! TDDL:node='TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005_RDS'*/ EXPLAIN s
elect `t_item`.`detail_url`,SUM(distinct `t_item`.`price`) from `t_item` group by `t_item`.`detail_url`
order by `t_item`.`detail_url` asc;
+----+-------------+--------+------+---------------+-----+---------+-----+---------+----------------------------------+
| id | select_type | table  | type | possible_keys | key | key_len | ref | rows    | Extra                            |
+----+-------------+--------+------+---------------+-----+---------+-----+---------+----------------------------------+
| 1 | SIMPLE     | t_item | ALL  | NULL          | NULL | NULL    | NULL | 1322263 | Using temporary; Using filesort |
+----+-------------+--------+------+---------------+-----+---------+-----+---------+----------------------------------+
1 row in set (0.01 sec)
```

When the preceding SQL statement is run in ApsaraDB RDS for MySQL, the message `Using temporary; Using filesort` is returned. It indicates that low SQL statement execution is caused by improper use of the index. In this case, you can correct the index and run the SQL statement again.

5.14.10. Handle DDL exceptions

When you run any data definition language (DDL) commands of PolarDB-X, PolarDB-X performs the corresponding DDL operation on all table shards.

Failures can be divided into two types:

1. A DDL statement fails to be executed in a database shard. DDL execution failure in any database shard may result in inconsistent table shard structures.
2. The system does not respond for a long time after a DDL statement is executed. When you perform a DDL statement on a large table, the system may make no response for a long time due to the long execution time of the DDL statement in a database shard.

Execution failures in database shards may occur for various reasons. For example, the table you want to create already exists, the column you want to add already exists, or the disk space is insufficient.

No response for a long time is generally caused by the long execution time of a DDL statement in a database shard. Taking ApsaraDB RDS for MySQL as an example, the DDL execution time depends mostly on whether the operation is an in-place (directly modifying the source table) or copy (copying data in the table) operation. An in-place operation only requires modification of metadata, while a copy operation reconstructs the whole table and also involves log and buffer operations.

To determine whether a DDL operation is an in-place or copy operation, you can view the returned value of "rows affected" after the operation is completed.

Example:

- Change the default value of a column (this operation is very fast and does not affect the table data at all):

  ```
  Query OK, 0 rows affected (0.07 sec)
  ```

- Add an index (this operation takes some time, but "0 rows affected" indicates that the table data is not replicated):

  ```
  Query OK, 0 rows affected (21.42 sec)
  ```

- Change the data type of column (this operation takes a long time and reconstructs all data rows in the table):

  ```
  Query OK, 1671168 rows affected (1 min 35.54 sec)
  ```

Therefore, before executing a DDL operation on a large table, perform the following steps to determine whether the operation is a fast or slow operation:

1. Copy the table structure to generate a cloned table.
2. Insert some data.
3. Perform the DDL operation on the cloned table.
4. Check whether the value of "rows affected" is 0 after the operation is completed. A non-zero value means that this operation reconstructs the entire table. In this case, you need to perform this

operation in off-peak hours.

Solution for failures

PolarDB-X DDL operations distribute all SQL statements to all database shards for parallel execution. Execution failure on any database shard does not affect the execution on other database shards. In addition, PolarDB-X provides the CHECK TABLE command to check the structure consistency of the table shards. Therefore, failed DDL operations can be performed again, and errors reported on database shards on which the operations have been executed do not affect the execution on other database shards. Make sure that all table shards ultimately have the same structure.

Procedure for handling DDL operation failures

1. Run the **CHECK TABLE** command to check the table structure. If the returned result contains only one row and the status is normal, **the table statuses are consistent**. In this case, go to Step 2. Otherwise, go to Step 3.
2. Run the **SHOW CREATE TABLE** command to check the table structure. If the displayed table structure is the same as the expected structure after the DDL statement is run, the DDL statement is run. Otherwise, go to Step 3.
3. Run the **SHOW PROCESSLIST** command to check the statuses of all SQL statements being executed. If any ongoing DDL operations are detected, wait until these operations are completed, and then perform Steps 1 and 2 to check the table structure. Otherwise, go to Step 4.
4. Perform the DDL operation again on PolarDB-X. If the **Lock conflict** error is reported, go to Step 5. Otherwise, go to Step 3.
5. Run the **RELEASE DBLOCK** command to release the DDL operation lock, and then go to Step 4.

The procedure is as follows:

1. Check the table structure consistency

   Run the CHECK TABLE command to check the table structure. When the returned result contains only one row and the displayed status is OK, **the table structures are consistent**.

   > **Notice** If no result is returned after you run CHECK TABLE, retry by using the CLI.

   ```
   mysql> check table `xxxx`;
   +----------------------------+-------+----------+----------+
   | TABLE                      | OP    | MSG_TYPE | MSG_TEXT |
   +----------------------------+-------+----------+----------+
   | TDDL5_APP.xxxx             | check | status   | OK       |
   +----------------------------+-------+----------+----------+
   1 row in set (0.05 sec)
   ```

2. Check the table structure

   Run the SHOW CREATE TABLE command to check the table structure. If **table structures are consistent** and **correct**, the DDL statement has been run.

```
mysql> show create table `xxxx`;
+---------+-----------------------------------------------------------------------------------------------------------------------------------+
| Table   | Create Table                                                                                                                      |
+---------+-----------------------------------------------------------------------------------------------------------------------------------+
| xxxx    | CREATE TABLE `xxxx` (
`id` int(11) NOT NULL DEFAULT '0',
`NAME` varchar(1024) NOT NULL DEFAULT '',
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by hash(`id`) tbpartition by hash(`id`) tbpartiti
ons 3                |
+---------+-----------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.05 sec)
```

3. Check the SQL statements being executed.

   If some DDL statement executions are slow and no response is received for a long time, you can run the SHOW PROCESSLIST command to check the status of all SQL statements being executed.

```
mysql> SHOW PROCESSLIST WHERE COMMAND ! = 'Sleep';
+---------------+-----------+-------------------+-------------+---------+-----------------------------------------------------------------------------------------------------------------------------------+-----------+-------------
---------+-----------+-------------+
| ID        | USER    | DB           | COMMAND    | TIME   | STATE                                 | INFO
| ROWS_SENT | ROWS_EXAMINED | ROWS_READ |
+---------------+-----------+-------------------+-------------+---------+-----------------------------------------------------------------------------------------------------------------------------------+-----------+-------------
---------+-----------+-------------+
| 0-0-352724126 | ifisibhk0 | test_123_wvvp_0000 | Query     |    15 | Sending data
| /*DRDS /xx.xxx.xx.88/ac47e5a72801000/ */select `t_item`.`detail_url`,SUM(`t_item`.`price`) from `t
_i |    NULL |       NULL |    NULL |
| 0-0-352864311 | cowxhthg0 | NULL           | Binlog Dump |    13 | Master has sent all binlog to slave; waiti
ng for binlog to be updated | NULL                                                                                                                                |    NULL |       NULL |    NU
LL |
| 0-0-402714566 | ifisibhk0 | test_123_wvvp_0005 | Query     |    14 | Sending data
| /*DRDS /xx.xxx.xx.88/ac47e5a72801000/ */select `t_item`.`detail_url`,`t_item`.`price` from `t_i    |
NULL |       NULL |    NULL |
| 0-0-402714795 | ifisibhk0 | test_123_wvvp_0005 | Alter     |    114 | Sending data
| /*DRDS /xx.xxx.xx.88/ac47e5a72801000/ */ALTER TABLE `Persons` ADD `Birthday` date             |    NU
LL |       NULL |    NULL |
......
+---------------+-----------+-------------------+-------------+---------+-----------------------------------------------------------------------------------------------------------------------------------+-----------+-------------
---------+-----------+-------------+
12 rows in set (0.03 sec)
```

The value in the TIME column indicates the number of seconds that the command has been executed. If a command execution is too slow, as shown in the figure, you can run the **KILL '0-0-402714795'** command to cancel the slow command.

> **Notice** In PolarDB-X, one logical SQL statement corresponds to multiple statements on database shards. Therefore, you may need to kill multiple commands to stop a logical DDL statement. You can determine the logical SQL statement to which a command belongs based on the INFO column in the SHOW PROCESSLIST result set.

4. Handle the lock conflict error

   PolarDB-X adds a database lock before performing a DDL operation and releases the lock after the operation. The KILL DDL operation may not release the lock. If you perform the DDL operation again, the following error message will be returned:

```
Lock conflict , maybe last DDL is still running
```

In this case, run **RELEASE DBLOCK** to release the lock. After the command is canceled and the lock is released, run the DDL statement again during off-peak hours or when the service is stopped.

Other problems

Clients cannot display the modified table structures.

To enable some clients to obtain table structures from system tables (such as COLUMNS or TABLES), PolarDB-X creates a shadow database in database shard 0 on your ApsaraDB RDS for MySQL instance. The shadow database name must be the same as the name of your PolarDB-X logical database. It stores all table structures and other information in the user database.

The client obtains the PolarDB-X table structure from the system table of the shadow database. During the processing of DDL exceptions, the table structure may be modified normally in the user database but not in the shadow database due to some reasons. In this case, you need to connect to the shadow database and perform the DDL operation on the table again in the database.

> **Notice** The CHECK TABLE command does not check whether the table structure in the shadow database is consistent with that in the user database.

5.14.11. Efficiently scan PolarDB-X data

PolarDB-X supports efficient data scanning and uses aggregate functions for statistical summary during full table scan.

The following describes common scanning scenarios:

- **Scan of tables without database or table shards**: PolarDB-X transmits the original SQL statement to the backend ApsaraDB RDS for MySQL database for execution. In this case, PolarDB-X supports any aggregate functions.
- **Non-full table scan**: PolarDB-X transmits the original SQL statement to each single ApsaraDB RDS for MySQL database for execution. For example, when the shard key in the WHERE clause is Equal, non-full table scan is performed. In this case, PolarDB-X also supports any aggregate functions.
- **Full table scan**: Currently, the supported aggregate functions are COUNT, MAX, MIN, and SUM. In addition, LIKE, ORDER BY, LIMIT, and GROUP BY are also supported during full table scan.
- **Parallel scan of all table shards**: If you need to export data from all databases, you can run the SHOW command to view the table topology and scan all table shards in parallel. For more information, see the following section.

Traverse tables by using a hint

1. Run the SHOW TOPOLOGY FROM TABLE_NAME command to obtain the table topology.

```
mysql:> SHOW TOPOLOGY FROM DRDS_USERS;
+------+------------------+-------------+
| ID   | GROUP_NAME       | TABLE_NAME  |
+------+------------------+-------------+
|  0   | DRDS_00_RDS      | drds_users  |
|  1   | DRDS_01_RDS      | drds_users  |
+------+------------------+-------------+
2 rows in set (0.06 sec)
```

By default, the non-partition table is stored in database shard 0.

2. Traverse each table for TOPOLOGY.
   i. Run the current SQL statement in database shard 0.

   ```
   /! TDDL:node='DRDS_00_RDS'*/ SELECT * FROM DRDS_USERS;
   ```

   ii. Run the current SQL statement in database shard 1.

   ```
   /! TDDL:node='DRDS_01_RDS'*/ SELECT * FROM DRDS_USERS;
   ```

> **Notice** We recommend that you run `SHOW TOPOLOGY FROM TABLE_NAME` to obtain the latest table topology before each scan.

Scan data in parallel

PolarDB-X allows you to run mysqldump to export data. However, if you want to scan data faster, you can establish multiple sessions for each table shard to scan tables in parallel.

```
mysql> SHOW TOPOLOGY FROM LJLTEST;
+------+----------------+------------+
| ID   | GROUP_NAME     | TABLE_NAME |
+------+----------------+------------+
|  0 | TDDL5_00_GROUP | ljltest_00 |
|  1 | TDDL5_00_GROUP | ljltest_01 |
|  2 | TDDL5_00_GROUP | ljltest_02 |
|  3 | TDDL5_01_GROUP | ljltest_03 |
|  4 | TDDL5_01_GROUP | ljltest_04 |
|  5 | TDDL5_01_GROUP | ljltest_05 |
|  6 | TDDL5_02_GROUP | ljltest_06 |
|  7 | TDDL5_02_GROUP | ljltest_07 |
|  8 | TDDL5_02_GROUP | ljltest_08 |
|  9 | TDDL5_03_GROUP | ljltest_09 |
| 10 | TDDL5_03_GROUP | ljltest_10 |
| 11 | TDDL5_03_GROUP | ljltest_11 |
+------+----------------+------------+
12 rows in set (0.06 sec)
```

As shown above, the table has four database shards, and each database shard has three table shards. Run the following SQL statement to operate on the table shards of the TDDL5_00_GROUP database:

```
/! TDDL:node='TDDL5_00_GROUP'*/ select * from ljltest_00;
```

> **Note** TDDL5_00_GROUP in HINT corresponds to the GROUP_NAME column in the execution results of the SHOW TOPOLOGY command. In addition, the table name in the SQL statement is the table shard name.

At this time, you can establish up to 12 sessions (corresponding to 12 table shards respectively) to process data in parallel.

5.15. Appendix: PolarDB-X terms

This topic lists common terms of PolarDB-X for your reference.

Term	Description	Remarks
Cloud Native Distributed Database PolarDB-X	PolarDB-X is a distributed database service that was independently developed by Alibaba to solve the bottlenecks of single-instance database services. PolarDB-X is compatible with MySQL protocols and syntax. It supports automatic sharding, smooth scale-out, auto scaling, and transparent read/write splitting, and provides O&M capabilities for distributed databases throughout their entire lifecycle.	-

Term	Description	Remarks
TDDL	Taobao Distributed Data Layer (TDDL) was developed by Alibaba and has become a preferred component for nearly 1,000 core applications of Alibaba.	-
PolarDB-X Console	PolarDB-X Console is designed for database administrators (DBAs) to isolate resources as required and perform operations, such as instance management, database and table management, read/write splitting configuration, smooth scale-out, monitoring data display, and IP address whitelist.	-
PolarDB-X Manager	PolarDB-X Manager is designed for global O&M personnel and DBAs to manage all PolarDB-X resources and monitor the system.	-
PolarDB-X Server	PolarDB-X Server is the service layer of PolarDB-X. Multiple server nodes make up a server cluster to provide distributed database services, including the read/write splitting, routed SQL execution, result merging, dynamic database configuration, and globally unique ID (GUID).	-
Load balancer	PolarDB-X server nodes are stateless, and therefore requests can be randomly routed to any PolarDB-X server node. The load balancer is used to complete this task. Server Load Balancer (SLB) is used for overall output by Apsara Stack. VIPServer is typically used for Alibaba middleware output.	-
Diamond	Diamond manages the configuration and storage of PolarDB-X. It provides the configuration functions for storage, query, and notification. In PolarDB-X, Diamond stores the source data of databases, and configuration data including the sharding rules, and switches.	-
Data Replication System	Data Replication System migrates and synchronizes data for PolarDB-X. Its core capabilities include full data migration and incremental data synchronization. Its derived features include smooth data import, smooth scale-out, and global secondary index. Data Replication System requires the support of ZooKeeper and PolarDB-X Rtools.	-
PolarDB-X instance (PolarDB-X instance)	A PolarDB-X instance consists of multiple PolarDB-X server nodes. A PolarDB-X instance can contain multiple PolarDB-X databases.	-
PolarDB-X instance ID (PolarDB-X instance ID)	An instance ID uniquely identifies an PolarDB-X instance.	-
Number of nodes on a PolarDB-X instance	The number of PolarDB-X server nodes in a PolarDB-X instance.	-

Term	Description	Remarks
VIP	The virtual IP addresses (VIPs) of the load balancer can be classified as: • 1. Public VIP, which is accessible from the Internet. It is used for testing. • 2. Private VIP, which is accessible only from the Alibaba Cloud internal network.	-
VPC	Virtual Private Cloud (VPC) is generally used on Alibaba Cloud.	-
Region	A region is a geographical location, such as East China. This concept is generally used for Alibaba Cloud.	-
Azone	A physical area with independent power grids and networks within one region, such as Hangzhou Zone A. This concept is generally used for Alibaba Cloud.	-
Logical SQL statement	A logical SQL statement is an SQL statement sent from an application to PolarDB-X.	-
Physical SQL statement	A physical SQL statement is an SQL statement obtained after PolarDB-X parses a logical SQL statement and sends it to ApsaraDB RDS for MySQL for execution.	Logical SQL statements and physical SQL statements may be the same or different. Logical and physical SQL statements may be in a one-to-one or one-to-many mapping.
QPS	The queries per second (QPS) is the average number of logical SQL statements executed by PolarDB-X per second in a statistical period,	instead of the number of transactions. Most control statements, such as COMMIT and SET, are not counted in QPS.
RT	The response time (RT) is the average response time (in milliseconds) of logical SQL statements executed by PolarDB-X in a statistical period. The RT of an SQL statement is calculated as follows: (Time when PolarDB-X writes the last packet of the result set) - (Time when PolarDB-X receives the SQL statement)	-
Physical QPS	The physical QPS is the average number of physical SQL statements that PolarDB-X executes on ApsaraDB RDS for MySQL per second in a statistical period.	-

Term	Description	Remarks
Physical RT	The physical RT is the average response time (in milliseconds) of physical SQL statements executed by PolarDB-X on ApsaraDB RDS for MySQL in a statistical period. The RT of a physical SQL statement is calculated as follows: (Time when PolarDB-X receives the result set returned by ApsaraDB RDS for MySQL) - (Time when PolarDB-X starts to obtain the connection to ApsaraDB RDS for MySQL)	This includes the time of establishing a connection to ApsaraDB RDS for MySQL or obtaining a connection from the connection pool, the network transmission time, and the time of executing the SQL statement by ApsaraDB RDS for MySQL.
Connections	The number of connections established between the application and PolarDB-X,	instead of the number of connections established between PolarDB-X and ApsaraDB RDS for MySQL.
Inbound traffic	The network traffic generated when the application sends SQL statements to PolarDB-X.	This traffic is irrelevant to the traffic used for interaction between PolarDB-X and ApsaraDB RDS for MySQL.
Outbound traffic	The network traffic generated when PolarDB-X sends the result set to the application.	This traffic is irrelevant to the traffic used for interaction between PolarDB-X and ApsaraDB RDS for MySQL.
Number of active threads (ThreadRunning)	The number of threads running on a PolarDB-X instance. This parameter can be used to indicate the load of the PolarDB-X instance.	-
Global	The total monitoring data of all databases on a PolarDB-X instance.	-
Memory usage	The Java Virtual Machine (JVM) memory usage of a PolarDB-X server process.	-

Term	Description	Remarks
Total memory usage	The memory usage of the machine where the PolarDB-X server node is located.	This metric is available only when PolarDB-X servers are deployed on ECS instances. Generally, this metric is used for Alibaba Cloud.
CPU utilization	The CPU utilization of the machine where a PolarDB-X server node is located.	This metric is available only when PolarDB-X servers are deployed on ECS instances. Generally, this metric is used for Alibaba Cloud.
System load	The load of the machine where a PolarDB-X server node is located.	This metric is available only when PolarDB-X servers are deployed on ECS instances. Generally, this metric is used for Alibaba Cloud.
Service port	The port used by PolarDB-X servers to provide MySQL-based services to external applications.	Generally, the port number is 3306. However, when multiple PolarDB-X nodes (mostly physical machines) are deployed on one machine, the port number will change accordingly.
Management port	The port used by PolarDB-X servers to provide management application program interfaces (APIs).	Generally, the port number is the service port number plus 100.
Start time	The time when PolarDB-X servers start.	-
Running time	The continuous running time of the PolarDB-X servers since the last startup time.	-
Total memory size	The maximum JVM memory size of a PolarDB-X server node.	-

Term	Description	Remarks
Memory usage	The JVM memory that is already used by the PolarDB-X server nodes.	-
Number of nodes	Required. The number of machines. A PolarDB-X instance is essentially a PolarDB-X cluster, and the number of nodes refers to the number of machines in the cluster.	-
Instance type	Required. The type of the instance, including dedicated and shared instances. A dedicated instance works in the exclusive mode. A shared instance works in the multi-tenant mode, which is generally used in Alibaba Cloud.	-
Machine type	Required. The type of the machine where a PolarDB-X server node is deployed. Valid values are Auto-selected, PHY, and ECS. The PolarDB-X inventory is divided into physical machine inventory and virtual machine inventory according to the type of machines where the PolarDB-X servers are deployed. The two types cannot be mixed because their deployment and O&M methods are different.	-
AliUid	Required. The UID of the instance. In Apsara Stack, this ID is provided by the account system in the deployment environment.	-
Backend port	The backend port of the VIP. For a PolarDB-X server node, this port is the service port of machine where the PolarDB-X server node is deployed.	-
Frontend port	The frontend port of the VIP for user access. Each VIP has a set of frontend ports and backend ports. The VIP forwards data from frontend ports to backend ports.	-
Private network/Internet	The network type of the VIP. Valid values: • Internet: the public VIP, which is accessible from the Internet. • Private network: the private VIP (including VPC VIP), which is accessible from private networks.	-
lbId	The ID of an SLB instance, which is the unique ID of VIP. A VIP is managed based on this ID.	-
Forwarding mode	The port forwarding mode of the VIP. The following modes are supported: • FNAT: This mode is recommended when the backend machine is a virtual machine or VPC needs to be supported. • NAT: This mode can be selected when the backend machine is a physical machine. Currently, this mode is only used on Alibaba Cloud. • Open FNAT: This mode is applicable only to Alibaba Cloud.	-
VPC ID	The ID of the destination VPC, that is, the VPC to be accessed.	-

Term	Description	Remarks
VSwitch ID	The ID of the destination VSwitch, which determines the CIDR block where the VPC VIP of the instance is in.	-
APPName	The app name of the destination PolarDB-X database. Each PolarDB-X database has a corresponding app name for loading configurations.	-
UserName	The user name used to log on to the destination PolarDB-X database.	-
DBName	The name of the destination PolarDB-X database you want to log on to.	-
IP address whitelist	Only the IP addresses specified in the IP address whitelist can access the PolarDB-X instance.	-
Read-only instance	ApsaraDB RDS for MySQL instances where physical databases reside are divided into the following two types based on whether data can be written into the instances: • Primary instance: Both read and write requests are allowed on such an instance. In Apsara Stack, ApsaraDB RDS for MySQL is supported. In Alibaba Cloud, ApsaraDB for RDS is supported. • Read-only instance: Only read requests are allowed on such an instance. In Apsara Stack, ApsaraDB RDS for MySQL is supported. In Alibaba Cloud, ApsaraDB for RDS is supported.	-
Read SQL statement	A type of SQL statements used to read data, such as the SELECT statement. PolarDB-X determines whether an SQL statement is a read-only SQL statement when it is not in a transaction. If the SQL statement is in a transaction, PolarDB-X treats it as a write SQL statement during read/write splitting.	-
Read/write splitting	If read-only ApsaraDB RDS for MySQL instances exist, you can configure in the PolarDB-X console to allocate read SQL statements to the primary and read-only instances proportionally. PolarDB-X automatically identifies the type of SQL statements and allocates them proportionally.	-
Smooth scale-out	On the basis of horizontal partitioning, the data distribution on ApsaraDB RDS for MySQL instances is dynamically adjusted for scale-out. Generally, scale-out is completed asynchronously without any modification to the business code.	-
Broadcast of small tables	You can synchronize the data in a single table in a database to all database shards in advance, to convert the cross-database JOIN query into a JOIN query that can be completed on physical databases.	-

Term	Description	Remarks
Horizontal partitioning	Horizontal partitioning distributes the data rows originally stored in one table to multiple tables based on specified rules to achieve horizontal linear scaling.	-
Partition mode	This mode allows you to create multiple database shards on an ApsaraDB RDS for MySQL instance. These database shards make up a PolarDB-X database. In this mode, all PolarDB-X functions can be used.	-
Non-partition mode	In this mode, a database that has been created on an ApsaraDB RDS for MySQL instance is used as a PolarDB-X database. In this mode, only PolarDB-X read/write splitting is allowed, while other PolarDB-X features such as database sharding and table sharding are not allowed.	-
Imported database	An existing database on the ApsaraDB RDS for MySQL instance selected for creating a PolarDB-X database. This is a unique concept for the creation of a PolarDB-X database.	-
Read policy	The ratio of read SQL statements assigned by PolarDB-X to the primary and read-only ApsaraDB RDS for MySQL instances.	-
Full table scan	If no shard field is specified in a SQL statement, PolarDB-X runs the SQL statement on all table shards and summarizes the results. You can disable this function because of its high overheads.	-
Shard key	A column in a logical table. PolarDB-X routes data and SQL statements to a physical table based on this column.	-
Data import	The operation of importing data from an existing ApsaraDB RDS for MySQL instance to a PolarDB-X database.	-
Full data migration	The operation of migrating all existing records from a database to PolarDB-X. An offset is recorded before full migration starts.	-
Offset	In a MySQL binary log file, each row represents a data change operation. The position of a line in the binary log file is called an offset.	-
Incremental data migration	The operation of reading all MySQL binary log records from the recorded offset, converting them into SQL statements, and then running them in PolarDB-X. Incremental migration continues before the switchover.	-
Switchover	A step of data import and smooth scale-out, which writes all the remaining incremental records from MySQL binary logs to PolarDB-X.	-
Cleanup	The last step of smooth scale-out, which cleans redundant data and configurations generated during smooth scale-out.	-

Term	Description	Remarks
Heterogeneous indexing	For table shards of a PolarDB-X database, the WHERE condition of a SQL statement for query must contain the shard key whenever possible. In this way, PolarDB-X routes the query request to a specific database shard, improving the query efficiency. If the WHERE condition of the SQL statement does not contain the shard key, PolarDB-X performs a full table scan. PolarDB-X provides heterogeneous indexing to solve this problem. The data in a database shard or table shard of a PolarDB-X instance is fully or partially synchronized to another table based on different shard keys. The destination table to which the data is synchronized is called a heterogeneous index table.	-
PolarDB-X sequence	A PolarDB-X sequence (a 64-digit number of the BIGINT data type in MySQL) aims to ensure that the data (for example, PRIMARY KEY and UNIQUE KEY) in the defined unique field is globally unique and in ordered increments.	-
PolarDB-X hint	To facilitate PolarDB-X usage, PolarDB-X defines some hints to specify special actions.	-