Alibaba Cloud
Apsara Stack Agility SE

User Guide

Version: 2003, Internal: V3.2.0

Issue: 20200618

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 due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall 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 the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. 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 bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, 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 contact Alibaba Cloud directly if you discover any errors in this document.

Document conventions

Style	Description	Example
	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.
	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.
	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.
	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]`

Style	Description	Example
{} or {a|b}	This format is used for a required value, where only one item can be selected.	switch {active|stand}

Contents

1 ASCM console

1.1 What is the ASCM console?

The Apsara Stack Cloud Management (ASCM) 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 ASCM 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 , allowing 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

ASCM 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 CloudMonitor.
2. Cloud resource creation: This part is designed to create resources as needed.
3. Cloud resource management: This part is designed to complete resource management operations, such as starting, using, and releasing resources and changing resource configurations.

1.2 User roles and permissions

This topic describes roles and their permissions.

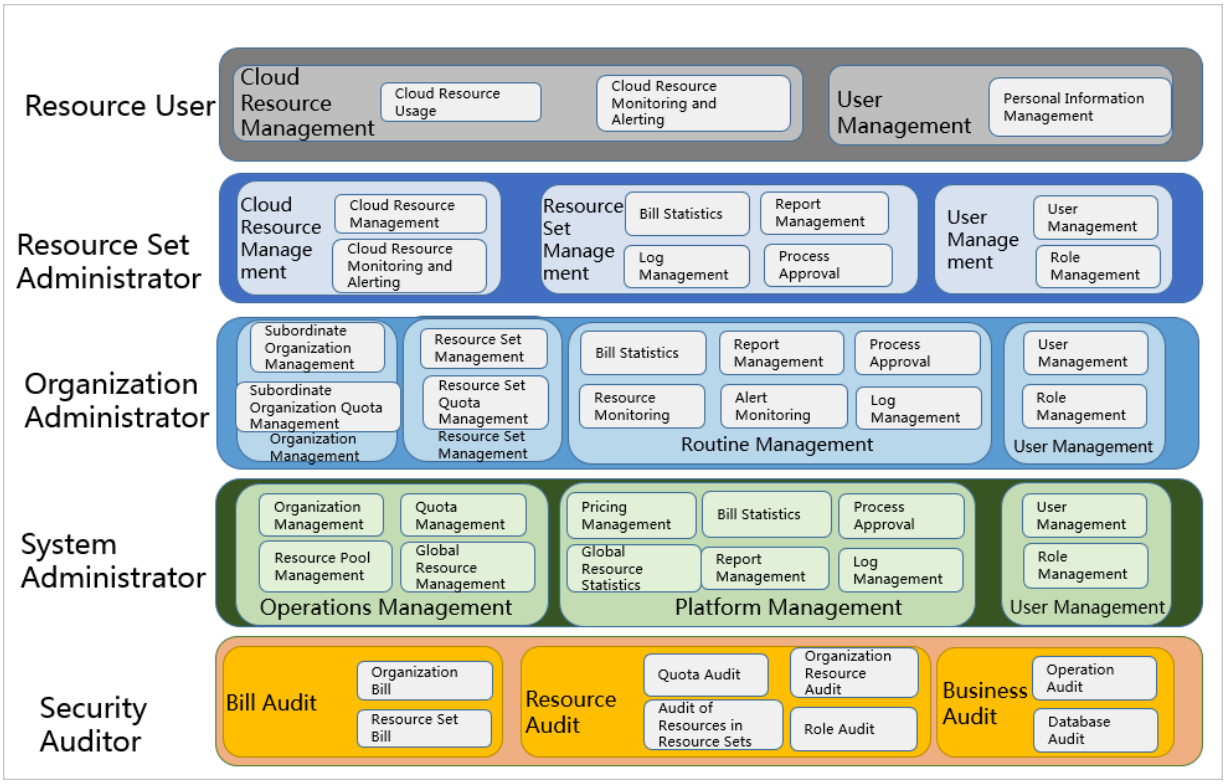


Table 1-1: Roles and permissions

Role name	Role permission
Resource user	This role has the permissions to view and modify resources in a resource set and create alarm 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 alarm rules for resources, and export reports.
Operations administrator	This role has read and write permissions on all resources.

Role name	Role permission
Security auditor	This role has read-only permissions on all resources.
Super administrator	This role has the permissions to initialize the system and create system administrators.

1.3 Log on to the ASCM console

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

Prerequisites

- Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel. The URL used to access the ASCM console is in the following format: https://[IP address or domain name of the ASCM console].
- We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to access the ASCM console. Press the Enter key.
2. Enter your username and password.

   The system has the following default user types: platform administrator (super), operations administrator (admin), and resource auditor (audito). The platform administrator is responsible for initial configurations of the ASCM console. The operations administrator can create other system users. The resource auditor is responsible for auditing global resources.

   Note:
   When you log on to the ASCM console for the first time, you must change the password of your username as prompted. Due to security concerns, your password must meet the minimum complexity requirements: The password must be 8 to 20 characters in length and must contain at least two of the following character types: uppercase letters, lowercase letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).
3. Click **Login** to go to the ASCM console homepage.

1.4 Web page introduction

The web page of the Apsara Stack Cloud Management (ASCM) console consists of the main menu bar, the information area of the currently logged-on user, and the operation area.

ASCM console page

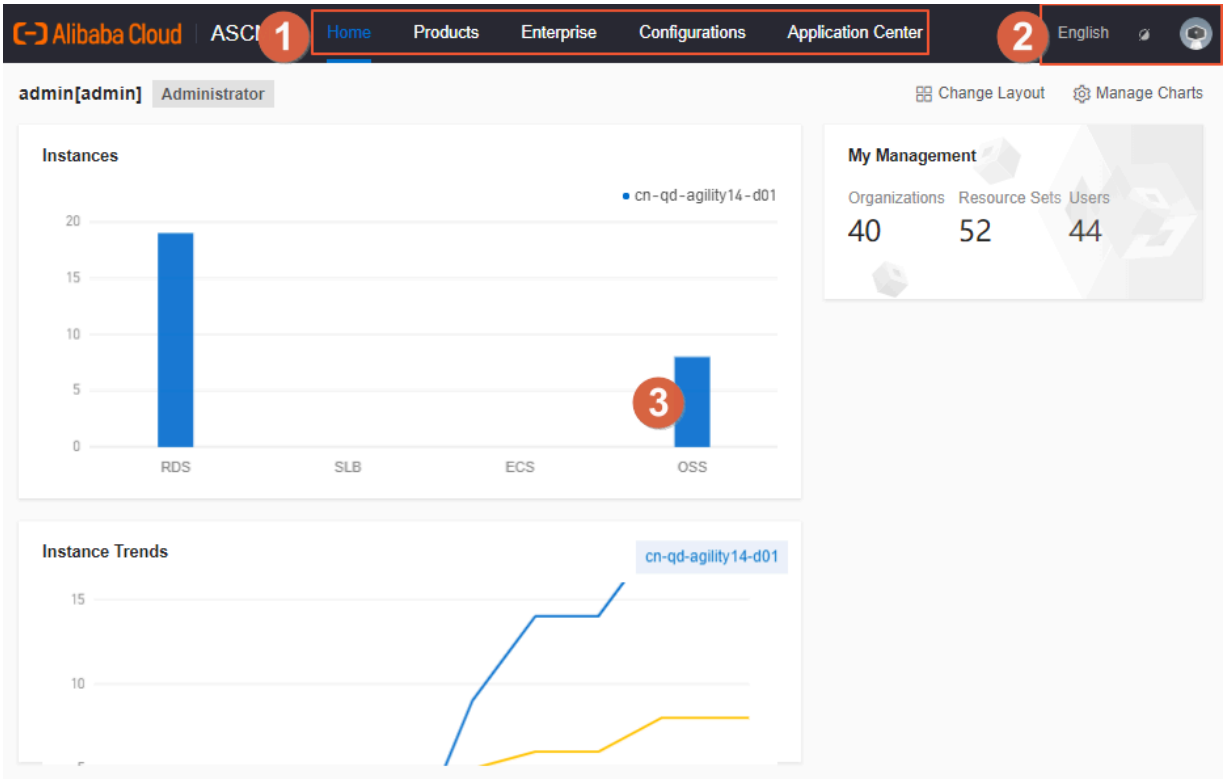


Table 1-2: Functional areas of the web page

Area		Description
1	Main menu bar	This area 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 products and resources. • **Enterprise**: manages organizations, resource sets, roles, users, and logon policies for enterprises. • **Configurations**: manages system configurations, password policies, specifications, and the message center. • **Security**: provides operation logs and system logs. **Note:** You must be a global organization security administrator, organization security administrator, security auditor, platform security administrator, or security system configuration administrator to view this menu.
2	Information area of the currently logged-on user	• English: allows you to switch between English, simplified Chinese, and traditional Chinese. • : allows you to select day or night mode. • User Information: Move the pointer over the icon of the currently logged-on 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.
3	Operation area	**Operation area**: the information display and operation area.

1.5 Initial configuration

1.5.1 Configuration description

Before using the Apsara Stack Cloud Management (ASCM) 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 ASCM 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 ASCM 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.
- Role

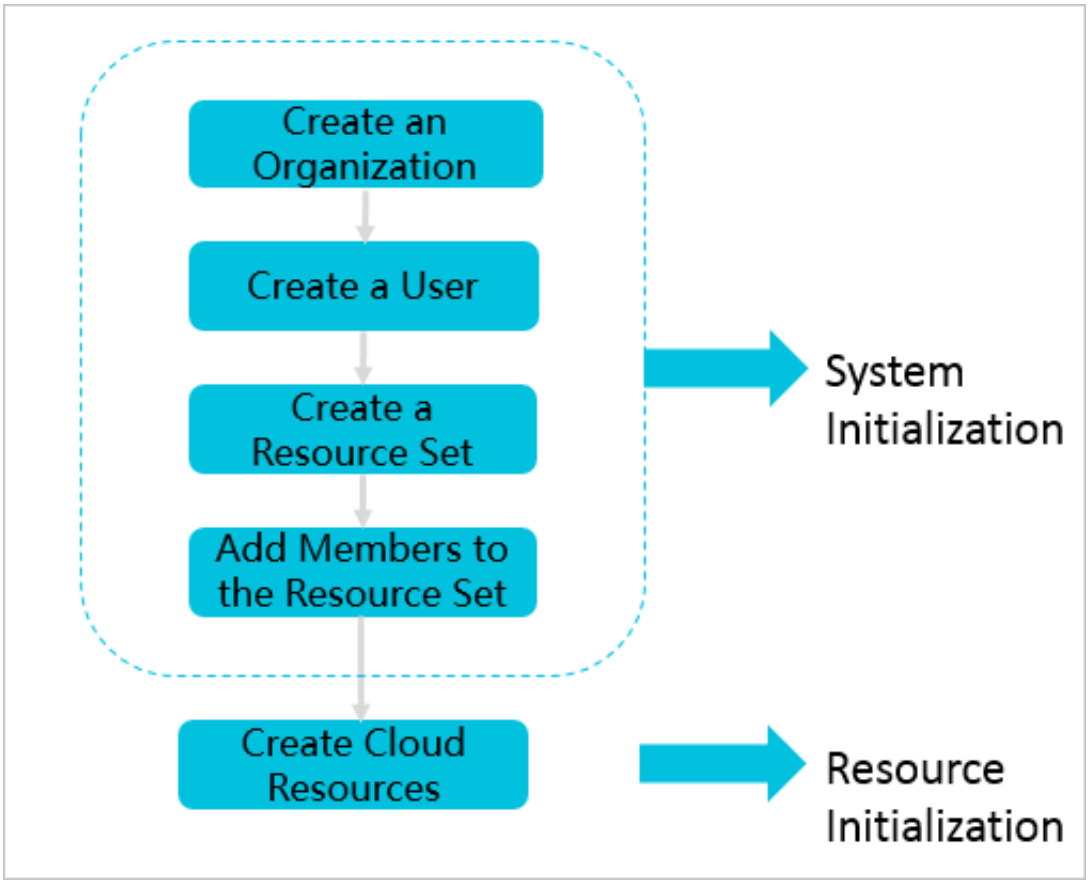
  A role is a set of access permissions. You can assign different roles to different users to meet different requirements for system access control.

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

1.5.2 Configuration process

This topic describes the initial configuration process.

Before using the Apsara Stack Cloud Management (ASCM) console, you must complete the initial system configurations as an administrator according to 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 members 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 Stack Cloud Management (ASCM) console uses charts to keep you up-to-date on the current usage of resources.

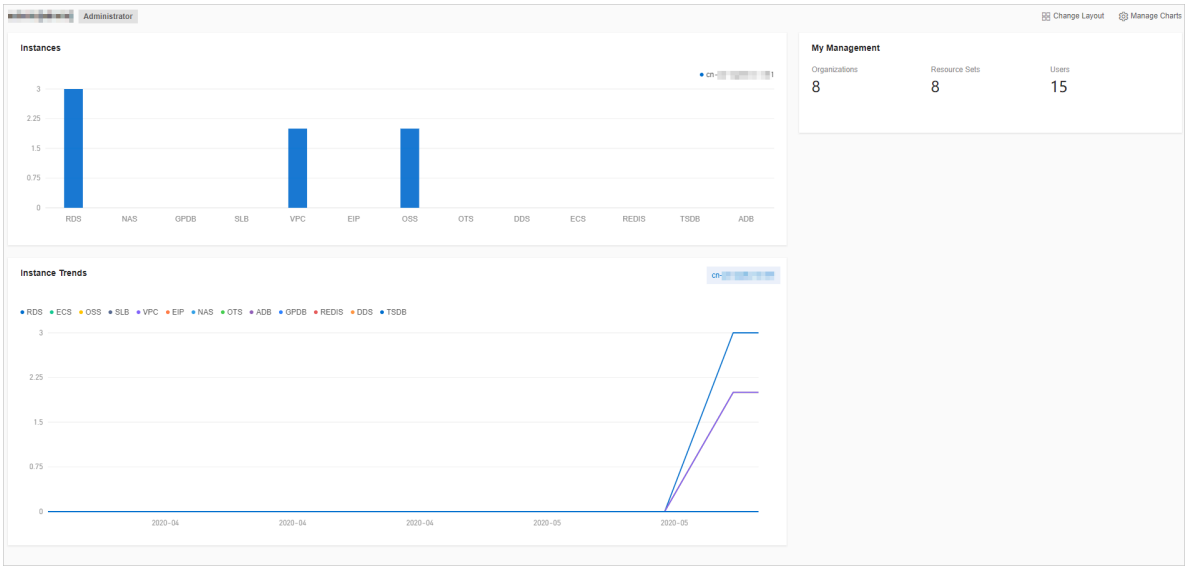
Context

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

Procedure

1. Log on to the ASCM console.

   By default, the **workbench** page appears when you log on to the ASCM 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 the target location.

   - **Instances**

     Shows the numbers of ECS instances, ApsaraDB for RDS instances, SLB instances, and OSS buckets 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 CloudMonitor overview

CloudMonitor provides real-time monitoring, alerting, and notification services for resources to protect your products and business.

CloudMonitor can monitor metrics for ApsaraDB RDS for MySQL and KVStore for Redis.

You can use the monitoring metrics of cloud services to configure alarm rules and notificati on 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 insufficie nt.

1.6.2.2 Metrics

This topic describes the monitoring metrics available for each service.

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

CloudMonitor can monitor resources of ApsaraDB RDS for MySQL and KVStore for Redis. The following table lists the metrics for each service.

Table 1-3: Metrics for ApsaraDB for RDS

Metric	Description	Apsara Stack service	Calculation formula
CPU utilization	Measures the CPU utilization of an RDS instance. Unit: %.	ApsaraDB for RDS	CPU utilization of an RDS instance/Total CPU cores of the RDS instance
Memory usage	Measures the memory usage of an RDS instance. Unit: %.	ApsaraDB for RDS	Used memory of an RDS instance/Total memory of the RDS instance
Disk usage	Measures the disk usage of an RDS instance. Unit: %.	ApsaraDB for RDS	None
IOPS utilization	Measures the number of I/O requests for an RDS instance per second. Unit: %.	ApsaraDB for RDS	Number of I/O requests for an RDS instance/Statistical period
Connection utilization	Measures the number of connections between an application and an RDS instance per second. Unit: %.	ApsaraDB for RDS	Number of connections between an application and an RDS instance per second/ Statistical period

Table 1-4: Metrics for KVStore for Redis

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

Metric	Description	Apsara Stack service	Unit
Number of failed operations per second	Measures the number of failed operations on a KVStore for Redis instance per second.	KVStore for Redis	N/A
Write bandwidth usage	Measures the percentage of total bandwidth used by write operations.	KVStore for Redis	%
Read bandwidth usage	Measures the percentage of total bandwidth used by read operations.	KVStore for Redis	%
Used QPS	Measures the number of queries per second (QPS).	KVStore for Redis	N/A
QPS usage	Measures the QPS utilization rate .	KVStore for Redis	%
Average response time	Measures the average response time.	KVStore for Redis	Milliseconds
Maximum response time	Measures the maximum response time.	KVStore for Redis	Milliseconds
Number of failed commands	Measures the number of failed commands.	KVStore for Redis	N/A
Hit rate	Measures the current hit rate.	KVStore for Redis	%
Inbound traffic	Measures the inbound traffic to a KVStore for Redis instance.	KVStore for Redis	Bytes
Inbound bandwidth usage	Measures the inbound bandwidth usage of a KVStore for Redis instance.	KVStore for Redis	%
Outbound traffic	Measures the outbound traffic from a KVStore for Redis instance.	KVStore for Redis	Bytes
Outbound bandwidth usage	Measures 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 ASCM 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 alarm overview

On the **Overview** page in CloudMonitor, you can view the alarm status statistics and alarm logs.

Procedure

1. Log on to the ASCM 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 **Overview**.
4. On the **Overview** page, view the alarm status statistics and alarm logs generated in the last 24 hours.

1.6.3.2 View alarm logs

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

Context

Alarm information contains information for all items that do not comply with your configured alarm rules.

Note:
The system can retain up to one million alarm items generated within the last three months.

Procedure

1. Log on to the ASCM 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, choose **Alarms** > **Alarm History**.

4. On the **Alarm Rule History List** page, you can filter alarm information by rule ID, rule name, product, and date.

   The following table describes the fields in the query result.

   Table 1-5: Alarm information fields

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

1.6.3.3 Alarm rules

1.6.3.3.1 Query alarm rules

After creating alarm rules, you can view your alarm rules on the **Alarm Rules** page.

Procedure

1. Log on to the ASCM 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 **Alarm Rules** in the **Actions** column corresponding to an instance to go to its **Alarm Rules** page.

   On the **Alarm Rules** page that appears, view the detailed information of alarm rules.

1.6.3.3.2 Create an alarm rule

You can create an alarm rule to monitor an instance.

Procedure

1. Log on to the ASCM 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 **Alarm 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 alarm rules.

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

Table 1-6: Parameters for creating an alarm rule

Parameter	Description
Product	The monitored cloud product.
Resource Range	The range of resources associated with the alarm rule.
Rule Description	The description of the alarm rule.
Add Rule Description	Click **Add Rule Description** to go to the rule configuration page. For more information, see Table 1-7: Parameters for adding rule description.
Effective Time	Only a single alarm is sent during each mute duration, even if the metric value exceeds the alarm rule threshold several times in a row.
Effective Period	An alarm is sent only when the threshold is crossed during the effective period.
HTTP CallBack	The callback URL when the alarm conditions are met.
Alarm Contact Group	The group to which alarms are sent.

Table 1-7: Parameters for adding rule description

Parameter	Description
Rule Name	The name of the alarm 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 alarm rule is triggered.
Threshold And Alarm Level	Different metrics have different reference thresholds.

7. Click **OK**.

1.6.3.3.3 Disable an alarm rule

You can disable one or more alarm rules as needed.

Procedure

1. Log on to the ASCM 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 **Alarm Rules** in the **Actions** column corresponding to an instance.
6. On the **Alarm Rules** page, choose **More** > **Disable** in the **Actions** column corresponding to the alarm rule to be disabled.
7. In the message that appears, click **OK**.

1.6.3.3.4 Enable an alarm rule

After an alarm rule is disabled, it can be re-enabled as needed.

Procedure

1. Log on to the ASCM 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 **Alarm Rules** in the **Actions** column corresponding to an instance.
6. On the **Alarm Rules** page, choose **More** > **Enable** in the **Actions** column corresponding to the alarm rule to be enabled.
7. In the message that appears, click **OK**.

1.6.3.3.5 Delete an alarm rule

You can delete alarm rules that are no longer needed.

Procedure

1. Log on to the ASCM 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 **Alarm Rules** in the **Actions** column corresponding to an instance.
6. On the **Alarm Rules** page, click **Delete** in the **Actions** column corresponding to the alarm rule to be deleted.
7. In the message that appears, click **OK**.

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 ASCM 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, move the pointer over the name of a parent organization, and click on the right.
5. Choose **Add Organization** from the shortcut menu.
6. In the dialog box that appears, 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 ASCM 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

   to query 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 ASCM 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 with operation permissions on an organization can modify the name of the organization.

Procedure

1. Log on to the ASCM 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, move the pointer over the name of an organization and click  on the right.
5. Choose **Edit Organization** from the shortcut menu.
6. In the dialog box that appears, modify the organization name.
7. Click **OK**.

1.7.1.5 Obtain the AccessKey pair of an organization

An administrator can obtain the AccessKey pair of an organization.

Procedure

1. Log on to the ASCM 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, move the pointer over the name of a level-1 organization and click on the right.
5. Choose **AccessKey** from the shortcut menu.
6. In the message that appears, view the AccessKey information of the organization.

1.7.1.6 Delete an organization

Administrators can delete organizations that are no longer needed.

Prerequisites

> **Note:**
> Before deleting an organization, make sure that the organization does not contain any users, resource sets, or subordinate organizations. Otherwise, the organization cannot be deleted.

Procedure

1. Log on to the ASCM 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, move the pointer over the name of an organization, and click on the right.
5. Choose **Delete Organization** from the shortcut menu.
6. In the message that appears, click **OK**.

1.7.2 Resource sets

1.7.2.1 Create a resource set

Before applying for resources, you must create a resource set.

Procedure

1. Log on to the ASCM 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-right corner of the page, click **Create**.

5. In the **Create Resource Set** dialog box that appears, 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 ASCM 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 ASCM 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 ASCM 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 Delete a resource set

Administrators can delete resource sets that are not needed.

Context

Ensure that the resource set to be deleted does not contain any members or resources.

Notice:
A resource set cannot be deleted if it contains resources or members.

Procedure

1. Log on to the ASCM 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 **Delete** from the shortcut menu.

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

1.7.3 Roles

1.7.3.1 Create a custom role

You can add custom roles in the Apsara Stack Cloud Management (ASCM) console to more efficiently grant permissions to users so that different personnel can work with different functions.

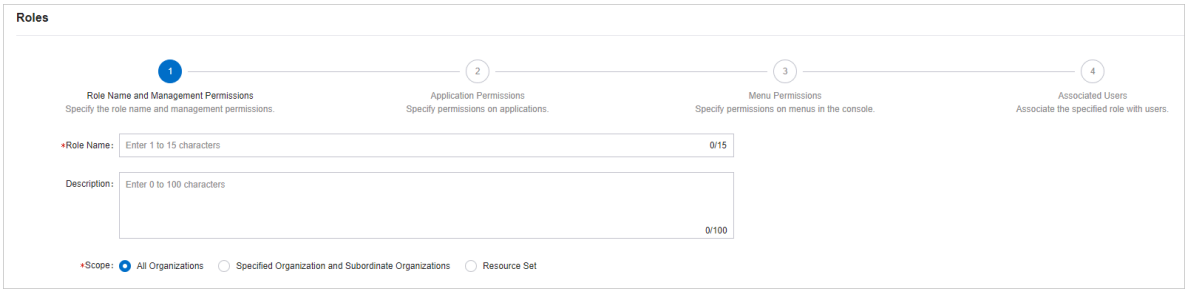
Context

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.

Before you add a custom role, note that the total number of custom and default roles cannot exceed 20.

Procedure

1. Log on to the ASCM 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 that appears, set the role name and management permissions.



The following table describes the role creation parameters.

Table 1-8: Role creation parameters

Parameter	Description
Role Name	The name of the role. The name can be up to 15 characters in length and can contain only letters and digits.

Parameter	Description
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 (_).
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 Set** The permissions apply to the resource sets assigned to the user.

6. Select the operation permissions that this role has based on your needs. Click **Next**.
7. In the **Application Permissions** step, select the operation permissions that this role has on the cloud products. Click **Next**.
8. In the **Menu Permissions** step, select the operation permissions that this role has on various menus. 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 ASCM 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

An administrator can modify the name and permissions of a custom role.

Context

Note:

Information about preset roles cannot be modified.

Procedure

1. Log on to the ASCM 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 custom role, and choose **Modify** from the shortcut menu to go to the **Roles** page.
5. You can modify the name, permissions, and associated users of a custom role.
   - Procedure of modifying a role name: Move the pointer over the role name and click to enter a new role name.
   - Procedure of modifying permissions: Click the **Management Permissions**, **Application Permissions**, or **Menu Permissions** tab, select or remove related permissions from the corresponding tab, and then click **Update**.
   - Procedure of binding 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**.

1.7.3.4 Disable a role

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

Procedure

1. Log on to the ASCM 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.

1.7.3.5 Enable a role

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

Procedure

1. Log on to the ASCM 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.

1.7.3.6 Delete a custom role

You can delete custom roles that are no longer needed.

Context

> **Note:**
> Default or preset roles cannot be deleted.

Procedure

1. Log on to the ASCM 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 **More** in the **Actions** column corresponding to a role, and choose **Delete** from the shortcut menu.
5. In the message that appears, click **OK**.

1.7.4 Users

1.7.4.1 System users

1.7.4.1.1 Create a user

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

Procedure

1. Log on to the ASCM 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. On the Users page, click **Create**.

5. In the dialog box that appears, 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.
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 roles 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 user's logon policy 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. Make sure that the entered mobile number is correct. **Note:** If the mobile number is changed, be sure to update it on the system in a timely manner.
Landline Number	Optional. The landline number of the user. It must be 4 to 20 characters in length and can contain only digits (0 to 9) and hyphens (-).
Email	The email address of the user. Emails about the usage and requests for resources will be sent to the email address. Make sure that the specified email address is correct. **Note:** If the email address is changed, be sure to update it on the system in a timely manner.

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

You can modify user information such as display name, mobile number, and email address to keep it up-to-date.

Procedure

1. Log on to the ASCM 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 **Edit** from the shortcut menu.
6. In the **Modify User Information** dialog box, enter the relevant information and 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 ASCM 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 ASCM 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 ASCM 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 View the initial password of a user

After a user is created, the system automatically 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 ASCM 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 the username that you want to query.
6. You can use one of the following methods to view the initial password of a user:
   - Click **View Initial Password** in the upper-right corner of the **Users** page to view the initial password.
   - Click **More** in the **Actions** column corresponding to the user, and choose **User Information** from the shortcut menu. On the user information page, click **View Password** to view the initial password.

1.7.4.1.7 Reset the password of a user

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

Prerequisites

The logon password of a user can be reset by only the user and those who have the permissions to create resource sets for the user.

Procedure

1. Log on to the ASCM 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 the user for which you want to reset the password.
6. Click **More** in the **Actions** column corresponding to the user, and choose **User Information** from the shortcut menu.
7. Click **Reset Password**.

   After the password is reset, a message is displayed, indicating that the password has been reset. If you want to view the initial password after password reset, click **View Password**.

1.7.4.1.8 Disable and enable a user

You can disable a user to prevent the user from logging on to the Apsara Stack Cloud Management (ASCM) console. Disabled users must be re-enabled before they can log on to the ASCM console again.

Context

By default, users are enabled when they are created.

Procedure

1. Log on to the ASCM 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 whose **Status** is **Enabled**, click **More** in the **Actions** column, and choose **Disable** from the shortcut menu to disable the user.
   - Select a user whose Status is **Disabled**, click **More** in the **Actions** column, and choose **Enable** from the shortcut menu to enable the user.

1.7.4.1.9 Delete a user

Administrators can delete users.

Procedure

1. Log on to the ASCM 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 ASCM 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 ASCM 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 Stack Cloud Management (ASCM) console, an administrator can create a logon policy 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, the user's logons will be restricted based on the logon time and IP addresses specified in the policy.

When providing services, the ASCM console automatically generates a default policy without restrictions on the logon time and IP addresses. The default policy cannot be deleted.

Procedure

1. Log on to the ASCM 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 that appears, set Name, Policy Properties, Time Period, and IP Address.

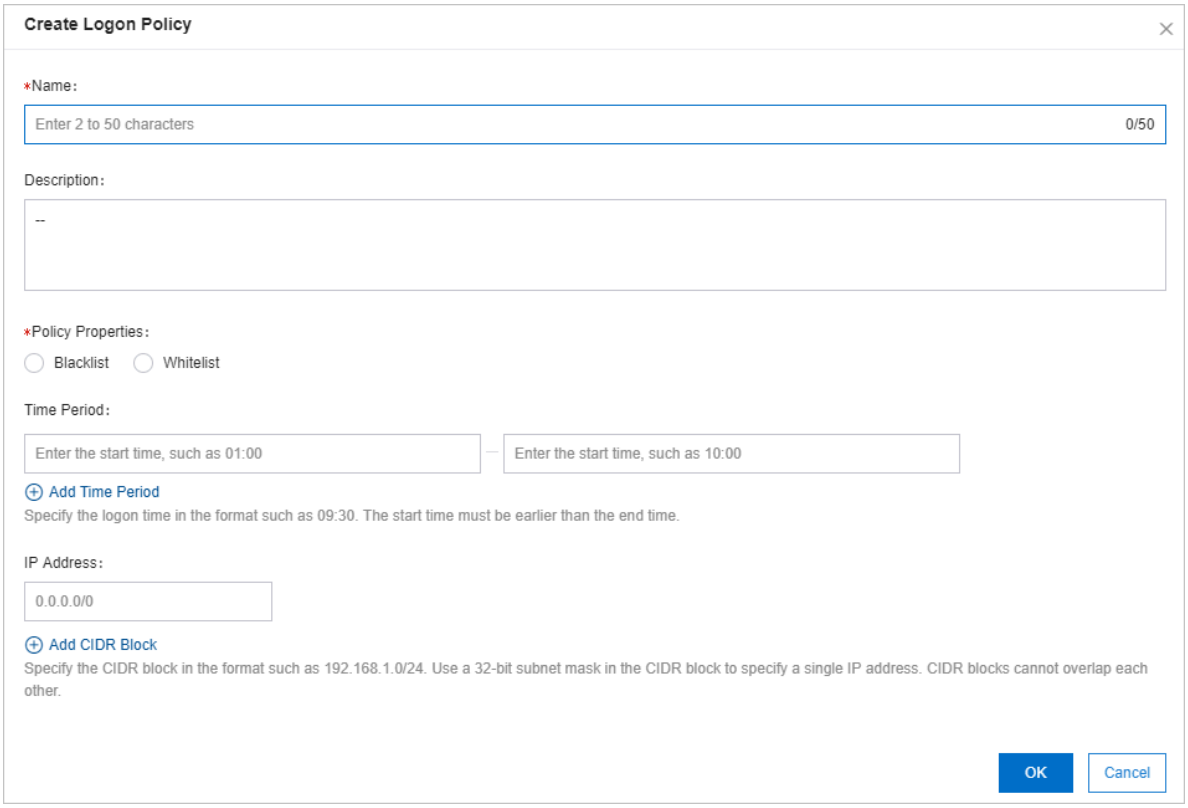


Table 1-9: Parameters for creating a logon policy

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.
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.
Time Period	The permitted logon time period. When this policy is configured, users can log on to the ASCM console only during the configured period. Specify the time in minutes in the 24-hour clock. Example: 16:32.

Parameter	Description
IP Address	The permitted CIDR block. • If **Policy Properties** is set to **Whitelist**, IP addresses within this CIDR block are allowed to log on to the ASCM console. • If **Policy Properties** is set to **Blacklist**, IP addresses within this CIDR block are not allowed to log on to the ASCM console.

1.7.5.2 Query a logon policy

When providing services, the Apsara Stack Cloud Management (ASCM) console automatically generates a default policy without restrictions on the logon time and IP addresses.

Context

When providing services, the Apsara Stack Cloud Management (ASCM) console automatica lly generates a default policy without restrictions on the logon time and IP addresses.

Procedure

1. Log on to the ASCM 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 ASCM 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 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 ASCM 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:

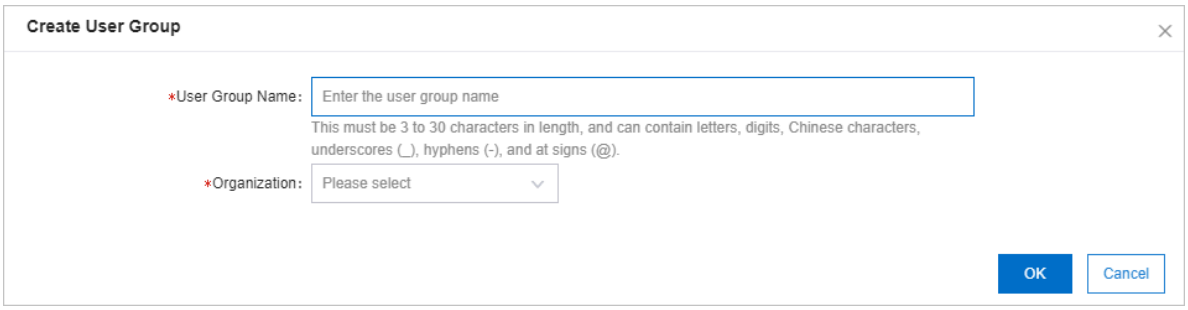
- A user group can only be bound to a single role.
- A role can be associated with multiple user groups.
- When a role is associated with a user group, the role permissions are automatically granted to users in the user group.

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 ASCM 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**.

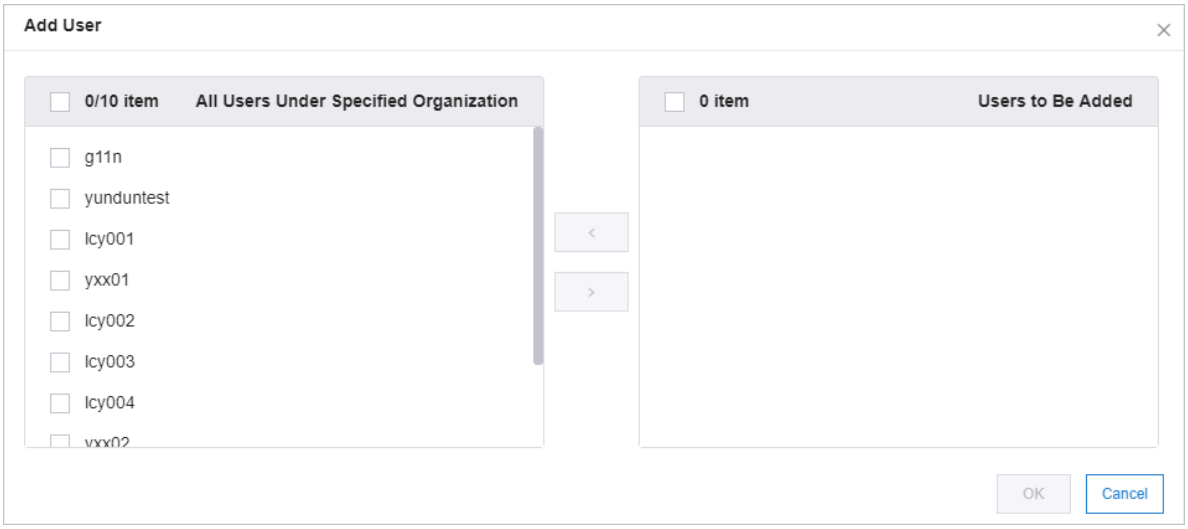
1.7.6.2 Add users to a user group

You can add users to a user group.

Procedure

1. Log on to the ASCM 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.



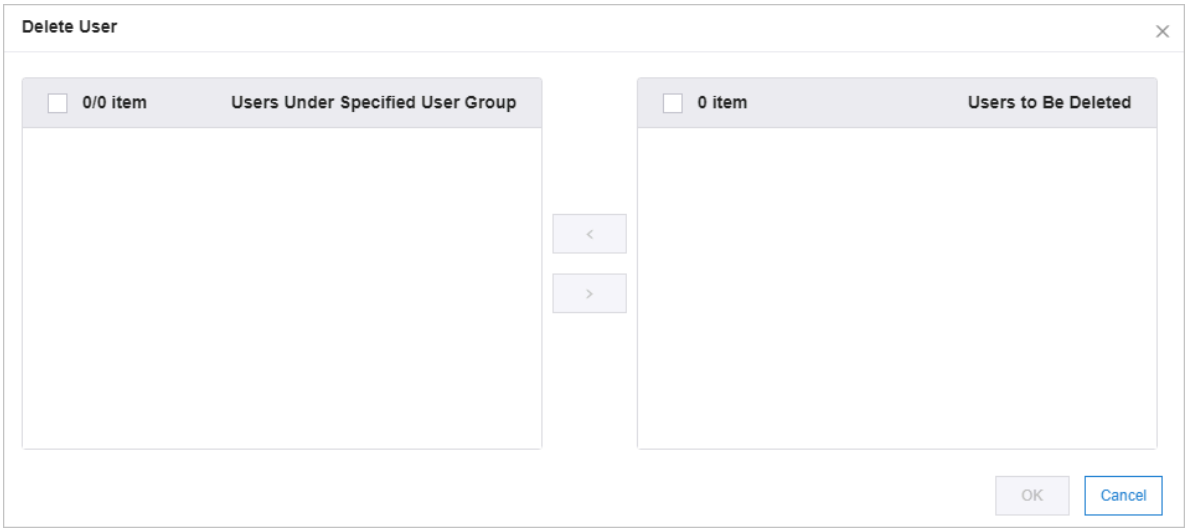
6. Click **OK**.

1.7.6.3 Delete users from a user group

You can delete users from a user group.

Procedure

1. Log on to the ASCM 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 ASCM 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 ASCM 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 ASCM 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 ASCM 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**.

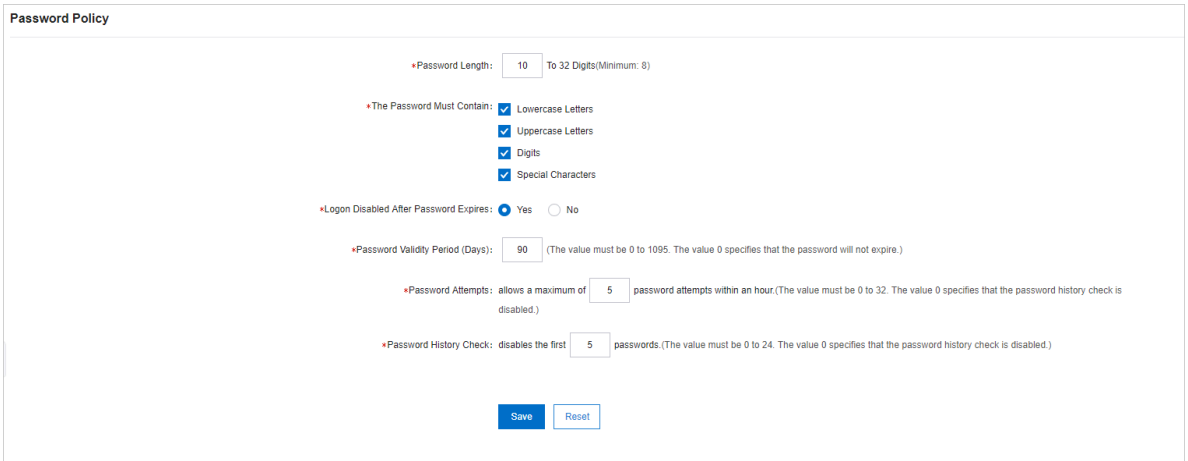
1.8 Configurations

1.8.1 Password policies

You can configure password policies for user logons.

Procedure

1. Log on to the ASCM 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 ASCM console for quick access.

Procedure

1. Log on to the ASCM 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. On the **Main Menu** page, click **Create** in the upper-right corner.
5. In the **Create** dialog box that appears, set the menu parameters.

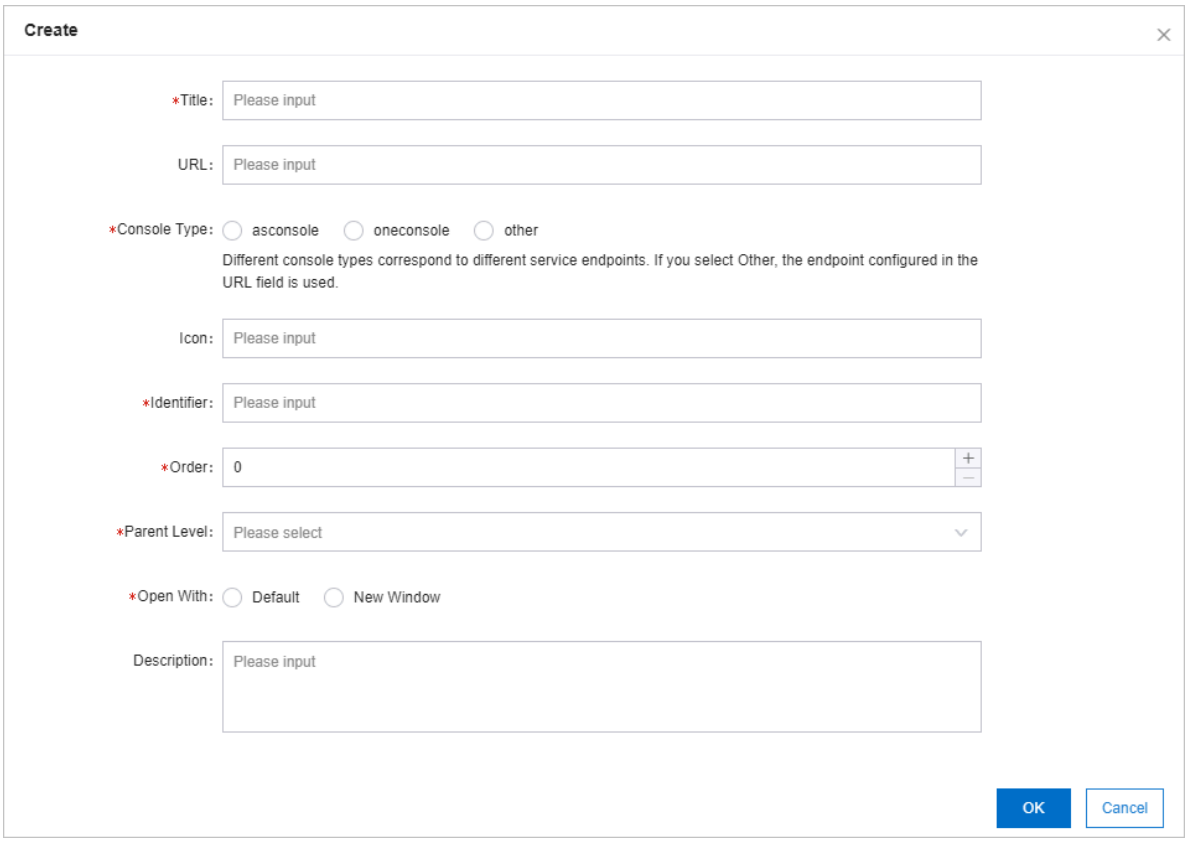


Table 1-10: Menu parameters

Parameter	Description
Title	The display name of the menu.
URL	The URL of the menu.

Parameter	Description
Console Type	Different console types correspond to different domain names. • **oneconsole**: You only need to enter the path in the URL field. The domain name is automatically matched. • **asconsole**: You only need 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.
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.

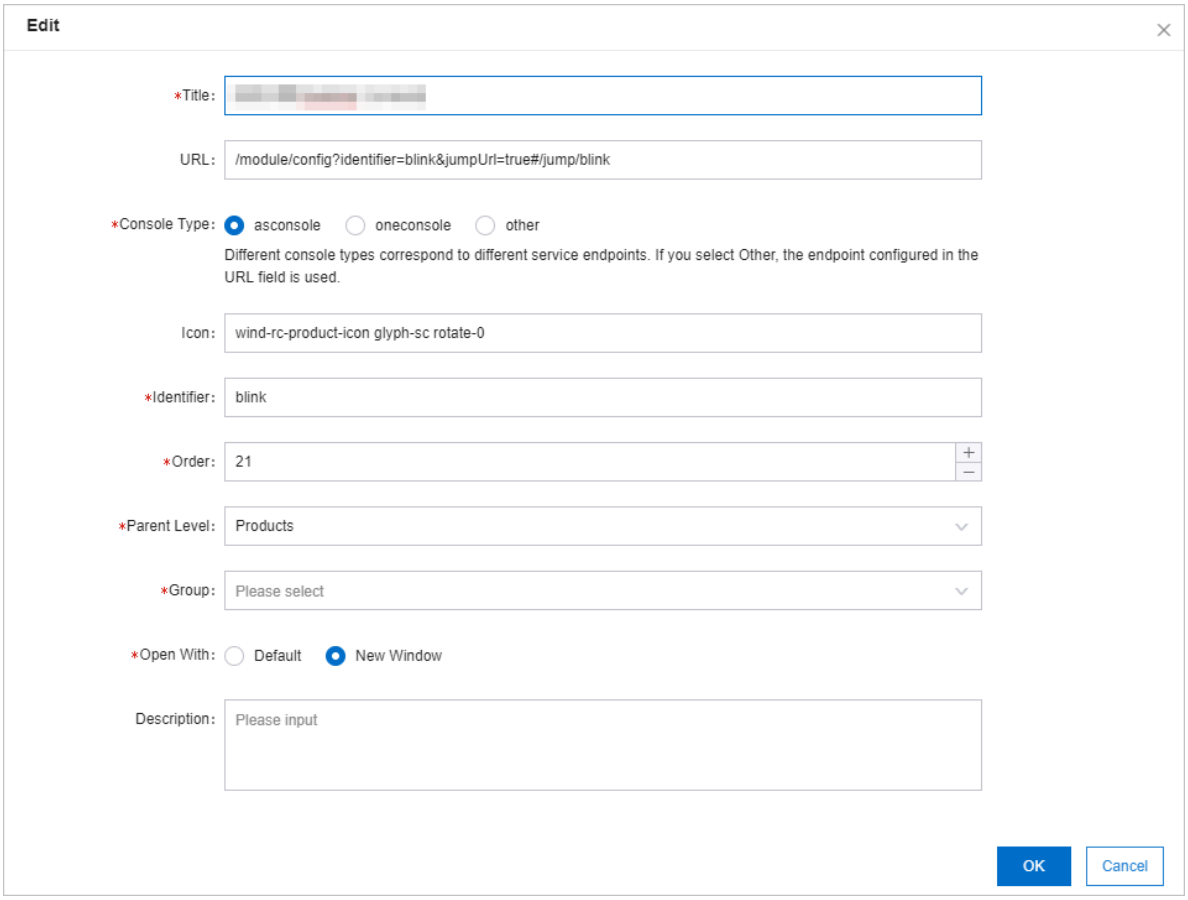
Prerequisites

Default menus cannot be modified.

Procedure

1. Log on to the ASCM 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 ASCM 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 ASCM 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 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.
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.

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 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 the database engine.

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.

Distributed Relational Database Service (DRDS)

Parameter	Description
Instance Type	The instance type that can be configured for 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.

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.

1.8.3.2 Create specifications

You can customize specifications for each resource type.

Procedure

1. Log on to the ASCM 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 ASCM 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. In the list of specifications, view information about the specifications.

1.8.3.4 Disable specifications

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

Procedure

1. Log on to the ASCM 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 Export specifications

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

Procedure

1. Log on to the ASCM 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 Stack Cloud Management (ASCM) console. You can also export operation logs to your personal computer.

Procedure

1. Log on to the ASCM 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.

   Table 1-11: 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.
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.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 permission s.

  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

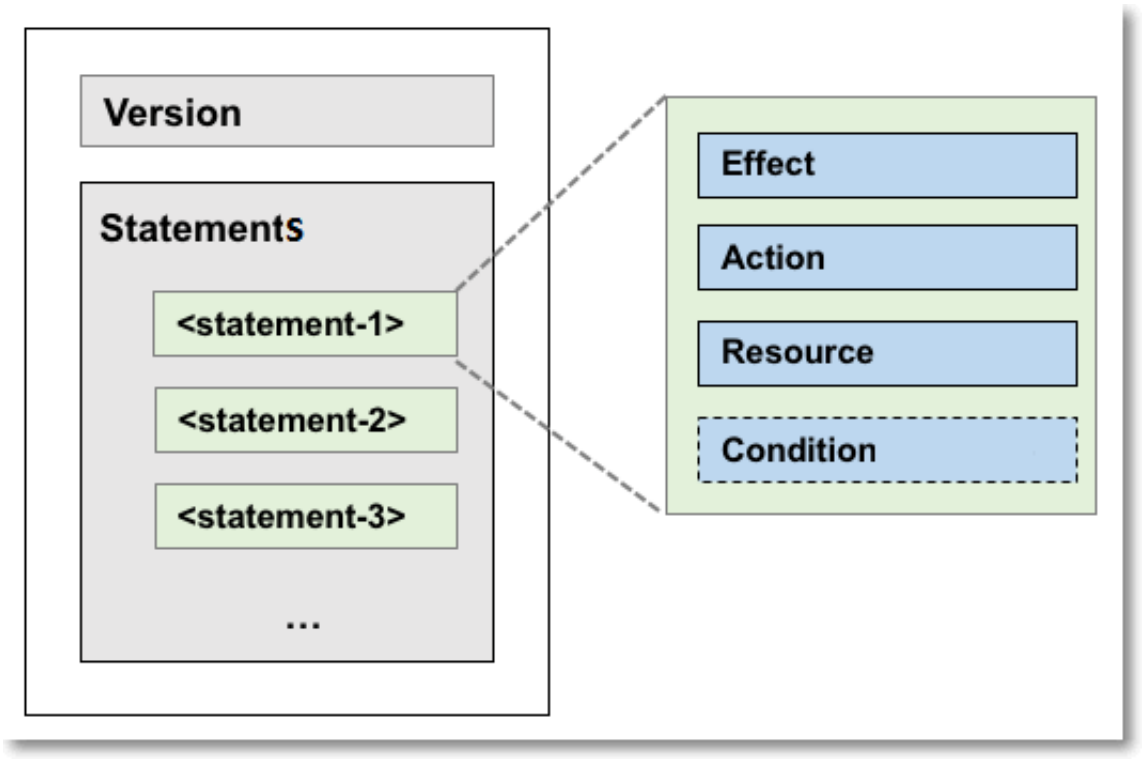
- Characters in a policy
  - The following characters are JSON tokens and are included in policies: { } [ ] " , :.
  - The following characters are special characters in the syntax and are not included in policies: = < > ( ) |.

- 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 ASCM 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 ASCM 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 that appears, set **Role Name** and **Description**.
6. Click **Create**.

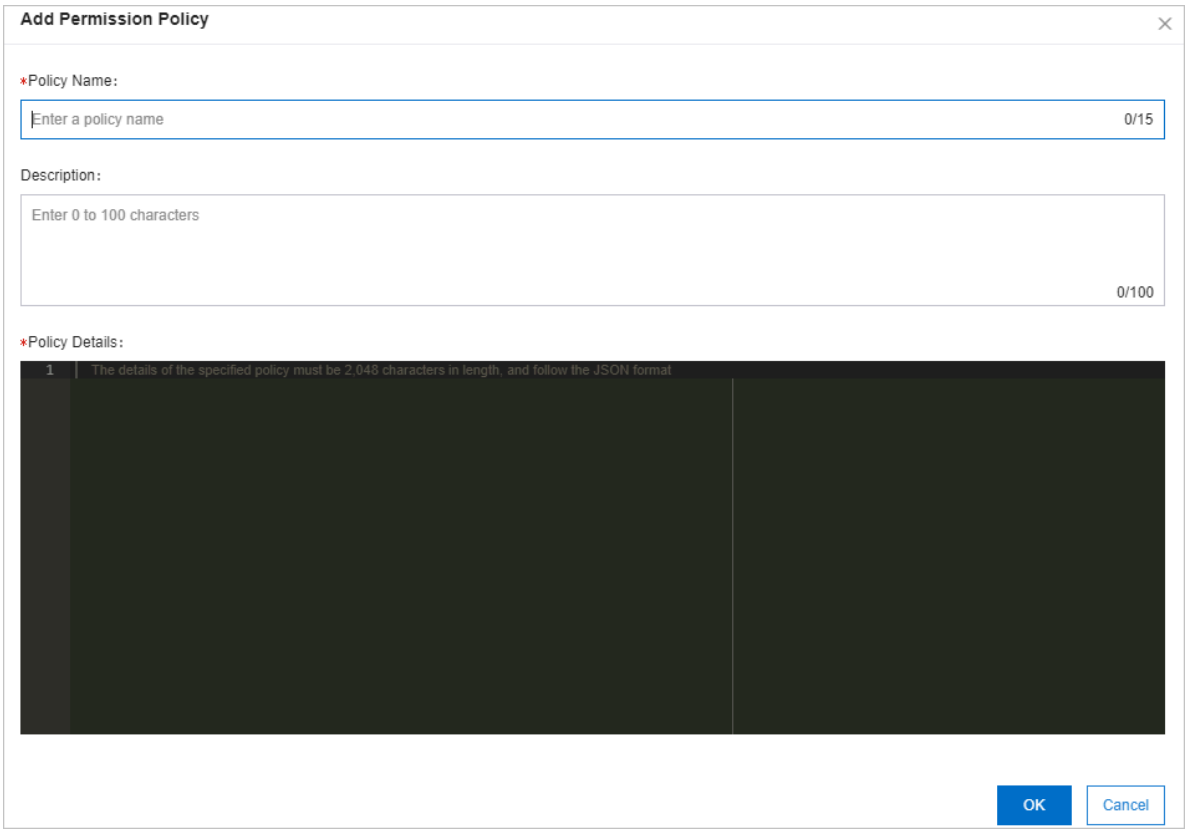
1.10.3.3 Add a permission policy

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 ASCM 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 ASCM 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 ASCM 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 Create a user group.

Procedure

1. Log on to the ASCM 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 ASCM 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 ASCM 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 ASCM 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 ASCM 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 ASCM 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 ASCM 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 ASCM 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.

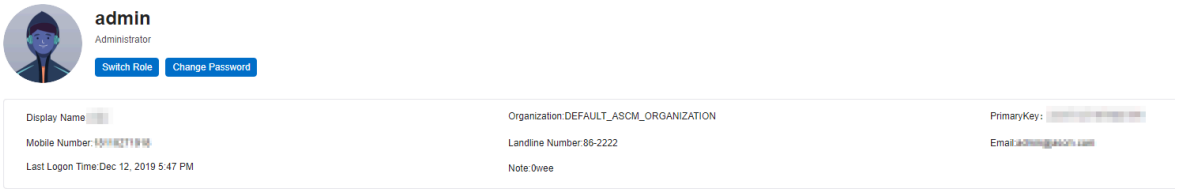
1.11 Personal information management

1.11.1 Modify personal information

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

Procedure

1. Log on to the ASCM 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.



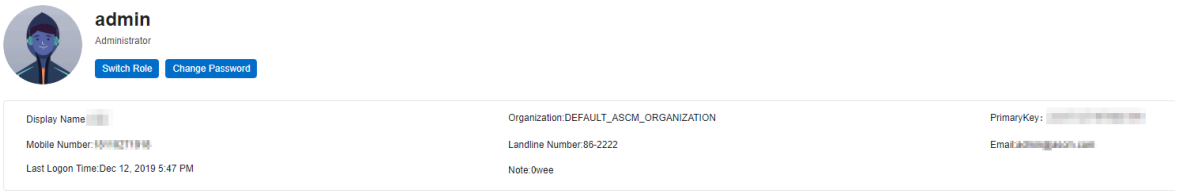
3. Click  next to the item you want to modify.
4. In the Modify User Information dialog box that appears, modify the relevant information.
5. Click **OK**.

1.11.2 Change your logon password

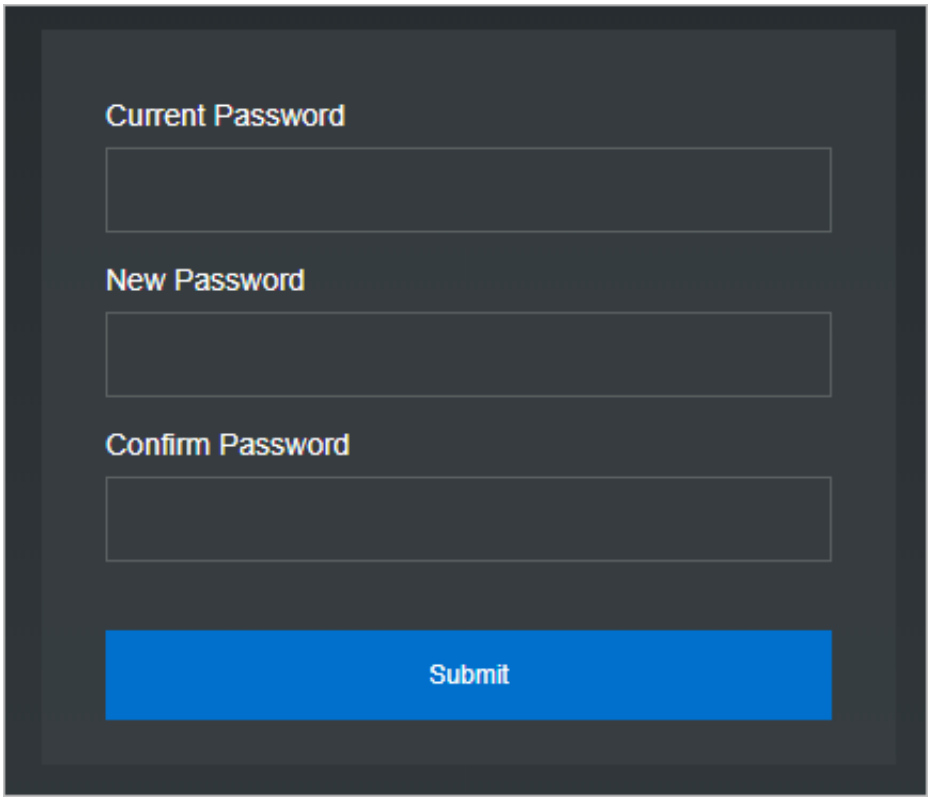
To improve security, you must change your logon password in a timely manner.

Procedure

1. Log on to the ASCM 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**.

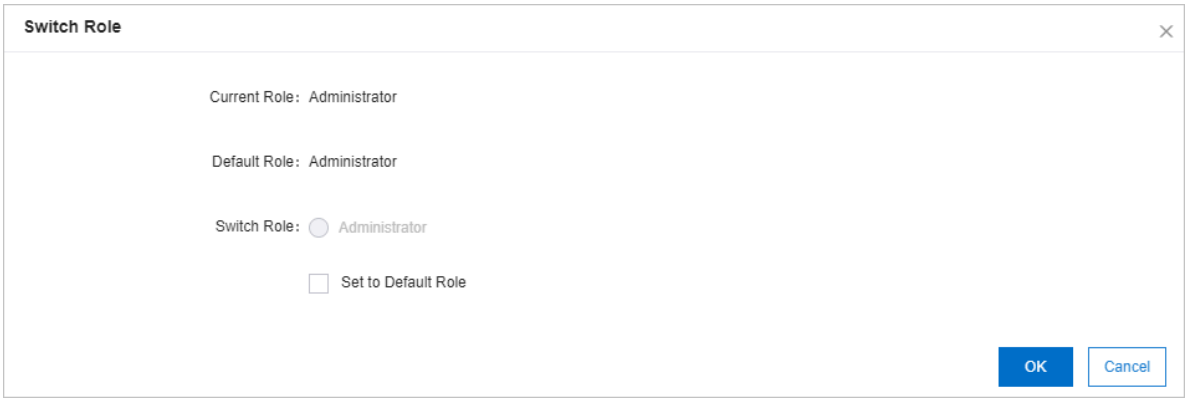
1.11.3 Switch the current role

You can switch the scope of your current role.

Procedure

1. Log on to the ASCM 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.



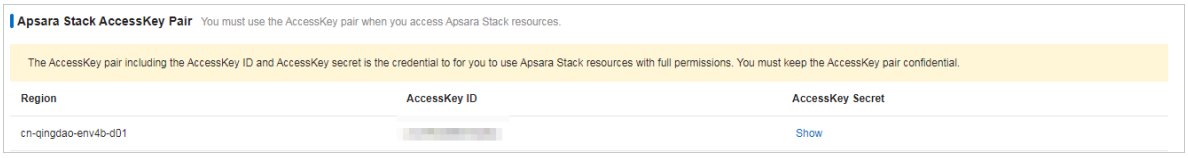
You can also switch back to the default role.

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 ASCM 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?

Alibaba Cloud Object Storage Service (OSS) is a secure, cost-effective, and highly reliable storage service that is capable of processing large amounts of data.

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

OSS uploads data files as objects to buckets. OSS is an object storage service that uses a key-value pair format. You can retrieve object content based on unique object keys.

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 attributes or metadata of a bucket or an object, and configure ACL for the bucket or the object.
- Perform basic and advanced OSS 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 their permission 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.

Item	Limit
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.
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

- Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel. The URL used to access the ASCM console is in the following format: https://[IP address or domain name of the ASCM console].
- A browser is available. We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to access the ASCM console. Press Enter.

2. Enter your username and password.

   Obtain the username and password for logging on to the console from the operations administrator.

   Note:

   When you log on to the ASCM console for the first time, you must change the password of your username as prompted. Due to security concerns, your password must meet the minimum complexity requirements: The password must be 8 to 20 characters in length and must contain at least two of the following character types: uppercase letters, lowercase letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

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

2.3.2 Create a bucket

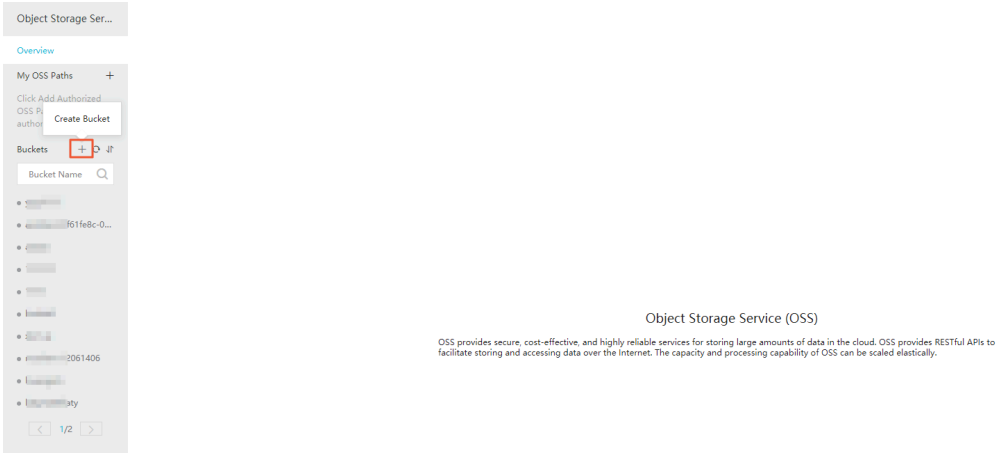
Objects uploaded to OSS are stored in a bucket. Before you upload an object to OSS, you need to create a bucket.

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. In the **Create OSS Bucket** dialog box that appears, configure parameters.



Note:

In the left-side navigation pane, click the **+** icon next to Bucket if there are buckets available. The **Create OSS Bucket** dialog box appears.

The following table describes the parameters for creating a bucket.

Table 2-1: Parameters for creating a bucket

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 over the internal network , select the region where your ECS instance is deployed.
Cluster	Select a cluster for the bucket. Two OSS clusters can be deployed 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 Alibaba Cloud OSS. • The bucket name cannot be changed after the bucket is created.
Storage Type	Set the value to **Standard**. Only Standard is supported.

Parameter	Description
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.
Bucket Access	Set the ACL for the bucket. The following options are available: • **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	Configure server-side encryption: • **No**: Data is not encrypted. • **AES256**: OSS server-side encryption uses AES256 to encrypt each object in the bucket with a different data key. Master keys used to encrypt data keys are rotated regularly.

3. Click **Submit**.

2.3.3 Upload objects

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

Context

You can upload an object of any format to a bucket. You can use the OSS console to upload an object no larger than 5 GB to a bucket. 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.
   - Do not refresh or close the upload page when objects are being uploaded. Otherwise, the upload tasks are interrupted and the upload object list is 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

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

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

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

> **Note:**
> To delete a bucket, ensure that all objects in the bucket are deleted, including parts generated from incomplete multipart upload operations. Otherwise, the bucket cannot be deleted.

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**.

> **Notice:**
> Deleted buckets cannot be recovered. Exercise caution when you perform this operation.

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

Before you modify the ACL of a bucket, you must complete the procedure instructed in Create a bucket, or at least one bucket exists in the current region.

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 the 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 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. Exercise caution when you configure this option.

 **Notice:**

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 through the bucket domain name.

Prerequisites

Before you configure static website hosting for a bucket, you must complete the procedure instructed in Create a bucket, or at least one bucket exists in the current region.

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 ending 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 the HTML object in the root folder of the bucket can be used.
   - **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 the HTML, JPG, PNG, BMP, or WebP object in the root folder can be used. The default 404 page is not enabled 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

Before you enable or disable logging for a bucket, you must complete the procedure instructed in Create a bucket, or at least one bucket exists in the current region.

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 a 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 directory where the access logs are generated and stored and the prefix of the access logs. 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

Before you configure hotlink protection for a bucket, you must complete the procedure instructed in Create a bucket, or at least one bucket exists in the current region.

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 in the OSS console for a bucket or configure whether to accept requests whose Referer field is empty. For example, you can add http://www.aliyun.com 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 a bucket to go to the bucket details page.
3. On the bucket details page, click the **Basic Settings** tab. Find the **Hotlinking Protection** section.

4. Click **Configure**. Configure the following parameters:
   - **Referer**: Add URLs to the whitelist. Referers are typically in URL format. Separate multiple Referers with new 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 fields, only HTTP or HTTPS requests which include an allowed 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

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

Context

OSS provides CORS in HTML5 to enable cross-origin access. When OSS receives a cross-origin request (or an OPTIONS request), it reads the CORS rules of the bucket and then checks the relevant permissions. OSS matches the rules one by one. When OSS finds the first match, it returns a corresponding header. If none of the rules match, OSS does not attach any CORS header.

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 **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 new 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 new 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.
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 can cache the response to a preflight (OPTIONS) request to a specific resource.

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

Before you manage lifecycle rules of a bucket, you must complete the procedure instructed in Create a bucket, or at least one bucket exists in the current region.

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 is automatically applied to all objects that match the rule. You can manage lifecycle rules to perform operations, such as object management and automatic part deletion.

- If an object matches a rule, data of the object is cleared 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 a 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 that have 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.

- **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 objects can be retained after they are last modified, and the operation to perform on these objects after they expire. Objects that meet the specified conditions are retained for the specified validity period after the objects are last modified. The specified operation is performed on these objects after they expire. 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 objects that are last modified expire and the operation to perform on these objects after they expire. All objects that are last modified before this date expire and the specified operation is performed on these objects. If you select Transit to Archive Storage Class and set Expiration Date to 2012-12-21, the backend application scans for objects that are last modified before December 21, 2012 and converts their storage class to Archive.
  - Disabled: The automatic object deletion or storage class conversion function is not enabled.
- **Fragment 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.

Notice:
In each lifecycle rule, you must configure at least object lifecycle or part lifecycle. In other words, you must select **Delete** or configure conversion actions for object lifecycle or select **Delete** for part lifecycle.

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.

2.4.9 Configure server-side encryption

Alibaba Cloud OSS performs server-side encryption on data stored in buckets to secure data.

Context

When you upload data, OSS encrypts the data before data is stored. When you download data, OSS decrypts the data and returns the original data. The returned HTTP request header declares that the data is encrypted on the server.

You can use one of the following methods to enable server-side encryption in the OSS console:

- Method 1: Enable server-side encryption when you create a bucket
- Method 2: Enable server-side encryption on the Basic Settings tab

Method 1: Enable server-side encryption when you create a bucket

1. Log on to the OSS console.
2. Click the + icon to create a bucket.

   Click **Create Bucket** if no buckets are available.

3. In the **Create OSS Bucket** dialog box that appears, configure corresponding parameters.

   Set **Server-Side Encryption** to **AES256**. For the settings of other parameters, see Create a bucket.

4. Click **Submit**.

Method 2: Enable server-side encryption on the Basic Settings tab

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of the target bucket.
3. Click the **Basic Settings** tab. Find the **Server-side Encryption** section.
4. Click **Configure**. Select **AES-256**.
5. Click **Save**.

> **Notice:**
> The configurations of the default encryption method for a bucket do not affect the existing objects in the bucket.

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 a bucket 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 a bucket 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 a bucket, 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.

Context

- 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.
- Parts cannot be read before they are combined into an object. Before you delete a bucket, delete all objects and parts in the bucket. Parts are generated during multipart upload.

Procedure

1. Log on to the OSS console.
2. In the left-side navigation pane, click the name of the target bucket.
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 Table 3-1: Limits on ApsaraDB RDS for MySQL.

Table 3-1: 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.
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 MylSAM 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.

Operation	Limit
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

- Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel. The URL used to access the ASCM console is in the following format: https://[IP address or domain name of the ASCM console].
- We recommend that you use the Google Chrome browser.

1. In the address bar, enter the URL used to access the ASCM console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password for logging on to the console from the operations administrator.

   Note:

   When you log on to the ASCM console for the first time, you must change the password of your username as prompted. Due to security concerns, your password must meet the minimum complexity requirements: The password must be 8 to 20 characters in length and must contain at least two of the following character types: uppercase letters, lowercase letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).
3. Click **Login** to go to the ASCM console homepage.

4. In the top navigation bar, choose Products > Database Services > ApsaraDB for RDS.

3.4 Quick Start

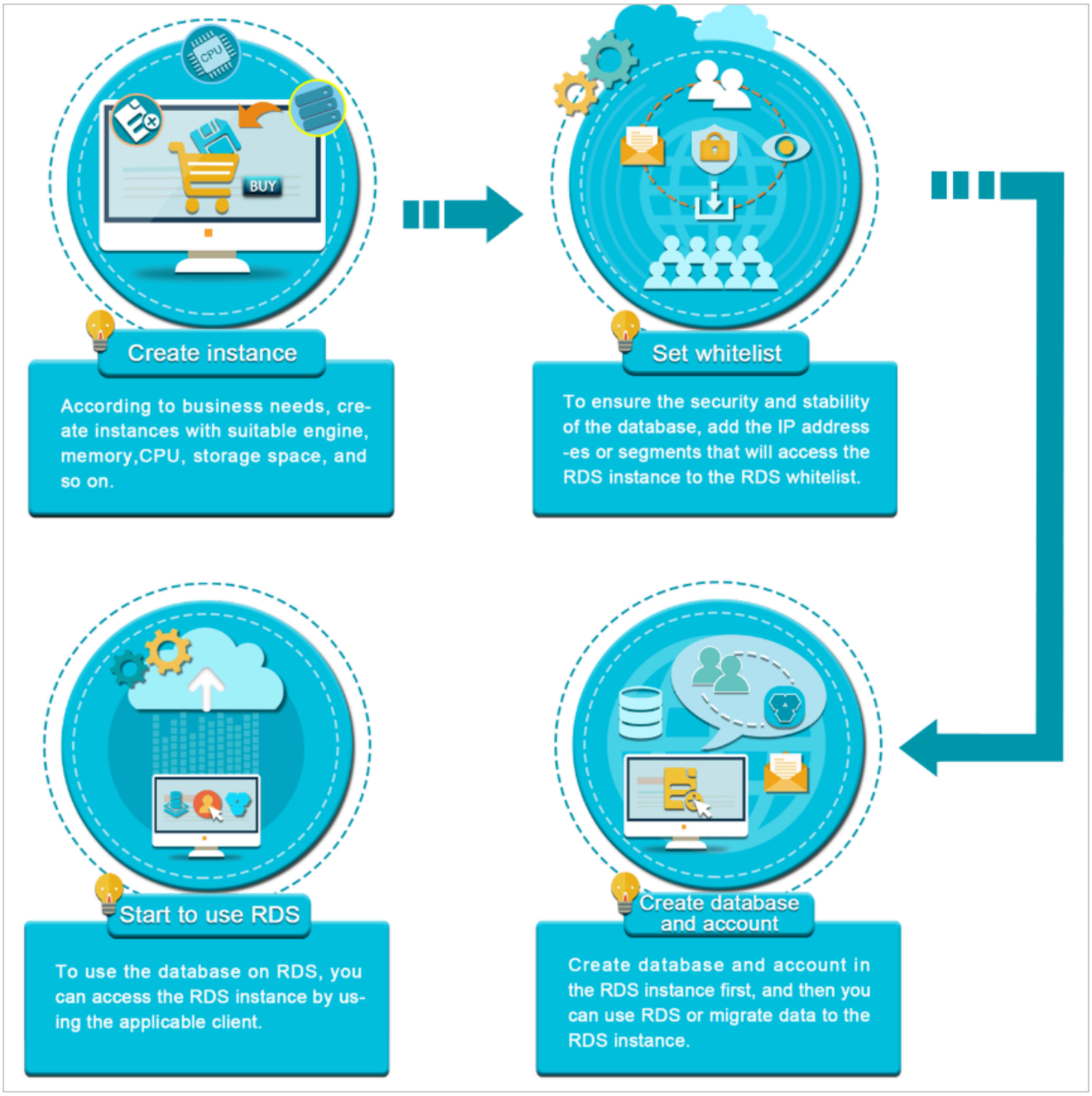
3.4.1 Procedure

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 Figure 3-1: Quick start flowchart.

Figure 3-1: 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

Before you create an RDS instance, you must apply for an Apsara Stack account.

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
Region	Region	The region where the RDS instance resides. Services in different regions cannot communicate over the internal network. After a region is selected, it cannot be changed.
Basic Settings	Organization	The organization to which the instance belongs.
	Resource Set	The resource set to which the instance belongs.

Category	Parameter	Description
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**.
	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, binlog files, and transaction files. The minimum storage space is 50 GB. You can adjust the storage space in 5 GB increments.  **Note:** For dedicated instances with local SSDs, the storage space is associated with the instance type.
Network Type	Network Access	The network access of the instance. Set the value to **Internal Network** or **Public Network**.
	Network Type	The network type of the instance, which is automatically set to **Classic Network**. Classic Network: Cloud services on a classic network are not isolated. Unauthorized access to a cloud service is blocked only by the security group or whitelist policy of the service.
	IP Whitelist	You can add IP addresses to allow them to connect to the RDS instance.

Category	Parameter	Description
Access Mode	Access Mode	RDS instances support two access modes: **Standard** and **Database Proxy**. • Standard: RDS uses SLB to eliminate the impact of instance high-availability switching on the application layer. This mode reduces the response time, but slightly increases the probability of transient disconnections and disables SQL interception. • Database Proxy: This mode prevents 90% of transient disconnections and intercepts SQL injection attacks based on semantic analysis. However, it increases the response time by over 20%.

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 only 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. Click the ID of an instance.
3. In the left-side navigation pane, click **Data Security**.

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

   Note:

   - To connect an ECS instance to an 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 also click **Create Whitelist** to create a new whitelist.

5. 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 to add to the whitelist.

   Note:

   If you add a new IP address or CIDR block to the **default** whitelist, the default address 127.0.0.1 is automatically deleted.

Create a whitelist

Note:

You can create up to 50 whitelists for an instance.

The procedure is as follows:

1. Log on to the ApsaraDB for RDS console.
2. Click the ID of an instance to go to the **Basic Information** page of the instance.
3. In the left-side navigation pane, click **Data Security**.
4. On the **Whitelist Settings** tab, click **Create Whitelist**.

5. 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.
IP Addresses	The IP addresses or CIDR blocks used to access the instance. • 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 to add to the whitelist.

6. Click **OK**.

3.4.3.2 Create a privileged account

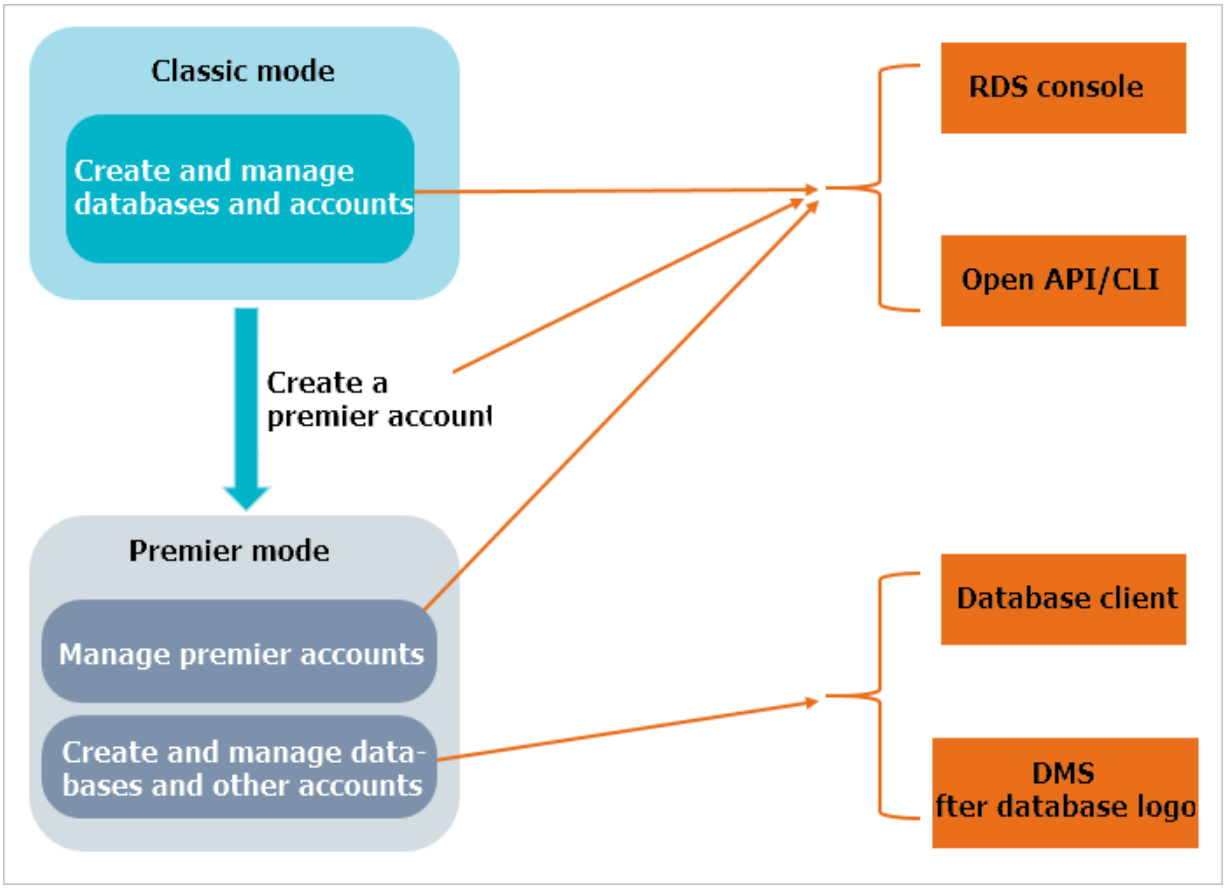
ApsaraDB for RDS supports two account management modes: classic and privileged. For ApsaraDB RDS for MySQL 5.6 or 5.7 instances, you can upgrade the account management mode from classic to privileged by creating a privileged account.

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.

Figure 3-2: 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.

Figure 3-2: Comparison of account management modes



Note:

- In an ApsaraDB RDS for MySQL 5.6 or 5.7 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, causing a transient disconnection of less than 30 seconds. To avoid service impacts from transient disconnections, choose an appropriate time and ensure that your applications can be automatically reconnected.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. Click the ID of an instance.

3. In the left-side navigation pane, click **Accounts**.
4. On the **Accounts** page, click **Create Account**.
5. Configure the following parameters.

Parameter	Description
Database Account	Enter an account name. The requirements are as follows: • The name must be 2 to 16 characters in length. • It must start with a letter and end with a letter or digit. • It can contain lowercase letters, digits, and underscores (_).
Account Type	Select **Privileged Account**.
Password	Enter an account password. The requirements are as follows: • 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	Enter the password again.
Description	Enter information about the account to facilitate subsequent management. The description can be up to 256 characters in length.

6. 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

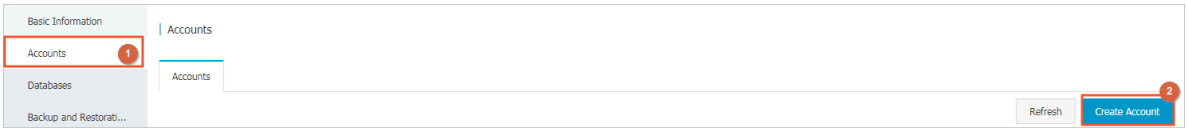
To migrate data from an on-premises database to an ApsaraDB for RDS instance, you must create a database and an account in the RDS instance identical to those in the on-premises database. Databases in an instance share all resources that belong to the instance. You can create up to 500 databases and 500 accounts for an ApsaraDB RDS for MySQL 5.6 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:

To ensure database security, set strong account passwords and change the passwords on a regular basis.

1. Log on to the ApsaraDB for RDS console.
2. Click the ID of an instance.
3. In the left-side navigation pane, click **Accounts**.
4. Click **Create Account**.



5. Configure the following parameters.

Parameter	Description
Database Account	Enter an account name. The requirements are as follows: • The name must be 2 to 16 characters in length. • It must start with a letter and end with a letter or digit. • It can contain lowercase letters, digits, and underscores (_).
Account Type	Select **Standard Account**.
Authorized Databases	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. **a.** Select one or more databases from the left-side box, and click **Add** to add them to the right-side box. **b.** In the right-side box, select **Read/Write** or **Read-only** for a 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 right-side box. The button may appear as **Set All to Read/Write**. **Note:** The button in the upper-right corner changes after you click. For example, after you click **Set All to Read/Write**, the button changes to **Set All to Read-only**.

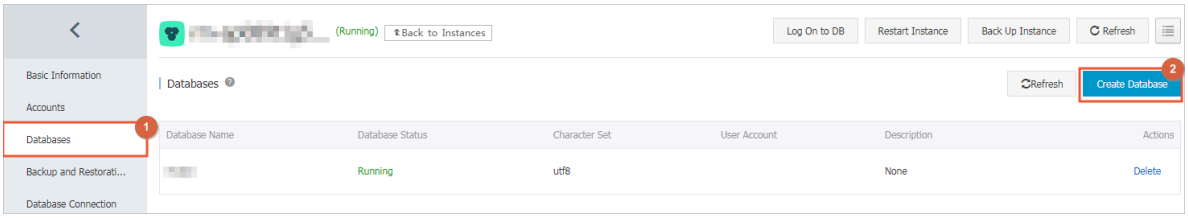
Parameter	Description
Password	Enter an account password. The requirements are as follows: • The password must be 8 to 32 characters in length. • It must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. • Special characters include ! @ # $ % ^ & * ( ) _ + - =
Re-enter Password	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.

6. 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. Click the ID of an instance.
3. In the left-side navigation pane, click **Databases**.
4. Click **Create Database**.



5. Configure the following parameters.

Parameter	Description
Database Name	• The database name must be 2 to 64 characters in length. • It must start with a letter and end with a letter or digit. • It can contain lowercase letters, digits, underscores (_), and hyphens (-). • Each database name must be unique in an instance.
Supported Character Set	Select utf8, gbk, latin1, utf8mb4, or all of them. If you want to use other character sets, select **all**, and then select the required character set from the list.

Parameter	Description
Description	Optional. Enter information about the database to facilitate subsequent management. The description can be up to 256 characters in length.

6. 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.

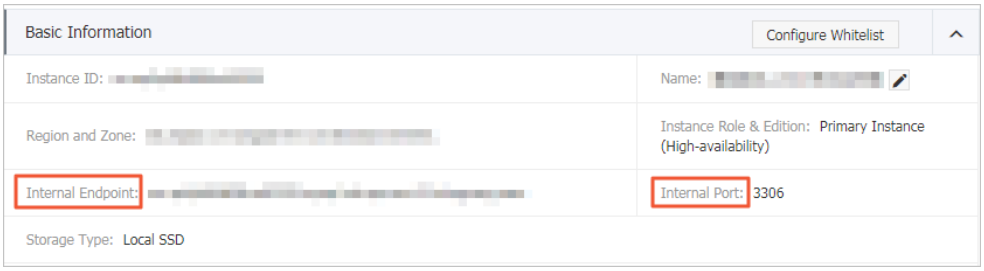
If you need to connect an ECS instance to an ApsaraDB for RDS instance, you must make sure that both instances are in classic networks or in the same VPC, and the IP address of the ECS instance is correctly configured in the RDS whitelist.

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)**.

Parameter	Description
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: **a.** Log on to the ApsaraDB for RDS console. **b.** Find the target instance and click its ID. **c.** 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

Before you create an ApsaraDB for RDS instance, apply for an Apsara Stack account.

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. The name must be 2 to 64 characters in length and can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter.
	Database Engine	The engine of the database. Select MySQL.
	Engine Version	The version of the database engine. Valid values: 5.6 and 5.7.
	Edition	The edition of the instance. The actual edition is displayed on the console.
	Instance Type	The type of the instance. Memory size determines the maximum number of connections and IOPS. The actual values are displayed on the console.
	Storage	The storage space of the instance, including the space for data, system files, binlog files, and transaction files. You can adjust the storage space in 5 GB increments.
Network Type	Network Type	The network type of the instance, which is automatically set to **Classic Network**. Classic Network: Cloud services on a classic network are not isolated. Unauthorized access to a cloud service is blocked only by the security group or whitelist policy of the service.
	IP Whitelist	You can add IP addresses to allow them to connect to the ApsaraDB for RDS instance.

Category	Parameter	Description
Access Mode	Access Mode	The access mode, which is automatically set to **Standard**. Standard: ApsaraDB for RDS uses SLB to eliminate the impact of high-availability switching between database engines on the application layer. This mode reduces the response time, but slightly increases the probability of transient disconnections 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 must be 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. Click the ID of an instance.
3. In the upper-right corner of the page, click **Restart Instance**.
4. In the **Restart Instance** message that appears, click **Confirm** to restart the instance.

3.5.4 Modify configurations

You can modify configurations of your instance, such as specifications 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. Click the ID of an instance or click **Manage** in the Actions column corresponding to the instance.
3. In the **Configuration Information** section of the **Basic Information** page, click **Change Specifications**.
4. In the dialog box that appears, click **Next**.
5. On the **Change Specifications** page, set **Edition**, **Instance Type**, and **Storage**.
6. 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 as needed.

Context

To ensure the stability of ApsaraDB for RDS instances, the backend system performs maintenance of 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.

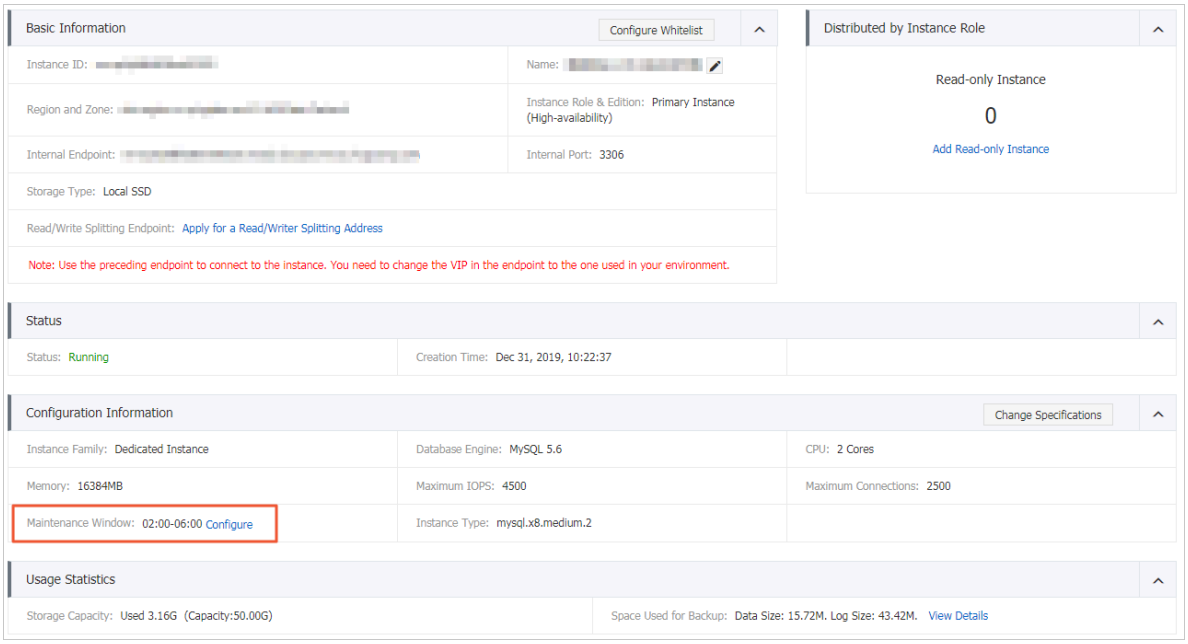
Precautions

- To ensure the stability of the maintenance process, the instance will enter the **Instance Maintaining** state before the maintenance time. When the instance is in this state, access to data in the database and query operations such as performance monitoring are not affected. However, apart from 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 you configure automatic reconnection policies for your applications to avoid service disruptions.

Procedure

1. Log on to the ApsaraDB for RDS console.

2. Click the ID of an instance or click **Manage** in the Actions column corresponding to the instance.
3. In the **Configuration Information** section, click **Configure** to the right of **Maintenance Window**.



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

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

3.5.6 Change the data replication mode

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

You can use the following methods to replicate data:

- **Semi-sync**

  After an application-initiated update is completed on the primary instance of a cluster , logs are synchronized to all secondary databases. This transaction is considered committed after at least one node in the cluster has received the logs. This way, there is no need to wait for the logs to be applied.

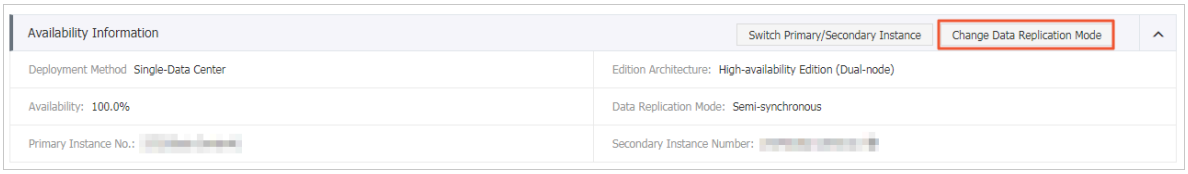
  If the secondary instances are unavailable or a network exception occurs between the primary and secondary instances, semi-synchronous replication will degrade to Async mode.

- **Async**

  When an application initiates an update request to add, delete, or modify data, the primary instance responds to the application immediately after completing the operation. 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 will remain consistent even if the primary instance is unavailable.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. Click the ID of an instance.
3. In the left-side navigation pane, click **Service Availability**.
4. Click **Change Data Replication Mode**.



5. In the dialog box that appears, select a data replication mode and click **OK**.

3.5.7 Release an instance

You can manually release instances as needed.

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. In the Actions column corresponding to the instance you want to release, choose **More** > **Release Instance**.
3. In the **Release Instance** message that appears, click **Confirm**.

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.

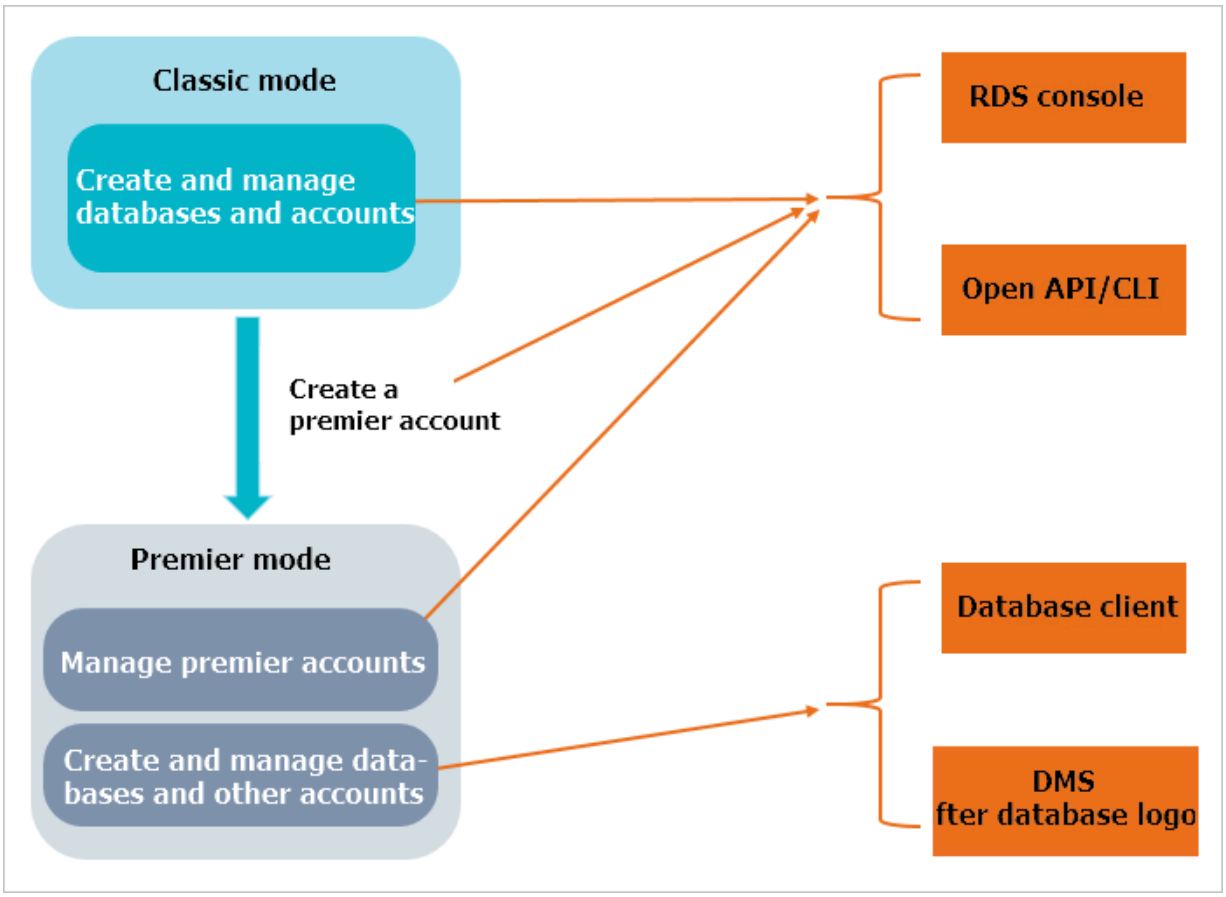
In the privileged mode, the first account that you create is the initial account. You must use the ApsaraDB for RDS console or API operations to create and manage the initial account. Log on to a database with your initial account. You can then use SQL statements or Data Management (DMS) to create and 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 Figure 3-3: Difference between standard and privileged accounts.

Figure 3-3: 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 your password

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

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

1. Log on to the ApsaraDB for RDS console.
2. Click the ID of an instance.
3. In the left-side navigation pane, click **Accounts**.
4. Find the target account and click **Reset Password** in the Actions column.
5. 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 instances at any time when using ApsaraDB for RDS.

> **Note:**
> You can modify the permissions of a standard account as needed. 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. Click the ID of an instance.
3. In the left-side navigation pane, click **Accounts**.
4. Find the target account and click **Modify Permissions** in the Actions column.

5. Configure the following parameters.

Parameter	Description
Authorized 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 Set All to Read/Write or Set All to Read-only to set the permissions of the account on all authorized databases. • **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.

6. Click **OK**.

3.6.4 Delete an account

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

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

1. Log on to the ApsaraDB for RDS console.
2. Click the ID of an instance.
3. In the left-side navigation pane, click **Accounts**.
4. Find the account you want to delete and click **Delete** in the Actions column.
5. In the message that appears, click **Confirm**.

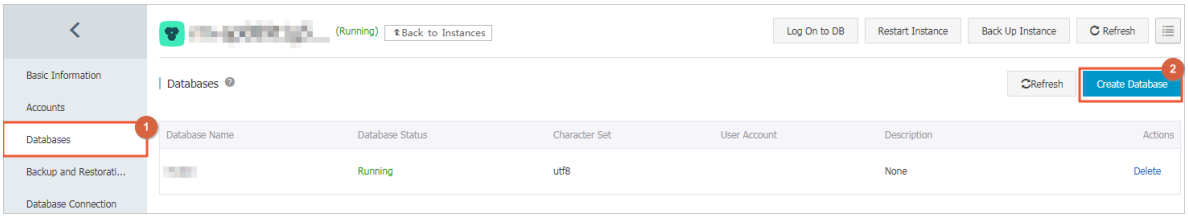
3.7 Databases

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. Click the ID of an instance.
3. In the left-side navigation pane, click **Databases**.
4. Click **Create Database**.



5. Configure the following parameters.

Parameter	Description
Database Name	• The database name must be 2 to 64 characters in length. • It must start with a letter and end with a letter or digit. • It can contain lowercase letters, digits, underscores (_), and hyphens (-). • Each database name must be unique in an instance.
Supported Character Set	Select utf8, gbk, latin1, utf8mb4, or all of them. 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.

6. Click **Create**.

3.7.2 Delete a database

You can delete out-of-date databases in the ApsaraDB for RDS console.

1. Log on to the ApsaraDB for RDS console.
2. Click the ID of an instance.
3. In the left-side navigation pane, click **Databases**.
4. Find the database you want to delete and click **Delete** in the Actions column.
5. 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. Click the ID of an instance.
3. In the left-side navigation pane, click **Database Connection**.
4. Click **Change Endpoint**.
5. In the dialog box that appears, set 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 65535.

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 for you to monitor the status of your instances.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. Click the ID of an instance.
3. In the left-side navigation pane, click **Monitoring and Alerts**.

4. On the **Monitoring and Alerts** page, select **Resource Monitoring** or **Engine Monitoring**, and select a time range to view the corresponding monitoring data. The specific metrics are described as follows.

Category	Metric	Description
Resource Monitoring	Disk Space (MB)	The disk space usage of the instance, including: • Instance size • Data usage • Log size • Temporary file size • Other system file size Unit: MB.
	IOPS (Input/Output Operations per Second)	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 (excluding 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.
Engine Monitoring	Transactions per Second (TPS)/ Queries per Second (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.

Category	Metric	Description
	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.
	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 SQL statements include: • Insert • Delete • Insert_Select • Replace • Replace_Select • Select • Update
	MySQL_RowDML	The number of operations that InnoDB performs per second, including: • 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, including: • 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

Category	Metric	Description
	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:

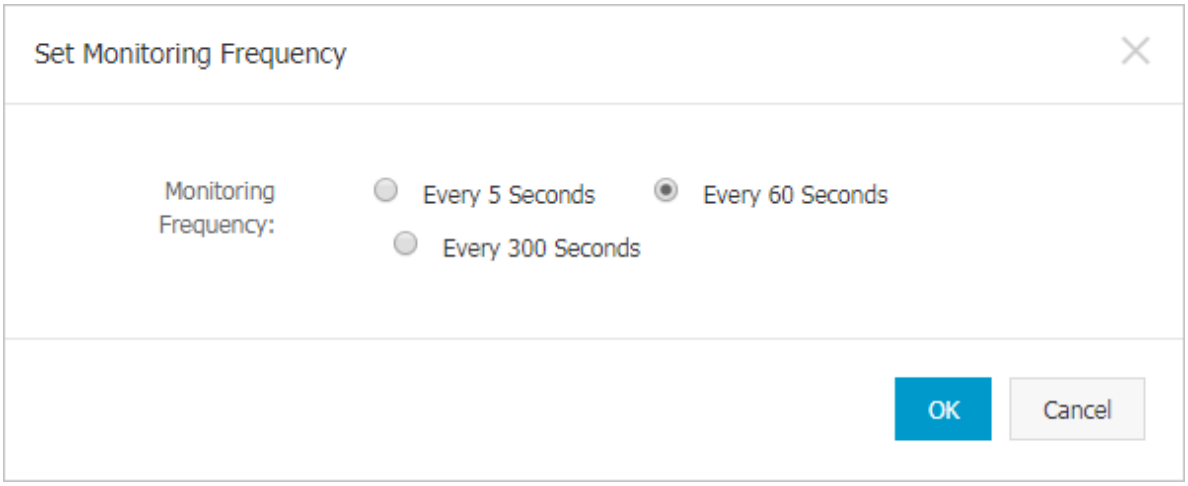
- Every 5 seconds for the first 7 days. After seven days, performance metrics are monitored every minute.
- Every 60 seconds for 30 days.
- Every 300 seconds for 30 days.

The following table lists the monitoring configuration policies in detail.

Instance type	Every 5 seconds	Every 60 seconds	Every 300 seconds
Basic Edition	Not supported	Supported for free	Default configuration
High-availability Edition and Enterprise Edition (formerly known as Finance Edition): less than 8 GB memory	Not supported	Supported for free	Default configuration
High-availability Edition and Enterprise Edition (formerly known as Finance Edition): at least 8 GB memory	Supported on a pay-as-you-go basis	Default configuration	Supported for free

1. Log on to the ApsaraDB for RDS console.
2. Click the ID of an instance.
3. In the left-side navigation pane, click **Monitoring and Alerts**.
4. On the **Resource Monitoring** tab, click **Set Monitoring Frequency**.

5. In the **Set Monitoring Frequency** dialog box, select the monitoring frequency you want.



6. Click **OK**.

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

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 only 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. Click the ID of an instance.
3. In the left-side navigation pane, click **Data Security**.
4. On the **Whitelist Settings** tab, click **Edit** corresponding to the **default** whitelist.

   Note:

   - To connect an ECS instance to an 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 also click **Create Whitelist** to create a new whitelist.
5. 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 to add to the whitelist.

   Note:
   If you add a new IP address or CIDR block to the **default** whitelist, the default address 127.0.0.1 is automatically deleted.

Create a whitelist

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

The procedure is as follows:

1. Log on to the ApsaraDB for RDS console.
2. Click the ID of an instance to go to the **Basic Information** page of the instance.
3. In the left-side navigation pane, click **Data Security**.

4. On the **Whitelist Settings** tab, click **Create Whitelist**.
5. 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.
IP Addresses	The IP addresses or CIDR blocks used to access the instance. • 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 to add to the whitelist.

6. 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 by using 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 by 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. Therefore, you may find that some binlog files 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.

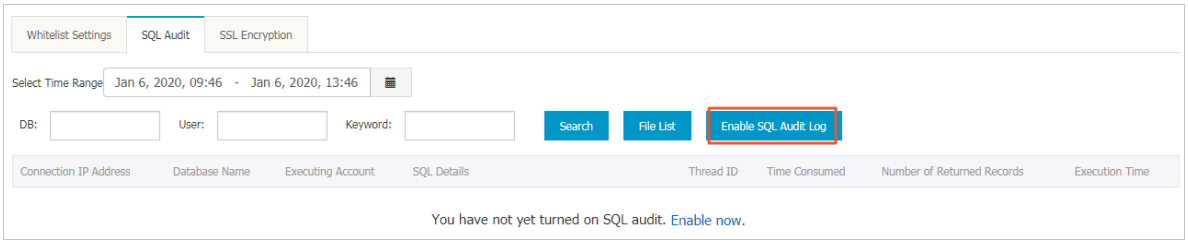
> **Note:**
> You can click **Upload Binlogs** on the **Backup and Restoration** page.

Precautions

- SQL audit is disabled by default. SQL audit does not affect instance performance.
- SQL audit records are retained for 30 days.
- 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. Click the ID of an instance.
3. In the left-side navigation pane, click **Data Security**.
4. Click the **SQL Audit** tab.



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

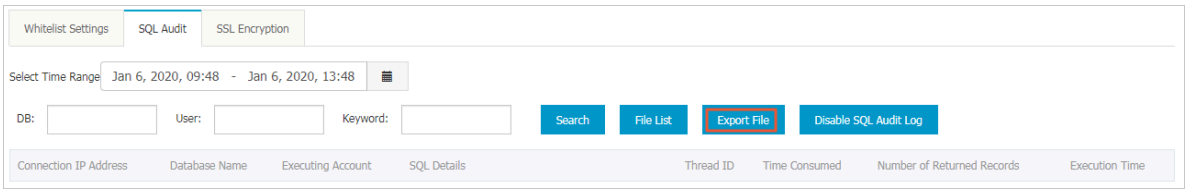
   After enabling SQL audit, you can query SQL information based on conditions such as time, database, user, and keyword.

Disable SQL audit

> **Note:**
> If SQL audit is disabled, all the SQL audit records are cleared. We recommend that you export and store the audit records locally before you disable SQL audit.

You can disable SQL audit when you do not need it to avoid charges. To disable SQL audit, follow these steps:

1. Log on to the ApsaraDB for RDS console.
2. Click the ID of an instance.
3. In the left-side navigation pane, click **Data Security**.
4. Click the **SQL Audit** tab.



5. Click **Export File** to export and store the SQL audit content locally.
6. After the file is exported, click **Disable SQL Audit Log**.
7. In the message that appears, click **Confirm**.

3.11 Database backup and restoration

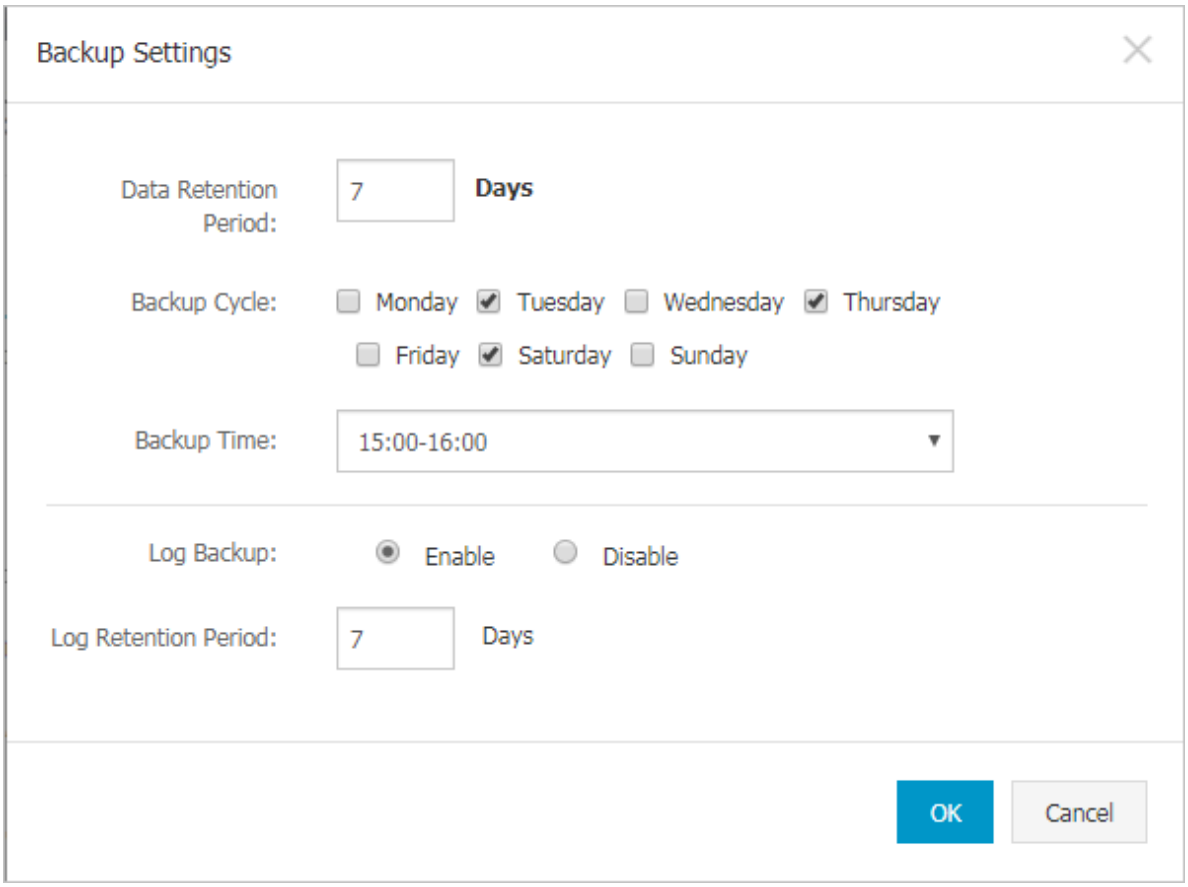
3.11.1 Automatic backup

RDS 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. Click the ID of an instance.
3. In the left-side navigation pane, click **Backup and Restoration**.
4. Click the **Backup Settings** tab.
5. 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 of this Wednesday, the system will still back up data during 19:00-20:00 this Wednesday.



6. 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.
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.

Parameter	Description
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.

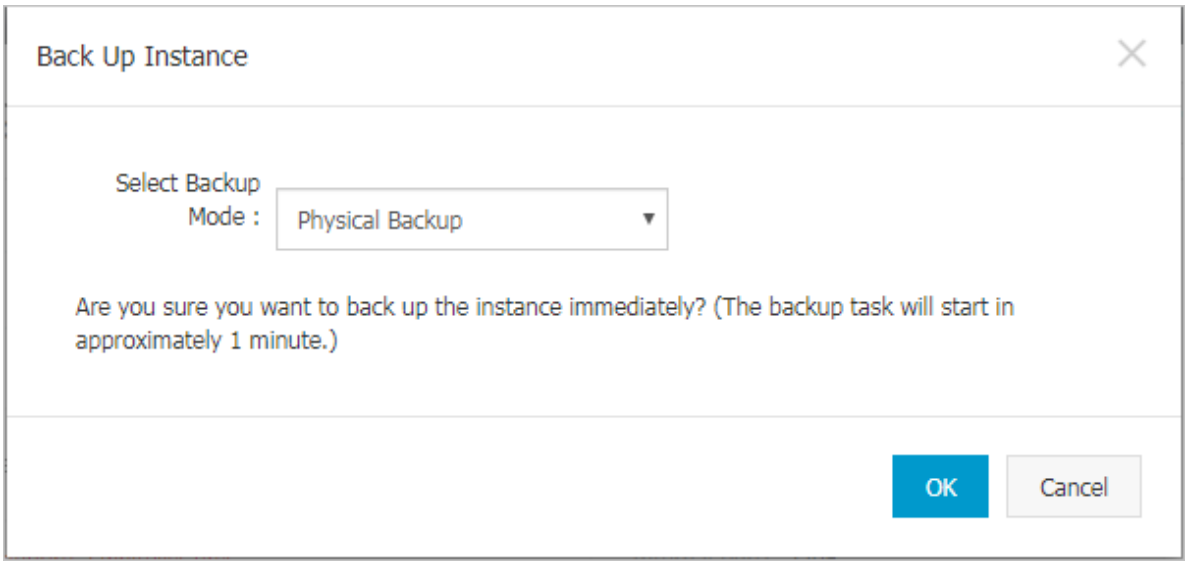
7. After you set the preceding parameters, click **OK**.

3.11.2 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. Click the ID of an instance.
3. In the upper-right corner, click **Back Up Instance**.



4. Set the backup mode and backup policy, and click **OK**.

Parameter	Value	Description
Select Backup Mode	**Physical Backup**	This mode dumps the physical files of the RDS database, such as data files, control files, and log files. In case the database fails, these files can be used to restore data.

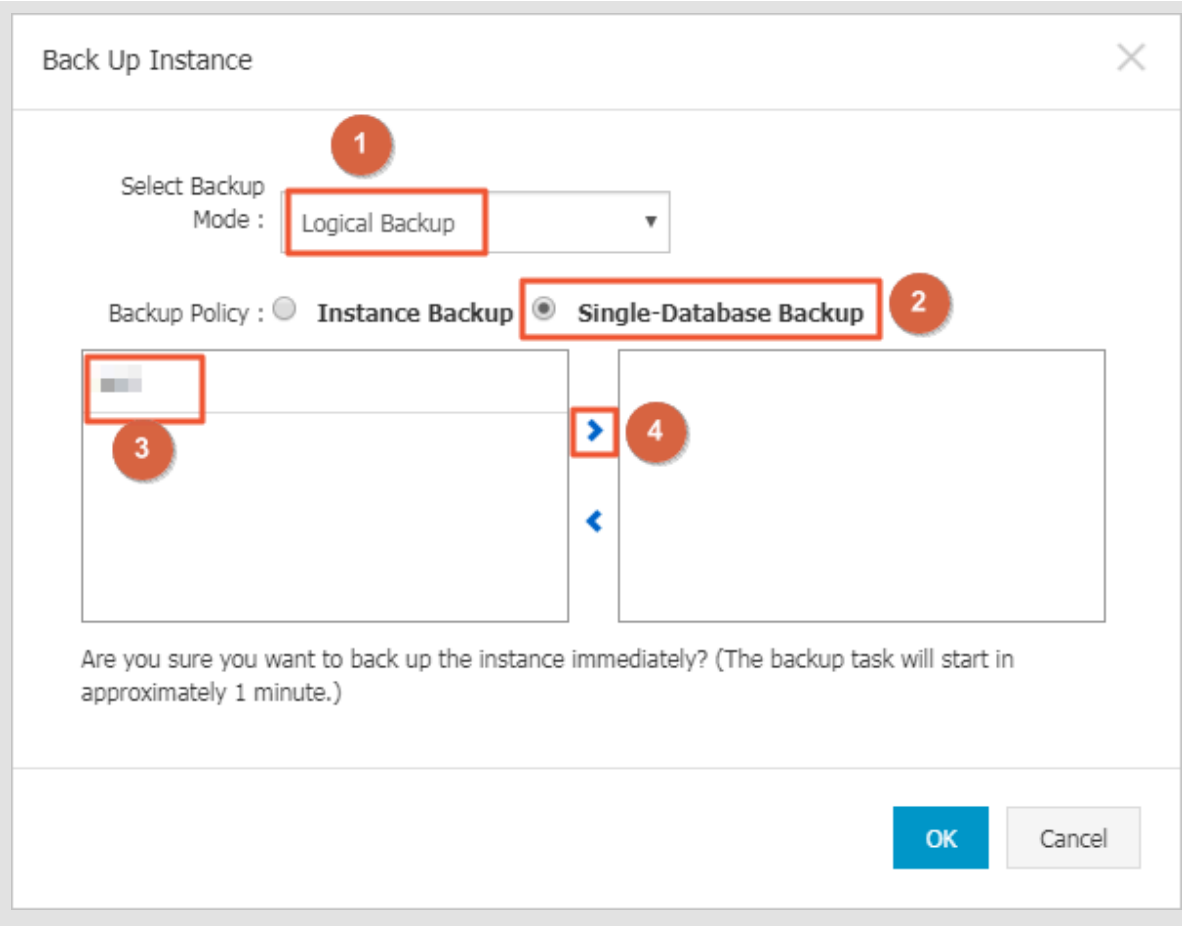
Parameter	Value	Description
	Logical Backup	This mode stores all schema definition statements and data insertion statements of the RDS database. You can execute these SQL statements to restore data. A database that is exactly the same as the original database is created. The **Logical Backup** mode provides the following backup policies: • **Instance Backup**: backs up an entire instance. • **Single-Database Backup**: backs up a single database.

5. Set the backup mode and backup policy, and click **OK**.

Parameter	Value	Description
Backup Mode	**Physical Backup**	This mode dumps the physical files of the RDS database, such as data files, control files, and log files. In case the database fails, these files can be used to restore data.
	Logical Backup	This mode stores all schema definition statements and data insertion statements of the RDS database. You can execute these SQL statements to restore data . A database that is exactly the same as the original database is created.

Note:

If you choose **Logical Backup** > **Single-Database Backup**, select the database to back up on the left, click **>** to add the database to the list on the right, and then click **OK**.



3.11.3 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 data of the primary instance. The copied data 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, such as the instance edition, zone, network type, instance type, and storage space, can be different. 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 that 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. Click the ID of an instance.
3. In the left-side navigation pane, click **Backup and Restoration**.
4. On the Data Backup tab, find the backup set you want to restore and click **Restore** in the Actions column.
5. In the dialog box that appears, select **Restore Database** and click **OK**.
6. On the **Restore RDS Instance** page, configure the following parameters.

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

Category	Parameter	Description
	Backup Set	Select 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 automatically set to **MySQL**.
	Engine Version	The version of the database engine, which is the same as that of the primary instance and automatically set.
	Edition	The edition of the database. The available database editions are displayed on the Restore RDS Instance page.
	Instance Type	The type of the cloned instance. **Note:** We recommend that you select the instance type and storage space that are higher than those of the primary instance. Otherwise, the data restoration may take a long time due to performance limitations.
	Storage	The storage space of the instance, including the space for data, system files, binary log files, and transaction files. You can adjust the storage space in 5 GB increments.
Network Type	Network Type	The network type of the instance, which is automatically set to **Classic Network**.

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

3.11.4 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

- Your RDS instance runs one of the following MySQL versions:
  - ApsaraDB RDS for MySQL 5.6 High-availability Edition
  - ApsaraDB RDS for MySQL 5.7 High-availability Edition (based on local SSDs)
- You can restore individual databases or tables only for instances that have less than 50,000 tables. If an instance has more than 50,000 tables, this function is not available.
- Restoration of individual databases or tables is enabled.

  Note:

  - After you enable the restoration of individual databases or tables, this function cannot be disabled and backup file formats will be changed.
  - By default, this function is enabled for new instances and cannot be disabled.
- If you 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 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 disconnections of ApsaraDB for RDS. Make sure that your application supports automatic reconnection.
- 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 backup file format. Pay attention to the space used for backup.
- You can select a maximum of 50 databases or tables at a time.

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 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 when 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 when you set **Restore Method** to By Time.
Databases and Tables to Restore	Select the databases or tables you want to restore.

Parameter	Description
Selected Databases and Tables	• The selected databases and tables are displayed. You can set 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**.

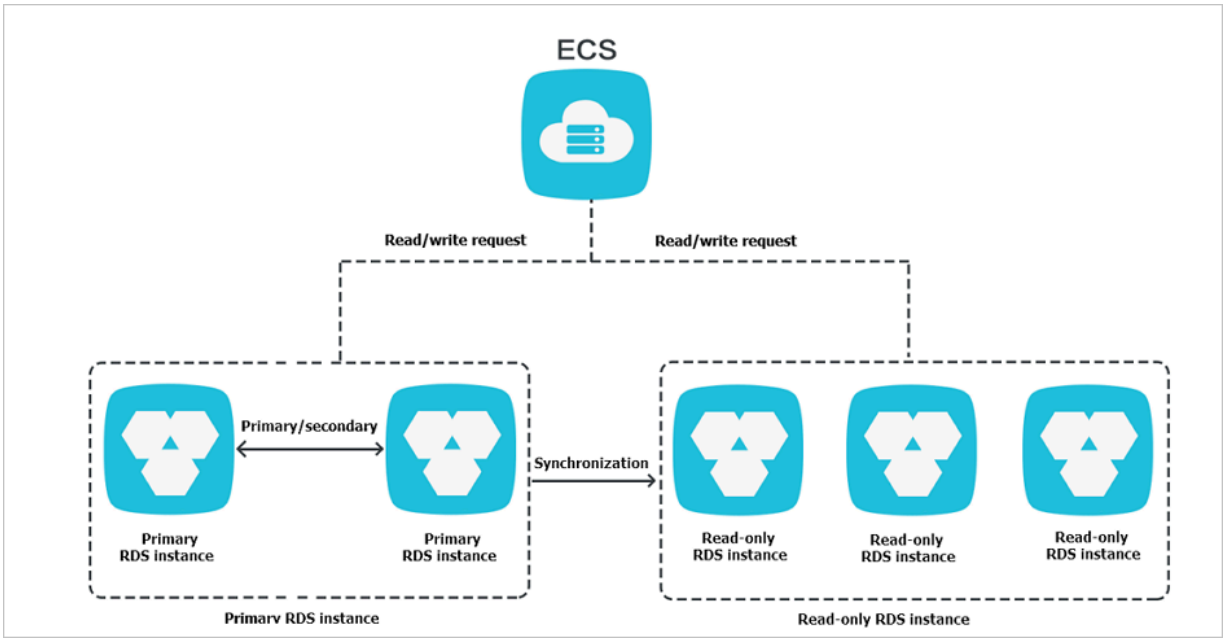
3.12 Read-only instances

3.12.1 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.

3.12.2 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. Click the ID of an instance.
3. On the **Basic Information** page, click **Add Read-only Instance** on the right.
4. 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 facilitate 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.

Category	Parameter	Description
Network Type	Network Type	The network type of the read-only instance, which is automatically set to **Classic Network**.

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

3.12.3 View the details of read-only instances

3.12.3.1 View instance details through a read-only instance

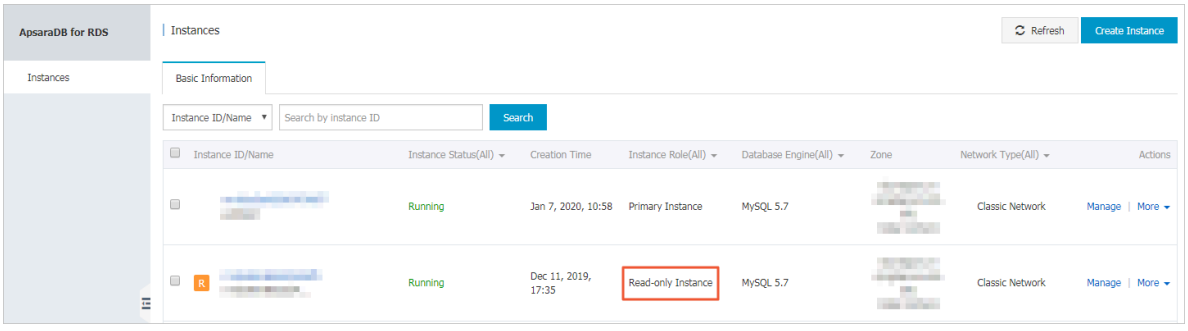
You can go to the read-only instance management page from the Instances page or the Read-Only Instance page of the primary instance. Read-only instances are managed in the same way as ordinary instances. The read-only instance management page shows the management operations that can be performed. This 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 that appears allows you to manage the read-only instance.

   In the instance list, **Instance Type** of read-only instances is displayed as **Read-only Instance**, as shown in Figure 3-4: View read-only instances.

Figure 3-4: View read-only instances



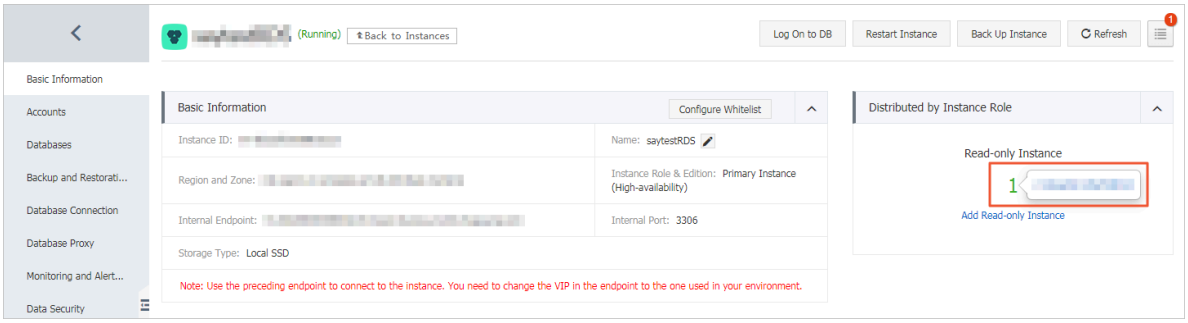
3.12.3.2 View instance details through the primary instance

You can go to the read-only instance management page from the Instances page or the Read-Only Instance page of the primary instance. Read-only instances are managed in the same way as ordinary 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 page of the primary instance.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. Click the ID of an instance.
3. 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.



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

3.13 Logs

This topic describes how to query error logs, slow query log details, and primary/secondary instance switching logs of an ApsaraDB for RDS instance.

1. Log on to the ApsaraDB for RDS console.
2. Click the ID of an instance to go to the **Basic Information** page.
3. In the left-side navigation pane, click **Logs**.
4. On the **Logs** page, click the **Error Logs**, **Slow Log Details**, or **Primary/Secondary Switching Logs** tab, select a time range, and then click **Search**.

Tab	Description
Error Logs	Records database running errors that occurred within the last month.

Tab	Description
Slow Log Details	Records SQL statements that took longer than one second to execute from 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 High-availability Edition instances.

3.14 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 provided by Alibaba Cloud. DTS supports data transmission between various types of data sources, such as relational databases.

DTS provides data transmission capabilities such as data migration and change tracking. DTS can be used in many scenarios, such as interruption-free data migration, geo-disaster recovery, cross-border data synchronization, and cache updates. DTS helps you build a data architecture that features high availability, scalability, and security.

- DTS allows you to simplify data transmission and focus on business development.
- DTS supports MySQL as the data source type.

4.2 Log on to the DTS console

This topic uses the Google Chrome browser as an example to describe how to log on to the DTS console.

Prerequisites

- Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel. The URL used to access the ASCM console is in the following format: https://[IP address or domain name of the ASCM console].
- We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to access the ASCM console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password for logging on to the console from the operations administrator.

   Note:
   When you log on to the ASCM console for the first time, you must change the password of your username as prompted. Due to security concerns, your password must meet the

minimum complexity requirements: The password must be 8 to 20 characters in length and must contain at least two of the following character types: uppercase letters, lowercase letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

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

4.3 Data migration

4.3.1 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. When you migrate data between user-created MySQL databases, you can select all of the supported migration types to ensure service continuity.

Prerequisites

- The version of the user-created MySQL database is 5.1, 5.5, 5.6, 5.7, or 8.0.
- The available storage space of the destination MySQL database is larger than the total size of the data in the source MySQL database.
- The service port of the user-created MySQL database is accessible over the Internet.

Migration types

DTS supports schema migration, full data migration, and incremental data migration between MySQL databases.

- Schema migration

  DTS migrates the schemas of the required objects to the destination database. DTS supports schema migration for the following types of objects: table, view, trigger, stored procedure, and function.

- Full data migration

  DTS migrates historical data of the required objects from the source MySQL database to the destination MySQL database. If you also select incremental data migration, non-transactional tables without primary keys are locked during full data migration. Data cannot be written to these locked tables, and the locking duration depends on the data

volume of the tables. The locks are released only after these tables are migrated. This ensures data consistency.

- Incremental data migration

  During incremental data migration, DTS synchronizes data changes to the destinatio n database. However, DTS does not synchronize data definition language (DDL) operations that are performed during data migration to the destination database.

Limits

- DDL operations that are performed during incremental data migration cannot be synchronized to the destination database.
- DTS does not support schema migration for events.
- If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.
- If you select incremental data migration, the binary logging feature of the source MySQL database must be enabled and the value of the binlog_format parameter must be set to row. If the version of the source MySQL database is 5.6 or later, the value of the binlog_row_image parameter must be set to full.

Preparations

Before you migrate data, create accounts for the source and destination MySQL databases . You must also grant the read/write permissions to the accounts. The following table lists the permissions that are required for database accounts when different migration types are used.

Table 4-1: Migration types and required permissions

Database type	Structure migration	Full data migration	Incremental data migration
Source MySQL database	SELECT	SELECT	• SELECT • REPLICATION SLAVE • REPLICATION CLIENT

Database type	Structure migration	Full data migration	Incremental data migration
Destination MySQL database	• SELECT • REPLICATION SLAVE • REPLICATION CLIENT	• SELECT • REPLICATION SLAVE • REPLICATION CLIENT	• SELECT • REPLICATION SLAVE • REPLICATION CLIENT

1. Run the following statement to create an account for an on-premises database:

   ```
   CREATE USER 'username'@'host' IDENTIFIED BY 'password';
   ```

   Parameters

   - username: the account that you want to create.
   - host: the host from which the account is authorized to log on to the database. To authorize a local user to log on to the database, set the value of this parameter to

localhost. To authorize the account to log on to the database from all hosts, set the value of this parameter to a percent sign (**%**).

- password: the logon password for the account.

For example, you can run the following statement to create an account named **William**. The password is **Changme123**. The account is authorized to log on to the database from all hosts.

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

2. Run the following statement to grant permissions to the account of the on-premises database:

```
GRANT privileges ON databasename.tablename TO 'username'@'host' WITH GRANT OPTION;
```

Parameters

- privileges: the operations that the account is authorized to perform, such as the SELECT, INSERT, and UPDATE operations. To authorize the account to perform all operations, set the value of this parameter to **ALL**.
- databasename: the database name. To authorize the account to perform operations on all databases, set the value of this parameter to an asterisk (*****).
- tablename: the table name. To authorize the account to perform operations on all tables, set the value of this parameter to an asterisk (*****).
- username: the account to which you want to grant permissions.
- host: the host from which the account is authorized to log on to the database. To authorize a local user to log on to the database, set the value of this parameter to **localhost**. To authorize the account to log on to the database from all hosts, set the value of this parameter to a percent sign (**%**).
- WITH GRANT OPTION: the permissions that authorize the account to use the GRANT statement. This parameter is optional.

For example, you can run the following statement to grant all permissions on tables and databases to the **William** account. The account is authorized to log on to the database from all hosts.

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

Note:

If you want to perform incremental data migration, you must enable binary logging for the on-premises database and set the relevant parameters.

3. Run the following statement to check whether binary logging is enabled for the on-premises database:

```
show global variables like "log_bin";
```

If the query result is log_bin=OFF, binary logging is disabled for the on-premises database. To ensure that incremental data generated during data migration is synchronized to the destination database, modify the following parameters in the my.cnf configuration file:

```
log_bin=mysql_bin binlog_format=row server_id=An integer greater than 1 binlog_row_image=full //If the version of the on-premises MySQL database is later than 5.6, you must set this parameter.
```

4. Run the following statement to restart the MySQL process:

```
$mysql_dir/bin/mysqladmin -u root -p shutdown $mysql_dir/bin/safe_mysqld &
```

The mysql_dir parameter indicates the installation directory of MySQL.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Migration**. On the Migration Tasks page, click **Create Migration Task** in the upper-right corner. In the **Create DTS Instances** dialog box, set the required parameters.

   The following table describes the required parameters.

   Table 4-2: Parameters

Parameter	Description
Feature	The feature specified by the system. In this case, the value is **Data Migration**.
Region	The region where the instance resides.
Instances to Create	The number of instances that you want to create.

3. Click **OK**.

4. Find the data migration task and click **Configure Migration Task** in the Actions column.

   On the **Create Migration Task** page, configure the source and destination databases.

   The following table describes the required parameters.

Section	Parameter	Description
N/A	**Task Name**	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification.
Source Database	**Instance Type**	The instance type of the source database. Select **User-Created Database with Public IP Address**.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**.
	Database Type	The type of the source database. Select **MySQL**.
	Hostname or IP Address	The endpoint that is used to connect to the source database.
	Port Number	The port number of the source database. The default port number for a MySQL database is 3306.
	Database Account	The account that is used to log on to the source database.
	Database Password	The password for the account that is used to log on to the source database.
Destination Database	**Instance Type**	The instance type of the destination database. Select **User-Created Database with Public IP Address**.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**.
	Database Type	The type of the destination database. Select **MySQL**.
	Hostname or IP Address	The endpoint that is used to connect to the destination database.
	Port Number	The port number of the destination database. The default port number for a MySQL database is 3306.

Section	Parameter	Description
	Database Account	The account that is used to log on to the destination database.
	Database Password	The password for the account that is used to log on to the destination database.

Note:
After the source and destination databases are configured, you can click **Test Connectivity** to verify whether the specified parameters are valid.

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

Note:
To change the name of an object that is migrated to the destination database, move the pointer over the object in the **Selected** section and click **Edit**.

Parameter	Description
Migration Types	• To perform only full data migration, select Schema Migration and Full Data Migration. **Note:** To ensure data consistency, do not write data into the source MySQL database during full data migration. • To migrate data with minimal downtime, select Schema Migration, Full Data Migration, and Incremental Data Migration.

Parameter	Description
Objects	Select objects from the **Available** section and click the right arrow (>) to move the objects to the **Selected** section. • You can select columns, tables, or databases as the objects to be migrated. • After an object is migrated to the destination MySQL database, the name of the object remains the same as that in the source MySQL database. You can change the names of the objects that are migrated to the destination MySQL database by using the object name mapping feature. For more information about how to use this feature, see Object name mapping. • If you use the object name mapping feature on an object, objects that are dependent on the object may fail to be migrated.

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

   Note:

   - Before you can start the data migration task, a precheck is performed. You can start the data migration task only after the task passes the precheck.
   - If the task fails to pass the precheck, click the info icon next to each failed item to view details. Troubleshoot the issues based on the cause of failure and perform a precheck again.

8. After the task passes the precheck, start the task again. After the task is started, check the migration status and progress on the **Migration Tasks** page.

4.3.2 Precheck items

4.3.2.1 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.

4.3.2.2 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.

4.3.2.3 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.

4.3.2.4 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.

4.3.2.5 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.

4.3.2.6 Permissions

Source database permissions

DTS checks whether the account of the source database has the required permissions to perform data migration. For more information, see Table 4-1: Migration types and required permissions.

Destination database permissions

DTS checks whether the account of the destination database has the required permissions to perform data migration. For more information, see Table 4-1: Migration types and required permissions.

4.3.2.7 Schema name conflict

DTS checks whether the destination database contains objects that have the same names as objects to be migrated. If the destination database contains objects that have the same names as 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 database and table 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.

4.3.2.8 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.

4.3.2.9 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.

4.3.2.10 Source database version

DTS checks whether the version of the source database is supported. The table Table 4-3: Source database types and versions lists the source database versions that are supported by DTS.

Table 4-3: 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.

4.3.3 Object name mapping

DTS provides the object name mapping feature. You can use this feature to change the name of an object to be 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:
We recommend that you do not use the object name mapping feature after a data migration task is started. Otherwise, data may fail to be migrated.

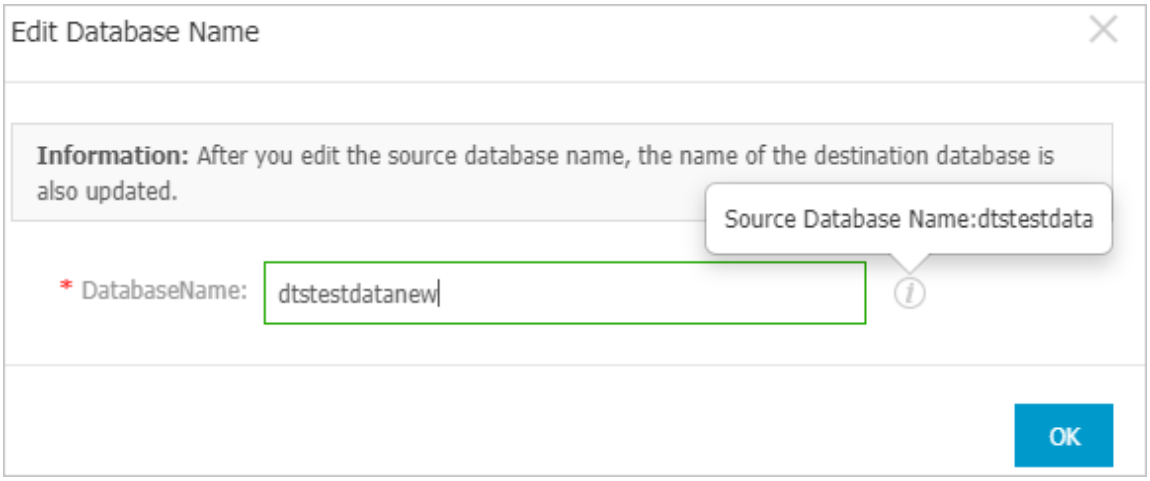
Procedure

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.

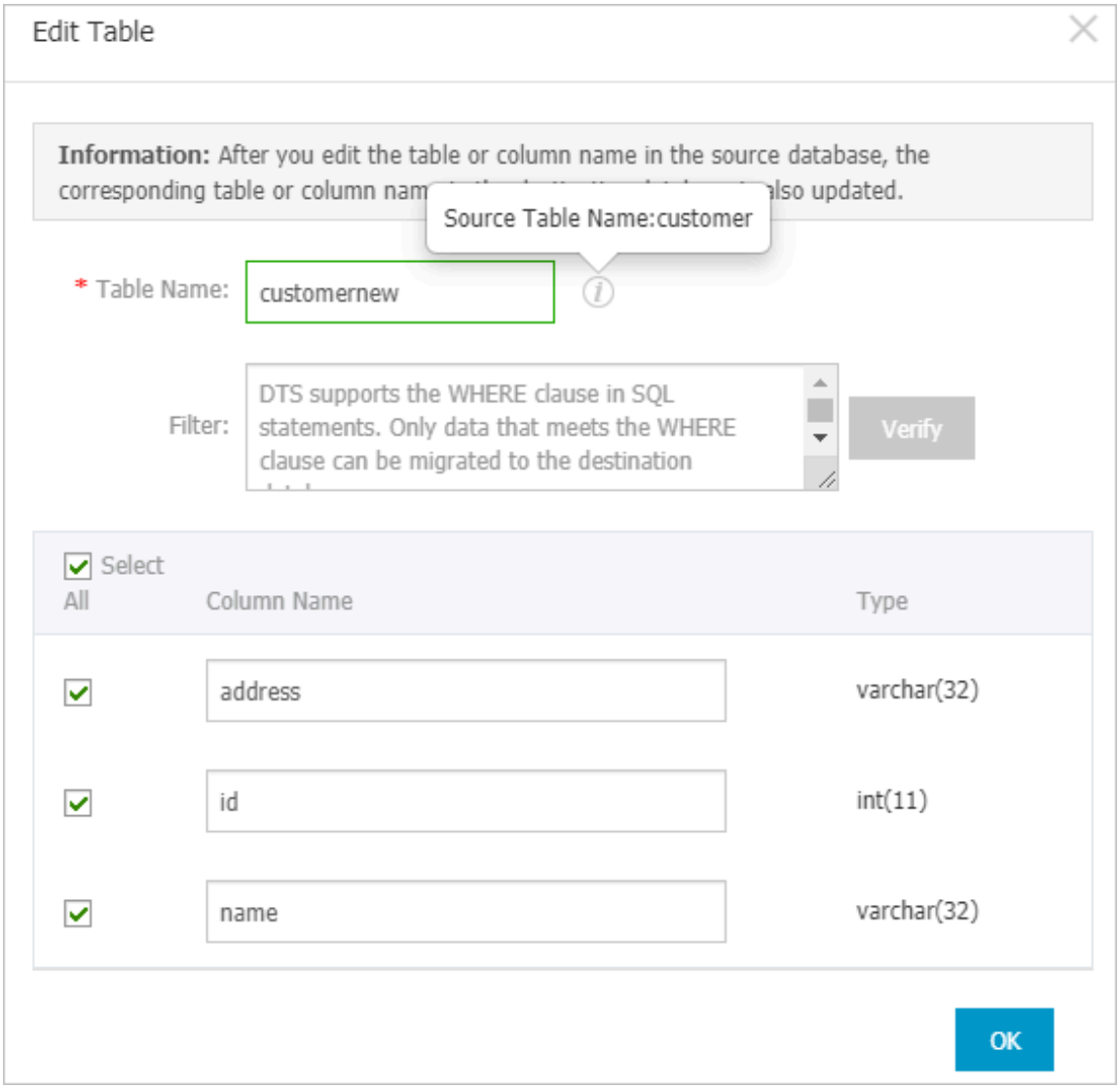
   - Database name mapping

     In the **Edit Database Name** dialog box that appears, enter the database name that you want to use in the destination instance.

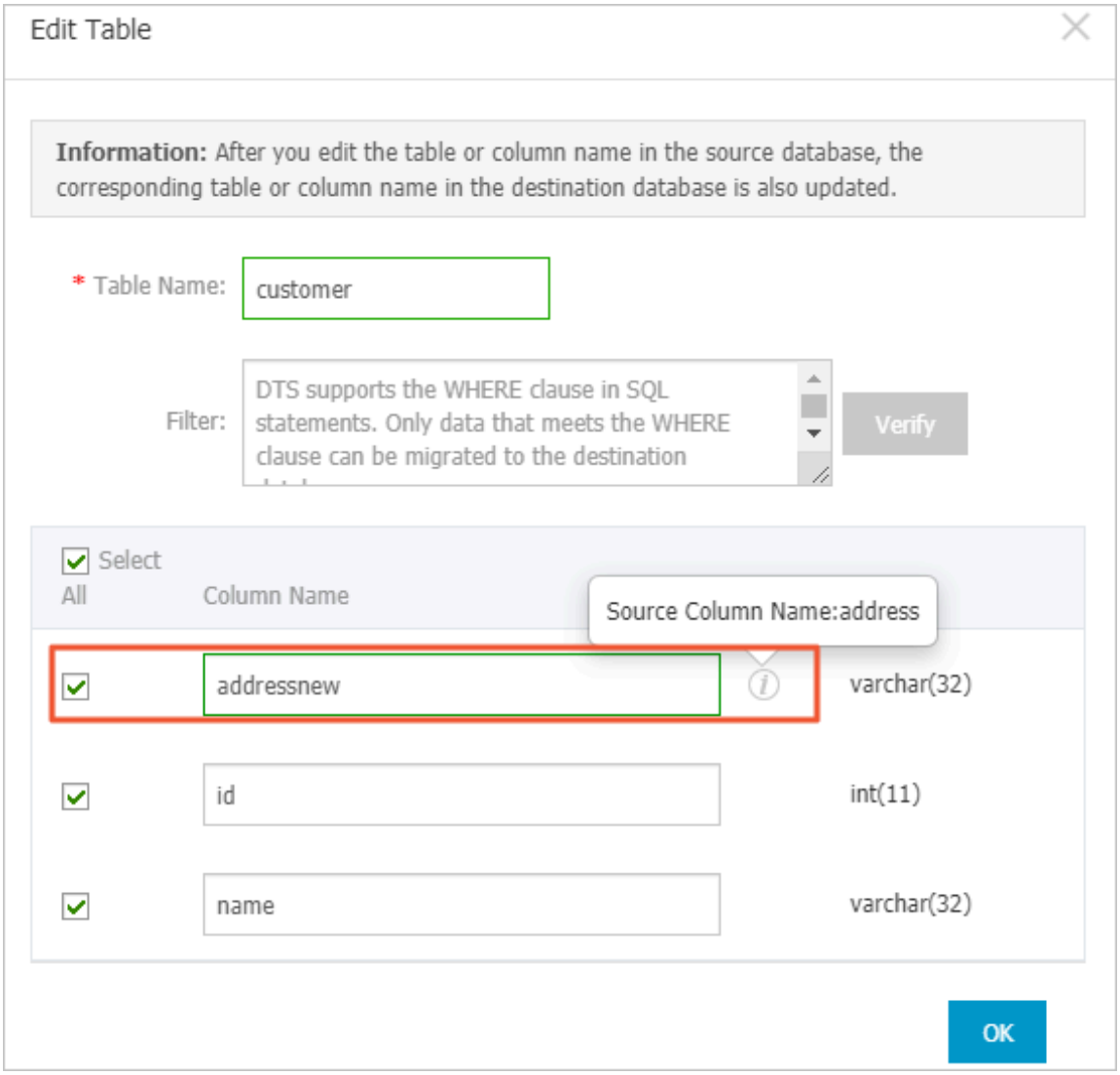
   - Table name mapping

     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.

4.3.4 Configure an SQL filter for filtering the data to be migrated

This section describes how to configure an SQL filter for filtering migration data when you create a migration task.

DTS allows you to configure an SQL filter to filter the table data to be migrated. The SQL filter applies only to the configured table. DTS filters the data in the table of the source database based on this filter. Only data that meets this filter can be migrated to the

destination database. This feature is applicable to multiple scenarios such as regular incremental data migration and table partitioning.

Functional restrictions

The SQL filter applies only to full data migration. If you select **Incremental Data Migration** as the migration type, the SQL filter does not apply.

Configure an SQL filter

You can configure an SQL filter in the **Migration Types and Tasks** step of migration task configuration.

If you want to configure an SQL filter for table migration, you must select a specific table instead of the entire database as the object to be migrated. The following part describes how to configure an SQL filter.

Configure an SQL filter

1. In the **Migration Types and Tasks** step, move the pointer over the table for which you want to create an SQL filter in the **Selected** area. The **Edit** button appears.
2. Click **Edit** to configure a filter.

Modify an SQL filter

Filters in DTS are the same as the standard SQL WHERE conditions for databases and support calculation and simple functions.

Enter an SQL filter in the text box as needed.

Now, you have configured an SQL filter.

4.3.5 Troubleshoot migration errors

DTS provides the feature of online troubleshooting in multiple stages to fix migration errors . These stages include:

- Schema migration

  DTS supports data migration between heterogeneous data sources. If you import data of unsupported types to the destination instance during a schema migration, the migration fails.

- Full data migration

  During full data migration, the migration task may fail because the destination RDS instance does not have sufficient space or required IP addresses have been deleted from

the whitelist. In this case, you can modify the task configurations and then restart the task.

DTS provides the online troubleshooting feature that allows you to resume a failed task when an error occurs during migration. The following sections describe how to troublesho ot errors that occur during schema migration and full data migration.

Troubleshoot errors occurred during schema migration

If a schema migration task fails, the task status changes to Migration Failed and the **Rectify** button appears.

Click **Rectify** next to a failed object.

Click **Rectify** next to each failed object. A troubleshooting dialog box appears.

Modify the schema definition based on the cause of failure. Click **Rectify** after you complete the modification and re-import the modified definition to the destination instance.

If the error persists after you click **Rectify**, the cause of failure changes to **Troubleshooting Failed** and the cause of troubleshooting failure is displayed. You need to continue troubleshooting based on the cause of troubleshooting failure until the troubleshooting is successful.

The details page of the schema migration appears after troubleshooting is successful, and the status of the object changes to Finished.

The task resumes after issues with all objects are rectified. For example, the task resumes by proceeding to the full data migration stage.

Troubleshoot errors occurred during full data migration

DTS provides the troubleshooting and retry feature for the following causes of failures:

- If you fail to connect to the source or destination database, retry the task after you ensure that the network connection is established.
- If a connection to the source or destination database times out, retry the task after you ensure that the network connection is established.
- If the destination RDS instance does not have sufficient space or the instance is locked, retry the task after you scale up the RDS instance or clean up the instance log space.
- If MyISAM of the source database is corrupted, retry the task after troubleshooting.

For other circumstances, if full data migration fails, DTS only offers the Ignore option. You can ignore the failed object and continue the migration of other objects.

If a full data migration task fails, the status of the task changes to **Migration Failed** and the **Rectify** button appears.

When a migration task fails, click **Rectify** next to a failed object.

If you encounter the preceding failures and the migration tasks can be retried, troubleshoot the errors as prompted. Then, click the Retry button on the full data migration details page to continue the data transfer in the task.

For other causes of failures, DTS only supports the **Ignore** operation to ignore the full data migration of the object. After you click Ignore, data of this object is not migrated, but data of other objects is migrated to the destination instance.

4.4 Change tracking

4.4.1 Track data changes from a user-created MySQL database

You can use DTS to track data changes in real time. This feature applies to the following scenarios: lightweight cache updates, business decoupling, asynchronous data processing, and real-time data synchronization of extract, transform, and load (ETL) operations. This topic describes how to track data changes from a user-created MySQL database.

Prerequisites

The version of the user-created MySQL database is 5.1, 5.5, 5.6, 5.7, or 8.0.

Supported source database type

You can track data changes only from a **user-created MySQL database with a public IP address**.

Note:

If your source database is a user-created MySQL database, you must create a database account and enable binary logging.

Create a change tracking task

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Change Tracking**.
3. On the **Change Tracking Tasks** page, click **Create Change Tracking Task** in the upper-right corner.

4. In the **Create DTS Instances** dialog box, specify a region and the number of change tracking instances, and then click **Create**.

   The following table describes the required parameters.

   Table 4-4: Parameters

Parameter	Description
Feature	The feature specified by the system. In this case, the value is **Change Tracking**.
Region	The region where the source instance resides.
Instance Type	Select **MySQL**.
Instances to Create	The number of change tracking instances that you want to create.

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

Configure the change tracking task

1. Find the change tracking task and click **Configure Channel** in the **Actions** column.
2. Configure the source database information 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 purchased the change tracking instance. You cannot change the value of this parameter.

Section	Parameter	Description
	Instance Region	The region of the source instance. The region is the same as the **source region** that you selected when you purchased the change tracking instance. 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.
	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 for the database account.
Consumer Network Type	Network Type	The network type of the change tracking instance. Only the **classic network** is supported.

3. Click **Set Whitelist and Next**.
4. In the **Create Change Tracking Account** message, click **Next** after the account is created.

Note:

In this step, DTS creates a database account for change tracking in the source instance.

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 selected objects, including 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 need to 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 only tracks data changes in this table. In this case, if you want to track data changes of a new table, you must add the table to the objects for change tracking. For more information, see Modify objects for change tracking.
Required Objects	Select objects from the **Required Objects** section and click the right arrow 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:

- 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.

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 initial change tracking, you can create consumer groups and consume tracked data.

4.4.2 Create consumer groups

Note

- You can create multiple consumer groups 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.
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**.

4.4.3 Manage consumer groups

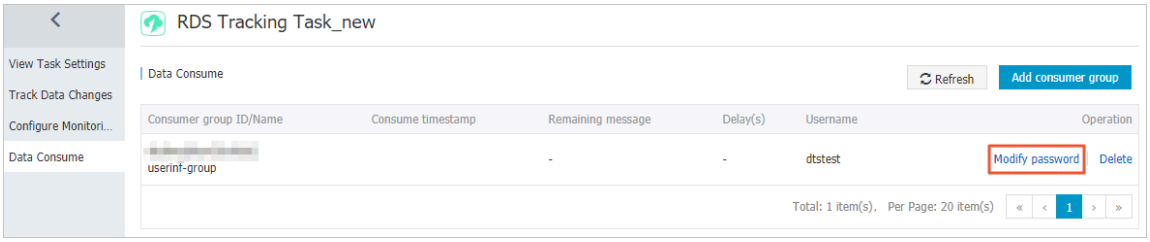
You can manage consumer groups of a change tracking instance in the DTS console. This topic describes how to modify the password of a consumer group and how to delete a consumer group.

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

   a) On the **Consume Data** page, find the target consumer group and click **Modify Password** in the **Actions** column.



   b) 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.

   c) 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.

a) On the **Consume Data** page, find the target consumer group and click **Delete** in the **Actions** column.

b) In the **Delete Consumer Group** message that appears, click **OK**.

4.4.4 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.

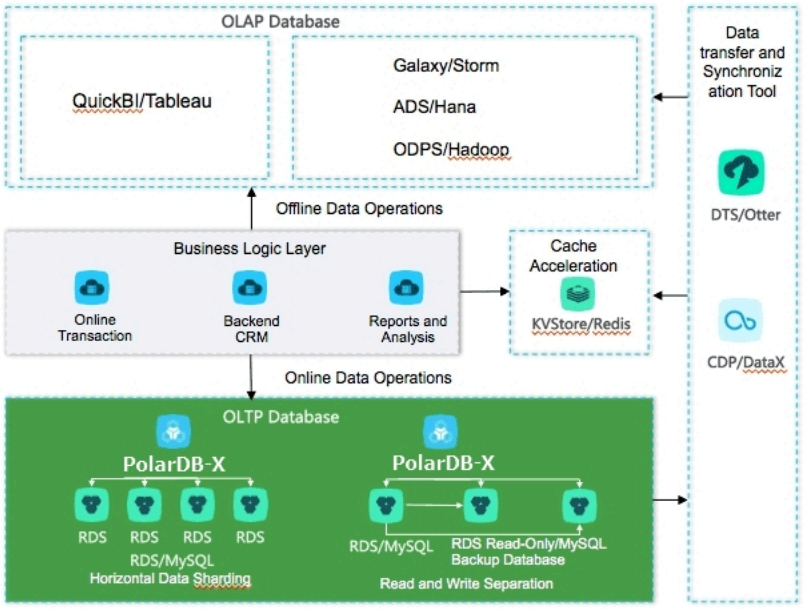
6. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.**

5 Cloud Native Distributed Database PolarDB-X

5.1 What is PolarDB-X?

Cloud Native Distributed Database PolarDB-X is a middleware service independently developed by Alibaba Group for scale-out of single-instance relational databases. It is compatible with 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 features lightweight (stateless), flexibility, stability, and high efficiency, and provides you with O&M capabilities throughout the lifecycle of distributed databases.

PolarDB-X is mainly used for operations on large-scale online data. By partitioning data in specific business scenarios, PolarDB-X maximizes the operation efficiency, meeting the requirements of online businesses on relational databases.



Problems solved

- **Capacity bottleneck of single-instance databases:** As the data volume and access volume increase, traditional single-instance databases encounter great challenges that cannot be completely solved by hardware upgrades. Distributed solutions use multiple

instances to work jointly, effectively resolving the bottlenecks of data storage capacity and access volumes.

- **Difficult scale-out of relational databases:** Due to the inherent attributes of distributed databases, data can be stored to different shards through smooth data migration, supporting the dynamic scale-out of relational databases.

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 only contains metadata. Specific data is stored in the physical database of the underlying storage instance. To get started with PolarDB-X, follow these 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 RDS instance exists, create one first. For more information about how to create and manage ApsaraDB RDS for MySQL instances, see User Guide of RDS .
3. After a PolarDB-X database is created, you also need to create tables in the PolarDB-X database like in a single-instance database. However, the syntax is different, mainly in the expression of data partitioning information in the PolarDB-X table creation statement. 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

- Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel. The URL used to access the ASCM console is in the following format: https://[IP address or domain name of the ASCM console].
- We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to access the ASCM console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password for logging on to the console from the operations administrator.

   Note:
   When you log on to the ASCM console for the first time, you must change the password of your username as prompted. Due to security concerns, your password must meet the minimum complexity requirements: The password must be 8 to 20 characters in length and must contain at least two of the following character types: uppercase letters, lowercase letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).
3. Click **Login** to go to the ASCM console homepage.
4. In the top navigation bar, choose **Products** > **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 **DRDS Instance Management** page, click **Create Instance** in the upper-right corner.
3. On the **Create DRDS Instance** page, set parameters as required.

   Table 5-1: Parameters for creating an instance describes the parameters.

Table 5-1: Parameters for creating an instance

Section	Parameter	Description
Region	**Organization**	The organization to which the instance belongs.
	Resource Set	The resource set to which the instance belongs.

Section	Parameter	Description
	Region	The region where the PolarDB-X instance resides. Services in different regions are not interconnected over the internal network. Once a region is selected, it cannot be changed.
	Zone	The zone where the PolarDB-X instance resides.
Basic Configurations	**Instance Type**	The type of the PolarDB-X instance. Select an instance type from the options available on the page.
	Instance Edition	The series of the PolarDB-X instance. Valid values include: • Standard • Enterprise • Starter
	Instance Specifications	The specifications of the PolarDB-X instance. The rules vary with instance series. Select the instance specifications from the options available on the page.
Network Type	**Network Type**	The network type of the instance. PolarDB-X instances support the following network types: • **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. • **VPC**: A Virtual Private Cloud (VPC) helps you to build an isolated network environment on Alibaba Cloud. You can customize routing tables, CIDR blocks, and gateways in a VPC. We recommend that you select VPC for higher security. **Note:** Make sure that the PolarDB-X instance has the same network type as the Elastic Compute Service (ECS) instance to which you want to connect. If the PolarDB-X and ECS instances have different network types, they cannot communicate over an internal network.

4. Click **Submit**.

After the instance is created, it is displayed in the instance list and its status changes to **Running**. An instance name uniquely identifies a PolarDB-X instance.

5.4.2 Change specifications

When using PolarDB-X, you can change the specifications of a PolarDB-X instance as required.

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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 on the right, click **Upgrade** or **Downgrade** to access the Change Specifications page.

   Note:
   Alternatively, choose **More** > **Downgrade** from the **Actions** column of the target instance on the **DRDS Instance Management** page.
5. On the Change Specifications page, set Instance Edition and Instance Specifications, and then click **Submit**.

   Wait for a few minutes and then check the new specifications of the instance in the instance list.

   Note:
   Specifications downgrade leads to transient disconnections between applications and PolarDB-X within a short period. 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 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 analysis directly on read-only or primary ApsaraDB RDS for MySQL instances. Multiple instance types are provided to handle highly concurrent access requests and reduce the response time (RT) for complex queries. Resource isolation

alleviates the load pressure on the primary instances and reduces the link complexity of the business architecture. It reduces the O&M and budget costs, eliminating the need for additional data synchronization.

Instance type

Concurrent read-only instances: For high-concurrency and high-traffic simple queries or offline data extraction, resource isolation protects you against highly concurrent queries, ensuring the stability of online business links.

Note:

For the businesses with primary PolarDB-X instances, 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 creating a read-only instance, you must create a primary 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 or delete databases in PolarDB-X read-only instances.
- PolarDB-X read-only instances cannot be cloned.
- PolarDB-X read-only instances support data definition language (DDL) statements but do not support data manipulation language (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. On the **DRDS Instance Management** page, find the target instance.

3. Click the instance ID or choose **More** > **Manage** from the Actions column of the instance to access the **Basic Information** page.
4. Click **Create DRDS Read-only Instance** on the right.
5. Set parameters in the Region, Basic Settings, and Network Type sections, and click **Submit**.

Table 5-2: Parameters for creating a read-only PolarDB-X instance

Section	Parameter	Description
Region	**Region**	The region where the PolarDB-X instance resides. Services in different regions are not interconnected over the internal network. Once a region is selected, it cannot be changed.
	Zone	The zone where the PolarDB-X instance resides.
Basic Configuration	**Instance Type**	The type of the PolarDB-X instance. Select an instance type from the options available on the page.
	Instance Edition	The series of the PolarDB-X instance. Valid values include: • Starter • Standard • Enterprise
	Instance Specifications	The specifications of the PolarDB-X instance. The rules vary with instance series. Select the instance specifications from the options available on the page.
	Describe	The description of the PolarDB-X instance. We recommend that you provide an informative description to simplify future management operations.

Section	Parameter	Description
Network Type	**Network Type**	The network type of the instance. PolarDB-X instances support the following network types: • **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. • **VPC**: A Virtual Private Cloud (VPC) helps you to build an isolated network environment on Alibaba Cloud. You can customize routing tables, CIDR blocks, and gateways in a VPC. We recommend that you select VPC for higher security. **Note:** Make sure that the PolarDB-X instance has the same network type as the Elastic Compute Service (ECS) instance to which you want to connect. If the PolarDB-X and ECS instances have different network types, they cannot communicate over an internal network.

6. It takes several minutes to create an instance. After the instance is created, it is displayed 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 the 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 those 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 PolarDB-X instance.
3. Click the instance ID or choose **More** > **Manage** from the Actions column of the target read-only PolarDB-X 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 PolarDB-X instance.

3. Click the instance ID or choose **More** > **Manage** from the Actions column of the target primary PolarDB-X instance to access the **Basic Information** page.
4. In the **Related Instance** section on the right of the **Basic Information** page, move the pointer over the number of read-only PolarDB-X instances to display the read-only PolarDB-X instance ID.
5. Click the ID of the target read-only PolarDB-X instance. The **Basic Information** page of the read-only instance PolarDB-X appears.

5.4.3.4 Release a read-only PolarDB-X instance

If you no longer need the read-only PolarDB-X instance, you can release it.

Prerequisites

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

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Choose **More** > **Release** from the Actions column of the target read-only PolarDB-X instance.

   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 is in the **Running** state.

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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. Perform this operation with caution.

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 is in the Running state.

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. In the PolarDB-X instance list, find the target PolarDB-X 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 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 recovery

PolarDB-X allows you to back up data of instances and databases and recover them by using the backup data. Instances can be automatically and manually backed up in a quick

or consistent manner. The instance recovery capability restores data to the new PolarDB-X and ApsaraDB RDS for MySQL instances based on the existing backup set.

Backup methods

For different scenarios, PolarDB-X provides quick backup and consistent backup, as well as corresponding recovery capabilities. The following table compares the two backup methods.

Method	Scenario	Advantage	Disadvantage
Quick backup	It applies to routine backup and recovery scenarios.	• It provides faster data backup and recovery. • It supports recovery at any time based on backup sets. • It supports all PolarDB-X instance versions.	It ensures data consistency only at a single ApsaraDB RDS for MySQL instance but not global data consistency in database and table sharding scenarios.

Method	Scenario	Advantage	Disadvantage
Consistent backup	It applies to backup and recovery for online core transaction businesses and the financial industry with a high-consistency requirement.	It ensures global data consistency in database and table sharding scenarios.	• It provides slower backup and recovery. • It supports recovery based on backup sets but not at any time. • It is supported only for PolarDB-X 5.3.8 and later. • During the backup, distributed transactions are locked within seconds for PolarDB-X instances. During the locking process, SQL execution response time (RT) may vary by milliseconds. Therefore, we recommend that you perform consistent backup during off-peak hours.

Limits

- The PolarDB-X automatic backup policy is disabled by default. You must manually enable it.
- The log backup capability of PolarDB-X depends 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.

- The backup and recovery feature of PolarDB-X depends on log backups. We recommend that you enable the log backup policy by default to avoid invalid backup sets.
- Data definition language (DDL) operations cannot be performed during the backup process.
- During the backup, the underlying ApsaraDB RDS for MySQL instances of the PolarDB-X instance must be normal to avoid a backup failure.
- Consistent backup and recovery is only supported by PolarDB-X 5.3.8 and later versions.
- All tables must have primary keys to ensure data accuracy during data backup and recovery.
- During consistent backup, distributed transactions are locked within seconds for PolarDB-X instances. During the locking process, the execution of non-transactional SQL statements and single-instance transactions are not affected, but the committing of distributed transactions is blocked and the SQL execution RT may vary by milliseconds. We recommend that you perform consistent backup during off-peak hours.
- Due to the inventory of PolarDB-X and ApsaraDB RDS for MySQL, PolarDB-X automatically adjusts the instance type and zone during instance recovery. We recommend that you confirm and adjust the instance type and zone after recovery.

5.4.6.2 Configure an automatic backup policy

PolarDB-X provides the automatic backup feature. This topic describes how to configure the automatic backup policy.

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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 the **Backup Policy** > **Edit**.
6. In the **Backup Policy** dialog box, set parameters based on the business requirements, and click **OK**.

> **Notice:**
> PolarDB-X instances cannot back up logs. The configured log backup policy is applied to all underlying ApsaraDB RDS for MySQL instances.

5.4.6.3 Configure local logs

Local logs can be used to accurately recover an instance or a database to the desired point in time through the backup and recovery capabilities or SQL flashback capability of PolarDB-X. This topic describes how to configure local logs.

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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 the **Local Log Settings** > **Edit**.
6. In the **Local Binlog Settings** dialog box, set parameters based on the business requirements, and click **OK**.

> **Notice:**
> The local log settings are applied to all underlying ApsaraDB RDS for MySQL instances.

5.4.6.4 Configure a manual backup policy

PolarDB-X also provides the manual backup capability so that you can 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. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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. 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** and **Consistent Backup**. For more information, see Backup methods.

> **Notice:**

If you select Consistent Backup, distributed transactions are locked within seconds and the response time (RT) may vary by sub-seconds. Therefore, we recommend that you perform this operation 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** and select the databases you want to back up.

7. Click **OK**.

5.4.6.5 Recover data

You can use the data recovery capability of PolarDB-X to recover 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 recover the data of an instance and a database to a specific point in time.

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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 upper-right corner of the page, click **Data Recovery (Original Clone Instance)**.
6. Set Recovery Method.
   - **By Time**: Recover data to the selected point in time. You need to set **Restoration Time** and **Recovery Level**.
   - **By Backup Set**: Recover data from the selected backup file.

   Note:
   You can also click **Recover** of the target backup set to recover data **by backup set**.
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 access the order confirmation page.
9. Confirm the order details and then click **Enable** to recover the data. You can view the data recovery progress in **Task Progress** in the upper-right corner.

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 recovery file

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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 an error 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 error SQL statement was executed.
Time Range	The time range during which the error SQL statement was executed . The start time is earlier than the start time when the error SQL statement was executed, while the end time is later than the time when the execution of the error SQL statement ended. To ensure efficient recovery, we recommend that the time range be limited to 5 minutes.
Table Name	The name of the table on which the error SQL statement was executed. This parameter is optional.
TRACE_ID	The unique TRACE_ID PolarDB-X allocates for each executed SQL statement. You can obtain the TRACE_ID of the error SQL statement through PolarDB-X SQL audit.
SQL Statement Type	The type of error 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 the binary log file, see Configure local logs.

   Note:

   - If no binary log file exists within the time range, the precheck fails and the system cannot recover the data for you.
   - If a binary log file exists within the time range, the precheck is successful and you can go 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 methods, see Rollback SQL statements and original SQL statements.
8. Click **Generate SQL** to generate an SQL flashback task. The statuses of the SQL flashback task that are running on the current instance are displayed at the bottom of the page.

What's next

After an SQL flashback task completes, the task information such as the exact match status and the number of recovered data rows is displayed. You can click **Download** next to the target SQL flashback task to download the corresponding recovery file.

Notice:
By default, the recovery 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
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 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 mode	Description	Advantage	Disadvantage
Exact match	The system exactly matches the event of an error SQL statement in the binary log file and generates a recovery file.	The recovery file only contains data that is deleted or modified by the error SQL statement. You can use the data directly to ensure the precision and efficiency of data recovery.	The following conditions 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 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 of the error SQL statement is provided.
Fuzzy match	The system matches the information about the error SQL statement in the binary log file, including the time range, table name, and SQL statement type. Then, the system generates a recovery file.	Fuzzy match is supported for all instances, regardless of the instance version or parameters.	Data that is deleted or modified by the error SQL statement cannot be accurately matched. The recovery file contains data changes made by other business SQL operations. You must filter the required data.

Enable exact match

> **Note:**
> Fuzzy match is enabled by default.

1. Log on to the Distributed Relational Database Service (DRDS) console and click the target PolarDB-X instance. In the left-side navigation pane, click Parameter Settings. For more information, see Set parameters.
2. Change the value of the **SQL exact flashback switch** parameter 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.

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.

3. Click the 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. On 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 **Enable**.
7. In the dialog box that appears, click **OK**.

5.4.6.7.3 Recover tables

This topic describes how to recover your tables from the table recycle bin.

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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. On the top of the **Table Recycle Bin** page, click the tab of the database in which the tables need to be recovered.
6. Click **Restore** in the **Actions** column of the target table.

5.4.6.7.4 Delete tables from the recycle bin

This topic describes how to delete unnecessary tables from the table recycle bin.

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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. On the top of the **Table Recycle Bin** page, click the tab of the database in which the tables need to be cleared.

6. Click **Delete** in the **Actions** column of the target table.

5.4.6.7.5 Disable the table recycle bin

If you no longer need the table recycle bin, you can disable it.

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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. On the top of the **Table Recycle Bin** page, click the tab of the database for which the table recycle bin needs to be disabled.
6. Click **Disable** to disable the table recycle bin 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 based on business needs.

Note:
Parameters cannot be set for read-only PolaDB-X instances.

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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 **Diagnostics and Optimization** > **Parameter Settings**. Click the Instance or Database tab to view parameters that you can modify for instances and databases, respectively. For more information about the parameters, see Parameter description.
5. Click  next to the parameter you want to modify, enter the target value, and click **OK**.

6. Click **Submit** in the upper-right corner to commit the modification.

> **Note:**
> To undo parameter modification, click **Cancel** in the upper-right corner.

Parameter description

Parameter	Level	Description
Slow SQL threshold	Instance	The threshold for slow SQL statements. SQL statements whose thresholds exceed this threshold are recorded in logical slow SQL logs.
Logical idle link timeout	Instance	The logical timeout period of the idle connection between user applications and PolarDB-X (unit: ms).
Maximum package size	Instance	The maximum network packet for the interaction between user applications and PolarDB-X (unit: byte).
Instance memory pool size limit	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 full table deletion or update.
Whether to open the recycle bin	Database	Specifies whether to enable the recycle bin for storing deleted PolarDB-X logical tables.
Temporary table size	Database	The size of the temporary table used during distributed queries in PolarDB-X (unit: row).
Number of join tables	Database	The maximum number of table shards that can be combined through JOIN when you query multiple table shards in a database.
Physical SQL timeout	Database	The timeout period of SQL statements for interaction between PolarDB-X and ApsaraDB RDS for MySQL (unit: ms). The value 0 indicates the timeout period is not limited.
SQL exact flashback switch	Database	Specifies whether to support SQL flashback exact match. It is disabled by default. After it is enabled, information about the queries is added to the binary log file used by the PolarDB-X database.

Parameter	Level	Description
Whether to enable logical INFORMATION_SCHEMA query	Database	Specifies whether to enable logical INFORMATION_SCHEMA query (not relying on the shadow database but returning the aggregation results of logical databases and tables). When it is disabled, the original status is restored (relying on the shadow database and returning the physical database and table information).
Transaction log cleanup start time period	Database	The period during which transaction log cleanup starts at a random time.
Library-level memory pool size limit	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 no limit.
Query-level memory pool size limit	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 no limit.
Whether CBO is enabled	Database	Specifies whether to enable the cost-based optimizer (CBO), including features such as Join Reorder and Hash Join.
Whether to enable the asynchronous DDL engine	Database	Specifies whether to enable the data definition language (DDL) engine. If you disable it, the execution logic of the original DDL engine remains.
Whether to enable asynchronous-only mode under asynchronous DDL engine	Database	Specifies whether to enable the asynchronous-only mode when the asynchronous DDL engine is enabled. • Enabled: The status is returned immediately after the client connects to PolarDB-X and executes the DDL statement. The execution status can be viewed only through asynchronous DDL management statements. • Disable: The synchronous mode remains. That is, the status is returned only after the client completes executing the DDL statement.

Parameter	Level	Description
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.
INFORMATION_SCHEMA.TABLES queries whether statistics are aggregated	Database	Specifies whether to aggregate statistics of INFORMATION_SCHEMA.TABLES queries. To ensure the performance, it is not aggregated by default.
Maximum number of physical sharding links	Database	The maximum number of connections between PolarDB-X and a single ApsaraDB RDS for MySQL shard.
Minimum number of physical sharding links	Database	The minimum number of connections between PolarDB-X and a single ApsaraDB RDS for MySQL shard.
Physical idle link timeout	Database	The idle time of the connection between PolarDB-X and ApsaraDB RDS for MySQL (unit: minute).

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. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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** from the left-side navigation pane.

5. On the **Instance Monitoring** page, select a monitoring dimension and the corresponding metrics to view details. For more information about monitoring metrics, see Monitoring metrics.

5.4.8.2 Monitoring metrics

Instance monitoring is divided into resource monitoring and engine monitoring. Engine monitoring metrics are classified into metrics at the PolarDB-X instance level and metrics at the PolarDB-X database level. When some engine monitoring metrics are abnormal, you can directly check the metrics of each database to locate the database with performance problems. The following table describes the metrics of these two types in details.

Monitoring item	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	-
Memory Usage (%)	Resource monitoring	The memory usage of 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.

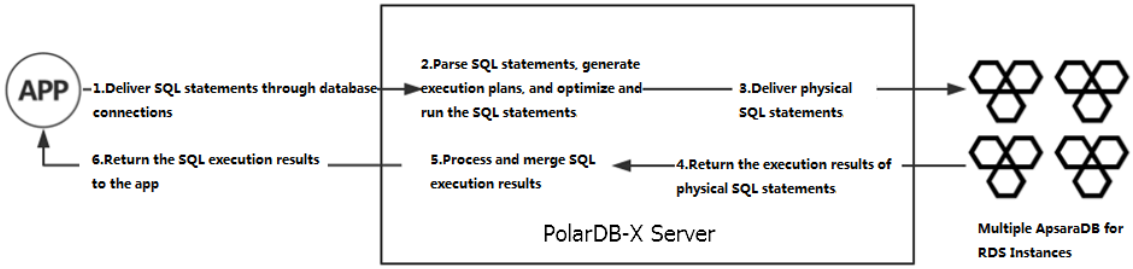
Monitoring item	Category	Description	Data collection cycle	Data retention period	Description
Outbound Traffic (Kbps)	Resource monitoring	The total outbound network traffic of PolarDB-X server nodes.	1 minute	3 days	Outbound network traffic is generated when a PolarDB-X instance sends a physical SQL statement 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	-
Physical QPS	Engine monitoring	The total number of SQL operations 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.

Monitoring item	Category	Description	Data collection cycle	Data retention period	Description
Logical RT (ms)	Engine monitoring	The average response time (RT) for processing each SQL statement by PolarDB-X.	5 seconds	7 days	If a logical SQL statement is partitioned into physical SQL statements for delivery, the logical RT of the SQL statement contains the RT of the physical SQL statements.
Physical RT (ms)	Engine monitoring	The average RT for transmitting SQL statements from PolarDB-X to ApsaraDB RDS for MySQL.	5 seconds	7 days	-
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 run SQL statements.	5 seconds	7 days	-

5.4.8.3 How metrics work

Before analyzing metrics, you need to understand the execution process of SQL statements on PolarDB-X.

Figure 5-1: PolarDB-X SQL execution flowchart



In the entire SQL execution process, the execution status of steps 2 through 4 is reflected in various metrics of PolarDB-X.

- In step 2, SQL parsing, optimization, and execution consume CPU resources. A more complex SQL statement (with a complex structure or ultra-long length) consumes more CPU resources. You can run the **TRACE** command to trace the SQL execution process. You can see the time consumed by an SQL statement during optimization. The longer time consumed indicates a higher CPU utilization.
- In step 3, the delivery 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 response time (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 processing SQL statements very slowly. You need to check the performance of the ApsaraDB RDS for MySQL instance.
- In step 5, the SQL execution results are processed and integrated. These operations convert the execution results of physical SQL statements. 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 at this stage, see Details about a low SQL statement.

5.4.8.4 Prevent performance problems

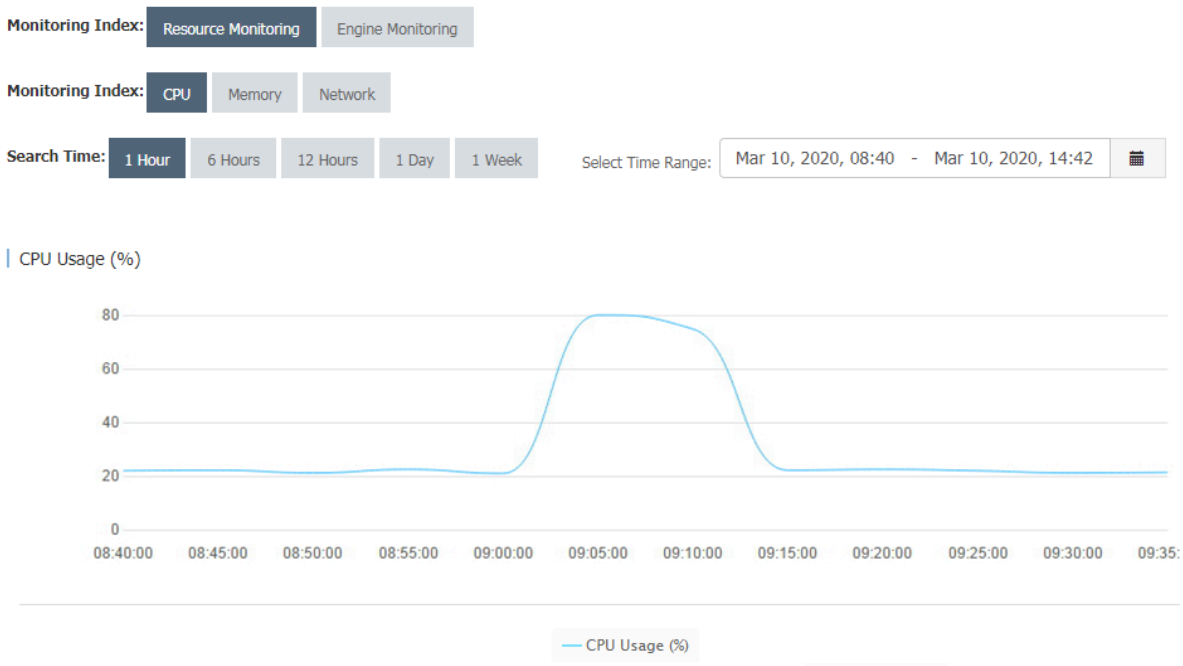
5.4.8.4.1 Example 1: PolarDB-X CPU utilization

Performance metrics change with the system business traffic.

The following describes the CPU utilization in two common cases:

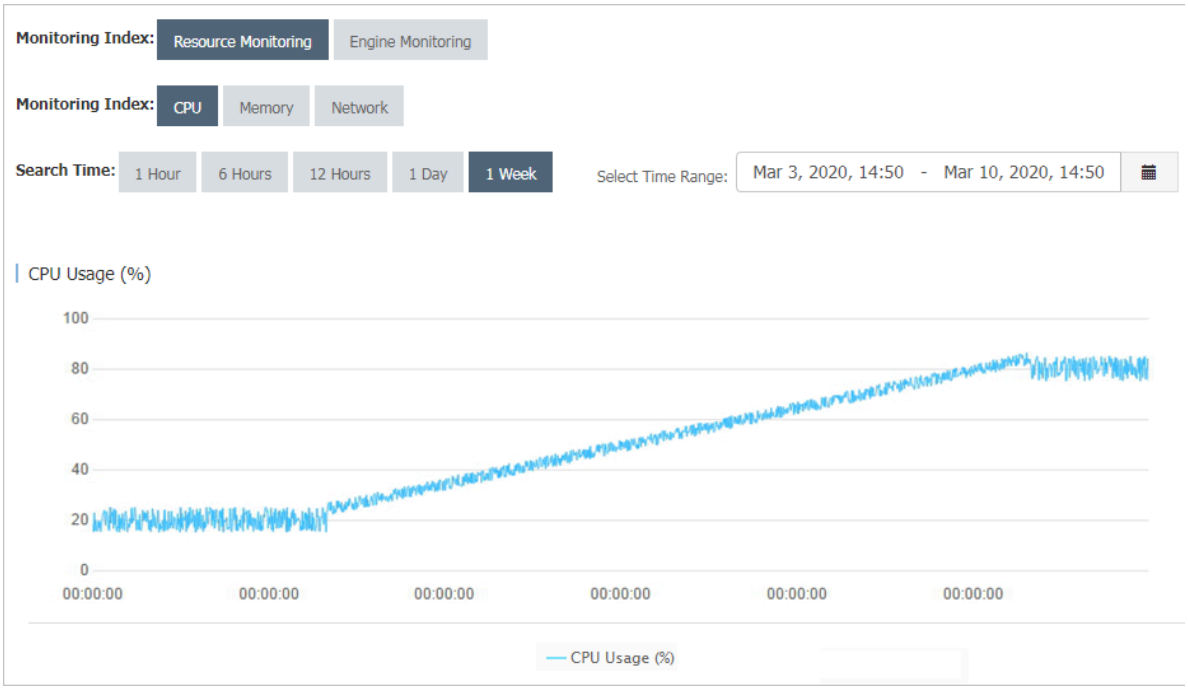
- An application has a shopping spree activity at 09:00 every morning. Therefore, the traffic of the system increases significantly at this time point. According to the monitoring data, the CPU utilization of the PolarDB-X instance increased from 20% to about 80% from about 09:00, with the traffic peak lasting about 10 minutes.

Figure 5-2: CPU utilization-1



- The system traffic keeps increasing with an application until it reaches a plateau. The monitored CPU utilization of the PolarDB-X instance also reflects this change.

Figure 5-3: CPU utilization-2



When the load on the PolarDB-X instance changes with the business, you must pay close attention to the changes in metrics. If the CPU utilization exceeds the threshold, you must upgrade the PolarDB-X 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 short 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 Example 2: Logical RT and physical RT

You can observe the difference between the logical response time (RT) and physical RT.

Logical RT refers to the time from when a PolarDB-X instance receives a logical SQL statement to when it returns data to an application. Physical RT refers to the time from when a PolarDB-X instance sends 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 run fast, while logical SQL queries take a long time to run. In this case, the logical RT and physical RT are as follows.

Figure 5-4: Logical RT

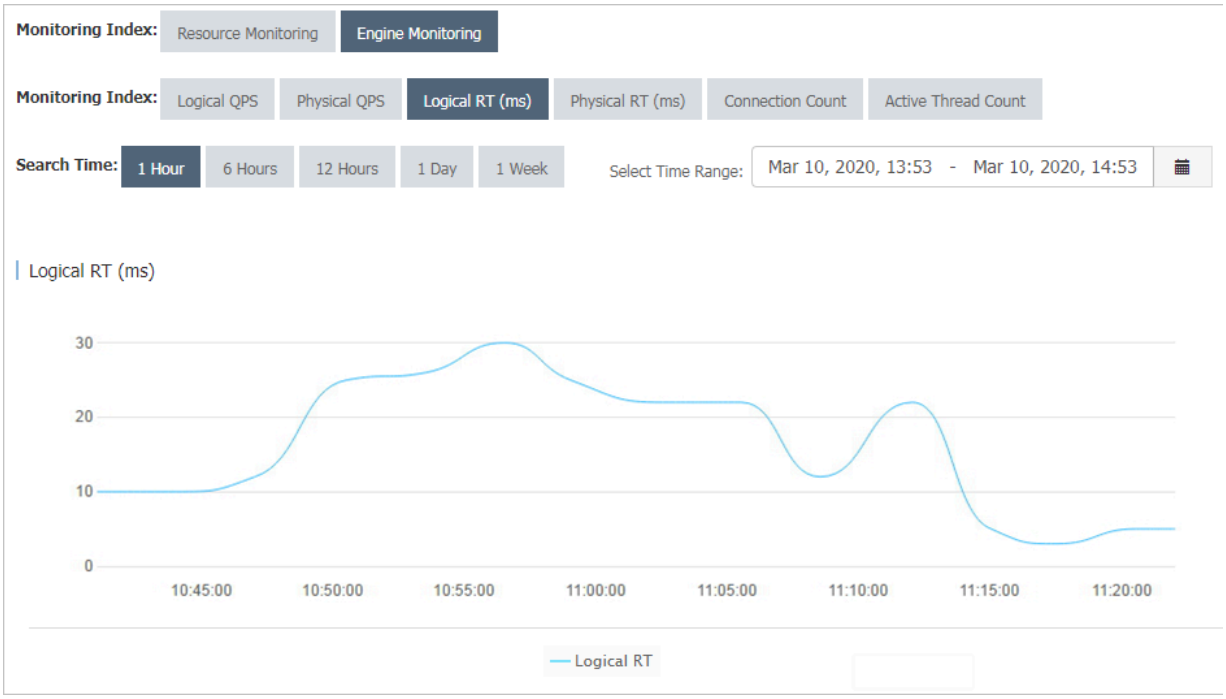
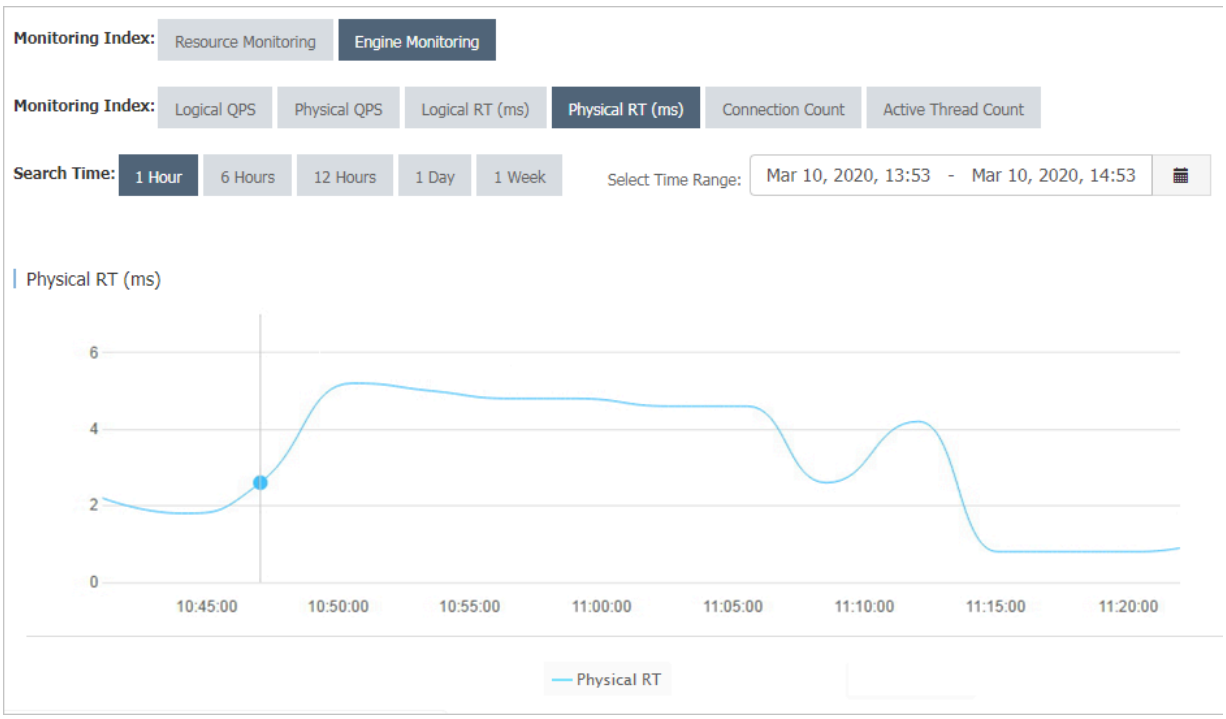


Figure 5-5: Physical RT



As shown in the preceding figures, the change trends of logical RT and physical RT in the two monitoring charts are basically the same, while logical RT fluctuates between 10 ms and 20 ms and physical RT fluctuates between 2 ms and 5 ms. This means that PolarDB-X has a heavy load, which can be solved by upgrading the PolarDB-X 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 Example 3: Logical QPS and physical QPS

You can observe the difference between the logical queries per second (QPS) and physical QPS.

According to the monitoring data, the logical QPS and physical QPS have the same trends, but the difference between the two is relatively large and in a certain proportion.

Figure 5-6: Logical QPS

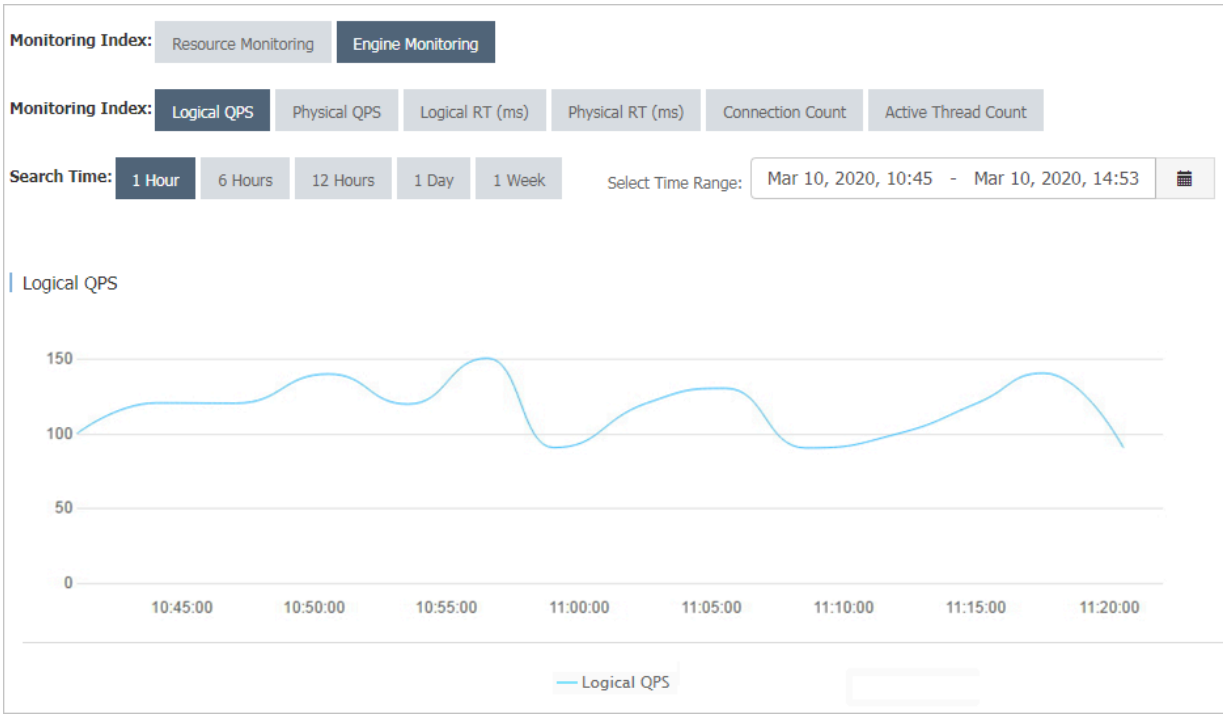
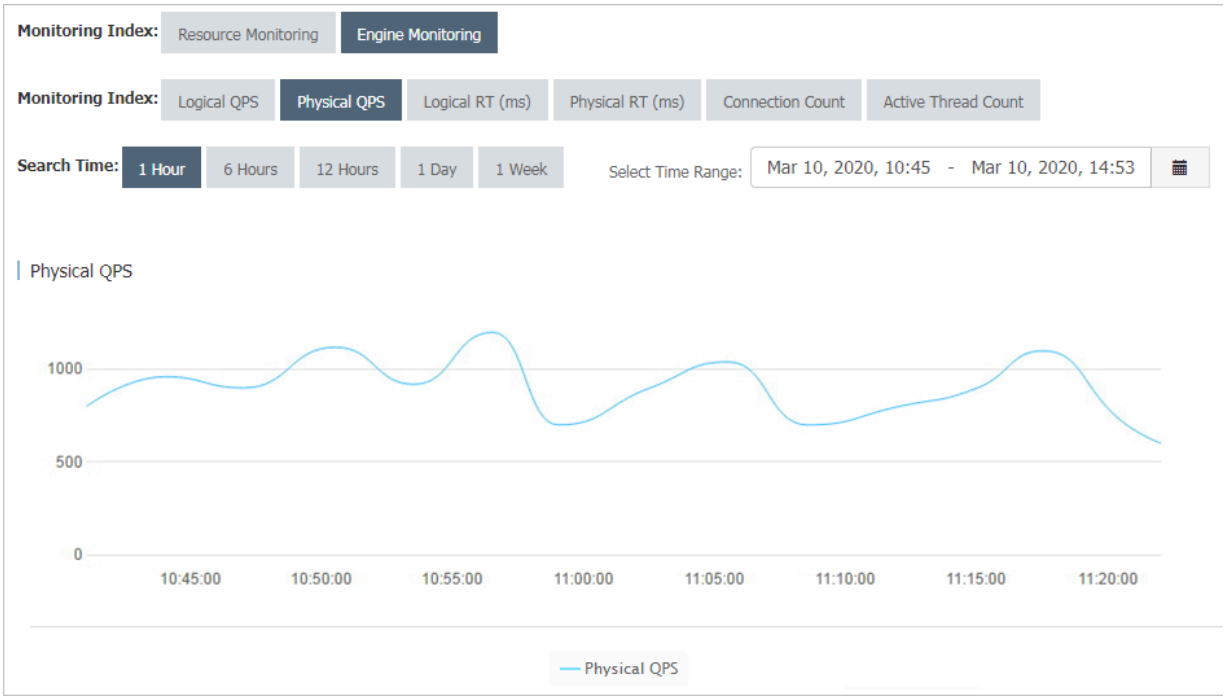


Figure 5-7: Physical QPS



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 by using 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 pushes the logical SQL statement down to the ApsaraDB RDS for MySQL instance for execution. According to the execution plan, 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 there are eight physical SQL statements:

```
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 Example 4: High memory usage

The overly high memory usage of the 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

You can view the PolarDB-X instance version in two ways.

View the instance version in the console

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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

Connect to the PolarDB-X instance through the MySQL command line and run the `SELECT version()` statement to view the version of the PolarDB-X instance.

```
mysql> select version();
+----------------------------+
| VERSION()                  |
+----------------------------+
| 5.6.29-TDDL-5.1.28-1320920 |
+----------------------------+
1 row in set (0.00 sec)
```

In the preceding statement, 5.1.28-1320920 indicates the version of the PolarDB-X instance.

5.5 Account management

5.5.1 Terms

This topic describes the terms of the PolarDB-X account and permission system.

The usage of the account and permission system in PolarDB-X is the same as that in MySQL. PolarDB-X supports statements such as GRANT, REVOKE, SHOW GRANTS, CREATE USER, DROP USER, and SET PASSWORD. Currently, 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 official documentation.

> **Notice:**
> Accounts created by using CREATE USER in PolarDB-X only exist in the PolarDB-X instance and will not be synchronized to the backend ApsaraDB RDS for MySQL instances.

Account

An account is specified by the user name and hostname in the username@'host' format . Accounts with the same user name 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: administrator account and 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 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. According to the preceding rules, the dreamdb database contains the dreamdb administrator account and the dreamdb_RO read-only account, while the andordb database contains the andordb administrator account and the andordb_RO read-only account.

Account rules

- Each administrator account has all permissions.
- Only the administrator account can create accounts and grant permissions. Other accounts can only be created and granted permissions by the administrator account.
- The administrator account is bound to a database and does not have permissions on other databases. It can only access the bound database, and cannot grant permissions of other databases to an account. For example, the easydb administrator account can only connect to the easydb database, and can only grant permissions of the easydb database or tables in the easydb database to an account.
- A read-only account has only the SELECT permission.

User name rules

- User names are case-insensitive.
- A user name must be 4 to 20 characters in length.
- A user name must start with a letter.
- A user name can contain letters and digits.

Password rules

- A password must be 6 to 20 characters in length.
- A password can contain letters, digits, and special characters (@ # $ % ^ & + =).

Hostname matching rules

- A hostname must be an IP address. It can contain underscores (_) and wildcards (%). A underscore (_) indicates a character and a wildcard (%) indicates no or more characters. Quote hostnames that contain wildcards with single quotations marks ('), for example, `lily@'30.9. %.%'` and `david@'%'`.
- If there are two user names that can be used to log on to the system, the user name with the longest prefix (the longest IP segment excluding wildcards) prevails. For example, if the `david@'30.9.12_.234'` and `david@'30.9.1%.234'` user names are available in the system, use `david@'30.9.12_.234'` to log on from the 30.9.127.234 host as david.
- When you enable the Virtual Private Cloud (VPC) access feature for a host, the IP address of the host changes. To avoid invalid configurations in the account and permission system, set the hostname to `'%'` to match any IP address.

Permission support

Permission support by level

- Global permission (not supported currently)
- Database-level permission (supported)
- Table-level permission (supported)
- Column-level permission (not supported currently)
- Subprogram-level permission (not supported currently)

Permissions

Currently, 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.
- CREATE INDEX and DROP INDEX statements are supported.
- 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') rather than a user name (username).
- An error occurs if the table does not exist during authorization.
- The database permissions in descending order are as follows: global permissions (not supported currently), database-level permissions, table-level permissions, and column-level permissions (not supported currently). 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 in the PolarDB-X console and by using SQL statements.

Create an account in the console

1. Log on to the PolarDB-X console.

2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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 **Account Management** page, click **Create Account** in the upper-right corner.
6. Configure the following parameters.

Parameter	Description
Database Account	Enter the account name. The account name must meet the following requirements: • The name must be 4 to 20 characters in length. • The name must start with a letter and end with a letter or digit . • The name can contain letters, digits, and underscores (_).
New Password	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 ! @#$%^&*()_+-=
Confirm New Password	Enter the password again.

Parameter	Description
Authorization Databases	You can grant permissions on one or multiple databases to the account. **a.** Select one or more databases in the left-side section, and click **Add** to add them to the right-side section. **b.** In the right-side section, select **Read/Write**, **Read-only**, **DDL Only**, or **DML Only**. **Note:** You can also grant permissions on multiple authorized 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 right-side section. The button in the upper-right corner changes as you click it. For example, after you click **Set All to Read-only**, the button is changed to **Set All to DDL Only**.

Create an account in the command line

Syntax rules:

```
CREATE USER user_specification [, user_specification] ...
  user_specification: user [ auth_option ]
  auth_option: IDENTIFIED BY 'auth_string'
```

Examples:

Create an account with the user name lily and password 123456, which can be used to log on only from 30.xx.xx.96.

```
CREATE USER lily@30.xx.xx.96 IDENTIFIED BY '123456';
```

Create an account named david with no password, which can be used to log on from any host.

```
CREATE USER david@'%';
```

5.5.3 Reset the password

When using PolarDB-X, you can reset the password of your database account in the PolarDB-X console or by using the command line.

Note:

- Accounts with ROOT permissions cannot be deleted or modified.
- For data security, we recommend that you change your password periodically.

Reset the password in the console

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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 **Reset Password**.
6. In the **Reset Account Password** dialog box, set **New Password** and **Confirm New Password**.

   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.
   - Special characters include ! @#$%^&*()_+-=
7. After you confirm that the password is correct, click **OK**.

Reset the password in the command line

Syntax rules:

```
SET PASSWORD FOR user = password_option
password_option: {
    PASSWORD('auth_string')
}
```

Examples:

Change the password of the account lily@30.xx.xx.96 to 123456.

```
SET PASSWORD FOR lily@30.xx.xx.96 = PASSWORD('123456')
```

5.5.4 Modify account permissions

You can modify the account permissions of your instances at any time when using PolarDB-X.

Precautions

- Privileged accounts cannot be modified.
- In the console, you can only grant data manipulation language (DML), data definition language (DDL), read-only, and read/write permissions to standard accounts. To grant more permissions, use the command line.

Modify account permissions in the console

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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 **Modify Permission**.
6. In the **Modify Permissions** dialog box, **grant** or **remove** the permissions on one or more databases to or from the account.
   - Authorize a database:

     Select one or more databases in the left-side section, and click **Add** to add them to the right-side section.
   - Remove an authorized database:

     Select one or more databases in the right-side section, and click **Remove** to remove them. The removed databases are displayed in the left-side section.
   - Modify permissions of an authorized database:

     In the right-side section, select **Read/Write**, **Read-only**, **DDL Only** or **DML Only**.

     Note:

You can also grant permissions on multiple authorized 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 right-side section.

The button in the upper-right corner changes as you click it. For example, after you click **Set All to Read-only**, the button is changed to **Set All to DDL Only**.

7. After the configuration is complete, click **OK**.

GRANT statements

Syntax rules:

```
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 in the GRANT statement does not exist and no IDENTIFIED BY information is provided, an error message indicating that the account does not exist is returned.
- 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 permission.

For example, in the easydb database, create an account named david, which can be used to log on from any host and has all the permissions on easydb.

Method 1: Create an account and 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 using one statement.

```
GRANT ALL PRIVILEGES ON easydb.* to david@'%' IDENTIFIED BY 'your#password';
```

In the easydb database, create an account named hanson, which can be used to log on from any host and has all the permissions on the easydb.employees table.

```
GRANT ALL PRIVILEGES ON easydb.employees to hanson@'%' IDENTIFIED BY 'your#
password';
```

In the easydb database, create an account named hanson, which can be used to log on only from 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';
```

In the easydb database, create a read-only account named actro, which can be used to log on from any host.

```
GRANT SELECT ON easydb.* to actro@'%' IDENTIFIED BY 'your#password';
```

REVOKE statements

Syntax rules:

- Delete the permissions at a certain 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 of the account at the database and table levels.

```
REVOKE ALL PRIVILEGES, GRANT OPTION
    FROM user [, user] ...
```

Examples:

Delete the CREATE, DROP, and INDEX permissions from hanson@'%' on the easydb.emp table.

```
REVOKE CREATE,DROP,INDEX ON easydb.emp FROM hanson@'%';
```

Delete all permissions from the account lily@30.xx.xx.96.

```
REVOKE ALL PRIVILEGES,GRANT OPTION FROM lily@30.xx.xx.96;
```

Notice:
GRANT OPTION must be added to the statement for compatibility with MySQL.

SHOW GRANTS statements

Syntax rules:

```
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 console

Note:
You can only delete standard accounts that are created in the console.

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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

Syntax rules:

```
DROP USER user [, user] ...
```

Examples:

Delete the account lily@30.xx.xx.96:

```
DROP USER lily@30.xx.xx.96;
```

5.6 Database management

5.6.1 Create a database

After creating a PolarDB-X instance, you need to create a database that contains one or more ApsaraDB RDS for MySQL instances. You can create a database in four steps.

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) permission. For more information, see Apsara Stack Console User Guide for the RAM management topic.

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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. On the **Enter Basic Information** page, enter the following information.

Parameter	Description
Partition Mode	Select **Horizontal Partitioning**.

Parameter	Description
Database Name	Specify a name for the PolarDB-X database. The name must meet the following requirements: • The name must be 2 to 24 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 (_). • The database name must be unique on the PolarDB-X instance.
Character Set	Select utf8, gbk, latin1, or utf8mb4.
PolarDB-X Link Password	Set the PolarDB-X connection password. The rules are as follows: • The password must be 8 to 30 characters in length • The password must contain three of the following character types: uppercase letters, lowercase letters, numbers, and underscores (_).
Confirm Password	Enter the password again.

6. Click **Next**.
7. On the **Select RDS Instance** page, you can click the **Buy New RDS Instance** or **Use Existing RDS Instance** tab.
   a) **Buy New RDS Instance**: Click the **Buy New RDS Instance** tab.
   b) Set **Storage Type**, **Edition**, and **Storage Capacity**.
   c) Click **Next**.
   a) **Use Existing RDS Instance**: Click the **Use Existing RDS Instance** tab.
   b) Click the ApsaraDB RDS for MySQL instances to be added on the left.
   c) Click > to move the selected instances to the **Selected RDS Instances** section on the right.
   d) Click **Next**.
8. After all prechecks are passed on the **Precheck** page, click **Next**.

   Note:
   If a precheck fails, rectify the configuration as prompted.
9. On the **Preview** page, click **Next** and 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 in database management in the console.

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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**. 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 using the MySQL client, note the following points:

> **Note:**
>
> - Some MySQL clients of earlier versions have limits on the user name length, which cannot be more than 16 characters. The PolarDB-X database name and user name are the same. If the database name exceeds 16 characters, an error is reported.
> - When using the MySQL client, you must add the `-c` parameter to the hint command. In PolarDB-X, HINT is implemented by using annotations. If the `-c` parameter is not added, the annotation is lost and the PolarDB-X hint is lost.

5.6.3 Perform smooth scale-out

When the underlying storage of the logical database reaches the physical bottleneck, for example, when the remaining disk space is about 30%, you can smoothly scale it out

to improve the performance. The smooth scale-out process is divided into four steps: configuration > migration > switchover > 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. In 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. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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**. 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 then click **Next**.
9. After all prechecks are passed on the **Precheck** page, click **Next**.

> **Note:**
> If a precheck fails, rectify the configuration as prompted.

10. On the **Select RDS Instance** page, you can click the **Buy New RDS Instance** or **Use Existing RDS Instance** tab.

    a) **Buy New RDS Instance**: Click the **Buy New RDS Instance** tab.

    b) Set **Storage Type**, **Edition**, and **Storage Capacity**.

    c) Click **Next**.

    a) **Use Existing RDS Instance**: Click the **Use Existing RDS Instance** tab.

    b) Click the ApsaraDB RDS for MySQL instances to be added on the left.

    c) Click > to move the selected instances to the **Selected RDS Instances** section on the right.

    d) Click **Next**.

11. On the **Preview** page, click **Start Scale-out**.

    Note:
    By default, the console evenly distributes the physical database shards to the ApsaraDB RDS for MySQL instances 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 details of the scale-out task.

Migration

Some physical database shards are migrated during smooth scale-out.

The migration does not change the data in the source database, and therefore it does not affect online services. Before the switchover, you can cancel the smooth scale-out operation through a rollback.

Note:

- This is because before the switchover, the current scale-out operation does not have a real impact on the data in the source database.
- During scale-out, the binary log files of the source ApsaraDB RDS for MySQL 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. Generally, the remaining disk space should be more than 30%. If the disk space cannot be guaranteed , you can submit a ticket to expand the ApsaraDB RDS for MySQL storage space.

- 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 data description language (DDL) tasks in the console or connect to the PolarDB-X instance to directly run DDL statements. Otherwise, the scale-out task may fail.
- Make sure that all tables in the source database have primary keys before scale-out.

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 this 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 process , 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 then **OK**.

   During the switchover process, a switchover task is generated and displayed in the Task Progress dialog box.
3. After the switchover is complete, the **Clean Up** button appears in the **Task Progress** dialog box, which means that the switchover task is complete.

Cleanup

In this step, the migrated database shards are deleted from the source ApsaraDB RDS for MySQL instance.

1. After switchover is complete, click **Clean Up** next to the target 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 complete, the smooth scale-out process ends. The new ApsaraDB RDS for MySQL instance becomes the storage node of the PolarDB-X logical database.
- Currently, you can implement smooth scale-out by migrating physical database shards. If no further scale-out is allowed after the number of database shards exceeds the capacity of a single ApsaraDB RDS for MySQL instance, you can submit a ticket to apply for increasing the number of database shards and scaling out the database. In this case, Hash calculation is performed again to reallocate data.
- The cleanup task deletes database shards that are no longer used after the current scale-out. You can back up the database shards before running 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. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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 **Data Indexes** and **Query Time**. Then, the corresponding monitoring information appears.

Note:

For more information about instance-level monitoring, see View monitoring information.

5.6.5 Set the IP address whitelist

PolarDB-X provides the access control function. Only IP addresses in the whitelist of a database can access the database.

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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**. The **Basic Information** page of the database appears.
6. On the **Basic Information** page of the database, choose **Data Security** > **Whitelist Settings** from the left-side navigation pane.
7. On the Whitelist Settings page, click **Manually Modify**.
8. Enter the IP address that is allowed to access the database, and click **OK**.

Note:

- The whitelist supports the following formats:
  - IP address, for example, 192.168.1.1.
  - CIDR IP address, for example, 192.168.1.1/24.
  - IP address with an asterisk (*) as the wildcard, for example, 192.168.1. *, indicating that access is allowed from any host with an IP address in the range of 192.168.1.1 to 192.168.1.254.
  - CIDR block, for example, 192.168.1.1-192.168.1.254.
- If you want to add multiple IP addresses or CIDR blocks, separate them with commas (,) without spaces, 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. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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

Context

When using a PolarDB-X instance, you need to access ApsaraDB RDS for MySQL. If the network configuration of the connected ApsaraDB RDS for MySQL instance changes, for example, if the zone is switched or the network type is changed from classic to Virtual Private Cloud (VPC), the network connection between the PolarDB-X instance and the ApsaraDB RDS for MySQL instance is broken. As a result, the PolarDB-X instance cannot access the ApsaraDB RDS for MySQL instance. In this case, you must manually fix the database shard link in the PolarDB-X console to restore the network connection from the PolarDB-X instance to the ApsaraDB RDS for MySQL instance.

Procedure

1. Log on to the PolarDB-X console.
2. On the **DRDS Instance Management** page, find the target instance.
3. Click the 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**. The **Basic Information** page of the database appears.
6. In the **Shortcuts** section, click **Fix Database Shard Connections**.
7. In the pop-up window, 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 Help statements

This topic describes all the auxiliary SQL commands of PolarDB-X and their descriptions.

SHOW HELP statements:

```
mysql> show help;
+----------------------------------------
+------------------------------------------------------
+---------------------------------------------+
| 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 from 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 user; 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 user    |
| explain execute SQL                | Report sql on physical db plan info                | explain
execute select count(*) from 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

SHOW RULE [FROM tablename]

Usage notes:

- show rule: shows the partitioning status of each logical table in a database.
- show rule from tablename: shows the partitioning status of a specified logical table in a database.

The following describes the meanings of 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 NULL.
- DB_PARTITION_POLICY: indicates the database sharding policy. Options 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 NULL.
- TB_PARTITION_POLICY: indicates the table sharding policy. Options are Hash and date policies such as MM, DD, MMDD, and WEEK.
- TB_PARTITION_COUNT: indicates the number of table shards.

```
mysql> show rule;
+------+-------------+-----------+-----------------+--------------------
+--------------------+-----------------+--------------------+--------------------+
| ID   | TABLE_NAME   | BROADCAST | DB_PARTITION_KEY | DB_PARTITION_POLICY | DB_PARTITI
ON_COUNT | TB_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 run this SQL statement to view the sharding rules of logical tables in a database. It displays more detailed information than the SHOW RULE command.

The following describes the meanings of important columns:

- BROADCAST: indicates whether the table is a broadcast table. 0 indicates "No" and 1 indicates "Yes".
- JOIN_GROUP: a reserved field.
- ALLOW_FULL_TABLE_SCAN: indicates whether to allow data querying 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, which is a full table scan.
- DB_NAME_PATTERN: The value 0 between {} in DB_NAME_PATTERN is a placeholder. When the SQL statement is run, this value is replaced by the value of DB_RULES_STR, with the number of digits 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 value 0 between {} in TB_NAME_PATTERN is a placeholder. When the SQL statement is run, this value is replaced by the value of TB_RULES_STR, with the number of digits 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 a set of database and table shard keys. When database sharding and table sharding coexist, the database shard key is placed before the table shard key.

- DEFAULT_DB_INDEX: indicates the database shard in which a single database and a single table are stored.

```
mysql> show full rule;
+------+-------------+-----------+------------+----------------------
+-----------------------------------------------
+-----------------------------------------------------+----------------
+----------------------------------+---------------
+-----------------------------------------------+
| ID   | TABLE_NAME   | BROADCAST | JOIN_GROUP | ALLOW_FULL_TABLE_SCAN | DB_NAME_PA
TTERN                          | DB_RULES_STR                          | TB_NAME_PATTERN |
TB_RULES_STR                 | PARTITION_KEYS | DEFAULT_DB_INDEX                      |
+------+-------------+-----------+------------+----------------------
+-----------------------------------------------
+-----------------------------------------------------+----------------
+----------------------------------+---------------
+-----------------------------------------------+
|   0 | dept_manager |       0 | NULL      |               0 | SEQ_TEST_1487767780814RGKKSEQ_
TEST_WNJG_0000_RDS   | NULL                                  | dept_manager    | NULL
          | NULL          | SEQ_TEST_1487767780814RGKKSEQ_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 | SEQ_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_TEST_1487767780814RGKKSEQ_TEST_WNJG_
0000_RDS |
+------+-------------+-----------+------------+----------------------
+-----------------------------------------------
+-----------------------------------------------------+----------------
+----------------------------------+---------------
+-----------------------------------------------+
3 rows in set (0.01 sec)
```

SHOW TOPOLOGY FROM tablename

You can run this SQL statement to view the topology of a specified logical table, that is, the database shards to which data in the logical table is partitioned and the table shards in each database shard.

```
mysql> show topology from emp;
+------+-------------------------------------------------+------------+
| 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 run this SQL statement to view the set of database shard keys and table shard keys , which are separated by commas (,). If the final result contains 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.

```
mysql> show partitions from emp;
+-----------+
| KEYS      |
+-----------+
| emp_no,id |
+-----------+
1 row in set (0.00 sec)
```

SHOW BROADCASTS

You can run this SQL statement to view the list of broadcast tables.

```
mysql> show broadcasts;
+------+------------+
| ID   | TABLE_NAME |
+------+------------+
|    0 | brd2       |
|    1 | brd_tbl    |
+------+------------+
2 rows in set (0.01 sec)
```

SHOW DATASOURCES

You can run this SQL statement to view the information about the underlying storage, including the database name, database group name, connection URL, user name, storage type, read/write weight, and connection pool information.

The following describes the meanings of 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, such as the primary and secondary databases after data replication through ApsaraDB RDS for MySQL. It is mainly used for read/write splitting and primary/secondary switchovers.
- URL: indicates the connection information of the underlying ApsaraDB RDS for MySQL instance.

- 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 many read requests, you can use the read/write splitting function of PolarDB-X to distribute the read traffic. 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 configured weight.
- WRITE_WEIGHT: indicates the write weight. For more information, see READ_WEIGHT.

```
mysql> show datasources;
+------+--------------------------+-----------------------------------------------
+-----------------------------------------------
+-------------------------------------------------------------------------------------+-----------
+-------+------+------+------+-------------+----------+-------------+--------------
+---------------------------------------------+-------------+-------------+
| 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_1487767780814RGKKSEQ_TEST_WNJG_0000_RDS | jdbc:mysql://rds1ur80kc
v8g3t6p3ol.mysql.rds.aliyuncs.com:3306/seq_test_wnjg_0000 | jnkinsea0 | mysql | 0    | 24
  | 72  | 15       | 5000    | 0        | 1        | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0000_
iiab | 10       | 10       |
|   1 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0001_iiab_2
 | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0001_RDS | jdbc:mysql://rds1ur80kc
v8g3t6p3ol.mysql.rds.aliyuncs.com:3306/seq_test_wnjg_0001 | jnkinsea0 | mysql | 0    | 24
  | 72  | 15       | 5000    | 0        | 1        | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0001_
iiab | 10       | 10       |
|   2 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0002_iiab_3
 | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0002_RDS | jdbc:mysql://rds1ur80kc
v8g3t6p3ol.mysql.rds.aliyuncs.com:3306/seq_test_wnjg_0002 | jnkinsea0 | mysql | 0    | 24
  | 72  | 15       | 5000    | 0        | 1        | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0002_
iiab | 10       | 10       |
|   3 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0003_iiab_4
 | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0003_RDS | jdbc:mysql://rds1ur80kc
v8g3t6p3ol.mysql.rds.aliyuncs.com:3306/seq_test_wnjg_0003 | jnkinsea0 | mysql | 0    | 24
  | 72  | 15       | 5000    | 0        | 1        | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0003_
iiab | 10       | 10       |
|   4 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0004_iiab_5
 | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0004_RDS | jdbc:mysql://rds1ur80kc
v8g3t6p3ol.mysql.rds.aliyuncs.com:3306/seq_test_wnjg_0004 | jnkinsea0 | mysql | 0    | 24
  | 72  | 15       | 5000    | 0        | 1        | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0004_
iiab | 10       | 10       |
|   5 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0005_iiab_6
 | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0005_RDS | jdbc:mysql://rds1ur80kc
v8g3t6p3ol.mysql.rds.aliyuncs.com:3306/seq_test_wnjg_0005 | jnkinsea0 | mysql | 0    | 24
  | 72  | 15       | 5000    | 0        | 1        | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0005_
iiab | 10       | 10       |
```

```
|   6 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0006_iiab_7
 | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0006_RDS | jdbc:mysql://rds1ur80kc
v8g3t6p3ol.mysql.rds.aliyuncs.com:3306/seq_test_wnjg_0006 | jnkinsea0 | mysql | 0    | 24
  | 72  | 15        | 5000    | 0         | 1          | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0006_
iiab | 10        | 10        |
|   7 | seq_test_1487767780814rgkk | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0007_iiab_8
 | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0007_RDS | jdbc:mysql://rds1ur80kc
v8g3t6p3ol.mysql.rds.aliyuncs.com:3306/seq_test_wnjg_0007 | jnkinsea0 | mysql | 0    | 24
  | 72  | 15        | 5000    | 0         | 1          | rds1ur80kcv8g3t6p3ol_seq_test_wnjg_0007_
iiab | 10        | 10        |
+------+---------------------------+-----------------------------------------------
+-----------------------------------------------
+-------------------------------------------------------------------------------+-----------
+-------+------+------+------+-------------+----------+-------------+--------------
+---------------------------------------------+-------------+-------------+
8 rows in set (0.01 sec)
```

SHOW NODE

You can run this SQL statement to view the accumulative number of read and write operations and accumulative read/write weights of a physical database.

The following describes the meanings of 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, not the configured percentage.
- SLAVE_READ_PERCENT: indicates the actual percentage of read-only queries processed by the secondary ApsaraDB RDS for MySQL instances, not the configured 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 read/write weight ratio has been changed, these values do not immediately reflect the latest read/write weight ratio, which appears after a long period of time has passed.

```
mysql> show node;
+------+-------------------------------------------------+------------------
+-----------------+--------------------+-------------------+
| 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 SQL tuning statements

SHOW [FULL] SLOW [WHERE expr] [limit expr]

SQL statements that take more than 1 second to execute are slow SQL statements. Slow logical SQL statements are the slow SQL statements sent from an application to a PolarDB-X instance.

- SHOW SLOW: You can run this SQL statement to view the 100 slowest logical SQL queries that are recorded since the PolarDB-X instance is started or the last time when CLEAR SLOW is executed.

  > **Note:**
  > The recorded 100 slowest SQL queries are stored in the PolarDB-X system. When the PolarDB-X instance is restarted or executes CLEAR SLOW, these queries will be discarded.

- SHOW FULL SLOW: You can run this SQL statement to view all the slow logical SQL queries that are recorded and persisted to the built-in database of the PolarDB-X instance since the PolarDB-X instance is started. The upper limit for the number of records is specified in the specifications of the PolarDB-X instance. The PolarDB-X instance scrolls to delete the earliest slow SQL statements. If the specifications of the PolarDB-X instance is 4-core 4 GB, a maximum of 10,000 slow SQL statements can be recorded, including slow logical and physical SQL statements. If the specifications of the PolarDB-X instance is 8-core 8 GB, a maximum of 20,000 slow SQL statements can be recorded, including slow logical and physical SQL statements. The same rule applies to other specifications.

The following describes the meanings of 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 time when the SQL statement is executed.
- AFFECT_ROW: For data manipulation language (DML) statements, this parameter indicates the number of affected rows. For query statements, this parameter indicates the number of returned records.

```
mysql> show slow where execute_time > 1000 limit 1;
+-----------+---------------------+--------------+------------+-----------+
| 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 1 second to execute are slow SQL statements. Slow logical SQL statements are the slow SQL statements sent from an application to a PolarDB-X instance.

- SHOW SLOW: You can run this SQL statement to view the 100 slowest logical SQL queries that are recorded since the PolarDB-X instance is started or the last time when CLEAR SLOW is executed.

  Note:
  The recorded 100 slowest SQL queries are stored in the PolarDB-X system. When the PolarDB-X instance is restarted or executes CLEAR SLOW, these queries will be discarded.

- SHOW FULL SLOW: You can run this SQL statement to view all the slow logical SQL queries that are recorded and persisted to the built-in database of the PolarDB-X instance since the PolarDB-X instance is started. The upper limit for the number of records is specified in the specifications of the PolarDB-X instance. The PolarDB-X instance scrolls to delete the earliest slow SQL statements. If the specifications of the PolarDB-X instance is 4-core 4 GB, a maximum of 10,000 slow SQL statements can be recorded, including slow logical and physical SQL statements. If the specifications of the PolarDB-X instance is 8-core 8 GB, a maximum of 20,000 slow SQL statements can be recorded, including slow logical and physical SQL statements. The same rule applies to other specifications.

The following describes the meanings of 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 time when the SQL statement is executed.
- AFFECT_ROW: For data manipulation language (DML) statements, this parameter indicates the number of affected rows. For query statements, this parameter indicates the number of returned records.

```
mysql> show physical_slow;
+----------------+---------------------------------+--------------------+-------------
+-----------------+-----------------------+----------------------+------------
+-----------------+
| GROUP_NAME     | DBKEY_NAME                   | START_TIME         | EXECUTE_TIME |
SQL_EXECUTE_TIME | GETLOCK_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 run this SQL statement to clear the 100 slowest logical SQL queries and the 100 slowest physical SQL queries that are recorded since the PolarDB-X instance is started or the last time when CLEAR SLOW is executed.

Note:

Both SHOW SLOW and SHOW PHYSICAL_SLOW can be executed to display the 100 slowest SQL statements. If CLEAR SLOW has not been executed for a long time, these SQL statements may have been recorded a long time ago. Therefore, after SQL tuning statements are executed, we recommend that you execute CLEAR SLOW. After the system runs for a while, check the tuning results of slow SQL statements.

```
mysql> clear slow;
Query OK, 0 rows affected (0.00 sec)
```

EXPLAIN SQL

You can run this SQL statement to view the execution plan of a specified SQL statement in the PolarDB-X. Note that this SQL statement is not truly executed.

Example:

You can run this SQL statement to view the execution plan of the SQL `select * from doctest` statement. The doctest table is stored in database shards according to values in the id column. According to the execution plan, the SQL statement will be routed to each database shard for execution, and the execution results will be aggregated.

```
mysql> explain select * from doctest;
+-----------------------------------------------+-----------------------------------
+--------+
| 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 run this SQL statement to view the execution plan of the SQL `select * from doctest where id = 1` statement. The doctest table is stored in database shards according to values in the id column. According to the execution plan, the PolarDB-X instance will calculate a specified database shard based on the shard key, which is id, directly route the SQL statement to the database shard, and aggregate the execution results.

```
mysql> explain select * from doctest where id = 1;
+-----------------------------------------------
+-------------------------------------------------------------------+--------+
| 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 run this SQL statement to view the execution plan of a specified SQL statement in the PolarDB-X. Note that this SQL statement is not truly executed.

```
mysql> explain detail select * from doctest where id = 1;
+-----------------------------------------------
+-----------------------------------------------------------------------------------------------------------------------------------
+--------+
| 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 run this SQL statement to view the execution plan of underlying storage. This statement is equivalent to the MySQL EXPLAIN statement.

```
mysql> explain execute select * from tddl_mgr_log limit 1;
+----+-------------+--------------+------+---------------+------+---------+------+------
+-------+
| 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 run these SQL statements to view the execution results of an SQL statement. Note that you must use TRACE SQL and SHOW TRACE together. The difference between TRACE SQL and EXPLAIN SQL is that TRACE SQL is truly executed.

For example, you can run these statements to view the execution results of the select 1 statement.

```
mysql> trace select 1;
+---+
```

```
| 1 |
+---+
| 1 |
+---+
1 row in set (0.03 sec)
mysql> show trace;
+------+----------+---------------+----------------------------------+--------------
+-------------------------+------+-----------+--------+
| ID   | TYPE     | GROUP_NAME    | DBKEY_NAME                       | TIME_COST(MS) | CONNECTION
_TIME_COST(MS) | ROWS | 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 run this SQL statement to check a data table. This SQL statement is mainly used when a table failed to be created by using a data definition language (DDL) statement.

- If the data table is a table shard, this SQL statement allows you to check whether any underlying physical table shard is missing and whether the columns and indexes of the underlying physical table are consistent.
- If the data table is a single-database non-partition table, this SQL statement allows you to check whether this table exists.

```
mysql>  check table tddl_mgr_log;
+-----------------------+-------+----------+----------+
| TABLE                 | OP    | MSG_TYPE | MSG_TEXT |
+-----------------------+-------+----------+----------+
| TDDL5_APP.tddl_mgr_log | check | status   | OK       |
+-----------------------+-------+----------+----------+
1 row in set (0.56 sec)
mysql> check table tddl_mg;
+-------------------+-------+----------+---------------------------------------+
| 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 run this SQL statement to obtain the information about a table. This command aggregates the data of all underlying physical table shards.

The following describes the meanings of 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, is a VARCHAR or BLOB field. The value Fixed indicates that the row length is constant, for example, is 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, in 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.

```
mysql> show table status like 'multi_db_multi_tbl';
+-------------------+--------+---------+------------+------+----------------+-------------
+-----------------+-------------+-----------+----------------+----------------------
+-------------+------------+-----------------+----------+----------------+---------+
| 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_OPT
IONS | 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)
```

The combination of the SHOW TABLE STATUS statement and the PolarDB-X SCAN hint allows you to view the data volume of each physical table shard.

```
mysql> /! TDDL:SCAN='multi_db_multi_tbl'*/show table status like 'multi_db_multi_tbl';
+---------------------+--------+---------+------------+------+----------------+-------------
+-----------------+-------------+-----------+----------------+--------------------
+-------------+------------+-----------------+----------+----------------+---------
+--------------+
| Name              | Engine | Version | Row_format | Rows | Avg_row_length | Data_lengt
h | 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 | 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

SHOW [FULL] STATS

You can run this SQL statement to view the overall statistics. The statistics are instantaneous values. Note that the results of `SHOW FULL STATS` vary with the version.

The following describes the meanings of important columns:

- QPS: the number of queries per second (QPS) sent from an application to the PolarDB-X instance. These queries are usually called logical QPS.
- RDS_QPS: the number of QPS sent from the PolarDB-X instance to an ApsaraDB RDS for MySQL instance. These queries are usually called 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 conflicts that occur on primary keys or unique keys per second.
- MERGE_QUERY_PER_SECCOND: the number of queries processed on multiple tables through database sharding and table sharding per second.
- 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 response time (RT) is usually called 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 called 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 have been processed by the PolarDB-X instance since startup.

- AGGREGATE_QUERY_PER_SECCOND: the number of aggregate SQL queries processed by the PolarDB-X instance per second.
- AGGREGATE_QUERY_COUNT: the total number of aggregate SQL queries that have been processed by the PolarDB-X instance.
- TEMP_TABLE_CREATE_PER_SECCOND: the number of temporary tables created in the PolarDB-X instance per second.
- TEMP_TABLE_CREATE_COUNT: the total number of temporary tables that have been created in the PolarDB-X instance since startup.
- MULTI_DB_JOIN_PER_SECCOND: the number of multi-database JOIN queries processed by the PolarDB-X instance per second.
- MULTI_DB_JOIN_COUNT: the number of multi-database JOIN queries that have been processed by the PolarDB-X instance since startup.

```
mysql> show stats;
+------+---------+----------+------------------+-----------------+----------------------
+--------------------+--------+-----------+-------------+--------------+---------------+
| QPS  | RDS_QPS | SLOW_QPS | PHYSICAL_SLOW_QPS | ERROR_PER_SECOND | MERGE_QUER
Y_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)
mysql> show full stats;
+------+---------+----------+------------------+-----------------+----------------------
+-----------------------+--------------------+-----------------------------+--------
+------------+-------------+--------------+---------------+----------------------
+----------------+---------------------------+----------------------
+-----------------------------+------------------------+-------------------------
+---------------------+-------+---------+-------------+------------+
| QPS  | RDS_QPS | SLOW_QPS | PHYSICAL_SLOW_QPS | ERROR_PER_SECOND | VIOLATION_
PER_SECOND | MERGE_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 | AGGREGATE_
QUERY_PER_SECOND | AGGREGATE_QUERY_COUNT | TEMP_TABLE_CREATE_PER_SECOND |
TEMP_TABLE_CREATE_COUNT | MULTI_DB_JOIN_PER_SECOND | MULTI_DB_JOIN_COUNT | CPU
  | FREEMEM | FULLGCCOUNT | FULLGCTIME |
+------+---------+----------+------------------+-----------------+----------------------
+-----------------------+--------------------+-----------------------------+--------
+------------+-------------+--------------+---------------+----------------------
+----------------+---------------------------+----------------------
+-----------------------------+------------------------+-------------------------
+---------------------+-------+---------+-------------+------------+
| 1.63 |   1.68 |    0.03 |          0.03 |          0.02 |             0.00 |             0.00 |             6 |
              0.01 | 157.13 |     51.14 |     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 run this SQL statement to view the capacity and performance information of a physical database. 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 describes the meanings of important columns:

- NAME: the PolarDB-X internal tag that represents a logical PolarDB-X database corresponding to the database shard. The value is different from the name of the logical PolarDB-X 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 logical PolarDB-X 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 logical PolarDB-X 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 returned by the MySQL SHOW GLOBAL STATUS command. For more information, see MySQL official documentation.

```
mysql> show db status;
+------+--------------------------+-------------------+------------------+------------
+--------+---------------+
| 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 run this SQL statement to view the capacity and performance information of a table shard in a physical database, which is also called 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 describes the meanings of important columns:

- NAME: the PolarDB-X internal tag that represents a logical PolarDB-X database corresponding to the database shard. The value is different from the name of the logical PolarDB-X 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 used for filtering in the statement, the TOTAL row indicates the total amount of capacity of the database shard. If the LIKE keyword is not used for filtering 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 used for filtering in the statement, the TOTAL row indicates the total amount of capacity of the table shard. If the LIKE keyword is not used for filtering 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 table shards obtained through filtering.
- 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 returned by the MySQL SHOW GLOBAL STATUS command. For more information, see MySQL official documentation.

```
mysql> show full db status like hash_tb;
+------+-------------------------+-------------------+------------------+---------------
+-----------+--------+---------------+
| 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 | 7.63% |            |
|  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 | 7.63% |            |
|  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 | 7.63% |            |
|  16 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0004 |
hash_tb_09    |  1.515625 | 7.63% |            |
|  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 | 7.63% |            |
|  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 | 7.63% |            |
|  22 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0006 |
hash_tb_13    |  1.515625 | 7.63% |            |
|  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 | 7.63% |            |
|  25 | drds_db_1516187088365daui | 100.100.64.1:59077 | drds_db_xzip_0007 |
hash_tb_15    |  0.359375 | 1.81% |            |
```

```
+------+--------------------------+------------------+------------------+---------------
+-----------+--------+----------------+
```

5.7.6 SHOW PROCESSLIST and KILL commands

Note:

- If the version of PolarDB-X is 5.1.28-1408022 or later, PolarDB-X support the SHOW PROCESSLIST and KILL commands for both 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 support the SHOW PROCESSLIST and KILL commands only for physical connections. For more information, see SHOW PROCESSLIST and KILL commands in earlier versions.

SHOW PROCESSLIST command

In a PolarDB-X instance, you can run the SHOW PROCESSLIST command to view information such as connections to the PolarDB-X instance and SQL statements that are being executed in the PolarDB-X instance.

Syntax

```
SHOW [FULL] PROCESSLIST
```

Examples

```
mysql> SHOW PROCESSLIST\G
     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 following describes the meanings of the fields in the result set:

- ID: the ID of the connection. The value is a long-type number.
- USER: the name of the user who sets up the connection.
- HOST: the IP address and port number of the host that sets up the connection.
- DB: the name of the database accessed by the connection.

- COMMAND: the usage state of the connection. Currently, this field can be set to the following values:
  - Query: the current connection is executing an SQL statement.
  - Sleep: 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 on the connection.
  - When the value of COMMAND is Sleep, this field indicates how long the connection has been in the idle state.
- STATE: currently, no meaning has been assigned for this field. The value is constantly empty.
- INFO:
  - When the value of COMMAND is Query, this field indicates the content of the SQL statement that is being executed on 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 1000 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 command

In a PolarDB-X instance, you can run the SHOW PHYSICAL_PROCESSLIST command 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` command may be truncated. In this case, you can run the `SHOW FULL PHYSICAL_PROCESSLIST` command 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` command. For more information, see SHOW PROCESSLIST Syntax.

 **Note:**

Different from ApsaraDB RDS for MySQL, the PolarDB-X instance returns a string instead of a number in the ID column of a physical connection.

```
mysql> SHOW PHYSICAL_PROCESSLIST\G
*************************** 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 command

The KILL command is used 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 do not have the permission to run the KILL command on 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 such as the MySQL command line and to connect to the PolarDB-X instance, and then run the KILL command on the PolarDB-X instance.

Syntax

```
KILL PROCESS_ID | 'PHYSICAL_PROCESS_ID' | 'ALL'
```

The KILL command can be used in the following ways:

- Run `KILL PROCESS_ID` 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` command.

  Running the `KILL PROCESS_ID` command in the PolarDB-X instance will terminate both logical and physical SQL statements that are being executed on this connection, and disconnect this connection.

  The PolarDB-X instance does not support the `KILL QUERY` command.

- Run KILL 'PHYSICAL_PROCESS_ID' 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 command.

  > **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 command.

  Examples:

  ```
  mysql> KILL '0-0-521570';
  Query OK, 0 rows affected (0.01 sec)
  ```

- Run KILL 'ALL' 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 some SQL statements, you can use the KILL 'ALL' command to terminate all the physical SQL statements that are being executed in the current logical PolarDB-X database.

  All physical SQL statements indicated by PROCESS that meet the following conditions can be terminated by running KILL 'ALL':

  - 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. In other words , the value of COMMAND is Query.

5.7.7 SHOW PROCESSLIST and KILL commands in earlier versions

> **Note:**
>
> - If the version of PolarDB-X is 5.1.28-1408022 or later, PolarDB-X support the SHOW PROCESSLIST and KILL commands for both 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 only supports the SHOW PROCESSLIST and KILL commands for physical connections. For more information, see this topic.

SHOW PROCESSLIST command

In a PolarDB-X instance, you can run the SHOW PROCESSLIST command to view information about all the SQL statements that are being executed on the ApsaraDB RDS for MySQL instances.

Syntax

```
SHOW [FULL] PROCESSLIST
```

When an SQL statement is excessively long, the responses of the SHOW PROCESSLIST command may be truncated. In this case, you can run the SHOW FULL PROCESSLIST command 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 command. For more information, see SHOW PROCESSLIST Syntax.

```
mysql> SHOW PROCESSLIST\G
*************************** 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 command

The KILL command is used to terminate an SQL statement that is being executed.

The instance connects to an ApsaraDB RDS for MySQL instance by using the username created by the instance on the ApsaraDB RDS for MySQL instance. Therefore, if you directly connect to the ApsaraDB RDS for MySQL instance, you do not have the permission to run the KILL command on a request initiated by the instance.

To terminate an SQL statement that is being executed on the instance, you must use tools such as the MySQL command line and to connect to the instance, and then run the KILL command on the instance.

Syntax

```
KILL 'PROCESS_ID' | 'ALL'
```

The KILL command can be used in the following ways:

- Run `KILL 'PROCESS_ID'` to terminate a specified SQL statement.

  The PROCESS_ID parameter is obtained from the ID column in the responses of the `SHOW PROCESSLIST` command.

  Note:

  Different from ApsaraDB RDS for 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 command.

  Examples

  ```
  mysql> KILL '0-0-521570';
  Query OK, 0 rows affected (0.01 sec)
  ```

- Run `KILL 'ALL'` 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 use the `KILL 'ALL'` command 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 running `KILL 'ALL'`:

  - The value of the User parameter for the physical SQL statement indicated by PROCESS is a username created by the instance in the ApsaraDB RDS for MySQL instance.
  - The physical SQL statement indicated by PROCESS is executing a query. In other words, the value of COMMAND is Query.

  PolarDB-X instances in earlier versions do not support the `KILL 'ALL'` command. An error will be reported if this command 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 influence execution plans of SQL statements by using relevant syntax, to specially optimize the SQL statements. PolarDB-X also provides special hint syntax.

For example, if you know the target data is stored in table shards in certain database shards and you need to route the 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 `*/`, namely, `+TDDL:node('node_name')`, is a PolarDB-X hint. The hint specifies the ApsaraDB RDS for MySQL database shard where the SQL statement is to 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, if you need to run an SQL statement that contains a PolarDB-X hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because PolarDB-X hints are based on the MySQL Comment Syntax. Otherwise, the client deletes the PolarDB-X hint and then sends the SQL statement to the server for execution, which causes the hint to fail to take effect. For more information, see MySQL Client Options.

PolarDB-X hint syntax

Basic syntax

```
/*+TDDL: hint_command [hint_command ...] */
/! +TDDL: hint_command [hint_command ...] */
```

PolarDB-X hints are based on the MySQL Comment Syntax. The hint statements are located between `/*` and `*/` or between `/!` and `*/`, and must begin with `+TDDL:`. The hint_command

parameter indicates a PolarDB-X hint command related to the specific operation. Multiple hint_command parameters are separated by spaces.

Examples

```
# Query the names of physical tables 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() functions are PolarDB-X hint commands. Hint commands are separated by spaces.

- Use one hint in an SQL statement:

  PolarDB-X allows you to use hints in data manipulation language (DML), data definition language (DDL), and data access language (DAL) statements. The following describes the syntax in detail.

  - For all statements that support hints, you can specify a hint at the beginning of the statements, for 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, for example:

    ```
    SELECT /*+TDDL: ... */  ...
    INSERT /*+TDDL: ... */  ...
    REPLACE /*+TDDL: ... */  ...
    UPDATE /*+TDDL: ... */  ...
    DELETE /*+TDDL: ... */  ...
    ...
    ```

    Note:

    Different hints may be applicable to different syntaxes. For more information about the applicable syntaxes, see the documentation of hint commands.

- Use multiple hint commands in an SQL statement:

  PolarDB-X allows you to use multiple hint commands in SQL statements that contain hints.

  ```
  SELECT /*+TDDL:node(0) slave()*/ ... ;
  ```

  PolarDB-X has the following limitations on the use of multiple hint commands:

  ```
  # A single SQL statement cannot contain multiple hint statements.
  SELECT /*+TDDL:node(0)*/ /*+TDDL:slave()*/ ... ;
  # An SQL statement that contains a hint cannot contain duplicate hint commands.
  SELECT /*+TDDL:node(0) node(1)*/ ... ;
  ```

PolarDB-X hint classification

PolarDB-X hints are classified into the following major categories according to 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

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 delay of several milliseconds. If you need to read changed data immediately after 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, 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 earlier versions, see Read/write splitting.

Syntax

```
/*+TDDL:
   master()
   | slave()
```

```
*/
```

With this custom hint, you can specify whether to run an SQL statement on a primary or read-only ApsaraDB RDS for MySQL instance. With the custom hint `/*+TDDL:slave()*/`, if a primary ApsaraDB RDS for MySQL instance is configured with 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 run the SQL statement.

Note:

- hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/`.
- In the MySQL command-line client, if you need to run an SQL statement that contains a hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because hints are based on the MySQL Comment Syntax. Otherwise, the client deletes the hint and then sends the SQL statement to the server for execution, which causes the hint to fail to take effect. For more information, see MySQL Client Options.

Examples

- Specify a primary ApsaraDB RDS for MySQL instance to run an SQL statement:

  ```
  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 for execution.
- Specify a read-only ApsaraDB RDS for MySQL instance to run an SQL statement:

  ```
  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 the allocated weight.

Note

- 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 for execution.
- The PolarDB-X hint `/*+TDDL:slave()*/` allows you to route the SQL statement randomly to a read-only ApsaraDB RDS for MySQL instance based on the configured weight for

execution. 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 run 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:

- hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/`.
- In the MySQL command-line client, if you need to run an SQL statement that contains a hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because hints are based on the MySQL Comment Syntax. Otherwise, the client deletes the hint and then sends the SQL statement to the server for execution, which causes the hint to fail 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 Specify a database shard to run an SQL statement

When running SQL commands in a PolarDB-X instance, you may find that some SQL statements are not supported by the PolarDB-X instance. In this case, you can use the NODE HINT provided by PolarDB-X, to route the SQL statements to one or more database shards for execution. In addition, if you need to query the data in a specified database shard or the data in a specified table shard in a known database shard, you can use the NODE HINT to directly route the SQL statement to the database shard for execution.

Note:
This topic is applicable to PolarDB-X 5.3 and later. For earlier versions, see Specify a database shard to run an SQL statement.

Syntax

The NODE HINT allows you to specify a database shard by using a shard name, to run the SQL statement in the database shard. A shard name uniquely identifies a database shard in a PolarDB-X instance. You can run the SHOW NODE statement to obtain the shard name.

Specify a database shard by using a shard name, to run an SQL statement

This custom hint allows you to specify one or more database shards to run an SQL statement.

Note:
If the hint for specifying a database shard is used in an INSERT statement that contains a sequence for the target table, the sequence will not take effect. For more information, see Limits and precautions for sequences.

- Specify one database shard to run an SQL statement:

  ```
  /*+TDDL:node('node_name')*/
  ```

  Specifically, node_name indicates the shard name. This PolarDB-X hint enables you to route the SQL statement to the database shard specified by node_name.

- Specify multiple database shards to run an SQL statement:

  ```
  /*+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 for execution. 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 for execution. Therefore, the specified shard names in the SQL statement must correspond to existing database shards.
- The NODE HINT can be used in data manipulation language (DML), data definition language (DDL), and data access language (DAL) statements.
- hints can be in the formats of /*+TDDL:hint_command*/ and /! +TDDL:hint_command*/.
- In the MySQL command-line client, if you need to run an SQL statement that contains a hint in the format of /*+TDDL:hint_command*/, add the -c parameter to the logon command, because hints are based on the MySQL Comment Syntax. Otherwise, the client deletes the hint and then sends the SQL statement to the server for execution, which causes the hint to fail to take effect. For more information, see MySQL Client Options.

Examples

The following shows the responses of the SHOW NODE statement for a logical database named drds_test in a PolarDB-X instance.

```
mysql> SHOW NODE\G
*************************** 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)
```

As you can see, 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 obtaining the shard name, you can use the PolarDB-X hint to specify the corresponding database shard to run the SQL statement.

- Specify database shard 0 to run an SQL statement:

```
SELECT /*TDDL:node('DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS')*/
 * FROM table_name;
```

- Specify multiple database shards to run an SQL statement:

```
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 in the database shards whose shard names are `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS` and `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0006_RDS`.

- View the execution plan of an SQL statement in database shard 0:

```
/*TDDL:node('DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS')*/
EXPLAIN SELECT * FROM table_name;
```

5.8.5 Scan all or some of database shards and table shards

In addition to routing an SQL statement to one or more database shards for execution, PolarDB-X provides the `SCAN HINT` to scan all or some of database shards and table shards. With the `SCAN HINT`, you can route an SQL statement to each database shard 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 of a specified logical table.

Note:

This topic is applicable to PolarDB-X 5.3 and later. For earlier versions, see Scan all database shards and table shards.

With the SCAN HINT, you can specify the following SQL execution manners:

- Run an SQL statement in all table shards in all database shards.
- Run an SQL statement in all table shards in a specified database shard.
- Run an SQL statement in the specified table shard in the specified database shard by calculating the name of the physical table based on conditions.
- Run an SQL statement in the specified table shard in the specified database shard by explicitly specifying the name of the physical table.

The `SCAN HINT` can be used in data manipulation language (DML) statements, data definition language (DDL) statements, and some data access language (DAL) statements.

Note:

- hints can be in the formats of /*+TDDL:hint_command*/ and /! +TDDL:hint_command*/.
- In the MySQL command-line client, if you need to run an SQL statement that contains a hint in the format of /*+TDDL:hint_command*/, add the -c parameter to the logon command, because hints are based on the MySQL Comment Syntax. Otherwise, the client deletes the hint and then sends the SQL statement to the server for execution, which causes the hint to fail 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 a specified database shard.
SCAN(NODE="node_list")              # Specify the database shard.
# Route an SQL statement to the specified table shard in the specified database shard by calculating the name of the physical table based on conditions.
SCAN(
  [TABLE=]"table_name_list"         # Specify the name of the logical table.
  , CONDITION="condition_string"    # Calculate the names of physical databases based on the content of TABLE and CONDITION.
  [, NODE="node_list"] )            # Filter the results obtained based on the content of CONDITION, to retain only the results of the specified physical database.
# Route an SQL statement to the specified table shard in the specified database shard by explicitly specifying the name of the physical table.
SCAN(
  [TABLE=]"table_name_list"         # Specify the name of the logical table.
  , REAL_TABLE=("table_name_list")  # Specify the name of the physical table. The same physical table names are applied to all physical databases.
  [, NODE="node_list"] )            # Filter the results obtained based on the content of CONDITION, to retain only the results of the specified physical database.
# Specify physical table names or logical table names.
table_name_list:
    table_name [, table_name]...
# Specify physical databases by using GROUP_KEY and GROUP_INDEX, which can be obtained by running the SHOW NODE statement.
node_list:
    {group_key | group_index} [, {group_key | group_index}]...
# Run an SQL WHERE statement. When using this syntax, you must specify conditions for each table, for example, t1.id = 2 and t2.id = 2.
condition_string:
    where_condition
```

Examples

- Run the following SQL statement in 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 tables corresponding to the logical table t1, and the result sets are merged and returned.

- Run the following SQL statement in all table shards in specified database shards:

```
SELECT /*+TDDL:scan(node='0,1,2')*/ COUNT(1) FROM t1
```

After this statement is executed, all physical tables corresponding to the logical table t1 in database shards 0000, 0001, and 0002 are calculated, the SQL statement is routed to the physical tables, and the result sets are merged and returned.

- Run the following SQL statement in specified table shards based on conditions:

```
SELECT /*+TDDL:scan('t1', condition='t1.id = 2')*/ COUNT(1) FROM t1
```

After this statement is executed, all physical tables that correspond to the logical table t1 and meet the conditions are calculated, the SQL statement is routed to the physical tables, and the result sets are merged and returned.

- Run the following SQL JOIN statement in 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 WHERE b.name = "test"
```

After this statement is executed, all physical tables that correspond to the logical tables t1 and t2 and meet the conditions are calculated, the SQL statement is routed to the physical tables, and the result sets are merged and returned.

Notice:

Before using this custom hint, you must ensure that the logical tables t1 and t2 are partitioned into the same number of database shards and the same number of table shards. Otherwise, the database shards calculated by the PolarDB-X instance based on the conditions are different, and an error will be returned.

- Run the following SQL statement in the specified table shards in database shards by explicitly specifying the names of the physical tables:

```
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 the table shards t1_00`` t1_01 in all database shards, and the result sets are merged and returned.

- Run the following SQL JOIN statement in the specified table shards in database shards by explicitly specifying the names of the physical tables:

```
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 the 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

- PolarDB-X supports global secondary indexes. The INDEX hint allows you to obtain query results from a specified GSI.
- The INDEX hint takes effect only for SQL SELECT statements.

Note:

This custom hint is applicable to only MySQL 5.7 and later and PolarDB-X 5.4.1 and 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})
*/
```

PolarDB-X INDEX hint can be used in two ways:

- `FORCE INDEX()`: This syntax is the same as that of MySQL FORCE INDEX.
- `INDEX()`: In this syntax, a global secondary index is specified using 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 global secondary index is not in the specified table.

Note:

- hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/`.
- In the MySQL command-line client, if you need to run an SQL statement that contains a hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because hints are based on the MySQL Comment Syntax. Otherwise, the

client deletes the hint and then sends the SQL statement to the server for execution, which causes the hint to fail 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 global secondary index `g_i_seller` by using FORCE INDEX 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 global secondary index `g_i_buyer` by using INDEX+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 affect execution plans of SQL statements by using relevant syntax, to specially optimize the SQL statements.

Overview of PolarDB-X hints

PolarDB-X provides special hint syntax.

For example, if you know the target data is stored in table shards in certain database shards and you need to route the 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 `*/`, namely, `TDDL:node in('node_name', ...)`, is a PolarDB-X hint. The hint specifies the ApsaraDB RDS for MySQL database shard where the SQL statement is to be executed.

Note:

- hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/`.
- In the MySQL command-line client, if you need to run an SQL statement that contains a hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because hints are based on the MySQL Comment Syntax. Otherwise, the client deletes the hint and then sends the SQL statement to the server for execution, which causes the hint to fail to take effect. For more information, see MySQL Client Options.

PolarDB-X hint syntax

Basic syntax:

```
/! TDDL:hint command*/
```

PolarDB-X hints are based on MySQL Comment Syntax. Therefore, an SQL statement that contains a PolarDB-X hint is located between `/!` and `*/`, and must begin with `TDDL:`. The `hint command` indicates a PolarDB-X hint command related to the specific operation. For example, a PolarDB-X hint is added to the following SQL statement to display 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

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 delay of several milliseconds. If you need to read changed data immediately after 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, to route SQL statements to a specified primary or read-only ApsaraDB RDS for MySQL instance.

Syntax

```
/! TDDL:MASTER|SLAVE*/
```

With this custom hint, you can specify whether to run an SQL statement on a primary or read-only ApsaraDB RDS for MySQL instance. With the custom hint `/!TDDL:SLAVE*/`, if a primary ApsaraDB RDS for MySQL instance is configured with 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 run the SQL statement.

Note:

- hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/`.
- In the MySQL command-line client, if you need to run an SQL statement that contains a hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because hints are based on the MySQL Comment Syntax. Otherwise, the client deletes the hint and then sends the SQL statement to the server for execution, which causes the hint to fail to take effect. For more information, see MySQL Client Options.

Examples

- Specify a primary ApsaraDB RDS for MySQL instance to run an SQL statement:

  ```
  /! TDDL:MASTER*/SELECT * FROM table_name;
  ```

  After the custom hint `/! TDDL:MASTER*/` is added at the beginning of the SQL statement, this SQL statement is routed to the primary ApsaraDB RDS for MySQL instance for execution.

- Specify a read-only ApsaraDB RDS for MySQL instance to run an SQL statement:

  ```
  /! TDDL:SLAVE*/SELECT * FROM table_name;
  ```

  After the custom hint `/! TDDL:SLAVE*/` is added at the beginning of the SQL statement, this SQL statement is randomly routed to a read-only ApsaraDB RDS for MySQL instance based on the allocated weight.

Note

- 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 for execution.
- The hint `/*+TDDL:slave()*/` allows you to route the SQL statement randomly to a read-only ApsaraDB RDS for MySQL instance based on the configured weight for execution. 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 run 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:

- hints can be in the formats of `/*+TDDL:hint_command*/` and `/! +TDDL:hint_command*/`.

- In the MySQL command-line client, if you need to run an SQL statement that contains a hint in the format of /*+TDDL:hint_command*/, add the -c parameter to the logon command, because hints are based on the MySQL Comment Syntax. Otherwise, the client deletes the hint and then sends the SQL statement to the server for execution, which causes the hint to fail 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:

- hints can be in the formats of /*+TDDL:hint_command*/ and /! +TDDL:hint_command*/.
- In the MySQL command-line client, if you need to run an SQL statement that contains a hint in the format of /*+TDDL:hint_command*/, add the -c parameter to the logon command, because hints are based on the MySQL Comment Syntax. Otherwise, the client deletes the hint and then sends the SQL statement to the server for execution, which causes the hint to fail 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 Specify a database shard to run an SQL statement

When running SQL commands in a PolarDB-X instance, you may find that some SQL statements are not supported by the PolarDB-X instance. In this case, you can use the custom hint provided by PolarDB-X to route the SQL statements to one or more database shards for execution. In addition, 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 for execution.

Syntax

This custom hint allows you to specify a database shard by using a shard name or the value of the database shard key, to run an SQL statement in the database shard. A shard name uniquely identifies a database shard in a PolarDB-X instance. You can run the SHOW NODE command to obtain the shard name.

Note:

If the hint for specifying a database shard is used in an INSERT statement that contains a sequence for the target table, the sequence will not take effect. For more information, see Limits and precautions for sequences.

- Specify a database shard by using a shard name, to run an SQL statement

  This custom hint allows you to specify one or more database shards to run an SQL statement.

  - Specify one database shard to run an SQL statement:

    ```
    /! TDDL:NODE='node_name'*/
    ```

    Specifically, node_name indicates the shard name. This PolarDB-X hint enables you to route the SQL statement to the database shard specified by node_name.

  - Specify multiple database shards to run an SQL statement:

    ```
    /! 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 for execution. Therefore, the specified shard names in the SQL statement must correspond to existing database shards.

- Specify a database shard by using the value of the database shard key, to run an SQL statement

  ```
  /! TDDL:table_name.partition_key=value [and table_name1.partition_key=value1]*/
  ```

  In this PolarDB-X hint, table_name indicates the name of a logical table, and this table is a partitioned table. In addition, partition_key indicates a shard key, and value indicates

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 the database shards and table shards where the SQL statement is to be executed, and routes the SQL statement to the corresponding database shards.

Note:

- hints can be in the formats of /*+TDDL:hint_command*/ and /! +TDDL:hint_command*/.
- In the MySQL command-line client, if you need to run an SQL statement that contains a hint in the format of /*+TDDL:hint_command*/, add the -c parameter to the logon command, because hints are based on the MySQL Comment Syntax. Otherwise, the client deletes the hint and then sends the SQL statement to the server for execution, which causes the hint to fail to take effect. For more information, see MySQL Client Options.

Examples

The following shows the responses of the SHOW NODE statement for a logical database named drds_test in a PolarDB-X instance.

```
mysql> SHOW NODE\G
*************************** 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)
```

As you can see, 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 obtaining the shard name, you can use the PolarDB-X hint to specify the corresponding database shard to run the SQL statement.

- Specify database shard 0 to run an SQL statement:

  ```
  /! TDDL:NODE='DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS'*/SELECT
   * FROM table_name;
  ```

- Specify multiple database shards to run an SQL statement:

  ```
  /! 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 in the database shards whose shard names are `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS` and `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0006_RDS`.

- View the execution plan of an SQL statement in 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 in the database shard corresponding to the shard name DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS will be returned.

- Specify a database shard by using the value of the database shard key, to run an SQL statement:

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 route the statement to a database shard for execution.

For example, the following shows the CREATE TABLE statement for creating two logical tables t1 and t2, which 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`) tbpartitions 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`) tbpartitions 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 in a PolarDB-X instance, an error will be returned indicating that this statement is not supported. In this case, you can add the PolarDB-X

hint to this SQL statement before submitting it to the PolarDB-X instance for execution. The SQL statements are as follows:

```
/! 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 of t1, with the id of the database shards being 1. You can run the following EXPLAIN command 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)
```

According to the result set of the EXPLAIN command, the SQL statement is rewritten into three statements, which are then routed to the database shards for execution. You can further specify a table shard by using the value of the table shard key, to narrow the execution scope of the SQL statement to a specified table shard.

```
 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 using this custom hint, you must ensure that the logical tables t1 and t2 are partitioned into the same number of database shards and the same number of table shards. Otherwise, the database shards calculated by the PolarDB-X instance based on the conditions are different, and an error will be 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:

- hints can be in the formats of /*+TDDL:hint_command*/ and /! +TDDL:hint_command*/.
- In the MySQL command-line client, if you need to run an SQL statement that contains a hint in the format of /*+TDDL:hint_command*/, add the -c parameter to the logon command, because hints are based on the MySQL Comment Syntax. Otherwise, the client deletes the hint and then sends the SQL statement to the server for execution, which causes the hint to fail 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

Note:

- When you use MySQL 5.7 or later and PolarDB-X 5.3.4 or later, XA distributed transactions are automatically enabled. The user experience of the XA distributed transactions is the same as that of single-database transactions in MySQL. No special commands are required to enable XA distributed transactions.
- When you use MySQL and a PolarDB-X instance in other versions, see Distributed transactions based on MySQL 5.6.

How it works

When you use MySQL 5.7 or later, the PolarDB-X instance processes distributed transactions based on the XA protocol by default.

Use method

The user experience of distributed transactions in a PolarDB-X instance is the same as that of single-database transactions in MySQL, for example, in terms of the following commands:

- `SET AUTOCOMMIT=0`: Start a transaction.
- `COMMIT`: Commit the current transaction.
- `ROLLBACK`: Roll 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 single-database transaction. If the SQL statement in the transaction is to modify the data of 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

How it works

The XA protocol for MySQL 5.6 is not mature. Therefore, the PolarDB-X instance independently implements two-phase commit (2PC) transaction policies for distributed transactions. When you use MySQL 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. When you use MySQL 5.7 or later and a PolarDB-X instance in 5.3.4 or later, see Distributed transactions based on MySQL 5.7.

Use method

If a transaction involves multiple database shards, you must declare the current transaction as a distributed transaction. If a transaction involves only a single database shard, you do not need to enable distributed transactions, but can process the transaction as a single-database transaction in MySQL. No additional operations are required.

To enable distributed transactions, do as follows:

After transactions are enabled, run `SET drds_transaction_policy = '...'`.

To enable 2PC transactions in the MySQL command-line client, run the following statements:

```
SET AUTOCOMMIT=0;
SET drds_transaction_policy = '2PC'; -- We recommend that you use MySQL 5.6 to run this
 command.
.... -- Here, you can run your business SQL statement.
COMMIT; -- You can alternatively write ROLLBACK.
```

To enable 2PC transactions by using the Java database connectivity (JDBC) API, write the code as follows:

```
conn.setAutoCommit(false);
try (Statement stmt = conn.createStatement()) {
    stmt.execute("SET drds_transaction_policy = '2PC'");
}
// ... Here, you can run 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.

Sample code:

```
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) throws SQLException {
        try (Statement stmt = con.createStatement()) {
```

```
            stmt.executeUpdate("SET drds_transaction_policy = '2PC'"); // A 2PC transaction is
used as an example.
        }
    }
}
```

After that, instantiate the preceding class in the Spring configuration. For example, you can write the code as follows:

```
<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-partition table

This topic describes how to create a single-database non-partition table.

Create a single-database non-partition table

```
CREATE TABLE single_tbl(
  id int,
  name varchar(30),
  primary key(id)
```

```
);
```

According to the node topology of the logical table, you can see that a single-database non-partition logical table is created in database 0.

```
mysql> show topology from single_tbl;
+------+-------------------------------------------------------------------+------------+
| ID   | GROUP_NAME                                                        | TABLE_NAME |
+------+-------------------------------------------------------------------+------------+
|    0 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000_RDS | single_tbl |
+------+-------------------------------------------------------------------+------------+
1 row in set (0.01 sec)
```

Specify parameters

You can also specify the select_statement parameter when creating a a single-database non-partition table. If you need to create table shards, you cannot specify this parameter.

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)]
    [table_options]
    [partition_options]
    select_statement
```

For example, you can run the following statement to create a single-database non-partition table single_tbl2 to store the data from the single_tbl table. In this case, no sharding is required.

```
CREATE TABLE single_tbl2(
  id int,
  name varchar(30),
  primary key(id)
) select * from single_tbl;
```

5.11.2.3 Create a non-partition table in database shards

This topic describes how to create a non-partition table in database shards.

Assume that eight database shards have been created. You can run the following command to create a non-partition table in the database shards by calculating the hash function based on the userId shard key.

```
CREATE TABLE multi_db_single_tbl(
  id int,
  name varchar(30),
  primary key(id)
```

```
) dbpartition by hash(id);
```

According to the node topology of the logical table, you can see that a table shard is created in each database shard. In other words, the table is only distributed to database shards.

```
mysql> show topology from multi_db_single_tbl;
+------+------------------------------------------------------------------
+--------------------+
| 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_tbl_10 |
|  11 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0003_RDS |
multi_db_multi_tbl_11 |
|  12 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0004_RDS |
multi_db_multi_tbl_12 |
|  13 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0004_RDS |
multi_db_multi_tbl_13 |
|  14 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0004_RDS |
multi_db_multi_tbl_14 |
|  15 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005_RDS |
multi_db_multi_tbl_15 |
|  16 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005_RDS |
multi_db_multi_tbl_16 |
|  17 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005_RDS |
multi_db_multi_tbl_17 |
|  18 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0006_RDS |
multi_db_multi_tbl_18 |
|  19 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0006_RDS |
multi_db_multi_tbl_19 |
```

```
|  20 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0006_RDS |
multi_db_multi_tbl_20 |
|  21 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS |
multi_db_multi_tbl_21 |
|  22 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS |
multi_db_multi_tbl_22 |
|  23 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS |
multi_db_multi_tbl_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:

  1. The order table of each service needs to be partitioned into database shards by buyer ID and order ID.
  2. 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) tbpartition 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 informatio n, 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 calculatin

g 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 |
```

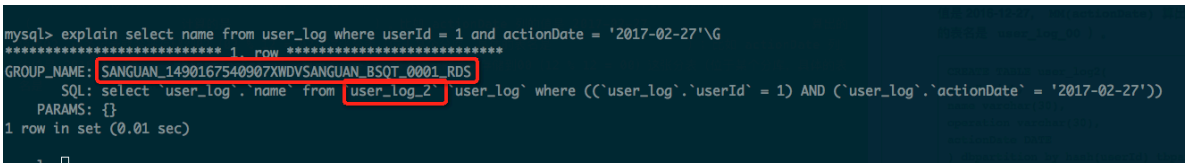
```
|  54 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS
 | user_log_5 |
|  55 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007_RDS
 | user_log_6 |
+------+------------------------------------------------------------------+------------+
56 rows in set (0.01 sec)
```

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_PARTITI
ON_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.

Figure 5-8: View the route of the SQL statement



- 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_PARTITI
ON_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_PARTITI
ON_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_PARTITI
ON_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_PARTITI
ON_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 for the sharding algorithm, the system uses the primary key as the shard key by default. The following illustrates 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

The BROADCAST clause is used to specify a broadcast table to be created. A broadcast table is replicated to each database shard and data consistency is ensured between the database shards by using a synchronization mechanism with a delay of several seconds. This feature allows you to route a JOIN operation from a Cloud Native Distributed Database

PolarDB-X (PolarDB-X) instance to an underlying ApsaraDB RDS for MySQL instance to prevent the JOIN operation from being performed in multiple databases. Overview describes how to optimize SQL statements by using broadcast tables.

The following is an example statement for 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

When creating table shards in database shards, you can also specify other attributes of the table shards in the MySQL CREATE TABLE statement. For example, you can specify other attributes as follows:

```
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 ALTER TABLE statement

The syntax of the ALTER TABLE statement used to modify a table is as follows:

```
ALTER [ONLINE|OFFLINE] [IGNORE] TABLE tbl_name
    [alter_specification [, alter_specification] ...]
    [partition_options]
```

In a PolarDB-X instance, you can use this data definition language (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 table shard, 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 DROP TABLE statement

The syntax of the DROP TABLE statement used to delete a table is as follows:

```
DROP [TEMPORARY] TABLE [IF EXISTS]
    tbl_name [, tbl_name] ...
    [RESTRICT | CASCADE]
```

The DROP TABLE statement in a PolarDB-X instance is the same as the DROP TABLE statement in a MySQL database. After the statement is executed, the system automatically deletes the corresponding physical table. For more information about the syntax, see MySQL DROP TABLE Statement.

For example, you can run the following statement to delete the user_log table:

```
DROP TABLE user_log;
```

5.11.5 FAQ about DDL statements

What can I do if an error occurs during table creation?

Data definition language (DDL) statements in a PolarDB-X instance are processed in a distributed manner. If an error occurs, the structures of all table shards are inconsistent from each other. Therefore, you need to perform manual cleanup.

Perform the following steps:

1. Check the basic error descriptions provided by the PolarDB-X instance, such as syntax errors. If the error message is too long, the system will prompt you to call the SHOW WARNINGS command to view the failure cause of each database shard.

2. Run the SHOW TOPOLOGY command to view the topology of physical tables.

```
SHOW TOPOLOGY FROM multi_db_multi_tbl;
+------+----------------+----------------------+
| 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 tablename command to check whether the logical table has been created. For example, the following response indicates that a physical table corresponding to the logical table multi_db_multi_tbl failed to be created.

```
mysql> check table multi_db_multi_tbl;
+-----------------------------------------------+-------+----------
+-------------------------------------------------------------------------------+
| TABLE                                | OP    | MSG_TYPE | MSG_TEXT
          |
+-----------------------------------------------+-------+----------
+-------------------------------------------------------------------------------+
| andor_mysql_qatest. multi_db_multi_tbl | check | Error    | Table 'corona_qatest_0.
multi_db_multi_tbl_02' doesn't exist |
+-----------------------------------------------+-------+----------
+-------------------------------------------------------------------------------+
1 row in set (0.16 sec)
```

4. Continue to create or delete the table in idempotent mode to create or delete the remaining physical tables.

```
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 in PolarDB-X data definition language (DDL) sharding functions.

Sharding function	Description	Support for database sharding	Support for table sharding
HASH	Performs a simple modulus operation.	Yes	Yes
UNI_HASH	Performs a simple modulus operation.	Yes	Yes
RIGHT_SHIFT	Shifts the value to the right.	Yes	Yes
RANGE_HASH	Performs double hashing .	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
YYYYDD	Performs hashing by year and date.	Yes	Yes
YYYYMM_OPT	Performs optimized hashing by year and month.	Yes	Yes

Sharding function	Description	Support for database sharding	Support for table sharding
YYYYWEEK_OPT	Performs optimized hashing by year and week.	Yes	Yes
YYYYDD_OPT	Performs optimized hashing by year and date .	Yes	Yes

Note:

When using database sharding and table sharding in PolarDB-X, note the following:

- In a PolarDB-X instance, the sharding method of a logical table is defined jointly by a sharding function and a shard key. The sharding function contains the number of shards to be created and the routing algorithm. The shard key also specifies the MySQL data type of the shard key.
- When the database sharding function is the same as the table sharding function and the database shard key is the same as the table shard key in a PolarDB-X instance, the same sharding method is used for database sharding and table sharding. This allows the PolarDB-X instance to uniquely locate one physical table in a physical database based on the value of the shard key.
- If the database sharding method and the table sharding method of a logical table are different and an SQL query does not contain both database shard key and table shard key, the PolarDB-X instance scans all database shards or all table shards when processing the SQL query.

Support for data types of PolarDB-X DDL sharding functions

Different PolarDB-X DDL sharding functions support different data types. The following table lists the support for various data types in PolarDB-X sharding functions (√ indicates supported and × indicates not supported).

Figure 5-9: 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 CREATE TABLE statement in MySQL, and additionally provides the drds_partition_options keyword to support database sharding and table sharding:

```
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

Requirements

- The shard key must be an integer or a string.
- This sharding function has no requirements on the version of a PolarDB-X instance. It supports all PolarDB-X instances by default.

Routing method

When the HASH function is run by using different shard keys for database sharding and table sharding, perform the remainder operation on the value of the database shard key based on the number of database shards. If the value of the shard key is a string, the string is converted to a hash value before route calculation. For example, HASH('8') is equivalent to 8 % D, where D indicates the number of database shards.

When the HASH function is run by using the same shard key for both database sharding and table sharding, perform the remainder operation on the value of the shard key based on the total number of table shards. For example, assume that two database shards are created, each database shard contains 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. If a key value is 15, the key value 15 is distributed to table shard 7 in database shard 1, because 15 % (2 x 4) = 7.

Scenarios

- HASH is applicable when database sharding is implemented by user ID or order ID.
- HASH is also applicable when the shard key is a string.

Examples

If you need to create a non-partition table in database shards by using the HASH function based on the ID column, you can use 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 HASH(ID);
```

Precautions

The HASH is a simple modulus operation. The output distribution of the HASH function can be even only when the values in the partition column are evenly distributed.

5.11.6.3 UNI_HASH

Requirements

- The shard key must be an integer or a string.
- The version of the PolarDB-X instance must be 5.1.28-1508068 or later. For more information about the PolarDB-X release notes, see View the instance version.

Routing method

When the UNI_HASH function is used for database sharding, perform a remainder operation on the value of the database shard key based on the number of database shards. If the value of the shard key is a string, the string is converted to a hash value before route calculation. For example, HASH('8') is equivalent to 8 % D, where D indicates the number of database shards.

When the UNI_HASH function is run by using the same shard key for both database sharding and table sharding, perform the remainder operation on the value of the database shard key based on the number of database shards first (this step is different from that in the HASH function). Then, the data is evenly distributed to the table shards in the database shard.

Scenarios

- UNI_HASH is applicable when database sharding is implemented by user ID or order ID.
- UNI_HASH is also applicable when the shard key is an integer or a string.
- UNI_HASH can be used when the following conditions are met: Two logical tables need to be partitioned into different numbers of table shards in database shards based on the same shard key. In addition, the two tables are frequently joined by using a JOIN statement based on the shard key.

Comparison with HASH

When you use the UNI_HASH function to create a non-partition table in database shards, the routing method is the same as that used in the HASH function. Specifically, the route is

calculated by performing the remainder operation on the key value of the database shard key based on the number of database shards.

When the UNI_HASH function is run by using the same shard key for both database sharding and table sharding, as the number of table shards changes, the database shard route calculated based on the same key value may also change.

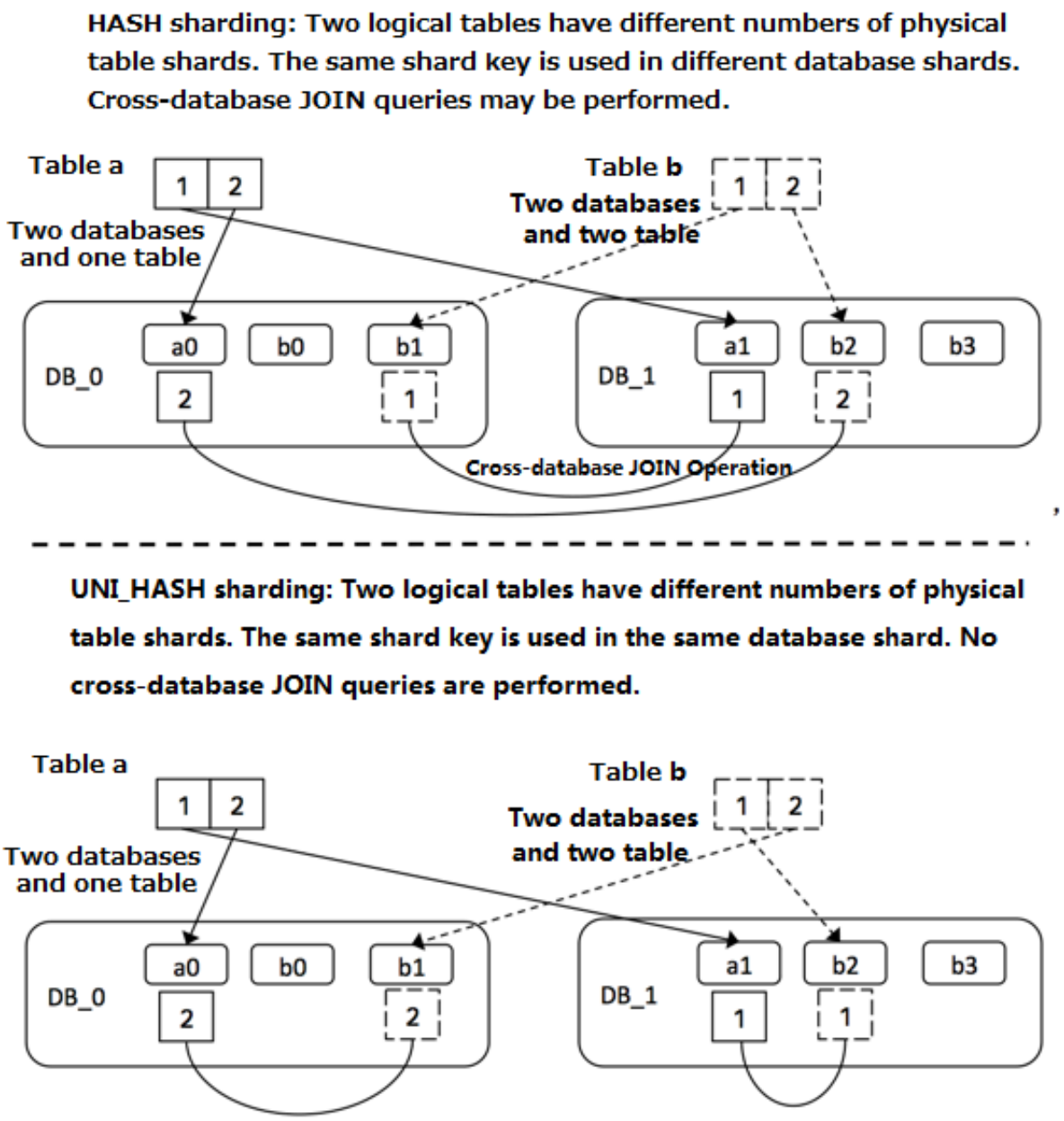
When the UNI_HASH function is run by using the same shard key for both database sharding and table sharding, the database shard route calculated based on the same key value is always the same regardless of the number of table shards.

If two logical tables need to be partitioned into different table shards in database shards based on the same shard key, when the two tables are joined by using the HASH function based on the shard key, multi-database join may occur. However, when the two tables are joined by using the UNI_HASH function based on the shard key, multi-database join does not occur.

Assume that you have two database shards and two logical tables, and each database shard in logical table a stores one table shard and each database shard in logical table b stores two table shards. The following figures separately show the results of a JOIN query

for logical tables a and b after the HASH function is used for sharding and the results of a JOIN query for logical tables a and b after the HASH function is used for sharding.

Figure 5-10: Comparison between HASH and UNI_HASH



Examples

If you need to create four table shards in each database shard by using the UNI_HASH function based on the ID column, you can run 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;
```

Precautions

The UNI_HASH is a simple modulus operation. The output distribution of the UNI_HASH function can be even only when the values in the shard column are evenly distributed.

5.11.6.4 RIGHT_SHIFT

Requirements

- The shard key must be an integer.
- The version of the PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Routing method

Shift the value of the database shard key to the right by a specified number of binary digits, and then perform the remainder operation on the obtained integer based on the number of database shards or table shards. In particular, you can specify the number of shifted digits by running a data definition language (DDL) statement.

Scenarios

RIGHT_SHIFT is applicable to improve the evenness of the hash results when the lower-digit parts of most shard key values are very similar to each other 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. Directly hashing the values of the shard keys outputs poor results. However, if you run the RIGHT_SHIFT(shardKey, 4) statement to shift the values of the shard keys to the right by four digits, to obtain 1234, 1235, 1246, and 1233, the hashing results are improved.

Examples

If 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 to obtain hash values, you can run 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;
```

Precautions

The number of shifted digits cannot exceed the number of digits occupied by the integer.

5.11.6.5 RANGE_HASH

Requirements

- The shard key must be a character or a number.
- The version of the 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 any shard key and then perform the remainder operation on the hash value based on the number of database shards. This completes the route computing. 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 a table needs to be partitioned by two shard keys but query is performed only based on the value of one shard key.

Examples

Assume that a PolarDB-X database is partitioned into eight physical databases. Our customer has the following requirements:

The order table of a business needs to be partitioned into database shards by buyer ID and order ID. 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,
    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;
```

Precautions

- 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.

5.11.6.6 MM

Requirements

- The shard key must be of the DATE, DATETIME, or TIMESTAMP type.
- MM is only applicable to table sharding.
- The version of the PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Routing method

Perform the remainder operation based on the month that corresponds to the time value of the database shard key to obtain the table shard subscript.

Scenarios

MM can be used to partition tables by month. The table shard name indicates a specific month.

Examples

Assume that we need to perform database sharding by ID, perform table sharding for the create_time column by month, and map every month to a physical table. The data definition language (DDL) statement is as follows:

```
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;
```

Precautions

When you partition tables with MM, ensure that each database shard has no more than 12 table shards because a year has 12 months.

5.11.6.7 DD

Requirements

- The shard key must be of the DATE, DATETIME, or TIMESTAMP type.
- DD is only applicable to table sharding.
- The version of the PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Routing method

Perform the remainder operation based on the day of the month that corresponds to the time value of the database shard key to obtain the table shard subscript.

Scenarios

DD can be used to partition tables based on a specified number of days in a month, that is, a date. The subscript of the table shard name indicates the day in a month. A month has 31 days at most.

Examples

Assume that we need to perform database sharding by ID, perform table sharding for the create_time column by day, and map every day to a physical table. The data definition language (DDL) statement is as follows:

```
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;
```

Precautions

When you partition tables with DD, ensure that each database shard has no more than 31 table shards because a month has 31 days at most.

5.11.6.8 WEEK

Requirements

- The shard key must be of the DATE, DATETIME, or TIMESTAMP type.
- WEEK is only applicable to table sharding.

- The version of the PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Routing method

Perform the remainder operation based on the day of a week that corresponds to the time value of the database shard key to obtain the table shard subscript.

Scenarios

WEEK can be used to partition tables based on days in a week. The subscript of the table shard name corresponds to each day of a week, from Monday to Sunday.

Examples

Assume that we need to perform database sharding by ID, perform table sharding for the create_time column by week, and map every day of a week (from Monday to Sunday) to a physical table. The data definition language (DDL) statement is as follows:

```
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;
```

Precautions

When you partition tables with WEEK, ensure that each database shard has no more than seven table shards because a week has seven days.

5.11.6.9 MMDD

Requirements

- The shard key must be of the DATE, DATETIME, or TIMESTAMP type.
- MMDD is only applicable to table sharding.
- The version of the PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Routing method

Perform the remainder operation based on the number of days in a year that corresponds to the time value of the database shard key to obtain the table sharding subscript.

Scenarios

MMDD can be used to partition tables based on the number of days in a year that corresponds to a date in that year. The subscript of the table shard name indicates the day in that year, with a maximum of 366 days in a year.

Examples

Assume that we need to perform database sharding by ID, create tables for the create_time column by date (month-day), and map every day of a year to a physical table. The data definition language (DDL) statement is as follows.

```
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;
```

Precautions

When you partition tables with MMDD, ensure that each database shard has no more than 366 table shards because a year has 366 days at most.

5.11.6.10 YYYYMM

Requirements

- The shard key must be of the DATE, DATETIME, or TIMESTAMP type.
- The version of the 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 months of the year in the time value of the database shard key and then perform the remainder operation on the hash value based on the number of database shards. This completes route computing.

For example, YYYYMM('2012-12-31 12:12:12') is equivalent to (2012 x 12 + 12) % D, where D indicates the number of database shards.

Scenarios

YYYYMM can be used to partition databases by year and month. We recommend that you use YYYYMM with tbpartition YYYYMM(ShardKey).

Assume that a PolarDB-X database is partitioned into eight physical databases. Our customer has the following requirements:

- Perform database sharding for a service by year and month.
- Distribute data from every month within two years to a separate table shard.
- Distribute a query with the database and table shard keys to a physical table shard of a physical database shard.

The preceding requirements can be met by using YYYYMM. For the requirement of distributing data from every month within two years to a table shard (that is, one table shard stores the data of one month), create at least 24 physical table shards because a year has 12 months. Create three physical table shards for each database shard because the PolarDB-X instance contains eight database shards. The data definition language (DDL) statement is as follows.

```
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;
```

Precautions

- YYYYMM does not support distributing 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 (for example, a cycle exists between 2012-03 and 2013-03), data from the same month may be routed to the same database or table shard, depending on the actual number of table shards.

5.11.6.11 YYYYWEEK

Requirements

- The shard key must be of the DATE, DATETIME, or TIMESTAMP type.
- The version of the 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 weeks of the year in the time value of the database shard key and then perform the remainder operation on the hash value based on the number of database shards. This completes route computing.

For example, YYYYWEEK('2012-12-31 12:12:12') is equivalent to (2013 x 52 + 1) % D, with the date 2012-12-31 falling on the first week of 2013, where D indicates the number of database shards.

Scenarios

YYYYWEEK can be used to partition databases by year and the number of weeks in a year. We recommend that you use YYYYWEEK with tbpartition YYYYWEEK(ShardKey).

Assume that a PolarDB-X database is partitioned into eight physical databases. Our customer has the following requirements:

- Perform database sharding for a service by year and by week.
- Distribute data from every week within two years to a separate table shard.
- Distribute a query with the database and table shard keys to a physical table shard of a physical database shard.

The preceding requirements can be met by using YYYYWEEK. For the requirement of distributing data from every week within two years to a table shard (that is, one table shard stores the data of one week), create at least 106 physical table shards because a year has roughly 53 weeks (rounded). Create 14 physical table shards for each database shard because the PolarDB-X instance contains eight database shards (14 x 8 = 112 > 106). We recommend that the number of table shards be an integer multiple of the number of database shards. The data definition language (DDL) statement is as follows:

```
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;
```

Precautions

- YYYYWEEK does not support distributing 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 (for example, a cycle exists between the first week of 2012 and the first week of 2013), data from the same week after a cycle may be routed to the same database shard or table shard, depending on the actual number of table shards.

5.11.6.12 YYYYDD

Requirements

- The shard key must be of the DATE, DATETIME, or TIMESTAMP type.
- The version of the 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 days of the year in the time value of the database shard key and then perform the remainder operation on the hash value based on the number of database shards. This completes route computing.

For example, YYYYDD('2012-12-31 12:12:12') is equivalent to (2012 x 366 + 365) % D, with 2012-12-31 as the 365th day of 2012, where D indicates the number of database shards.

Scenarios

Database sharding is performed by year and the number of days in a year. We recommend that you use YYYYDD with tbpartition YYYYDD(ShardKey).

Assume that a PolarDB-X database is partitioned into eight physical databases. Our customer has the following requirements:

- Perform database sharding for a service by year and day.
- Distribute data from every week within two years to a separate table shard.
- Distribute a query with the database and table shard keys to a physical table shard of a physical database shard.

The preceding requirements can be met by using YYYYDD. For the requirement of distributing data from every day within two years to a table shard (that is, one table shard stores the data of one day), create at least 732 physical table shards because a year has up to 366 days. Create 92 physical table shards for each database shard because the PolarDB-X instance contains eight database shards (732/8 = 91.5, rounded to 92). We recommend that the number of table shards be an integer multiple of the number of database shards. The data definition language (DDL) statement is as follows:

```
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;
```

Precautions

- YYYYDD does not support distributing data from every day in every year to a separate table shard. Instead, the number of table shards must be fixed.
- After a cycle of a specific date (for example, a cycle exists between 2012-03-01 and 2013-03-01), data from the same date may be routed to the same database shard or table shard, depending on the actual number of table shards.

5.11.6.13 YYYYMM_OPT

Requirements

- The shard key must be of the DATE, DATETIME, or TIMESTAMP type.
- The year and month of user data increase naturally over time, rather than randomly.
- The version of the PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

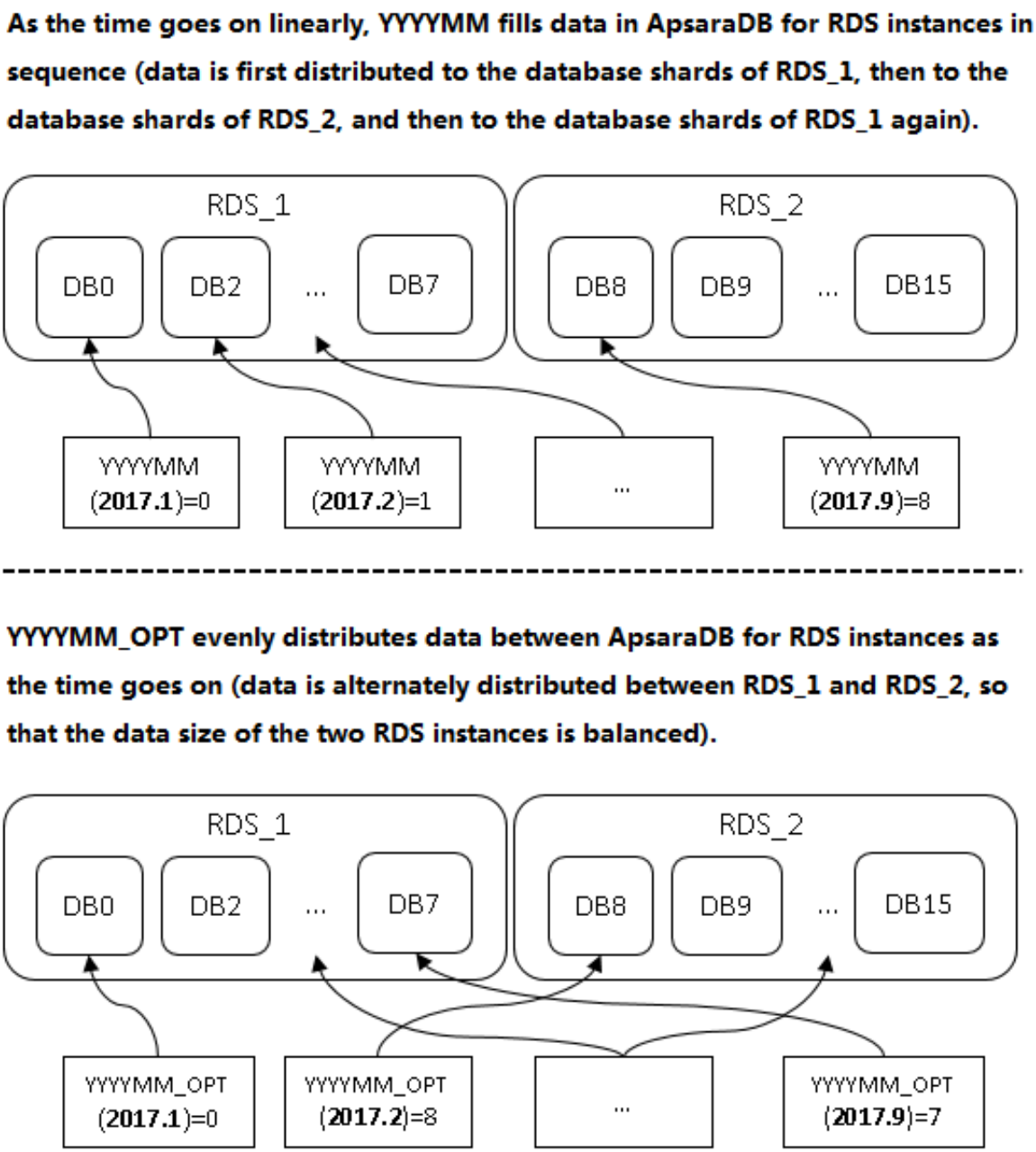
Optimizations

Compared with YYYYMM, YYYYMM_OPT maintains the even distribution of data among ApsaraDB RDS for MySQL instances as the timeline increases.

For example, assume that two ApsaraDB RDS for MySQL instances are attached to a PolarDB-X instance, with 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 figure shows the mappings when YYYYMM and YYYYMM_OPT are used for database sharding, respectively.

Figure 5-11: Comparison between YYYYMM and YYYYMM_OPT



- YYYYMM_OPT distributes data evenly to each ApsaraDB RDS for MySQL instance, helping to maximize the performance of each ApsaraDB RDS for MySQL instance.

- How to choose between YYYYMM and YYYYMM_OPT:
  - YYYYMM_OPT can be used to distribute data evenly to each ApsaraDB RDS for MySQL instance if the time of service data generation increases sequentially and the data volume does not differ much between the time points.
  - YYYYMM is applicable if the time of data generation increases randomly rather than sequentially.

Routing method

- Calculate the hash value based on the year and months of the year in the time value of the database shard key and then perform the remainder operation on the hash value based on the number of database shards. This completes route computing.
- The hash calculation based on the database and table shard key considers the data distribution among the ApsaraDB RDS for MySQL instances that connect to the PolarDB-X instances.

Scenarios

- Databases and tables need to be partitioned by year and month, respectively.
- Data must be evenly distributed to each ApsaraDB RDS for MySQL instance that connects to the PolarDB-X instance.
- The time of the shard key increases sequentially rather than randomly, and the data volume is relatively average from month to month. For example, the number of monthly journal logs increases every month, and the log data is not concentrated on the same ApsaraDB RDS for MySQL instance.

Precautions

- YYYYMM_OPT does not support distributing 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 (for example, a cycle exists between 2012-03 and 2013-03), data from the same month may be routed to the same database or table shard, depending on the actual number of table shards.

5.11.6.14 YYYYWEEK_OPT

Requirements

- The shard key must be of the DATE, DATETIME, or TIMESTAMP type.
- The version of the PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Optimizations

- Compared with YYYYWEEK, YYYYWEEK_OPT maintains the even distribution of data among ApsaraDB RDS for MySQL instances as the timeline increases. The effect is similar to YYYYMM_OPT.
- YYYYWEEK_OPT distributes data evenly to each ApsaraDB RDS for MySQL instance, helping to maximize the performance of each ApsaraDB RDS for MySQL instance.
- How to choose between YYYYWEEK and YYYYWEEK_OPT:
  - YYYYWEEK_OPT can be used to distribute data evenly to each ApsaraDB RDS for MySQL instance if the time of service data increases sequentially and the data volume does not differ much between time points.
  - YYYYWEEK is applicable if the time of data generation increases randomly rather than sequentially.

Routing method

- Calculate the hash value based on the year and weeks of the year in the time value of the database shard key and then perform the remainder operation on the hash value based on the number of database shards. This completes route computing.
- The hash calculation based on the database and table shard key considers the data distribution among the ApsaraDB RDS for MySQL instances that connect to the PolarDB-X instances.

Scenarios

- Databases and tables are partitioned by year and week, respectively.
- Data must be evenly distributed to each ApsaraDB RDS for MySQL instance that connects to the PolarDB-X instance.
- The time of the shard key increases sequentially rather than randomly, and the data volume is relatively average from week to week. For example, the number of weekly journal logs increases every week, and the log data is not concentrated on the same ApsaraDB RDS for MySQL instance.

Precautions

- YYYYWEEK_OPT does not support distributing 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 (for example, a cycle exists between the first week of 2012 and the first week of 2013), data from the same week after a cycle may be routed to the same database shard or table shard, depending on the actual number of table shards.

5.11.6.15 YYYYDD_OPT

Requirements

- The shard key must be of the DATE, DATETIME, or TIMESTAMP type.
- The version of the PolarDB-X instance must be 5.1.28-1320920 or later. For more information, see View the instance version.

Optimizations

- Compared with YYYYDD, YYYYDD_OPT maintains the even distribution of data among ApsaraDB RDS for MySQL instances as the timeline increases. The effect is similar to YYYYMM_OPT.
- YYYYDD_OPT distributes data evenly to each ApsaraDB RDS for MySQL instance, helping to maximize the performance of each ApsaraDB RDS for MySQL instance.
- How to choose between YYYYDD and YYYYDD_OPT:
  - YYYYDD_OPT can be used to distribute data evenly to each ApsaraDB RDS for MySQL instance if the time of service data generation increases sequentially and the data volume does not differ much between time points.
  - YYYYDD is applicable if the time of data generation increases randomly rather than sequentially.

Routing method

- Calculate the hash value based on the year and days of the year in the time value of the database shard key and then perform the remainder operation on the hash value based on the number of database shards. This completes route computing.
- The hash calculation based on the database and table shard key considers the data distribution among the ApsaraDB RDS for MySQL instances that connect to the PolarDB-X instances.

Scenarios

- Databases and tables need to be partitioned by year and by day, respectively.
- Data must be evenly distributed to each ApsaraDB RDS for MySQL instance that connects to the PolarDB-X instance.

- The time of the shard key increases sequentially rather than randomly, and the data volume is relatively average from day to day. For example, the number of daily journal logs increases every day, and the log data is not concentrated on the same ApsaraDB RDS for MySQL instance.

Precautions

- YYYYDD_OPT does not support distributing data from every day in every year to a separate table shard. Instead, the number of table shards must be fixed.
- After a cycle of a specific date (for example, a cycle exists between 2012-03-01 and 2013-03-01), data from the same date may be routed to the same database shard or table shard, depending on the actual number of table shards.

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 NOCYCLE;
```

```
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_SEQ.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

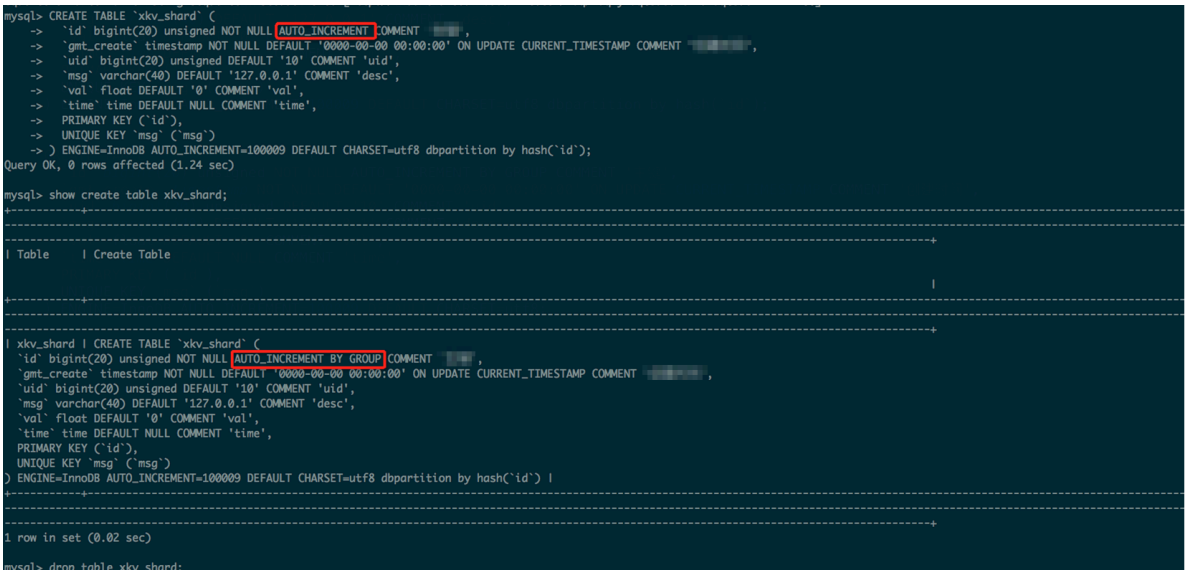
The sequence type is displayed for the auto-increment column of a table shard or broadcast table.

```
SHOW CREATE TABLE <name>
```

Examples

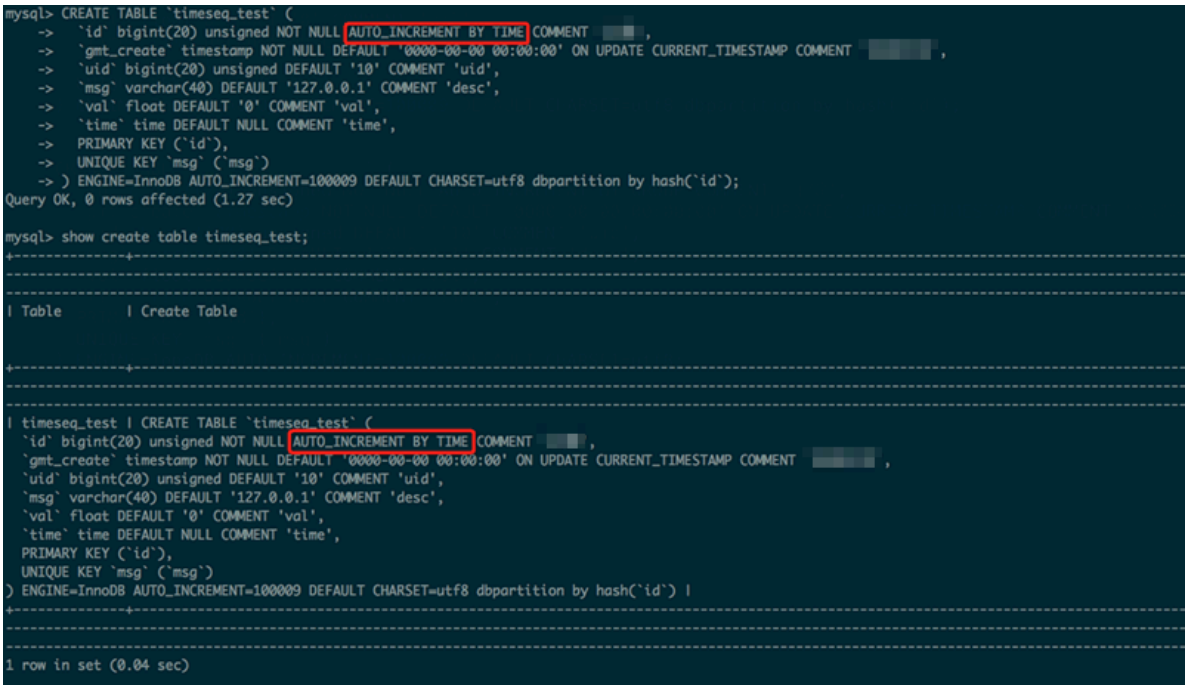
- If **AUTO_INCREMENT** is set but the sequence type is not specified when a table is created, a group sequence is used by default.

  Figure 5-12: Example 1



- When creating a table, set **AUTO_INCREMENT** and specify a time-based sequence as the primary key value.

  Figure 5-13: 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 for sequences

This topic describes the limits and precautions for sequences.

Limits and precautions

- When a time-based sequence is used in the auto-increment column of a table, the column must be of the BIGINT type.
- START WITH must be set when the sequence is changed to another type.
- When the INSERT statement is executed on a PolarDB-X database in non-partition mode where only one ApsaraDB RDS for MySQL database is bound or on a database in partition mode that has only one table but no broadcast table, PolarDB-X automatically optimizes and sends the statement, and bypasses the part of the optimizer that allocates the sequence value. In this case, INSERT INTO ... VALUES (seq.nextval, ...) is not supported. We recommend that you use the ApsaraDB RDS for MySQL auto-increment column feature instead.
- If the hint for a specific database shard is used by the INSERT statement such as INSERT INTO ... VALUES ... or INSERT INTO ... SELECT... and the target table uses a sequence, PolarDB-X bypasses the optimizer and directly sends the statement so that the sequence does not take effect. The target table creates an ID by using the auto-increment feature of the ApsaraDB RDS for MySQL table.
- The auto-increment ID allocation method for the same table must be kept consistent, which may be based on PolarDB-X sequences or the auto-increment column of the ApsaraDB RDS for MySQL. If both of the two allocation methods are used for the same table, duplicate IDs may be created and making location difficult.

Troubleshoot primary key conflicts

Assume that data is directly written to ApsaraDB RDS for MySQL and that the related primary key value is not the sequence value created by PolarDB-X. If PolarDB-X automatically creates a primary key and writes it to the database, this primary key may conflict with that of the directly written data. This problem can be resolved as follows:

1. View the existing sequences by using the PolarDB-X-specified SQL statement. The sequence prefixed with `AUTO_SEQ_` is an implicit sequence. This sequence is generated when a table is created with the `AUTO_INCREMENT` parameter.

```
mysql> SHOW SEQUENCES;
+------------------------+-------+-------------+------------+-----------+-------+-------+
| 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 contains conflicts and its primary key is ID, then retrieve the maximum primary key value of this table from PolarDB-X:

```
mysql> SELECT MAX(id) FROM t_item;
+-----------+
| 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 auto-increment primary key created by the subsequent INSERT statement.

```
mysql> ALTER SEQUENCE AUTO_SEQ_USERS START WITH 9000;
Query OK, 1 row affected (0.01 sec)
```

5.14 Best practices

5.14.1 Select a shard key

A shard key is a field for database sharding and table sharding, which is used to create sharding rules during horizontal partitioning. PolarDB-X partitions a logical table

horizontally into the physical database shards on each ApsaraDB RDS for MySQL instance based on the shard key.

The primary principle of sharding is to identify the business logic-specific subject of data in a table as much as possible and confirm that most (or core) database operations are performed based on this subject. Then, use the subject-related field as the shard key to perform database sharding and table sharding.

The business logic-specific subject is related to business scenarios. The following typical scenarios include business logic-specific subjects that can be used as shard keys:

- User-oriented Internet applications are operated to meet user requirements. Users are the business logic-specific subject and the user-related field can be used as the shard key.
- Seller-oriented e-commerce applications are operated to meet seller requirements. Sellers are the business logic-specific subject and the seller-related field can be used as the shard key.
- Game-oriented applications are operated to meet gamer requirements. Gamers are the business logic-specific subject and the gamer-related field can be used as the shard key.
- Internet of Vehicles (IoV) applications are operated based on vehicle information. Vehicles are the business logic-specific subject and the vehicle-related field can be used as the shard key.
- Tax-oriented applications are operated based on taxpayer information to provide frontend services. Taxpayers are the business logic-specific subject and the taxpayer-related field can be used as the shard key.

In other scenarios, you can also use the appropriate subject of business logic as the shard key.

For example, in a seller-oriented e-commerce application, the following single table must be horizontally partitioned:

```
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)
```

```
)
```

The sellerId field is used as the shard key because seller is the business logic-specific subject. In the case of database sharding but no table sharding, the distributed data definition language (DDL) statement for table creation is as follows:

```
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 business logic-specific subject can be used as the shard key, use the following methods to select an appropriate shard key:

- Determine the shard key based on the distribution and access of data. Distribute the data in a table to different physical database shards and table shards as evenly as possible. This method is applicable to scenarios with massive analytical queries, in which query concurrency stays at 1.
- Determine the shard key for database sharding and table sharding by combining fields of the numeric (string) type and time type. This method is applicable to log retrieval.

For example, a log system records all user operations and needs 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 identifier with the time field to create a shard key for partitioning the table by week. The distributed DDL statement for table creation is as follows:

```
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 shard key selection and table shard forms, see DDL statements.

Notice:

Avoid using hotspot data as the shard key 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/(Number 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

single-instance relational databases such as MySQL. PolarDB-X focuses on the network I/O overheads in a distributed environment and pushes SQL operations down to the underlying database shards (such as ApsaraDB RDS for MySQL) for execution, thereby reducing the network I/O overheads and improving the SQL execution efficiency.

PolarDB-X provides commands for obtaining the SQL execution information to help SQL optimization, for example, EXPLAIN commands for obtaining SQL execution plans and TRACE commands for obtaining SQL execution processes and overheads. This topic describes the basic concepts and common commands related to SQL optimization in PolarDB-X.

Execution plan

An SQL execution plan (or execution plan) is a set of ordered operation 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. Through execution plan analysis, you can know whether PolarDB-X or ApsaraDB RDS for MySQL has generated optimal execution plans for SQL statements and whether further optimization can be made.

During SQL statement execution, based on the basic information of the SQL statement and related tables, the PolarDB-X optimizer determines on which database shards the SQL statement should be executed, and the specific SQL statement form, execution policy, and data merging and computing policy for each database shard. 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 commands to display execution plans at different levels or with different levels of detail.

The following table briefly describes the EXPLAIN commands in PolarDB-X.

Table 5-3: EXPLAIN command description

Command	Description	Example
EXPLAIN { SQL }	Displays the summary execution plan of SQL statements at the PolarDB-X layer, including the database shards on which the SQL statement is run, physical statements, and general parameters.	EXPLAIN SELECT * FROM test
EXPLAIN DETAIL { SQL }	Displays the detailed execution plans of SQL statements at the PolarDB -X layer, including the statement type, concurrency, returned field information, physical tables, and database groups.	EXPLAIN DETAIL SELECT * FROM test
EXPLAIN EXECUTE { SQL }	Displays the execution plan of the underlying ApsaraDB RDS for MySQL instance, which is equivalent to the EXPLAIN statement of MySQL .	EXPLAIN EXECUTE SELECT * FROM test

Execution plan at the PolarDB-X layer

The following table describes the fields in the results returned for the execution plan at the PolarDB-X layer.

Table 5-4: Description of fields in execution plans at the PolarDB-X layer

Field	Description
GROUP_NAME	The name of the PolarDB-X database shard. The suffix identifies the specific database shard. The value is consistent with the result of the SHOW NODE command.
SQL	The SQL statement run 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:

   - Aggregate function involving multiple database shards.
   - Distributed JOIN queries involving multiple shards.
   - Complex subqueries.

   Examples:

```
mysql> EXPLAIN SELECT * FROM test;
+-----------------------------------------------+-----------------------------------------
+--------+
| 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)
```

   The group names displayed in the GROUP_NAME field can be found in the returned result of SHOW NODE:

```
mysql> SHOW NODE;
+----+-----------------------------------------------+------------------+-----------------
+--------------------+--------------------+
| 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 by PolarDB-X in custom format.

   Examples:

```
mysql> EXPLAIN DETAIL SELECT COUNT(*) FROM test;
+----------------------------------------------+--------------------------------
+--------+
| 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 in the SQL statement field indicates the database shard on which the SQL statement is run. PolarDB-X finally merges the results returned by the database shards.

Execution plans at the ApsaraDB RDS for MySQL layer

The execution plans at the ApsaraDB RDS for MySQL layer are the same as the native MySQL execution plan. For more information, see official 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 ApsaraDB RDS for MySQL database shard.

   If the query condition contains a shard key, directly run the EXPLAIN EXECUTE command to display the execution plan on the corresponding database shard. Examples:

```
mysql> EXPLAIN EXECUTE SELECT * FROM test WHERE c1 = 1;
+----+-------------+-------+-------+---------------+---------+---------+-------+------
+-------+
| 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 involves multiple shards (for example, its condition does not contain a shard key), the EXPLAIN EXECUTE command returns an execution plan on a random ApsaraDB RDS for MySQL database shard.

   To view the execution plan of an SQL statement on a specified database shard, you can add a hint. Examples:

```
mysql> /! TDDL:node='TESTDB_1478746391548CDTCTESTDB_OXGJ_0000_RDS'*/
EXPLAIN SELECT * FROM test;
+----+-------------+-------+------+---------------+-------+---------+-------+------
+-------+
| 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 database shards.

   You can run SCAN HINT to display the execution plans of SQL statements on all database shards:

```
mysql> /! TDDL:scan='test'*/EXPLAIN SELECT * FROM test;
+----+-------------+-------+------+---------------+-------+---------+-------+------
+-------+
| 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:

a. In hint mode, PolarDB-X only replaces the table names in case of database or table sharding, and then directly sends the logical SQL statement to ApsaraDB RDS for MySQL for execution. It will not process the result.

b. Execution plans obtained by using an EXPLAIN command are generated by static analysis and are not actually executed in databases.

TRACE command

The TRACE command in PolarDB-X can track the SQL execution process and the overheads in each stage. It can be used together with the execution plan to facilitate SQL statement optimization.

The TRACE command contains two related commands: TRACE and SHOW TRACE, which must be used together.

5.14.4 SQL optimization methods

5.14.4.1 Overview

This topic describes the SQL optimization principles 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 ApsaraDB RDS for MySQL instances is called push-down computing. Push-down computing reduces data transmission, decreases overheads at the network layer and PolarDB-X layer, and improves the execution efficiency of SQL statements.

Therefore, the basic principle for SQL statement optimization in PolarDB-X is as follows: Push down as many computations as possible to ApsaraDB RDS for MySQL instances.

Push-down computations include:

- JOIN connections
- Filter conditions, such as `WHERE` or `HAVING` conditions
- Aggregate computing, such as `COUNT` and `GROUP BY`
- Sorting, such as `ORDER BY`
- Deduplication, such as `DISTINCT`

- Function computing, such as the NOW() function
- Subqueries

> **Notice:**
> The preceding list only describes possible forms of push-down computations. It does not mean that all clauses or conditions or combinations of clauses or conditions can be pushed down for computing.

SQL statements of different types and containing different conditions can be optimized in different ways. The following uses some examples to describe how to optimize SQL statements:

- Single-table SQL optimization
  - Filter condition optimization
  - Optimization of the number of returned rows for a query
  - Grouping and sorting optimization
- JOIN query optimization
  - Optimization of push-down JOIN queries
  - Optimization of distributed JOIN queries
- Subquery optimization

5.14.4.2 Single-table SQL optimization

Single-table SQL optimization must follow the following principles:

- Make sure that the SQL statements contain 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 should be as small as possible (far fewer than the number of shards, and remain unchanged as the business grows).
- If SQL statements do not contain a shard key, use only one of DISTINCT, GROUP BY, and ORDER BY in the same SQL statement.

Filter condition optimization

PolarDB-X partitions data horizontally by the shard key. Therefore, the filter condition must contain the shard key as much as possible so that PolarDB-X can push queries down to specific database shards based on the shard key values, to avoid scanning all tables in the PolarDB-X instance.

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:

```
mysql> SELECT * FROM test WHERE c2 = 2;
+----+----+
| c1 | c2 |
+----+----+
|  2 |  2 |
+----+----+
1 row in set (0.05 sec)
```

The corresponding execution plan is as follows:

```
mysql> EXPLAIN SELECT * FROM test WHERE c2 = 2;
+-----------------------------------------------
+-------------------------------------------------------------------+--------+
| 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 PolarDB-X query speed.

For example, a query on the test table contains a range filter condition with the shard key c1:

```
mysql> SELECT * FROM test WHERE c1 > 1 AND c1 < 4;
+----+----+
| c1 | c2 |
+----+----+
|  2 |  2 |
|  3 |  3 |
+----+----+
2 rows in set (0.04 sec)
```

The corresponding execution plan is as follows:

```
mysql> EXPLAIN SELECT * FROM test WHERE c1 > 1 AND c1 < 4;
```

```
+-----------------------------------------------
+-----------------------------------------------------------------------------------------------
+--------+
| GROUP_NAME                          | SQL                                                                                                  |
PARAMS |
+-----------------------------------------------
+-----------------------------------------------------------------------------------------------
+--------+
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0002_RDS | select `test`.`c1`,`test`.`c2`
from `test` where ((`test`.`c1` > 1) AND (`test`.`c1` < 4)) | {}    |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0003_RDS | select `test`.`c1`,`test`.`c2`
from `test` where ((`test`.`c1` > 1) AND (`test`.`c1` < 4)) | {}    |
+-----------------------------------------------
+-----------------------------------------------------------------------------------------------
+--------+
2 rows in set (0.00 sec)
```

The equivalence condition is executed faster than the range condition. Examples:

```
mysql> SELECT * FROM test WHERE c1 = 2;
+----+----+
| c1 | c2 |
+----+----+
|  2 |  2 |
+----+----+
1 row in set (0.03 sec)
```

The corresponding execution plan is as follows:

```
mysql> EXPLAIN SELECT * FROM test WHERE c1 = 2;
+-----------------------------------------------
+-----------------------------------------------------------------------+--------+
| GROUP_NAME                          | SQL                                                                   | PARAMS |
+-----------------------------------------------
+-----------------------------------------------------------------------+--------+
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0002_RDS | select `test`.`c1`,`test`.`c2`
from `test` where (`test`.`c1` = 2) | {}    |
+-----------------------------------------------
+-----------------------------------------------------------------------+--------+
1 row in set (0.00 sec)
```

In addition, if you want to insert data into a table shard, the inserted field must contain a shard key.

For example, the data inserted into the test table contains the shard key c1:

```
mysql> INSERT INTO test(c1,c2) VALUES(8,8);
Query OK, 1 row affected (0.07 sec)
```

Optimization of the number of returned rows for a query

When PolarDB-X runs a query that contains **LIMIT** [offset,] row_count, PolarDB-X actually reads records before offset in order and directly discards them. In this way, when the value

of offset is large, the query is slow even if the value of row_count is small. For example, run the following SQL statement:

```
SELECT *
FROM sample_order
ORDER BY sample_order.id
LIMIT 10000, 2
```

Although only the 10000th and 10001st records are returned, it takes about 12 seconds to run the SQL statement because PolarDB-X actually reads 10,002 records.

```
mysql> SELECT * FROM sample_order ORDER BY sample_order.id LIMIT 10000,2;
+--------------+------------+--------------+--------------+------------+
| 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)
```

The corresponding execution plan is as follows:

```
mysql> EXPLAIN SELECT * FROM sample_order ORDER BY sample_order.id LIMIT 10000,2;
+------------------+-------------+------------+
| 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 `sample_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 `sample_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 `sample_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 `sample_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 `sample_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 `sample_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 `sample_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 `sample_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 IN to match the actual records. The modified SQL query is as follows:

```
SELECT *
FROM sample_order o
WHERE o.id IN (
    SELECT id
    FROM sample_order
    ORDER BY id
    LIMIT 10000, 2 )
```

The purpose 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 an ID, PolarDB-X can also push down such an IN query to different database shards through rule-based calculation, avoiding full table scan and unnecessary network I/O operations. Observe the effect of the modified SQL query:

```
mysql> SELECT *
   -> FROM sample_order o
   -> WHERE o.id IN ( SELECT id FROM sample_order ORDER BY id LIMIT 10000,2 );
+--------------+------------+--------------+--------------+------------+
| 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.

The corresponding execution plan is as follows:

```
mysql> EXPLAIN SELECT *
   -> FROM sample_order o
   -> WHERE o.id IN ( SELECT id FROM sample_order ORDER BY id LIMIT 10000,2 );
+-----------------------------------------------
+-----------------------------------------------------------------------------------------------------------------
+--------+
| 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 DISTINCT, GROUP BY, and ORDER BY, try to ensure that the fields after DISTINCT, GROUP BY, and ORDER BY are the same and the fields are shard keys. In 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 amount of data and sort the data in a temporary table, thereby maximizing the system performance.

5.14.4.3 JOIN query optimization

JOIN queries in PolarDB-X are classified into push-down JOIN queries and non-push-down JOIN queries (distributed JOIN queries). The optimization policies for these two types of JOIN queries are different.

Optimize push-down JOIN queries

Push-down JOIN queries are classified into the following types:

- JOIN queries between single tables (non-partition tables).
- The tables involved in the JOIN query contain the shard key in the filter condition and use the same sharding algorithm (that is, the data calculated by the sharding algorithm is distributed to the same shard).
- Tables involved in the JOIN query use the shard key as the JOIN condition and use the same sharding algorithm.
- JOIN query between broadcast tables (or small table broadcast) and table shards.

In PolarDB-X, optimize JOIN queries to push-down JOIN queries that can be executed on database shards.

Take a JOIN query between a broadcast table and table shards as an example. The broadcast table is used as the JOIN driving table (the left table in the JOIN query is called the driving table). The PolarDB-X broadcast table stores the same data in each database shard. When the broadcast table is used as the JOIN driving table, the JOIN query between this broadcast table and table shards can be converted into single-database JOIN queries and combined for computing to improve query performance.

For example, a JOIN query is performed on the following three tables, among which the sample_area table is the broadcast table, and the sample_item and sample_buyer tables are table shards. The query execution time is about 15s:

```
mysql> 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;
+------+
| name |
+------+
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
| BJ |
+------+
10 rows in set (14.88 sec)
```

If you adjust the JOIN query order and move the broadcast table to the farthest left as the JOIN driving table, the JOIN query is pushed down to a single database shard in the PolarDB -X instance:

```
mysql> 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;
+------+
| 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, which is a significant improvement to the query performance.

 **Notice:**

The broadcast table achieves data consistency through the synchronization mechanism on database shards, with a latency of several seconds.

Optimize distributed JOIN queries

If a JOIN query cannot be pushed down (that is, the JOIN condition and filter condition do not contain the shard key), PolarDB-X must complete part of the computing in the query. Such a query is a distributed JOIN query.

Tables in a distributed JOIN query 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 computing 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 computing after filtering.

In most cases, Nested Loop and its derived algorithms are used in JOIN computing at the PolarDB-X layer. If sorting is required for JOIN queries, the Sort Merge algorithm is used. When the Nested Loop algorithm is used, a smaller data size in the left table in a JOIN query indicates a smaller number of queries performed by PolarDB-X on the right table. If the right table has indexes or contains a small amount of data, the JOIN query is even faster. In PolarDB-X, the left table of a distributed JOIN query is called the driving table. To optimize a distributed JOIN query, 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 query as an example. The query takes about 24 seconds:

```
mysql> 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 < 1733635660;
+---------------------------------+--------+
| 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 query is an INNER JOIN query, with the actual size of the intermediate data involved in JOIN computing unknown. In this case, perform COUNT() on the o table and t table respectively to obtain the actual data size.

For the o table, o.sellerId < 1733635660 in the WHERE condition is only related to the o table. Then, extract and add it to the COUNT() condition of the o table. The following query result is returned:

```
mysql> SELECT COUNT(*) FROM sample_order o WHERE o.sellerId < 1733635660;
+----------+
| 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, which can be extracted directly for the COUNT() query:

```
mysql> SELECT COUNT(*) FROM sample_item i WHERE i.id = 242002396687;
+----------+
| 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 a large table and the t table is a small table. Use the small table as the driving table of the distributed JOIN query. The result of the adjusted JOIN query is as follows:

```
mysql> 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;
+----------------------------------+--------+
| 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 |
+----------------------------------+--------+
```

```
10 rows in set (0.15 sec)
```

The query execution time decreases from about 24 seconds to 0.15 seconds, with the query performance significantly improved.

5.14.4.4 Subquery optimization

When optimizing an SQL statement that contains subqueries, push the subqueries down to database shards as much as possible to reduce the computing workload at the PolarDB-X layer.

For this purpose, you can try two optimization methods:

- Rewrite subqueries into multi-table JOIN queries, and optimize the JOIN queries.
- Use the shard key in the JOIN condition or filter condition so that PolarDB-X can push the query down to a specific database shard to avoid full table scan.

The following subquery is used 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 query:

```
SELECT o. *
FROM sample_order o LEFT JOIN sample_seller s ON o.sellerId = s.id
WHERE s.id IS NULL
```

5.14.5 Select connection pools for an application

A database connection pool is used to manage database connections in a centralized manner, so as to improve application performance and reduce database loads.

- **Reuse resources**: Connections can be reused to avoid high performance overheads caused by frequent connection creation and release. Resource reuse can also improve the system stability.
- **Improve the system response efficiency**: After the connection initialization is complete, all requests can directly use the existing connections, which avoids the overheads of connection initialization and release and improves the system response efficiency.
- **Avoid connection leakage**: The connection pool forcibly revokes connections based on the preset de-allocation policy to avoid connection resource leakage.

We recommend that you use a connection pool to connect applications and databases for service operations. For Java programs, we recommend that you use the Druid connection pool.

The following is an example of standard Druid Spring configuration:

```
   <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method
="init" destroy-method="close">
       <property name="driverClassName" value="com.mysql.jdbc.Driver" />
       <! -- Basic attributes URL, user, 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 size, minimum value, and maximum value -->
       <property name="maxActive" value="20" />
       <property name="initialSize" value="3" />
       <property name="minIdle" value="3" />
       <! -- maxWait indicates the time-out 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" />
       <! -- SQL statement used to check whether connections are available -->
       <property name="validationQuery" value="SELECT 'z'" />
       <! -- Whether to enable idle connection check -->
       <property name="testWhileIdle" value="true" />
       <! -- Whether to check the connection status before obtaining a connection -->
       <property name="testOnBorrow" value="false" />
       <! -- Whether to check the connection status before releasing a connection -->
       <property name="testOnReturn" value="false" />
   </bean>
```

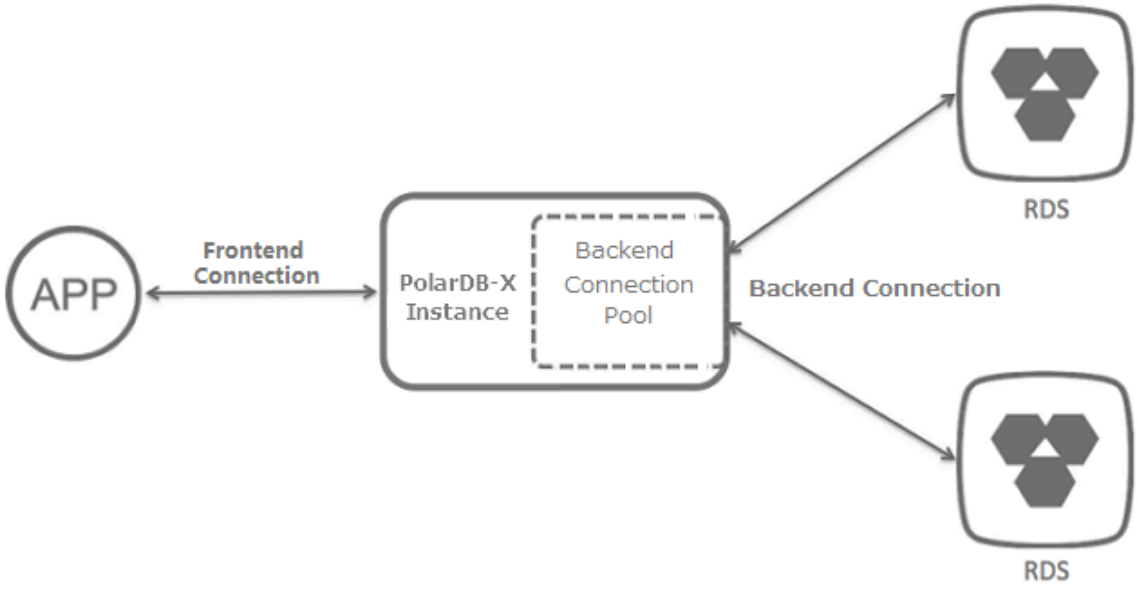
5.14.6 Connections to PolarDB-X instances

When an application connects to a PolarDB-X instance for operation, there are two types of connections 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 in a backend ApsaraDB RDS for MySQL instance.

Figure 5-14: PolarDB-X instance connection diagram



Frontend connection

Theoretically, the number of frontend connections is limited by the available memory size and the number of network connections to the nodes of the PolarDB-X instance. However, in actual application scenarios, when an application connects to a PolarDB -X instance, the nodes of the PolarDB-X instance usually manage a limited number of connections to perform requested operations, and do not maintain a large number of concurrent persistent connections (for example, tens of thousands of concurrent persistent connections). Therefore, the number of frontend connections that a PolarDB-X instance can accept can be considered to be unlimited.

Considering that the number of frontend connections is unlimited and a large number of idle connections are allowed, this method applies to scenarios where a large number of servers are deployed and need to maintain their connections to the PolarDB-X instance.

Note:
Although the number of frontend connections is considered as unlimited, operation requests obtained from frontend connections are actually executed by internal threads of the PolarDB-X instance through backend connections. Due to the limited number of

internal threads and backend connections, the total number of concurrent requests that can be processed by the PolarDB-X instance 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 databases 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 supported by the ApsaraDB RDS for MySQL instance. 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)
```

For example, a user has purchased an ApsaraDB RDS for MySQL instance and a PolarDB-X instance of the following types:

- The ApsaraDB RDS for MySQL instance has eight physical database shards, four cores, and 16 GB memory, supporting a maximum number of 4,000 connections.
- The PolarDB-X dedicated instance has 32 cores and 32 GB memory, with each PolarDB-X node having two cores and 2 GB memory (that is, the 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. In actual use, 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 smaller 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 SQL statement execution requests, the PolarDB-X nodes process the requests asynchronously and obtain backend connections through the internal backend connection pool, and then run optimized SQL statements on one or more physical databases.

PolarDB-X nodes process requests asynchronously 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, rather than the number of concurrent connections.

Although the number of frontend connections is considered to be unlimited, the maximum number of connections maintained in the backend connection pool of a PolarDB-X instance is limited. For more information, see "Backend connections." Therefore, note the following points in actual application 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, which reduces the overall concurrent processing capability and increases the response time (RT).
- Monitor and optimize or remove slow SQL queries run in the PolarDB-X instance, to prevent them from occupying too many backend connections. Otherwise, the PolarDB-X instance or the ApsaraDB RDS for MySQL instance is under greater processing pressure, which may lead to reduced concurrent processing capability, increased RT, or higher SQL execution failure rate due to execution timeout. For troubleshooting and optimization of slow SQL queries, see Troubleshoot slow SQL statements in PolarDB-X and Overview.
- Under normal use of connections and execution of queries, 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 Perform instance upgrade

Database performance can be measured by the response time (RT) and queries per second (QPS). RT reflects the performance of a single SQL statement. This type of performance problem can be solved through SQL optimization. PolarDB-X upgrade expands the capacity

to improve performance, and is suitable for database access services 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 any PolarDB-X or ApsaraDB RDS for MySQL node can create a bottleneck in the overall performance. This topic describes how to observe the performance metrics of a PolarDB-X instance and upgrade the PolarDB-X instance to solve the performance bottleneck. 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 the performance bottleneck of a PolarDB-X instance

The QPS and CPU performance of a PolarDB-X instance are in positive correlation. When a PolarDB-X instance encounters a performance bottleneck, the CPU utilization of the PolarDB-X instance remains high.

Observe the CPU utilization

1. On the **Basic Information** page of the PolarDB-X instance, choose **Monitoring and Alerts > Instance Monitoring** from the left-side navigation pane.
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 above 80%**, the PolarDB-X instance faces a performance bottleneck. If there is no bottleneck for the ApsaraDB RDS for MySQL instance, the current PolarDB-X instance specifications cannot meet the QPS performance requirements of the business. In this case, the PolarDB-X instance needs to be upgraded.

For more performance-related service monitoring scenarios and methods for configuring the PolarDB-X CPU utilization alert, see View monitoring information.

Upgrade PolarDB-X

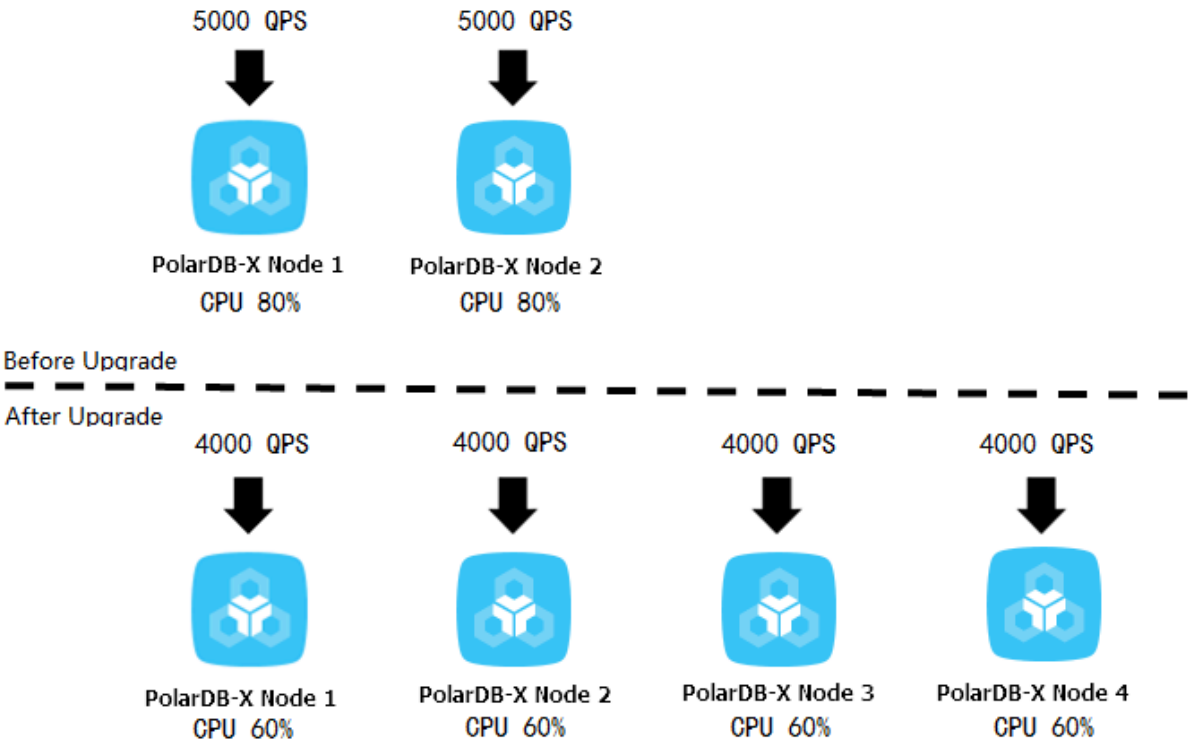
QPS is an important metric for determining whether the PolarDB-X instance specifications can meet the business requirements. Each type of instance specifications corresponds to a reference QPS value.

Some special SQL statements require more computing (such as temporary table sorting and aggregate computing) in PolarDB-X. In this case, the QPS supported by each PolarDB-X instance is lower than the standard value in its type.

PolarDB-X upgrade improves the processing performance of a PolarDB-X instance by adding nodes to share the QPS. As PolarDB-X nodes are stateless, this upgrade method linearly improves the performance of PolarDB-X instances.

For example, service A requires QPS of about 15 thousand. The current PolarDB-X instance has a 4-core virtual CPU (vCPU), 4 GB memory, and two nodes, supporting QPS of only 10 thousand. After finding that the CPU utilization of the PolarDB-X instance remains high, we upgraded the instance to 8-core vCPU and 8 GB memory, with each node handling about 4,000 QPS. Then, the performance meets service requirements, and the CPU utilization also drops to a reasonable level, as shown in the following figure.

Figure 5-15: PolarDB-X upgrade



For more information about how to upgrade a PolarDB-X instance, see Change specifications.

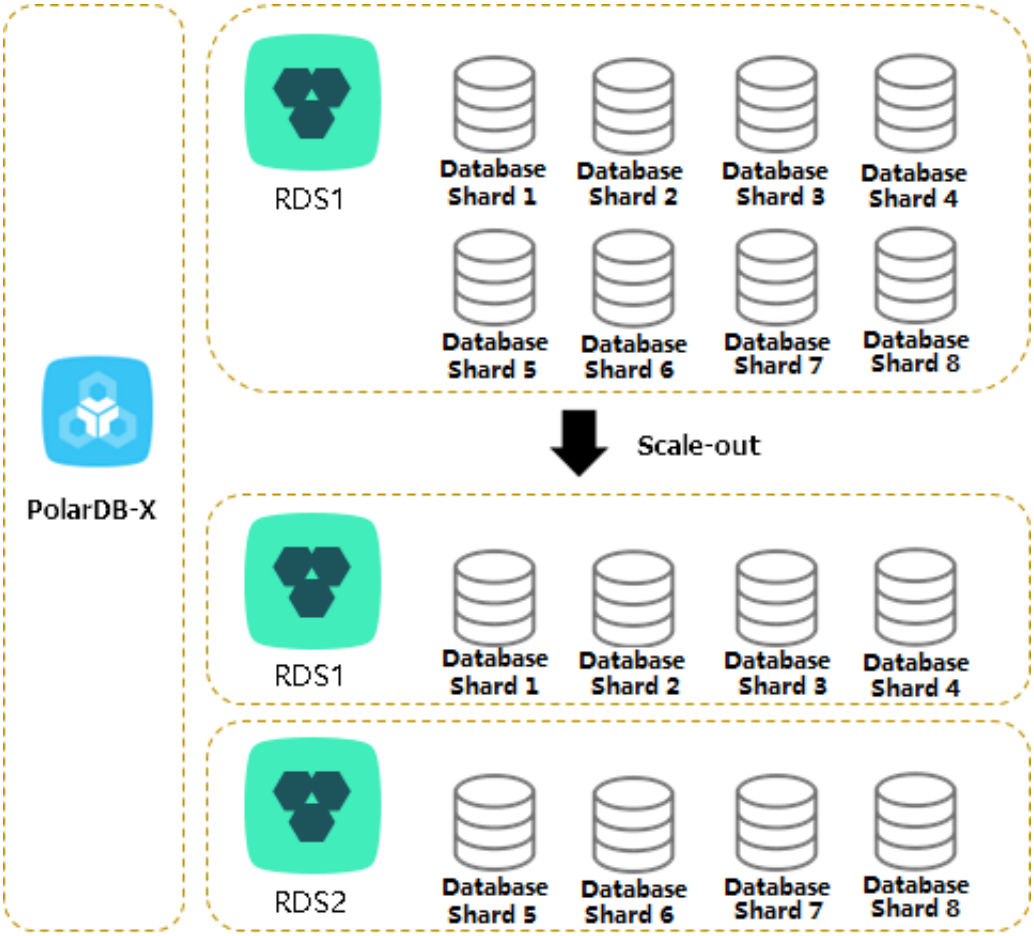
5.14.8 Perform scale-out

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 through PolarDB-X smooth scale-out to increase the PolarDB-X database capacity when the following conditions are met: 1. The input/output operations per second (IOPS), CPU utilization, disk space, and other metrics of the ApsaraDB RDS for

MySQL instance reach their bottlenecks. 2. The bottlenecks cannot be removed through SQL optimization or ApsaraDB RDS for MySQL upgrade (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. For example, before scale-out, all the eight databases are deployed in one ApsaraDB RDS for MySQL instance. After scale-out, the eight databases are deployed in two ApsaraDB RDS for MySQL instances, and the pressure on a single ApsaraDB RDS for MySQL instance is significantly reduced, as shown in the following figure.

Figure 5-16: PolarDB-X scale-out



After multiple scale-out operations, if the number of ApsaraDB RDS for MySQL instances is equal to the number of database shards, you need to create another PolarDB-X instance and ApsaraDB RDS for MySQL databases with the expected capacity, and then migrate data to further increase the data capacity. This process 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 properly when creating 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 the ApsaraDB RDS for MySQL documentation.

IOPS and CPU utilization

If you find that the IOPS or CPU utilization remains above 80% for a long time or you frequently receive alerts, follow these steps:

1. Optimize SQL statements. Generally, you can solve the high CPU utilization problem by this method.
2. If the problem persists, upgrade the ApsaraDB RDS for MySQL instance. For more information, see the ApsaraDB RDS for MySQL 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 space usage continues increasing as more data is inserted. We recommend that you keep the remaining disk space above 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 update transactions there are, the larger the occupied space is.

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 databases on multiple ApsaraDB RDS for MySQL instances through 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.

Note 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 data definition language (DDL) tasks in the console or connect to the PolarDB-X instance to directly run DDL SQL statements. Otherwise, the scale-out task may fail.
- Scale-out requires that the source database table have a primary key. If the source database 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 three to five 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 through rollback before the switchover.
- Scale-out 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 run as a slow SQL statement. Slow SQL statements in PolarDB-X are classified into slow logical SQL statements and slow physical SQL statements. In PolarDB-X, an SQL statement is run step by step on PolarDB-X and ApsaraDB RDS for MySQL nodes. Large execution loss on any node will result in slow SQL statements.

- Slow logical SQL statements are slow SQL statements sent by an application to PolarDB-X.
- Slow physical SQL statements are slow SQL statements sent by PolarDB-X to ApsaraDB RDS for MySQL.

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` command shows slow logical SQL statements, that is, SQL statements sent by an application to PolarDB-X.

The result set of the SHOW FULL SLOW command contains the following columns:

- TRACE_ID: the unique identifier of the SQL statement. A logical SQL statement and the physical SQL statements generated by the logical SQL statement have the same TRACE_ID. The TRACE_ID is also sent as a comment to ApsaraDB RDS for MySQL.
- HOST: the IP address of the client that sends the SQL statement.

  Notice:
  The client IP address may not be obtained when the network type is Virtual Private Cloud (VPC).
- START_TIME: the time when PolarDB-X starts running the SQL statement.
- EXECUTE_TIME: the time consumed by PolarDB-X to run 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 run.

The SHOW FULL PHYSICAL_SLOW command shows the slow physical SQL statements, that is, SQL statements sent by PolarDB-X to ApsaraDB RDS for MySQL.

The result set of SHOW FULL PHYSICAL_SLOW contains the following columns:

- TRACE_ID: the unique identifier of the SQL statement. A logical SQL statement and the physical SQL statements generated by the logical SQL statement have the same TRACE_ID. The TRACE_ID is also sent as a comment to ApsaraDB RDS for MySQL.
- GROUP_NAME: the name of a database group. Grouping aims to manage multiple groups of databases with identical data, such as the primary and secondary databases after data replication through ApsaraDB RDS for MySQL, which are mainly used for read/write splitting and primary/secondary switchover.
- DBKEY_NAME: the name of the database shard on which the SQL statement is run.
- START_TIME: the time when PolarDB-X starts running the SQL statement.
- EXECUTE_TIME: the time consumed by PolarDB-X to run the SQL statement.
- SQL_EXECUTE_TIME: the time consumed by PolarDB-X to call ApsaraDB RDS for MySQL to run this SQL statement.
- GETLOCK_CONNECTION_TIME: the time that PolarDB-X takes to get connections from the connection pool. If the value is large, the ApsaraDB RDS for MySQL connections have been 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 run SHOW PROCESSLIST for troubleshooting.

- CREATE_CONNECTION_TIME: the time consumed by PolarDB-X to establish a connection to ApsaraDB RDS for MySQL. 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 run.

Example 1

The following example describes how to locate the execution of a slow SQL statement on PolarDB-X and between PolarDB-X and ApsaraDB RDS for MySQL.

1. You can use certain conditions, such as the execution time and SQL string match, to obtain the specified slow SQL statement:

```
mysql> show full slow where `SQL` like '%select sleep(50)%';
+-----------------+-----------+-------------------------+--------------+------------
+------------------+
| 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)
```

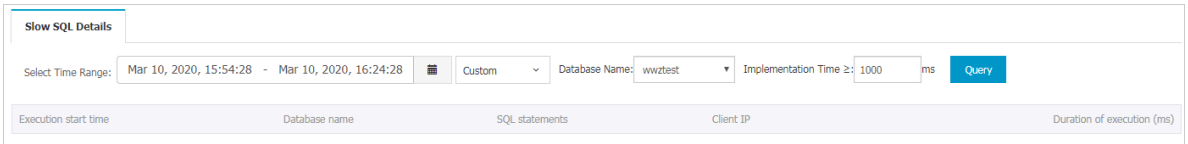
2. Based on the **TRACE_ID** of the slow logical SQL statement, run `SHOW FULL PHYSICAL_SLOW` to obtain the physical execution information of this SQL statement.

```
mysql> show full physical_slow where trace_id = 'ae0e565b8c00000';
+-----------------+-----------------------------------------------------
+-----------------------------------------------+-----------------------+--------------
+------------------+------------------------+-----------------------+------------
+------------------+
| TRACE_ID        | GROUP_NAME                                | DBKEY_NAME                            |
START_TIME              | EXECUTE_TIME | SQL_EXECUTE_TIME | GETLOCK_CONNECTION_TIME |
CREATE_CONNECTION_TIME | AFFECT_ROW | SQL             |
+-----------------+-----------------------------------------------------
+-----------------------------------------------+-----------------------+--------------
+------------------+------------------------+-----------------------+------------
+------------------+
| ae0e565b8c00000 | PRIV_TEST_1489167306631PJAFPRIV_TEST_APKK_0000_RDS |
rdso6g5b6206sdq832ow_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 **TRACE_ID**.

Figure 5-17: Slow query logs



Example 2

This example describes how to locate the original SQL statement in PolarDB-X based on the slow SQL statement located in ApsaraDB RDS for MySQL.

1. Based on the slow SQL query log in ApsaraDB RDS for MySQL, **TRACE_ID** of the slow SQL statement is ae0e55660c00000.
2. Based on the **TRACE_ID** obtained in Step 1, run `SHOW FULL PHYSICAL_SLOW` to obtain the physical execution information of this SQL statement.

```
mysql> show full physical_slow where trace_id = 'ae0e55660c00000';
+-----------------+-----------------------------------------------------
+-------------------------------------------------+-------------------------+--------------
+------------------+-------------------------+-----------------------+------------
+------------------+
| TRACE_ID        | GROUP_NAME                                | DBKEY_NAME                            |
START_TIME           | EXECUTE_TIME | SQL_EXECUTE_TIME | GETLOCK_CONNECTION_TIME
| CREATE_CONNECTION_TIME | AFFECT_ROW | SQL            |
+-----------------+-----------------------------------------------------
+-------------------------------------------------+-------------------------+--------------
+------------------+-------------------------+-----------------------+------------
+------------------+
| ae0e55660c00000 | PRIV_TEST_1489167306631PJAFPRIV_TEST_APKK_0000_RDS |
rdso6g5b6206sdq832ow_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

Generally, you can locate a slow SQL statement in two ways: Obtain historical information about slow SQL statements from slow SQL statement records, or run `SHOW PROCESSLIST` to display the real-time execution information about slow SQL statements.

You can troubleshoot slow SQL statements as follows:

1. Locate slow SQL statements.
2. Locate nodes with performance loss.
3. Troubleshoot the performance loss.

Note:
During troubleshooting, we recommend that you use the MySQL command line `mysql -hIP -PPORT -uUSER -pPASSWORD -c` to create the connection. Be sure to add `-c` to prevent the MySQL client from filtering out the comments (default operation) and therefore affecting the execution of HINT.

- View slow SQL statement records

  Run the following command to query top 10 slow SQL statements. This command can query logical SQL statements in PolarDB-X. One logical SQL statement corresponds to SQL statements of one or more databases or tables of the ApsaraDB RDS for MySQL instance. For more information, see Details about a low SQL statement.

```
mysql> SHOW SLOW limit 10;
+-----------------+-------------+-----------------------+-------------+------------
+-------------------------------------------------------------------+
| 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) from 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, run **SHOW PROCESSLIST** 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 run.

```
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      |    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 describes each column:

- ID: the ID of the connection.
- USER: the user name of the database shard in which this SQL statement is run.
- DB: the specified database. If no database is specified, the value is NULL.
- COMMAND: the type of the command being executed. SLEEP indicates an idle connection. For more information about other commands, 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 SQL statement being executed. The SQL statement may be too long to be displayed completely. You can derive the complete SQL statement based on information such as service parameters.

In the current example, the following slow SQL statement is identified:

```
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_COST(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   |
    |    2 |      2.378 | Query        | TEST_123_1488766060743ACTJSANGUAN_TEST_1
23_WVVP_0003_RDS | rdso6g5b6206sdq832ow_test_123_wvvp_0003_hbpz | 15          | 1.
59                   |    1 | select `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  | NULL   |
    |    3 |      2.731 | Query        | TEST_123_1488766060743ACTJSANGUAN_TEST_1
23_WVVP_0000_RDS | rdso6g5b6206sdq832ow_test_123_wvvp_0000_hbpz | 11          | 1.
78                   |    1 | select `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  | NULL   |
    |    4 |      2.933 | Query        | TEST_123_1488766060743ACTJSANGUAN_TEST_1
23_WVVP_0004_RDS | rdso6g5b6206sdq832ow_test_123_wvvp_0004_hbpz | 15          | 1.
48                   |    1 | select `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  | NULL   |
```

```
   |   5 |     3.111 | Query      | TEST_123_1488766060743ACTJSANGUAN_TEST_1
23_WVVP_0001_RDS | rdso6g5b6206sdq832ow_test_123_wvvp_0001_hbpz | 15         | 1.
56              |   1 | select `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 | NULL   |
   |   6 |     3.323 | Query      | TEST_123_1488766060743ACTJSANGUAN_TEST_1
23_WVVP_0007_RDS | rdso6g5b6206sdq832ow_test_123_wvvp_0007_hbpz | 15         | 1.
54              |   1 | select `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 | NULL   |
   |   7 |     3.496 | Query      | TEST_123_1488766060743ACTJSANGUAN_TEST_1
23_WVVP_0006_RDS | rdso6g5b6206sdq832ow_test_123_wvvp_0006_hbpz | 18         | 1.
30              |   1 | select `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 | NULL   |
   |   8 |     3.505 | Query      | TEST_123_1488766060743ACTJSANGUAN_TEST_1
23_WVVP_0005_RDS | rdso6g5b6206sdq832ow_test_123_wvvp_0005_hbpz | 423507     | 1
.97             |   1 | select `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 | NULL   |
   |   9 |     3.686 | Query      | TEST_123_1488766060743ACTJSANGUAN_TEST_1
23_WVVP_0002_RDS | rdso6g5b6206sdq832ow_test_123_wvvp_0002_hbpz | 14         | 1.
47              |   1 | select `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 | 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  select `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`) tbpartitions 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; waiting for binlog to be updated | NULL
                                |      NULL |         NULL |      NULL |
```

```
| 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                |     NULL |        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.

   a. Run the current SQL statement in database shard 0.

   ```
   /! TDDL:node='DRDS_00_RDS'*/ SELECT * FROM DRDS_USERS;
   ```

   b. 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.	-
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.	-

Term	Description	Remarks
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.	-
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.	-

Term	Description	Remarks
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.

Term	Description	Remarks
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.	-
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.

Term	Description	Remarks
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.	-
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.	-

Term	Description	Remarks
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.	-

Term	Description	Remarks
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.	-
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.	-

Term	Description	Remarks
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.	-
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.	-

Term	Description	Remarks
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.	-