ALIBABA CLOUD

Alibaba Cloud

Cloud Native Distributed Database PolarDB-X

User Guide

Document Version: 20220526

Alibaba Cloud

Legal disclaimer

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

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

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

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

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

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

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

Document conventions

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

Table of Contents

1.Instance Management

1.1. Overview of instances

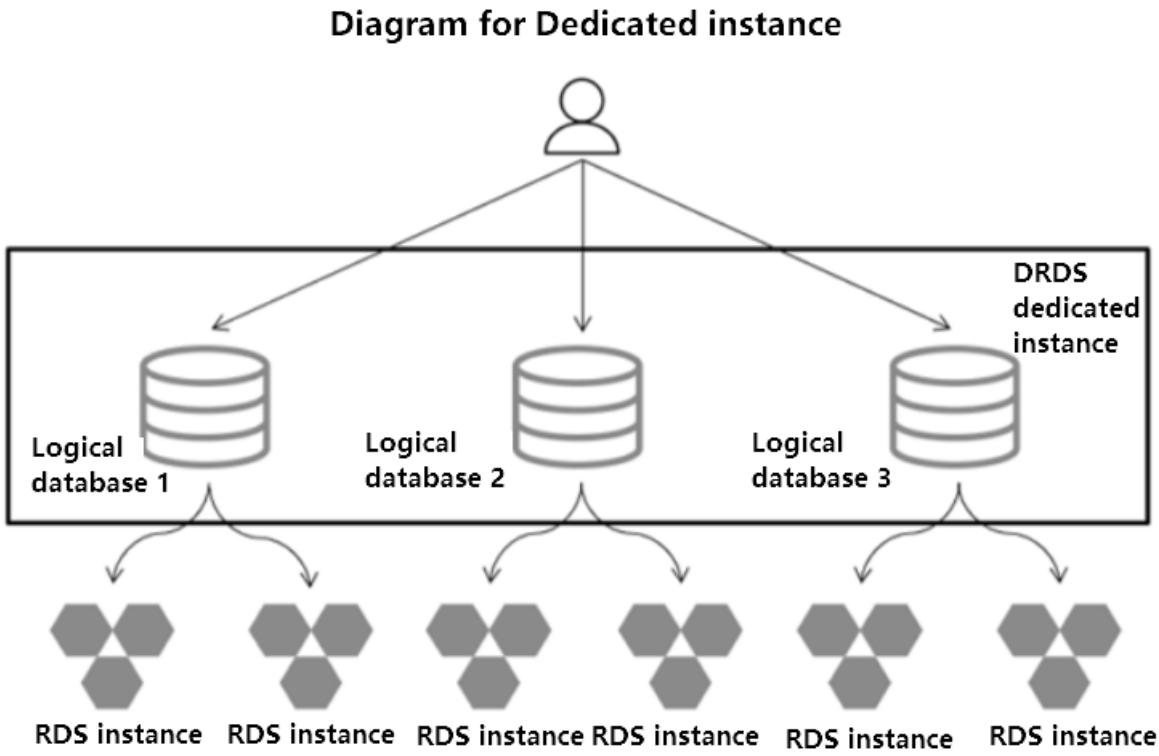
A PolarDB-X 1.0 instance consists of the compute layer and the storage layer resources. The compute layer is a distributed cluster that consists of multiple Distributed Relational Database Service (DRDS) server nodes and business data is stored in multiple ApsaraDB RDS for MySQL instances attached to the PolarDB-X 1.0 instance. Currently, all PolarDB-X 1.0 are dedicated instances.

Primary instances

PolarDB-X 1.0 primary instances are used for scaling of single-instance relational databases and can process real-time online transactions with high concurrency.

Dedicated instances

Dedicated instances offer exclusive physical resources.



1.2. Change the instance configurations

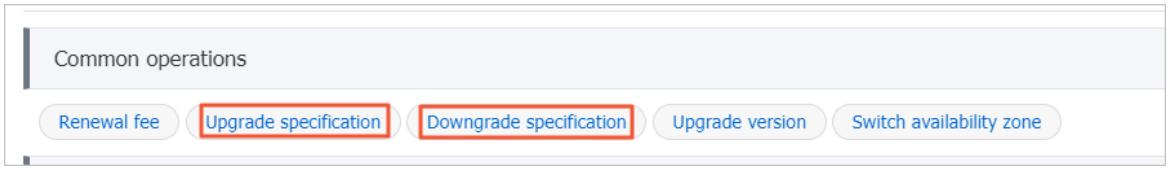
One PolarDB-X 1.0 is a distributed instance that consists of multiple compute nodes. Each node provides the features, such as Structured Query Language (SQL) routing, data merging, and data aggregation. You can change the node specification of the PolarDB-X 1.0 by changing instance configurations. You can upgrade the configurations during peak hours to share queries per second (QPS) of the business traffic. You can downgrade the configurations during off-peak hours to avoid wasting resources.

Notes

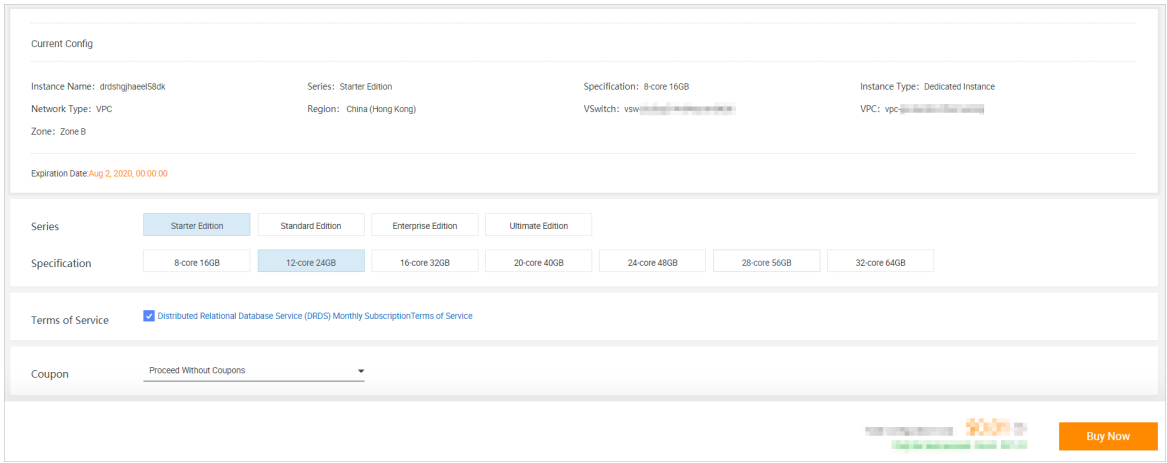
- Transient connection errors may occur in a short time when you downgrade the configurations due to the disconnection between the application and the instance. Make sure that your applications have the reconnection mechanism.
- When a persistent connection is in use, the newly added nodes after you upgrade the configurations cannot immediately receive traffic. We recommend that you restart Elastic Compute Service (ECS) instances in phased mode. Specifically, restart the ECS instances that host few services first, and then observe the service status to confirm that no problems occur before you restart all the other ECS instances.

Procedure

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, click **Instances**.
4. Find the target instance and click the instance ID to go to the **Basic Information** page.
5. In the **Common Actions** area, click **Upgrade** or **Downgrade** to go to the Upgrade/Downgrade page.



6. On the Upgrade/Downgrade page, select the instance series and the specification that you want, read and select terms of the service, and then click **Buy Now**.



7. After you complete the payment, wait for about five minutes to activate the service. Then, you can view the new instance configurations on the **Instances** page.

1.3. Upgrade the version

You can upgrade your PolarDB-X 1.0 instance to the latest version to immediately experience the new features. This topic describes how to upgrade the instance version of PolarDB-X 1.0 in the console.

Notes

- Before you upgrade your instance, perform comprehensive verification on an instance of the recommended version in advance to prevent SQL compatibility issues. We recommend that you

initiate a change evaluation for verification.

- You cannot perform other operations during the upgrade, such as database creation and smooth scale-out.
- You can roll back an upgraded instance within 24 hours, but you cannot change the configurations. To roll back the instance after the rollback time expires, you must Submit a ticket.
- Transient connection errors occur and a few errors are reported during the upgrade. We recommend that you perform the upgrade during off-peak hours.

Procedure

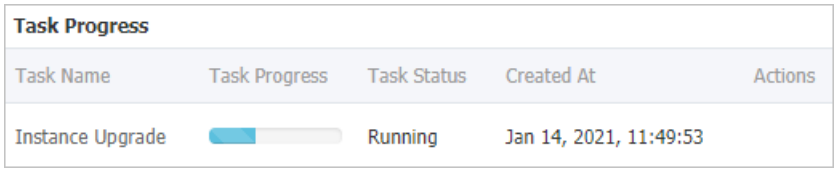
1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, click **Instances**.
4. Find the target instance and click the instance ID to go to the **Basic Information** page.
5. In the **Common Operations** area, click **Version Upgrade**.
6. In the dialog box that appears, configure the following parameters.

Parameter	Description
Performance Evaluation	You can choose to **Open** or **Close** performance evaluation. **Note** Performance evaluation has specific impact on the instance performance. We recommend that you evaluate the instance performance during off-peak hours.
Version series	Select the version that you want in the **Version series** drop-down list for the upgrade.

7. Click **Change assessment**.

   Note If you do not find the **Change assessment** button in the console, it indicates that this feature is not supported in the current region where the instance resides. You can skip the following steps and click **One-click Change** to upgrade your version.

8. In the upper-right corner of the **Basic Information** page of the instance, click the icon to check the evaluation task progress.



1.4. Switch zones

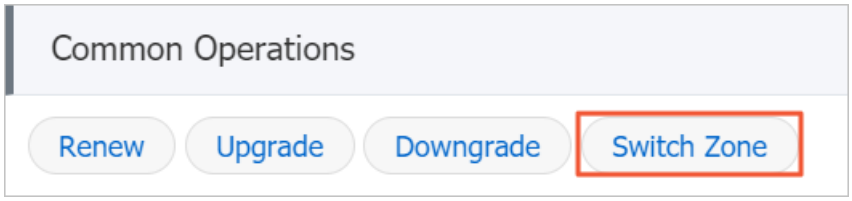
PolarDB-X 1.0 allows you to switch zones for instances. You can switch the zone of a PolarDB-X 1.0 instance if an inappropriate zone is selected for the instance or the quota in the original zone is insufficient.

Note

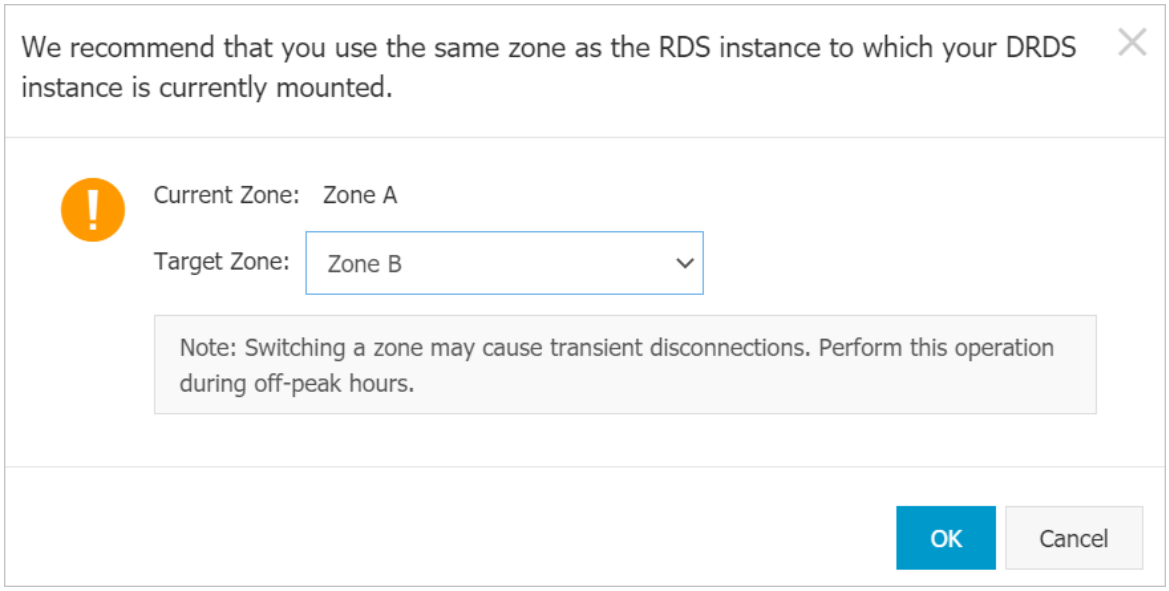
- Typically, it takes about one to two minutes to switch zones. We recommend that you switch zones during off-peak hours to avoid business interruptions.
- A transient disconnection within seconds may occur when you switch zones. Make sure that your application client supports automatic reconnection.

Procedure

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, click **Instances**.
4. Find the target instance and click the instance ID to go to the **Basic Information** page.
5. In the **Common Actions** section, click **Switch Zone**.



6. In the dialog box that appears, select a zone from the drop-down list, and click **OK**.



2.Database Management

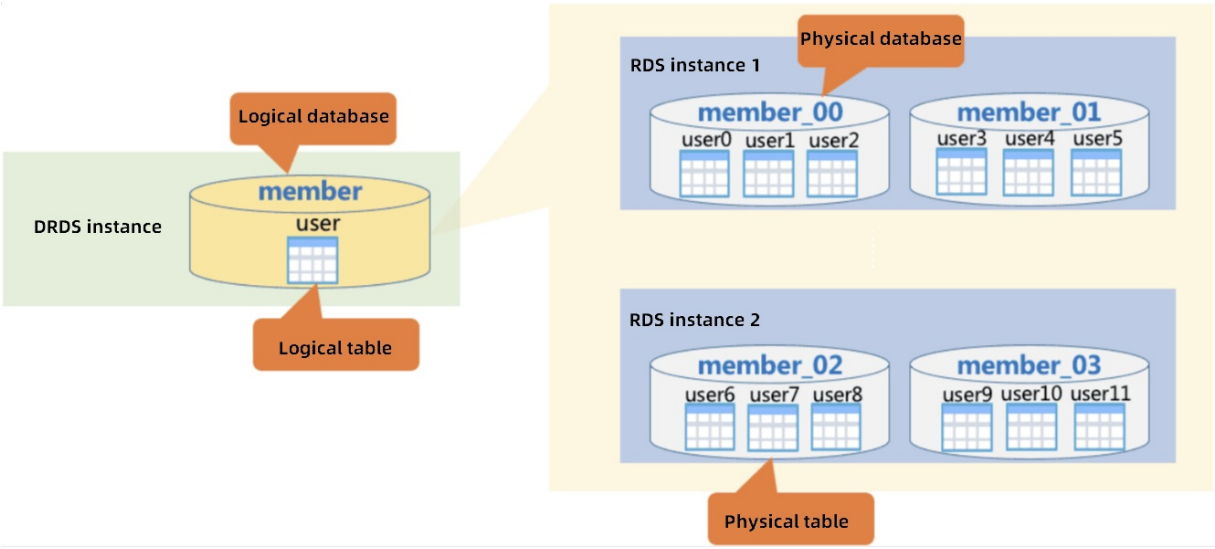
2.1. Overview

This topic introduces PolarDB-X 1.0 databases.

In theory, PolarDB-X 1.0 databases are similar to MySQL and Oracle databases.

In practice, PolarDB-X 1.0 provides a complete set of API operations, allowing business applications to directly connect to PolarDB-X 1.0 databases. This is the same as using a single-instance database.

In physical storage, the underlying storage of PolarDB-X 1.0 uses one or more ApsaraDB RDS for MySQL instances, and a PolarDB-X 1.0 database consists of the databases provided by one or more ApsaraDB RDS for MySQL instances.



The operations of database management on a single instance includes creating databases, viewing database details, configuring read/write splitting, managing tables, managing database shards, configuring whitelists, importing data, and viewing monitoring information.

2.2. View database information

The DRDS console allows you to view database details, delete databases, reset the password, and manage users with read-only access.

Context

The procedure is as follows:

Procedure

1. On the DRDS console, click **Databases** in the left-side navigation pane.
2. On the **Databases** page, click the name of the database you want to view.

   The **Basic Information** page appears.

   On the **Basic Information** page, you can view the database name, status, work mode, and so on.

The command-line URL is used to log on to the database. DRDS is fully compatible with MySQL, so you can operate DRDS databases from a MySQL client. Copy and paste the **command-line URL** to the operating system client and enter the password to log on to the DRDS database.

Result

Note:

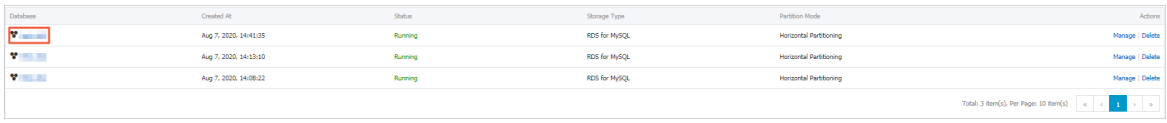
- Some MySQL clients of old versions require that the username contain no more than 16 characters. The DRDS database name and username are the same. If you set the database name to a string of more than 16 characters during database creation, an error is returned.
- When you operate a DRDS database from a MySQL client, you must add the "-c" parameter to the command line if you use hints. In DRDS, hints are implemented by commenting. If the "-c" parameter is not added, the comment is lost, causing loss of the DRDS hint.

2.3. Fix database shard connections

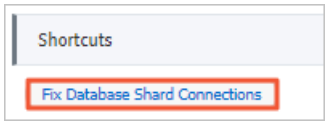
When you use a PolarDB-X 1.0 instance, you must access the mounted ApsaraDB RDS for MySQL instances. If the network configuration of a connected ApsaraDB RDS for MySQL instance changes, for example, if the zone is switched, the network connection between the PolarDB-X 1.0 instance and the ApsaraDB RDS for MySQL instance is broken. As a result, the PolarDB-X 1.0 instance cannot access the ApsaraDB RDS for MySQL instance. In this case, you must manually fix the database shard connection to restore the network connection. This topic describes how to manually fix database shard connections in the PolarDB-X 1.0 console.

Procedure

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, click **Instances**.
4. On the page that appears, find the target instance and click the instance ID.
5. In the left-side navigation pane, choose **Configuration and Management > Databases**.
6. On the page that appears, find the target database and click the database name.



7. On the **Basic Information** page of the target database, click **Fix Database Shard Connections** in the **Shortcuts** section.



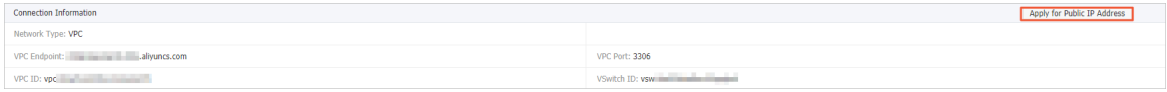
8. In the dialog box that appears, click **OK**.

2.4. Apply for or release a public IP address

By default, a PolarDB-X 1.0 instance has an internal network address, which allows applications and MySQL clients on Alibaba Cloud Elastic Compute Service (ECS) instances to connect to databases on PolarDB-X 1.0 instances. If you need to access the PolarDB-X 1.0 instance from a non-Alibaba-Cloud network, you must apply for a public IP address for the PolarDB-X 1.0 instance. This topic describes how to apply for and release a public IP address.

Apply for a public IP address

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, click **Instances**.
4. Find the target instance and click the instance ID to go to the **Basic Information** page.
5. In the upper-right corner of the **Connection Information** area, click **Apply for Public IP Address**.

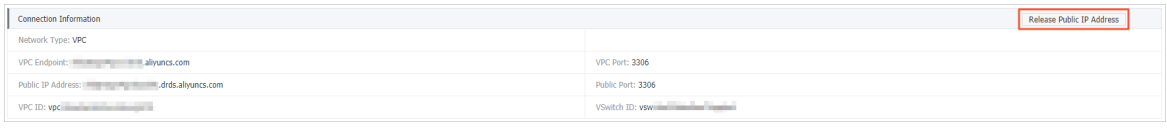


6. In the dialog box that appears, click **OK**.

> **Note**
>
> - To ensure data security, add the IP address of your application to the IP address whitelist of the target database on the instance. For more information, see Set an IP address whitelist.
> - To ensure data security, release the public IP address if it is no longer required. For more information, see Release a public IP address.

Release a public IP address

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, click **Instances**.
4. Find the target instance and click the instance ID to go to the **Basic Information** page.
5. In the upper-right corner of the **Connection Information** area, click **Release Public IP Address**.



6. In the dialog box that appears, click **OK**.

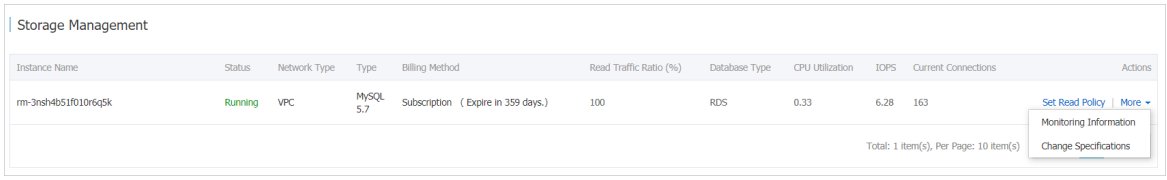
2.5. Storage management

This topic describs the storage management function.

Procedure

1. Go to the Basic Information page of a database, and then click **Storage Management** in the left-side navigation pane. On the Storage Management page, you can perform regular O&M operations

for dedicated ApsaraDB for RDS (RDS) instances, such as upgrading or downgrading specifications, adding or releasing read-only instances, and setting read policy.



2. To set read/write splitting for an RDS instance, find the target instance and click **Set Read Policy** in the Actions column. In the dialog box that appears, select a read policy, set Read Traffic Ratio, and then click **OK**.
   - On the Capacity Management page, you can only set the read/write ratio on a per RDS instance basis. If your PolarDB-X 1.0 database covers more than one RDS instance, you must set a read/write ratio for each RDS instance.
   - When an RDS read-only instance needs to be released, you must set its read weight to 0 in the PolarDB-X 1.0 console. Otherwise, traffic will still be routed to this instance, causing execution failures.
   - Read/write splitting is implemented on a per PolarDB-X 1.0 database basis. If one RDS read-only instance is used by multiple PolarDB-X 1.0 databases, you must set the read weight of this instance to 0 in each PolarDB-X 1.0 database.
3. After the settings are completed, connect to the PolarDB-X 1.0 database and run the SHOW NODE command to view the actual distribution of read traffic.

2.6. Adjust shard keys

This paper introduces the operation of managing data tables on PolarDB-X 1.0 console.

Prerequisites

You must connect to a database and use SQL statements to create tables before you manage tables in the console. For more information, see Connect to database and Basic SQL operations.

Function introduction

Adjust shard keys

In a PolarDB-X 1.0 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. You can consider that the database sharding and table sharding methods of a logical table in PolarDB-X 1.0 are the same, only when they use the same sharding function and shard key. The same sharding method enables PolarDB-X 1.0 to locate a unique physical database shard and physical table shard based on the value of the shard key. If the database sharding and table sharding methods of a logical table are different and an SQL query does not contain both the database shard key and the table shard key, PolarDB-X 1.0 scans all database shards or all table shards when it processes the SQL query.

Disable full table scan

By default, PolarDB-X 1.0 enables the full table scan feature for newly created tables. However, frequent use of the full table scan feature consumes database resources and reduces query efficiency. PolarDB-X 1.0 supports manually disable full table scan in the console.

Query table schema

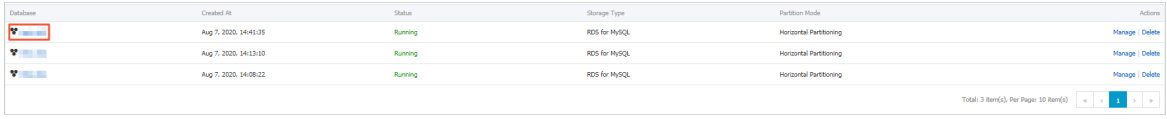
You can manage tables in the PolarDB-X 1.0 console and view table schema details.

Delete tables

Tables cannot be directly deleted in the PolarDB-X 1.0 console. You can log on to a database and use SQL statements to delete tables. For more information, see Connect to database and Basic SQL operations.

Enter the data tables interface

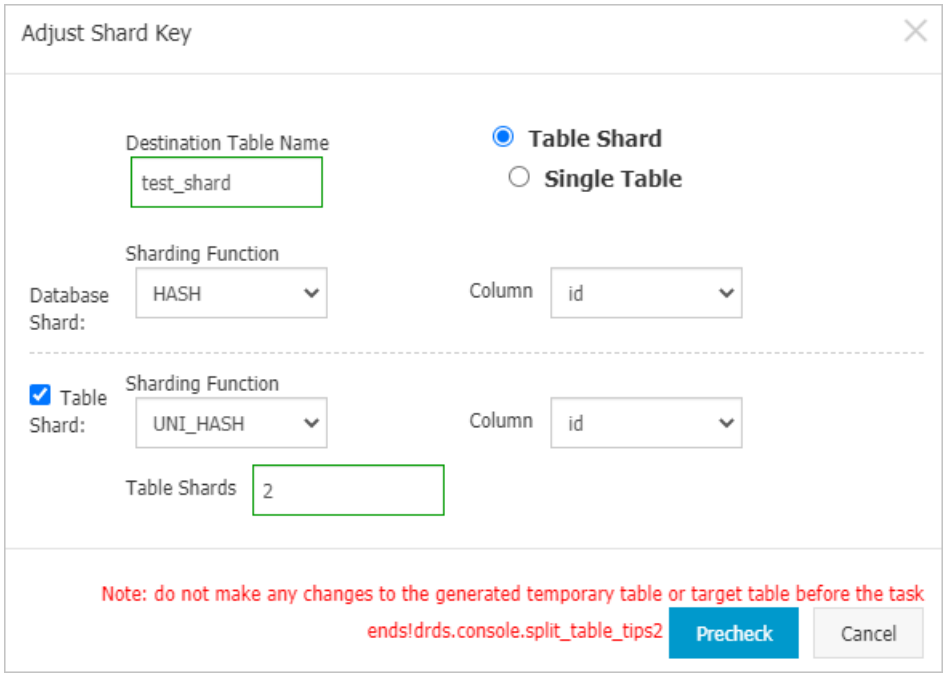
1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, click **Instances**.
4. On the page that appears, find the target instance and click the instance ID.
5. In the left-side navigation pane, choose **Configuration and Management > Databases**.
6. On the page that appears, find the target database and click the database name.



7. In the left-side navigation pane, choose **Table Management > Tables**.

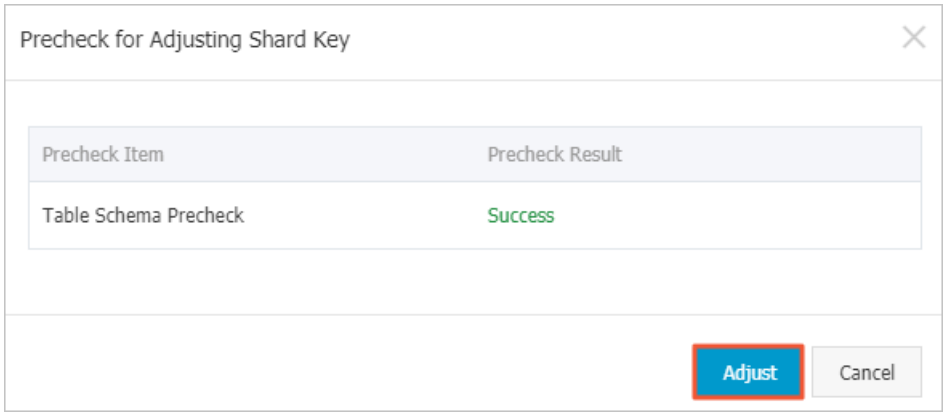
Adjust shard keys

1. On the tables page, find the target table and click **Adjust Shard Key** in the **Actions** column.
2. In the dialog box that appears, adjust the shard key settings as needed, and click **Precheck**.



> **Note**
>
> - Do not modify the generated temporary or destination table before the precheck task is completed.
> - Do not adjust the shard key for tables that have global secondary indexes. For more information about shard key selection and table shard forms, see CREATE TABLE and Overview.

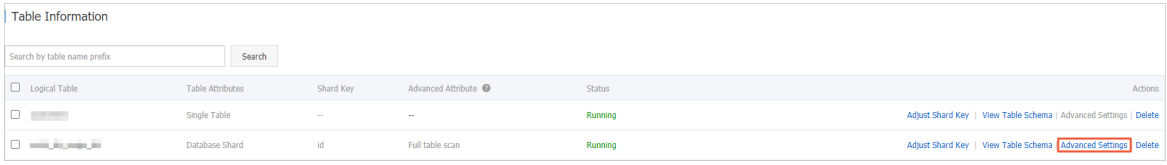
3. In the dialog box that appears, click **Adjust** after the precheck results are displayed.



Disable full table scan

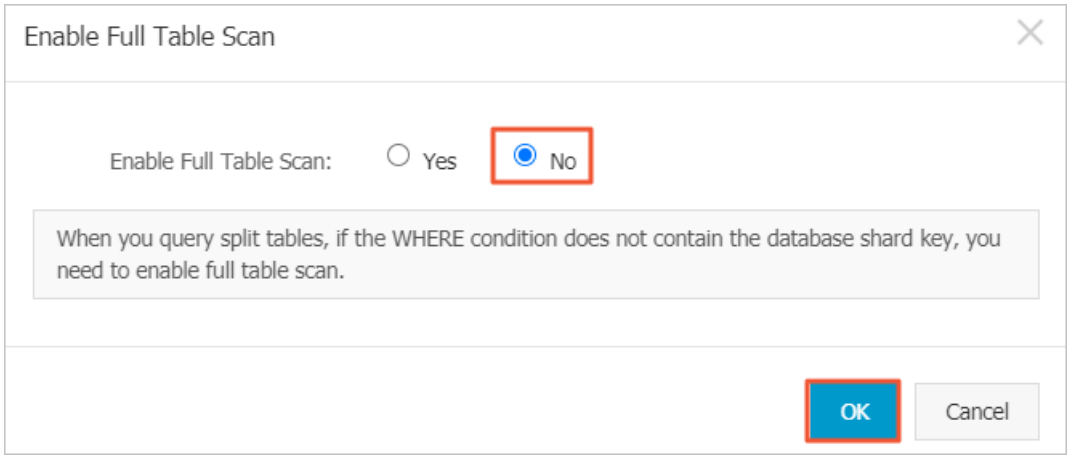
> **Note**
>
> - The target table must have a specified shard key. For more information about how to set a shard key, see Adjust shard keys.
> - If the full table scan feature is not enabled for a PolarDB-X 1.0 table shard, you must include the shard key in the WHERE clause to access the table. Otherwise, PolarDB-X 1.0 will return the TDDL-4510 error code.

1. On the tables page, find the target table and click **Advanced Settings** in the **Actions** column.



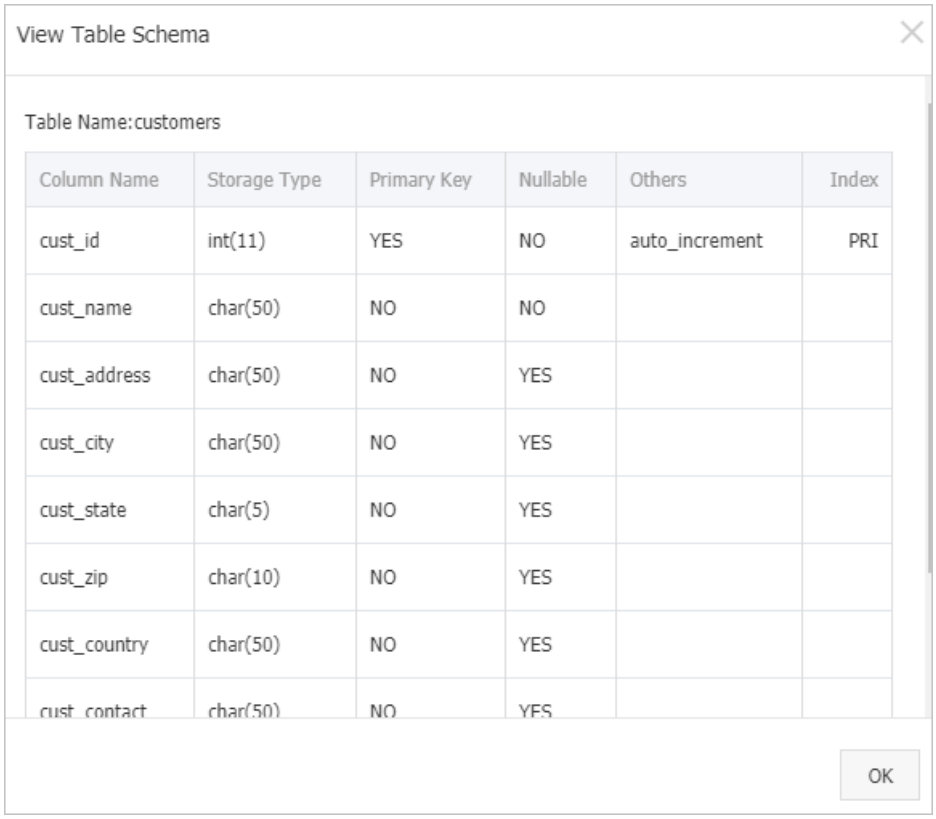
> **Note** The full table scan feature is supported only when **Table Attribute** is set to **Database Shard**. You can click Adjust shard keys to start database sharding.

2. In the dialog box that appears, select **No** and click **OK**.



Query table schema

On the tables page, find the target table and click **View Table Schema** in the **Actions** column. In the dialog box that appears, you can view table schema details, such as the column name, data type, and whether a column is a primary key.

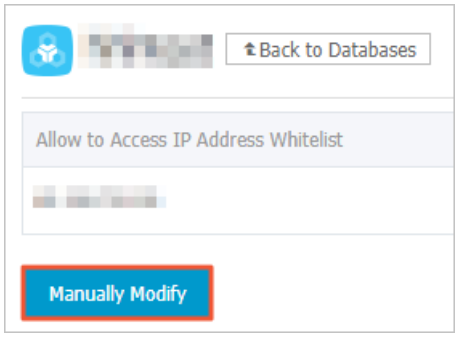


2.7. Set an IP address whitelist

PolarDB-X 1.0 allows you to set an IP address whitelist. Only IP addresses in the whitelist can access the PolarDB-X 1.0 databases. This topic describes how to set an IP address whitelist.

Procedure

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, click **Instances**.
4. Find the target instance, and click the instance ID.
5. In the left-side navigation pane, choose **Configuration and Management > Databases**.
6. Find the target database, and click the database name.
7. In the left-side navigation pane, choose **Data Security > Whitelist Settings**.
8. Click **Edit**. In the dialog box that appears, enter an IP address that is allowed to access the database.



Note

- After a database is created, all IP addresses are allowed to access the database by default. To ensure data security, we recommend that you only add the public IP addresses or CIDR blocks of your web servers to the whitelist.
- You can enter an IP address such as 192.168.0.1 or a CIDR block such as 192.168.0.0/24, 192.168.10.*, or 192.168.10.2-192.168.10.30.
- You can add up to 300 IP addresses or CIDR blocks to each IP address whitelist.
- Multiple IP addresses are separated with commas (,) such as 192.168.0.1,192.168.0.0/24.

9. Click **OK**.

3.Manage accounts

PolarDB-X 1.0 provides the account management feature. This topic describes how to create an account, modify account permissions, and reset the account password in the console.

Precautions

- The account management feature is only available on instances of 5.3.6 10460044 and later.
- In the console, you can only authorize DML, DDL, read-only, and read/write permissions to standard accounts. To grant more permissions, use SQL statements. For more information, see Manage accounts and permissions.

Account types and permissions

- PolarDB-X 1.0 instances support the following two types of database accounts.

Account type	Description
Privileged account	◦ You can create or manage privileged accounts by using SQL statements only. ◦ You can create only one privileged account on each instance, and can use this privileged account to manage all standard accounts and databases on the instance. ◦ A privileged account is granted with more permissions to enable personalized and refined management over permissions. For example, you can grant different users the permissions to query different tables. ◦ A privileged account has all the permissions on all the databases on the instance and can disconnect all accounts.
Standard account	◦ You can create or manage standard accounts in the console, by calling API operations, or executing SQL statements. ◦ You can create one or more standard accounts on each instance. The allowed maximum number of standard accounts depends on the kernel engine of the instance. ◦ You must manually grant standard accounts the permissions on specific databases. ◦ You cannot use a standard account to create or manage other accounts, or disconnect other accounts from databases.

- The following table shows the support for SQL operations by different account types with different permissions.

Account type	Permission	SELECT	INSERT	UPDATE	DELETE	INDEX	ALTER	CREATE	DROP	GRANT
	DDL	Not supported	Not supported	Not supported	Not supported	Supported	Supported	Supported	Supported	Not supported

Account type	Permission	SELECT	INSERT	UPDATE	DELETE	INDEX	ALTER	CREATE	DROP	GRANT
Standard account	DML	Supported	Supported	Supported	Supported	Not supported	Not supported	Not supported	Not supported	Not supported
	Read-only	Supported	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
	Read/write	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Not supported
Privileged account	Root	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported

Create an account

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, click **Instances**.
4. Find the target instance and click its ID.
5. In the left-side navigation pane, click **Account Management**.
6. In the upper-right corner of the page, click **Create an Account**.

> **Note** You can create only a standard account in the console. If you need to create a privileged account, use SQL statements. For more information.

7. In the dialog box that appears, set the following parameters.

Parameter	Description

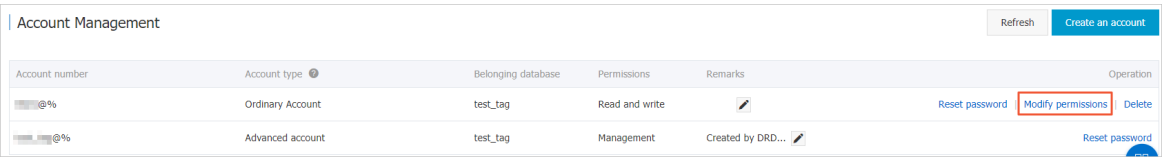
Parameter	Description
Database Account	Enter the account name. **Note** An account name must meet the following requirements: ○ The account name must be 2 to 16 characters in length and can contain lowercase letters, digits, and special characters. ○ The account name must start with a letter and end with a letter or digit. ○ The account name cannot be the same as the name of an existing account. ○ After the account is created, the full name of the account is composed of the name you entered and the host name. The host name is `%` by default, which means that this account is allowed to log on to the database from all hosts.
New Password	Enter the account password. **Note** An account password must meet the following requirements: ○ The password must be 8 to 32 characters in length. ○ The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. ○ The following special characters are allowed: ! @#$%^&*()_+-=
Confirm New Password	Enter the password again.

Parameter	Description
Authorization Database	You can grant permissions on one or more databases to the account. i. Select one or more databases and click **Authorization>** to move them from the **Databases List** box on the left to the **Authorized Databases** box on the right. ii. In the **Authorized Databases** area on the right, select the permissions on the target databases. **Note** ◦ The default permission is **Read-only**. You can also modify the permission to **Read/Write**, **DDL Only** or **DML Only**. ◦ If you need to grant the same permission on multiple databases, you can click the corresponding button next to Permissions in the upper-right corner of the **Authorized Databases** area, such as **Set All to Read/Write**.

8. Click **OK**.

Modify the permissions of a standard account

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, click **Instances**.
4. Find the target instance and click its ID.
5. In the left-side navigation pane, click **Account Management**.
6. Find the target account and click **Modify Permission** in the **Actions** column.



Note A privileged account has all the permissions on all databases, therefore, you do not need to **Modify Permission** for a privileged account.

7. In the dialog box that appears, set the following parameters.

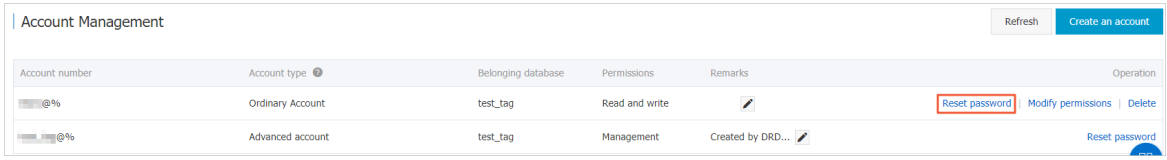
Parameter	Description

Parameter	Description
Authorization Database	You can modify the permissions on one or more databases for the account. i. Select one or more databases and click **Authorization>** or **<Remove** to add them to or remove them from the account. ii. In the **Authorized Databases** box on the right, select the permissions on the target databases. **Note** ○ You can also change the permission to **Read-only**, **Read/Write**, **DDL Only** or **DML Only**. ○ If you need to grant the same permission on multiple databases, you can click the corresponding button next to Permissions in the upper-right corner of the **Authorized Databases** area, such as **Set All to Read/Write**.

8. Click **OK**.

Reset the account password

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, click **Instances**.
4. Find the target instance and click its ID.
5. In the left-side navigation pane, click **Account Management**.
6. Find the target account and click **Reset Password** in the **Actions** column.



7. In the dialog box that appears, set the following parameters.

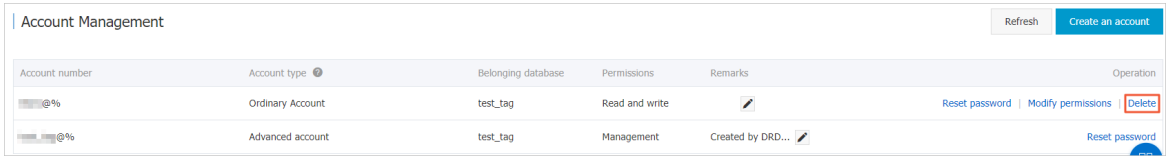
Parameter	Description

Parameter	Description
New Password	Enter the account password. **Note** An account password must meet the following requirements: ○ The password must be 8 to 32 characters in length. ○ The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. ○ The following special characters are allowed: ! @#$%^&*()_+-=
Confirm New Password	Enter the password again.

8. Click **OK**.

Delete an account

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, click **Instances**.
4. Find the target instance and click its ID.
5. In the left-side navigation pane, click **Account Management**.
6. Find the target account and click **Delete** in the **Actions** column.



Note The console does not support deleting **privileged accounts**. However, you can delete a privileged account by using SQL statements. For more information.

7. In the dialog box that appears, click **OK**.

4.Monitoring and Alerts

4.1. Monitor instances

PolarDB-X 1.0 provides the monitoring feature to help you check the status of your instances. This topic describes how to view performance monitoring data of a PolarDB-X 1.0 instance on the **Instance Monitoring** page in the console.

Monitoring metrics of instances

Metric category	Monitoring metric	Parameter	Description
Resource	CPU	`cpu`	The average CPU utilization of the compute node in the PolarDB-X 1.0 instance.
	Memory	`old`	The memory usage of the compute node in the PolarDB-X 1.0 instance. Memory usage fluctuations are normal.
	Network	`netin`: inbound traffic	The total inbound traffic of the compute node in the PolarDB-X 1.0 instance. Unit: Kbit/s. Inbound network traffic is generated when a storage node returns data to the compute node.
		`netout`: outbound traffic	The total outbound traffic of the compute node in the PolarDB-X 1.0 instance. Unit: Kbit/s. Outbound network traffic is generated when the compute node sends physical SQL statements to the storage node or the compute node returns data to applications.
Engine	Logical QPS	`lqps`	The total number of SQL statements that are processed by the instance per second.
	Physical QPS	`pqps`	The total number of SQL operations that are sent from the compute node to the storage node per second.
	Logical RT	`lrt`	The average response time (RT) for processing each SQL statement by the PolarDB-X 1.0 instance.
	Physical RT	`prt`	The average RT for processing SQL statements that are sent from the compute node to the storage node.
	Active Connections	`con`	The total number of connections between an application and the PolarDB-X 1.0 instance.
	Active Threads	`thread`	The number of threads that are used to execute SQL statements in the PolarDB-X 1.0 instance.

> **Note**
> - The data collection cycle is 1 minute for the metrics of the **Resource** category and 5 seconds for the metrics of the **Engine** category in the preceding table.
> - You can view the monitoring data that is generated within seven days.

Procedure

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. On the **Instances** page, find the instance and click the instance ID.
4. In the left-side navigation pane, choose **Monitoring and Alerting > Instance Monitoring**.
5. On the **Instance Monitoring** page, specify the following parameters.

Parameter	Description
Monitoring Index	The metric category. Select **Resource** or **Engine**.
Monitoring Index	The monitoring metrics of the instance. Select the monitoring metric that you want to view. For more information about monitoring metrics, see Monitoring metrics of instances.
Query Time	The query time. You can specify the query time as 1 hour, 6 hours, 12 hours, one day, or one week. You can also customize a query time range. The minimum time range is 1 minute and the maximum time range is one week.

4.2. Monitor databases

PolarDB-X 1.0 provides the monitoring feature to help you check the status of your instances. This topic describes how to view performance monitoring data of a single database in a PolarDB-X 1.0 instance on the **Database Monitoring** page in the console.

Monitoring metrics of databases

Monitoring metric	Parameter	Description
QPS	`qps`	The total number of SQL statements that are processed by the database in the PolarDB-X 1.0 instance per second.
Active Connections	`con`	The total number of connections between an application and the database in the PolarDB-X 1.0 instance.
Active Threads	`thread`	The number of threads that are used by the database in the PolarDB-X 1.0 instance to execute SQL statements.

> **Note**
> - The data collection cycle is 1 minute for each metric in the preceding table.
> - You can view the monitoring data that is generated within seven days.

Procedure

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. On the **Instances** page, find the instance and click the instance ID.
4. In the left-side navigation pane, choose **Monitoring and Alerting > Database Monitoring**.
5. On the **Database Monitoring** page, specify the following parameters.

Parameter	Description
Database	The database whose monitoring metrics you want to view. Select the database from the **Database** drop-down list.
Data Indexes	The monitoring metrics of the database. Select the monitoring metric that you want to view. For more information about monitoring metrics, see Monitoring metrics of databases.
Query Time	The query time. You can specify the query time as 1 hour, 6 hours, 12 hours, one day, or one week. You can also customize a query time range. The minimum time range is 1 minute and the maximum time range is one week.

4.3. Monitor storage nodes

PolarDB-X 1.0 provides the monitoring feature to help you check the status of your instances. This topic describes how to view performance monitoring data of storage nodes in a PolarDB-X 1.0 instance on the Storage Monitoring page in the console.

Monitoring metrics of storage nodes

Metric category	Monitoring metric	Parameter	Description
Resource	**CPU and Memory**	`MySQL_MemCpuUsage`	The CPU utilization and memory usage of the storage node in the instance.
	Disk Size	`MySQL_DetailedSpaceUsage`	The usage of the total space, data space, log space, temporary space, and system space of the storage node in the instance. Unit: MB.
	IOPS	`MySQL_IOPS`	The input/output operations per second (IOPS) of the storage node in the instance.

Metric category	Monitoring metric	Parameter	Description
	Connections	`MySQL_Sessions`	The number of active connections and the total number of connections of the storage node in the instance.
	Network Traffic	`MySQL_NetworkTraffic`	The inbound and outbound traffic of the storage node per second in the instance. Unit: KB.
Engine	**TPS/QPS**	`MySQL_QPSTPS`	The average number of times that SQL statements are executed per second and the average number of transactions that run per second on the storage node in the instance.
	InnoDB Cache	`MySQL_InnoDBBufferRatio`	The read hit ratio, usage, and dirty ratio of the InnoDB buffer pool on the storage node in the instance.
	InnoDB Read/Write	`MySQL_InnoDBDataReadWriten`	The average amount of data read by InnoDB per second and the average amount of data written by InnoDB per second on the storage node in the instance. Unit: KB.
	Cached Requests	`MySQL_InnoDBLogRequests`	The average number of reads from the InnoDB buffer pool per second and the average number of writes to the InnoDB buffer pool per second on the storage node in the instance.
	InnoDB Log	`MySQL_InnoDBLogWrites`	The average number of log write requests per second, the average number of physical writes to log files per second, and the average number of times that the fsync operation is performed for log files per second on the storage node in the instance.
	Temporary Tables	`MySQL_TempDiskTableCreates`	The number of temporary tables that are automatically created on hard disks during statement execution on the storage node in the instance.
	COMDML	`MySQL_COMDML`	The average number of times that DELETE statements are executed per second, the average number of times that INSERT statements are executed per second, the average number of times that INSERT_SELECT statements are executed per second, the average number of times that REPLACE statements are executed per second, the average number of times that REPLACE_SELECT statements are executed per second, the average number of times that SELECT statements are executed per second, and the average number of times that UPDATE statements are executed per second on the storage node in the instance.

Metric category	Monitoring metric	Parameter	Description
	RowDML	`MySQL_RowDML`	The average number of rows read from InnoDB tables per second, the average number of rows updated in InnoDB tables per second, the average number of rows deleted from InnoDB tables per second, the average number of rows inserted into InnoDB tables per second, and the average number of physical writes to log files per second on the storage node in the instance.
	MyISAM Read/Write	`MySQL_MyISAMKeyReadWrites`	The average number of reads from the buffer pool by MyISAM per second, the average number of writes to the buffer pool by MyISAM per second, the average number of reads from hard disks by MyISAM per second, and the average number of writes to hard disks by MyISAM per second on the storage node in the instance.
	MyISAM Key	`MySQL_MyISAMKeyBufferRatio`	The average usage, read hit ratio, and write hit ratio of the MyISAM key buffer per second on the storage node in the instance.

> **Note**
> - The data collection cycle is 1 minute for the metrics of the **Resource** category and 5 seconds for the metrics of the **Engine** category in the preceding table.
> - You can view the monitoring data that is generated within seven days.

Procedure

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. On the **Instances** page, find the instance and click the instance ID.
4. In the left-side navigation pane, choose **Monitoring and Alerting > Storage Monitoring**.
5. On the **Storage Monitoring** page, specify the following parameters.

Parameter	Description
Database	The database whose monitoring metrics you want to view. Select the database from the **Database** drop-down list.
Monitoring Index	The metric category. Select **Resource** or **Engine**.
Monitoring Index	The monitoring metrics of the storage resources. Select the monitoring metric that you want to view. For more information about monitoring metrics, see Monitoring metrics of storage nodes.
Query Time	The query time. You can specify the query time as 1 hour, 6 hours, 12 hours, one day, or one week. You can also customize a query time range. The minimum time range is 1 minute and the maximum time range is one week.

4.4. Manage alert rules

This topic describes how to create and manage rules of threshold value alerts in the PolarDB-X 1.0 console. The alert feature helps you detect exceptions of instances, databases, and storage nodes of PolarDB-X 1.0 and handle the exceptions in time. This topic uses instance monitoring as an example to describe how to manage alert rules.

Create an alert rule

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. On the **Instances** page, find the instance for which you want to create an alert rule and click the instance ID.
4. In the left-side navigation pane, choose **Monitoring and Alerting > Instance Monitoring**.
5. On the **Instance Monitoring** page, click **Set Alert Rules**.

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

7. On the page that appears, specify the alert rule parameters based on your business requirements.

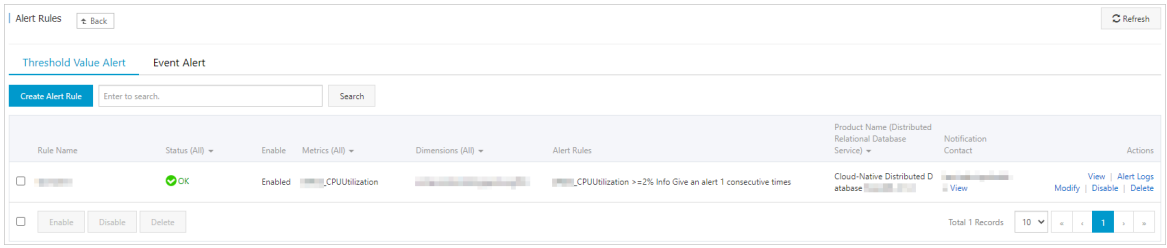
> **Note** In the **Related Resource** section, select **Cloud-Native Distributed Database PolarDB-X 1.0** from the **Product** drop-down list. For more information about how to specify other parameters, see Create an alert rule.

8. Click **Confirm**.

Manage alert rules

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. On the **Instances** page, find the instance that you want to manage and click the instance ID.
4. In the left-side navigation pane, choose **Monitoring and Alerting > Instance Monitoring**.
5. On the **Instance Monitoring** page, click **Set Alert Rules**.

6. On the **Threshold Value Alert** tab of the **Alert Rules** page, manage the existing alert rules by performing the following operations:



- Click **View** in the **Actions** column to view the basic information about the alert rule.
- Click **Alert Logs** in the **Actions** column to view alert history.
- Click **Modify** in the **Actions** column to modify the alert rule.
- Click **Disable** in the **Actions** column to disable the alert rule.
- Click **Delete** in the **Actions** column to delete the alert rule.
- Click **View** in the **Notification Contact** column to view the alert group, the alert contacts, and the alert notification method.

5.Query details about slow SQL queries

In PolarDB-X 1.0, SQL statements that are executed for more than 1 second are defined as slow SQL queries. Slow SQL queries consist of logical and physical slow SQL queries. This topic describes the definitions and query methods of the two types of slow SQL queries.

Definitions of slow SQL queries

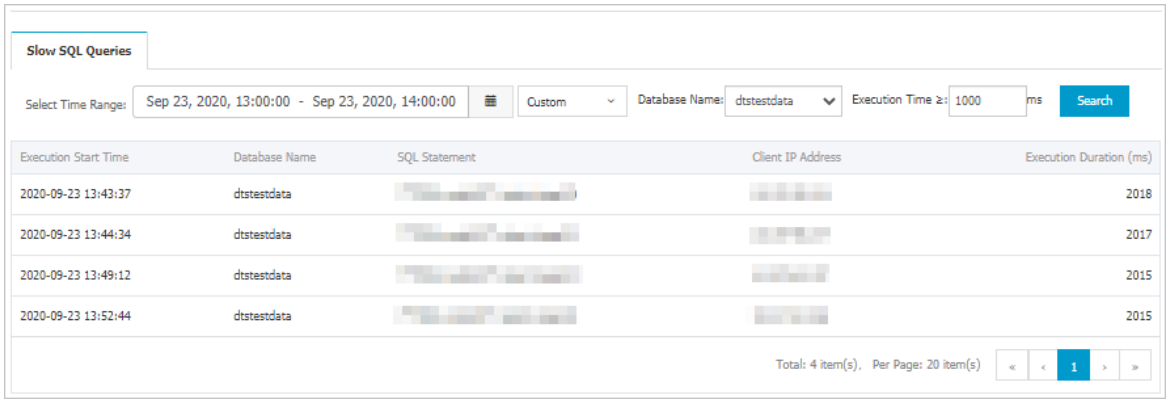
- Logical slow SQL queries: slow SQL queries that are sent from a client to PolarDB-X 1.0. The slow queries queried in the PolarDB-X 1.0 console are all logical slow SQL queries.
- Physical slow SQL queries: slow SQL queries that are sent from PolarDB-X 1.0 to the storage layer MySQL. All the slow SQL queries queried at the storage layer MySQL are physical slow SQL statements.

Precautions

Each PolarDB-X 1.0 node stores the details about a maximum of 5,000 slow SQL queries. The details about extra slow SQL queries are deleted in a rolling way.

Procedure

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, click **Instances**.
4. Find the target instance and click the instance ID to go to the **Basic Information** page.
5. In the left-side navigation pane, click **Slow SQL Queries**.
6. Specify **Select Time Range**, **Database Name**, and **Execution Time**, and then click **Search** to view the information about slow SQL queries.



7. You can also view the information about the corresponding physical slow SQL queries at the storage layer MySQL by performing the following steps:

> **Note** Only physical slow SQL queries in PolarDB for MySQL that serves as the storage layer can be queried.

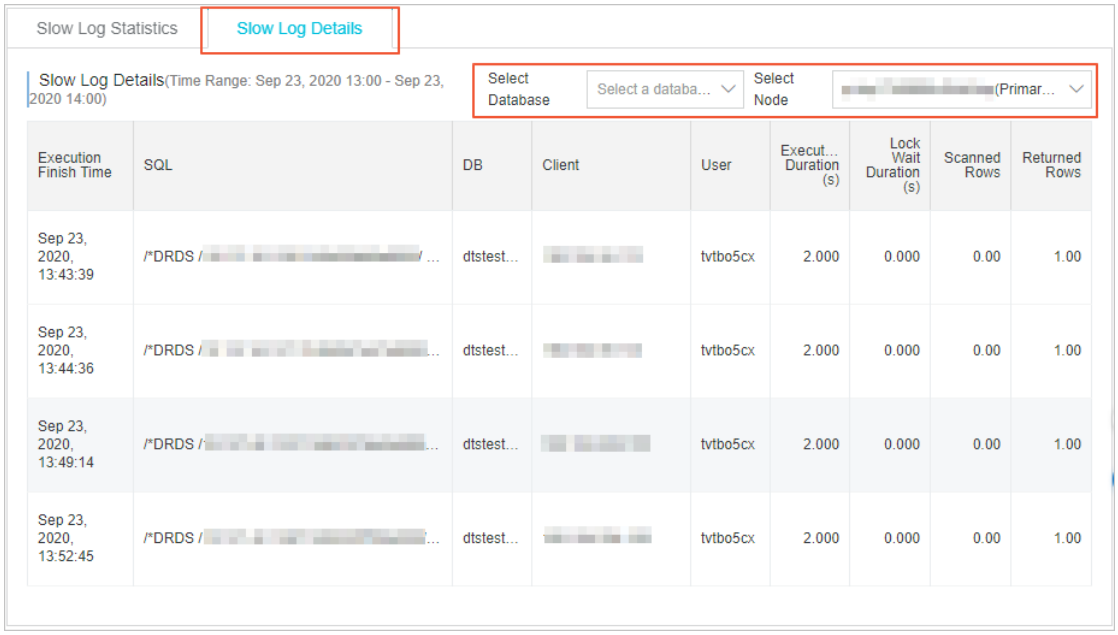
   i. In the left-side navigation pane, click **Databases**. Select the required database.

ii. Click **Storage Management** and click the required cluster. The **Overview** page of MySQL at the storage layer appears.

iii. In the left-side navigation pane, click **Slow SQL Query**.

iv. To query **Slow Log Trend**, **Slow Log Statistics**, and **Slow Log Details** within a specified time range, specify time in the upper-right corner and click **Search**.

v. Click **Slow Log Details** and select the required database and the primary node to view details about slow query logs



Note For more information about slow SQL analysis and diagnostics features, see Slow SQL query.

6.Set parameters

In the PolarDB-X 1.0 console, you can view instance and database parameters and modify parameter values as needed.

Note

- If your instance is read-only, you can set only **Database** parameters, and cannot set **Instance** parameters.
- Some parameters are only supported for instances of specific versions. For more information, see Parameter description.

Parameter description

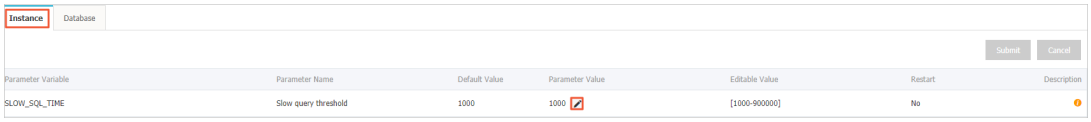
Level	Parameter	Description	Supported version
Instance	SLOW_SQL_TIME	The threshold for slow SQL queries. SQL queries that exceed this threshold are recorded in slow logical SQL logs.	5.3.4-15378085 or later.
	LOGIC_IDLE_TIMEOUT	The timeout period for an idle connection between the client and the PolarDB-X 1.0 instance. Unit: milliseconds.	5.3.4-15378085 or later.
	MAX_ALLOWED_PACKET	The maximum size of a packet between the client and the PolarDB-X 1.0 instance. Unit: bytes.	5.3.4-15378085 or later.
	MAX_POOL_SIZE	The maximum number of connections between a PolarDB-X 1.0 database and a single ApsaraDB for RDS database shard.	All versions.
	MIN_POOL_SIZE	The minimum number of connections between a PolarDB-X 1.0 database and a single ApsaraDB for RDS database shard.	All versions.
	IDLE_TIMEOUT	The timeout period for an idle connection between a PolarDB-X 1.0 database and an ApsaraDB for RDS database shard.	All versions.

Level	Parameter	Description	Supported version
Database	SOCKET_TIMEOUT	The timeout period for an SQL query sent from PolarDB-X 1.0 database to an ApsaraDB for RDS database.	All versions.
	FORBID_EXECUTE_DML_ALL	Specifies whether to prohibit full table deletion and update.	All versions.
	TEMP_TABLE_MAX_ROWS	The size of an intermediate temporary table used during distributed queries in PolarDB-X 1.0.	5.3.4-15378085 or later.
	MERGE_UNION_SIZE	The maximum number of table shards that can be combined by executing the UNION statement when you query multiple table shards in a database.	5.3.4-15378085 or later.
	PURE_ASYNC_DDL_MODE	Specifies whether to enable the pure asynchronous-mode when the asynchronous data definition language (DDL) engine is enabled. • When this mode is enabled, the client connects to the PolarDB-X 1.0 instance and immediately returns when a DDL statement is executed. You can use DDL management statements to view the execution status. • If this mode is disabled, the synchronous mode remains. The client does not return until the execution of a DDL statement is completed.	5.3.4-15378085 or later.
	ENABLE_RANDOM_PHY_TABLE_NAME	Specifies whether a random name is used for a physical table.	5.4.4 or later.

Procedure

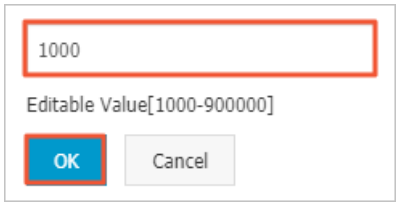
1. Log on to the PolarDB-X 1.0 console.

2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, click **Instances**.
4. Find the target instance and click its ID.
5. In the left-side navigation pane, choose **Diagnostics and Optimization > Parameter Settings**. Then, you can click the **Instance** or **Database** tab to modify the values of corresponding parameters.
   - Modify the values of **Instance** parameters
     a. On the **Instance** tab, find the target parameter, and click in the **Parameter Value** column.



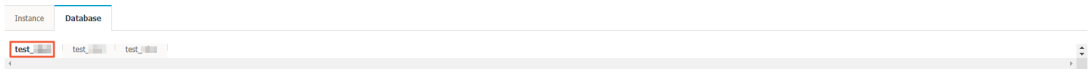
> **Note** You can click in the **Description** column to view the description of the target parameter.

     b. In the dialog box that appears, enter a value and click **OK**.

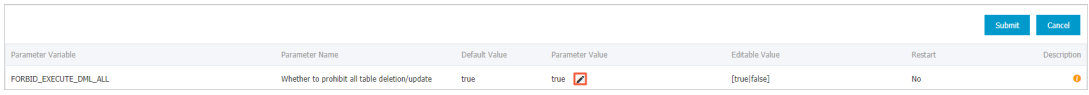


> **Note** The value you entered must be in the **Valid Values** range. If the value is out of the range, when you click **Submit**, an error message appears.

   - Modify the values of **Database** parameters
     a. Click the **Database** tab, find the target database, and then click the target database tab.

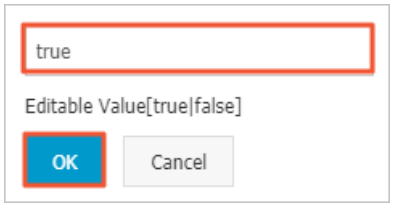


     b. Find the target parameter, and click in the **Parameter Value** column.



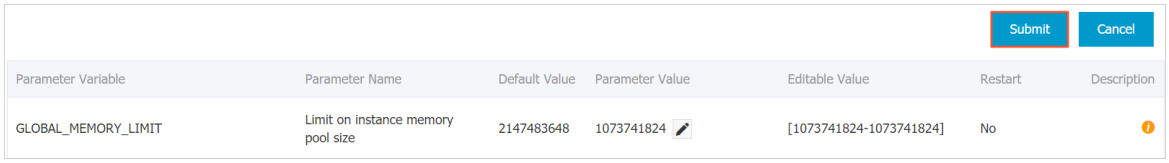
> **Note** You can click in the **Description** column to view the description of the target parameter.

c. In the dialog box that appears, enter a value and click **OK**.



> **Note** The value you entered must be in the **Valid Values** range. If the value is out of the range, when you click **Submit**, an error message appears.

6. After you modify a **Parameter Value**, click **Submit**.



> **Note** For parameters whose **Restart** column is **Yes**, after you click **Submit** to submit new values of **Parameter Value**, you must restart the instance to make the new values take effect.

7.Restore data

7.1. Backup and recovery

This topic will introduce two ways and methods of backup and recovery.

Overview

PolarDB-X 1.0 supports instance- and database-level backup and restoration. PolarDB-X 1.0 instances support fast backup and consistent backup in automatic and manual modes. Instance-level restoration refers to restoring data to new PolarDB-X 1.0 and ApsaraDB for RDS (RDS) instances based on existing data backup sets.

Backup methods

PolarDB-X 1.0 provides fast backup and consistent backup. These methods help you restore data based on your business scenarios. The following table compares the two backup methods.

Backup mode	Advantages	Disadvantages	Scenarios
Fast backup	• Provides fast backup and restoration. • Supports restoration based on backup sets and restoration to a specified time point. • Supports all instance versions.	Ensures data consistency only within a single RDS instance in sharding scenarios, but does not ensure global data consistency.	Applicable to backup and restoration for routine business.

Backup mode	Advantages	Disadvantages	Scenarios
Consistent backup	Ensures global data consistency in sharding scenarios.	• Features slow backup and restoration. • Supports only restoration based on backup sets, but does not support restoration to a specified time point. • Supports only 5.3.8 and later versions. • Locks distributed transactions on PolarDB-X 1.0 instances for seconds during backup. During the locking process, the response time (RT) for executing SQL statements includes sub-second jitters. We recommend that you perform consistent backup during off-peak hours.	Applicable to backup and restoration for the financial industry and core online transactions that require high data consistency.

Limits and notes

- The automatic backup policy of PolarDB-X 1.0 is disabled by default. You must enable it manually. For more information about how to enable it, see Set an automatic backup policy.
- The log backup feature of PolarDB-X 1.0 depends on underlying RDS instances. The log backup policy configured in the PolarDB-X 1.0 console is automatically synchronized to all the underlying RDS instances. Do not modify the configured policy in the RDS console so that the backup sets keep effective.
- The backup and restoration feature of PolarDB-X 1.0 depends on log backup. We recommend that you enable the log backup policy by default so that the backup sets keep effective.
- Do not perform Data Definition Language (DDL) operations during backup, which may lead to instance-level backup and restoration failure.
- Make sure the RDS instances mounted to a PolarDB-X 1.0 instance are normal during backup so that the backup can succeed.
- Make sure all tables have primary keys to keep data accuracy during consistent backup and restoration.
- Affected by the changes to the PolarDB-X 1.0 and RDS inventory, PolarDB-X 1.0 automatically adjusts the instance type and zone during instance-level restoration. You can confirm and change the instance type and zone after the restoration is completed to avoid affecting your business.
- Instance-level restoration is billed based on the actual billing method and instance type. For more information about the billing method, see PolarDB-X 1.0 specifications and pricing and ApsaraDB for RDS specifications and pricing.

Set an automatic backup policy

1. Log on to the PolarDB-X 1.0 console and select a region.
2. Click Instances in the left-side navigation pane and click an instance name on the page that appears.
3. In the left-side navigation pane, choose **Data Recovery > Backup and Recovery.**.
4. Click the **Backup Policy** Tab to view the current automatic backup policy. The backup policy of PolarDB-X 1.0 is disabled by default. If this is the first time you access this page, the backup policy takes effect after you set it.
5. Click **Edit** to set the automatic backup policy, as shown in the preceding figure. If you select Database for Backup Level, you must select a database to be backed up.
6. Set the parameters for the automatic backup policy, and then click **OK**.

> **Note** A PolarDB-X 1.0 instance does not support log backup, and the configured log backup policy will be applied to all underlying RDS instances.

Configure local logs

1. Log on to the PolarDB-X 1.0 console and select a region.
2. Click Instances in the left-side navigation pane and click an instance name on the page that appears.
3. In the left-side navigation pane, choose **Data Recovery > Backup and Recovery.**.
4. Click the **Local Log Settings tab**.
5. Click **Edit**. In the Local Binlog Settings dialog box, modify the local log settings.
6. After you complete the setting, click **OK**.

> **Note** The local log settings will be applied to all underlying RDS instances.

Proactive backup

1. Log on to the PolarDB-X 1.0 console and select a region.
2. Click Instances in the left-side navigation pane and click an instance name on the page that appears.
3. In the left-side navigation pane, choose **Data Recovery > Backup and Recovery.**.
4. Click **Data Backup**.
5. Set Backup Method and Backup Level. You can select Consistent Backup or Fast Backup for Backup Method. For more information about the differences between the two backup methods, see
6. Click **OK** to submit the backup task.
7. Click the **Data Backup Set** tab to view the available backup sets. You can view the data backup progress in the Task Progress list in the upper-right corner of the page.

Restore data

1. Log on to the PolarDB-X 1.0 console and select a region.
2. Click Instances in the left-side navigation pane and click an instance name on the page that

appears.

3. In the left-side navigation pane, choose **Data Recovery > Backup and Recovery.**.
4. In the upper right corner, click **Data Recovery(Original Clone Instance)**.
5. In the dialog box that pops up, select the recovery method and configure other recovery parameters.
   - If you want to restore data by backup set, choose the method as By Backup Set and select a valid backup set in the backup set list.
   - If you want to restore data to a specified time point, choose the method as By Time and select a specified time and recovery level.
6. Click **Precheck** to check whether a valid backup set is available for data restoration. If the precheck fails, the data cannot be restored.
7. When the precheck success, in the Verification for Recovery page, click **Enable**.
8. When the precheck success, in the Verification for Recovery page, click **Enable** to start data restoration.
9. You can view the restoration progress in the Task Progress list in the upper-right corner of the page. After the data is restored, you can view the restored instance on the PolarDB-X 1.0 Instance Management page.

7.2. SQL Flashback

Background

PolarDB-X 1.0 provides SQL flashback on SQL statements for incorrect operations, which supports row-level data restoration.

After you execute the SQL statement for an incorrect operation (such as INSERT, UPDATE, or DELETE) on a PolarDB-X 1.0 instance, you can use SQL flashback based on the SQL statement of the incorrect operation to find the matched event in BINLOG files. Then, you can generate and download the corresponding restoration file, and restore data as required.

SQL flashback supports fuzzy match and exact match to locate lost data and automatically selects a matching policy for SQL statements for incorrect operations. For more information, see Terms.

SQL flashback provides two types of SQL statements for data restoration in different scenarios. You can choose an appropriate SQL statement type for data restoration based on your business scenario.

Advantages

- **Easy operation:** It is easy to configure SQL flashback. To find lost data, you only need to enter a little information about the SQL statements for incorrect operations.
- **Fast speed and light weight:** SQL flashback does not depend on the backup policies of ApsaraDB for RDS (RDS) instances. You can quickly restore data by enabling RDS log backup before executing SQL statements for incorrect operations.
- **Flexible restoration methods:** SQL flashback provides rollback SQL statements and original SQL statements for different scenarios.
- **Exact match at the SQL statement level:** SQL flashback supports exact match of incorrect operation data at the SQL statement level, which can increase the accuracy of restored data.

Limits

- SQL flashback depends on the retention time of RDS BINLOG files. Therefore, you must enable RDS log backup. RDS BINLOG files are only retained for a period of time. We recommend that you use SQL flashback to generate a restoration file as soon as possible after an incorrect operation is executed.
- A restoration file generated by SQL flashback is stored for seven days by default. We recommend that you download the file as soon as possible after it is generated.
- Exact match of SQL flashback requires the following conditions:
  - The PolarDB-X 1.0 instance version is 5.3.4-15378085 or later.
  - The RDS instance version used by PolarDB-X 1.0 is 5.6 or later.
  - Exact match of SQL flashback is enabled before SQL statements for incorrect operations are executed.
  - The TRACE_ID of the SQL statement for an incorrect operation is provided.
- To ensure the accuracy of restored data, PolarDB-X 1.0 enables exact match for databases created on instances of 5.3.4-15378085 and later versions. After exact match is enabled, the information about the SQL statements will be contained in RDS BINLOG files by default. This requires additional RDS storage space. If you need exact match, we recommend that you upgrade your PolarDB-X 1.0 instance to enable exact match. For more information, see Enable exact match.

Generate a restoration file

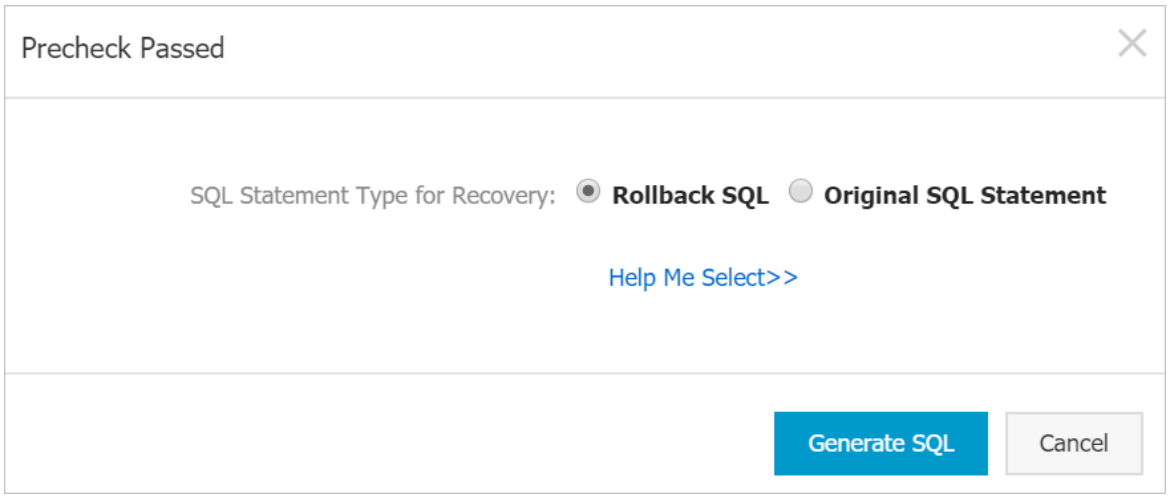
1. Log on to the PolarDB-X 1.0 console.
2. On the left-upper corner, choose the region where the target instance locating.
3. On the **Instances** page, find the target instance and click on its ID.
4. On the left navigation pane, cick on **Data Recovery>SQL flashback**.
5. On the **SQL flashback** page, enter the following settings.

Parameters	Descriptions
Database	Specify the PolarDB-X 1.0 database name.
Table name	Specify the table name on which the mistaken SQL statement is executed. This field is optional.
Time Range	Specify the approximate time range for the execution of the mistaken SQL statement. The beginning of the time range is earlier than the time when the SQL statement execution starts. The end of the time range is later than the time when the SQL execution ends. To ensure efficient restoration, we recommend that you set the time range as precisely as possible, it would be better if you can limit the time to 5 minutes.
TRACE_ID	PolarDB-X 1.0 allocates a unique TRACE_ID for each SQL statement. This field is optional.
SQL Statement Type	Choose one or several types from INSERT, UPDATE, DELETEE.

6. Click **Precheck**.

- If no RDS BINLOG file exists within the specified time range, the precheck would fail and the data cannot be restored.
- If RDS BINLOG files exist within the specified time range, the precheck would pass.

7. In the dialog box that pops up, you can choose the **SQL Statement Type for Recovery** as **Rollback SQL** or **Original SQL Statement**.
8. Click **Generate SQL**.



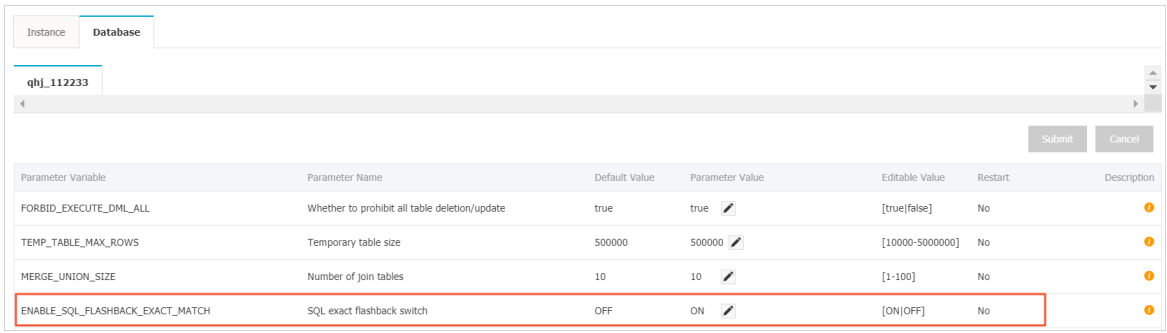
9. When the SQL falshback task finished, you can check if this task is exact match or not and view the numbers of recovered rows.

   On the **Action** column, you can click **Download** to download the corresponding restoration file for data restoration. The restoration file is retained for 7 days by default. If you need, please download it as soon as possible.

Enable exact match

Prerequisite: Your PolarDB-X 1.0 instance version must be 5.3.4-15378085 or later. If your PolarDB-X 1.0 instance version is earlier than the required version, we recommend that you upgrade it first.

1. Log on to the PolarDB-X 1.0 console and go to the Parameter Settings page of the specified instance. For more information, see Set parameters.
2. Set Exact Match for SQL flashback of the corresponding database to ON.



Terms

Exact match and fuzzy match

SQL flashback searches RDS BINLOG files for a matched event based on the information of the SQL statement for an incorrect operation, which you need to provide, and then generates a corresponding restoration SQL statement. Affected by the instance version and parameter settings, SQL flashback supports two matching policies for BINLOG events: exact match and fuzzy match. You do not need to select a specific matching policy. The SQL flashback feature of PolarDB-X 1.0 automatically detects and selects the optimal matching policy, and notifies you of the policy after the flashback task is completed.

Exact match

The SQL flashback feature searches the event that exactly matches the SQL statement for an incorrect operation in RDS BINLOG files to generate a restoration file.

Advantage: The restoration file only contains data incorrectly deleted or modified due to the execution of a SQL statement, which you can directly use to ensure the accuracy and efficiency of data restoration.

Disadvantage: Exact match requires the following conditions:

- The PolarDB-X 1.0 instance version must be 5.3.4-15378085 or later.
- The RDS instance version used by PolarDB-X 1.0 is 5.6 or later.
- Exact match of SQL flashback is enabled before SQL statements for incorrect operations are executed. After exact match of SQL flashback is enabled, the information about the SQL statements will be contained in RDS BINLOG files by default, which requires additional RDS storage space.
- The TRACE_ID of the SQL statement for an incorrect operation is provided.

> **Note** Exact match of SQL flashback is enabled by default for new databases on PolarDB-X 1.0 instances of 5.3.4-15378085 and later versions. We recommend that you upgrade instances of earlier versions and then manually enable exact match. For more information, see Enable exact match.

Fuzzy match

The SQL flashback feature searches RDS BINLOG files for matching events based on the information of SQL statements for incorrect operations, such as the time range, table name, and SQL statement type, and then generates a restoration file.

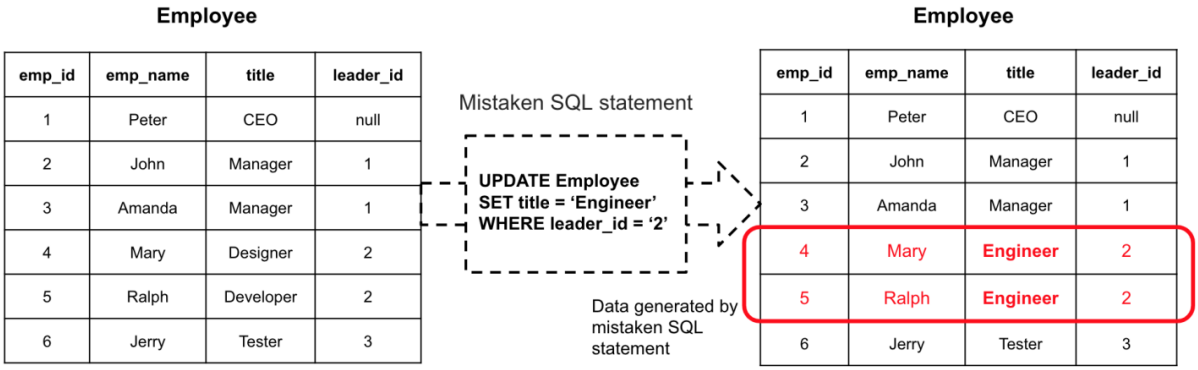
Advantage: Fuzzy match does not depend on the instance version or parameter settings. Therefore, it is supported on all instances.

Disadvantage: Fuzzy match cannot match accurate data that is deleted or modified by mistake. The restoration file contains data modified by other business SQL statements. You need to filter the required data from the restoration file.

Rollback SQL statements and original SQL statements

To support different business scenarios, PolarDB-X 1.0 SQL flashback allows you to generate rollback SQL statements and original SQL statements for data restoration. Before you generate a restoration SQL file, you must select a restoration method based on your business scenario.

The following figure shows how the SQL flashback feature generates a rollback SQL statement and an original SQL statement for an incorrect operation.



SQL statement types for Recovery

Rollback SQL Statement: It is generated after the BINLOG files are traversed for matching events in a reverse order and reverse operations are executed for the INSERT, UPDATE, and DELETE events.	**UPDATE Employee SET title = 'Developer' WHERE emp_id = '5'** **UPDATE Employee SET title = 'Designer' WHERE emp_id = '4'**
Original SQL Statement: It is generated after the BINLOG files are traversed for matching events in a sequential order** and images are constructed for all records of the INSERT, UPDATE, and DELETE events.	**INSERT INTO Employee(emp_id, emp_name, title, leader_id) values ('4', 'Mary', 'Designer', '2')** **INSERT INTO Employee(emp_id, emp_name, title, leader_id) values ('5', 'Ralph', 'Developer', '2')**

Rollback SQL statements

A rollback SQL statement is generated after the BINLOG files are traversed for matching events in a reverse order and reverse operations are executed for the INSERT, UPDATE, and DELETE events.

- The reverse operation of INSERT is equivalent to DELETE.
- The reverse operation of DELETE is equivalent to REPLACE.
- The reverse operation of UPDATE is equivalent to UPDATE SET value= the value before the update.

How to use: Execute the SQL statement in the **original table** to restore data by overwriting existing data.

Scenario: If the data to be restored has no conflict or some conflicted data can be overwritten, we recommend that you generate a rollback SQL statement.

Original SQL statements

An original SQL statement is generated after the BINLOG files are traversed for matching events in a sequential order and images are constructed for all records of the INSERT, UPDATE, and DELETE events.

- The image of INSERT is equivalent to INSERT.
- The image of DELETE is equivalent to INSERT.
- The image of UPDATE is equivalent to the value before INSERT.

How to use: Execute the SQL statement in a **temporary table** to write the original data, compare it with the existing data, analyze the conflict, and determine the final data.

Scenario: If the data to be restored conflicts with the existing data, we recommend that you generate an original SQL statement.

7.3. Table recycle bin

This topic introduces the operation of the table recycle bin.

Background

The table recycle bin feature of PolarDB-X 1.0 allows you to restore table data that is deleted by mistake.

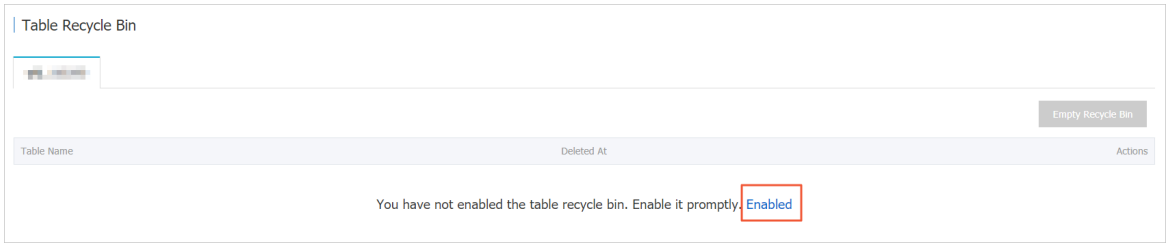
After you enable the table recycle bin feature for PolarDB-X 1.0, the tables that you deleted by using the `DROP TABLE` command are moved to the recycle bin and become invisible to you. After the tables are moved to the recycle bin for two hours, they are automatically cleared and cannot be restored. You can view, restore, and clear deleted tables in the recycle bin.

Limits and notes

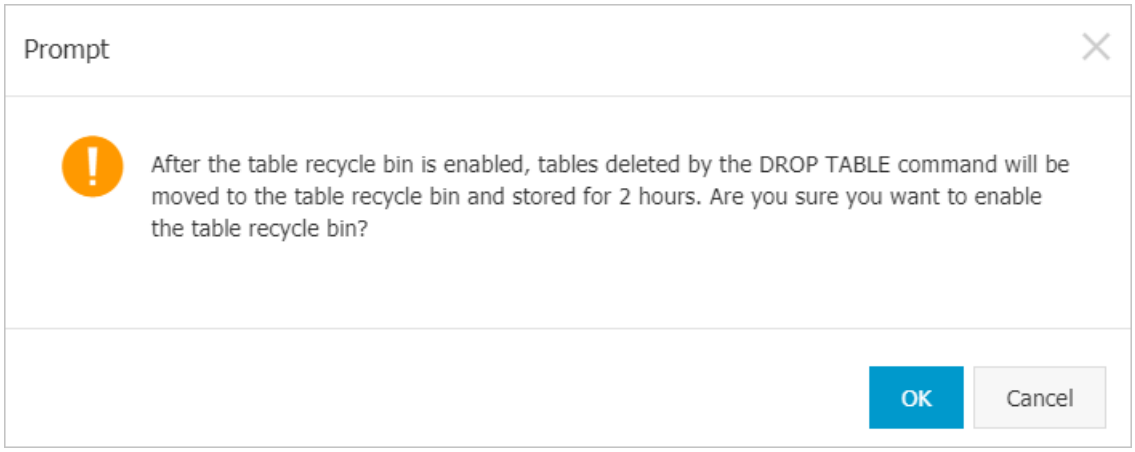
- The table recycle bin feature is only supported on PolarDB-X 1.0 instances of 5.3.3-1670435 and later versions.
- The table recycle bin feature for PolarDB-X 1.0 is disabled by default. For more information about how to enable it, see Enable the table recycle bin feature.
- The table recycle bin feature for PolarDB-X 1.0 is not applicable to tables that you delete by using the `TRUNCATE TABLE` command.
- Tables in the recycle bin still occupy the storage space of ApsaraDB for RDS (RDS) instances before they are automatically cleared. To quickly release the space, you can go to the recycle bin for manual clearing.
- PolarDB-X 1.0 allows you to manage the table recycle bin by using command lines.

Enable the table recycle bin feature

1. Log on to the PolarDB-X 1.0 console and choose the region where the target instance locating.
2. On the **Instances** page, find the target instance and click on its ID.
3. On the left navigation pane, cick **Data Recovery>Table Recycle Bin**.
4. On the **Table Recycle Bin** page, click the target database name tab. Click **Enabled**.

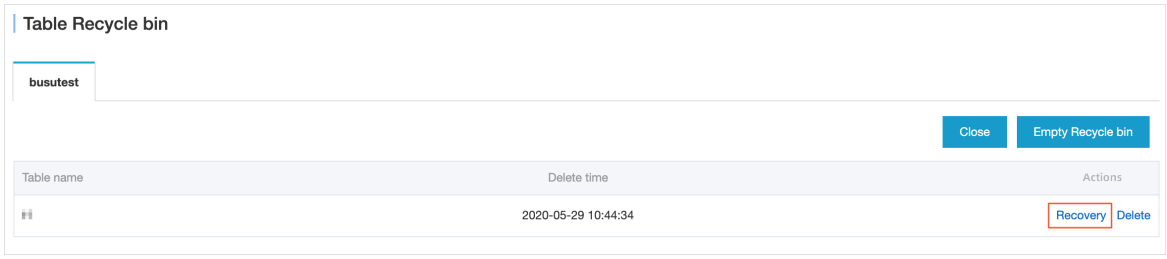


5. In the dialog box that prompts, click **OK**.



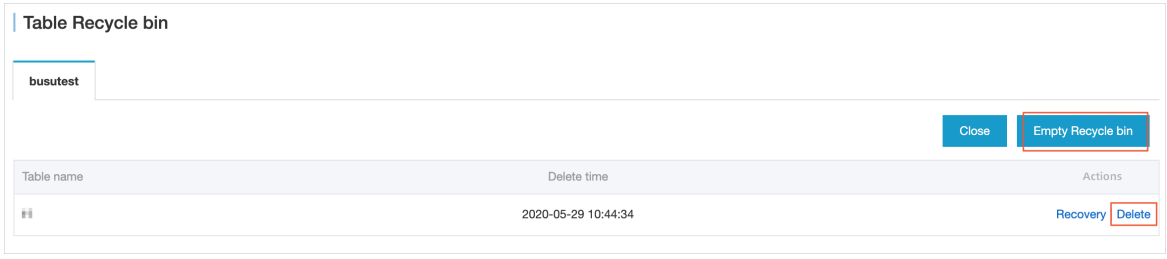
Restore a data table

1. On the **Table Recycle Bin** page, click the target database name tab to view the list of tables.
2. In the table list, find the target table, and click **Recovery** in the **Actions** column.



Clear a data table

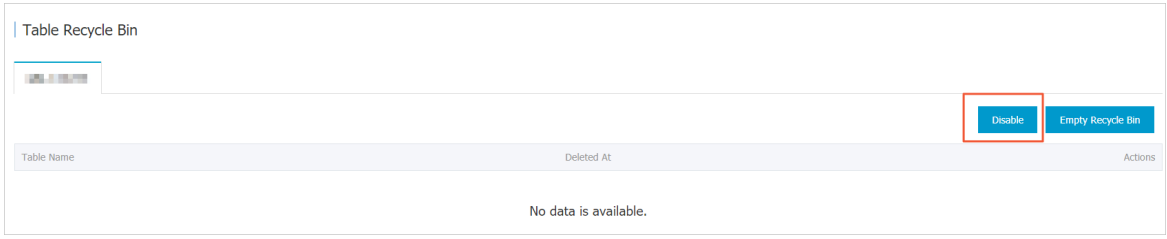
1. On the **Table Recycle Bin** page, click the target database name tab to view the list of tables.
2. If you want to delete a specified table, find the table in table list, and click **Delete** in the **Actions** column.



If you want to clear all data tables from the recycle bin, click **Empty Recycle Bin**.

Disable the table recycle bin feature

1. On the **Table Recycle Bin** page, click the target database name tab.
2. Click **Disable**.



8.Pending events

PolarDB-X 1.0 allows you to manage pending events. If pending events occur, the system uses emails, messages in the console, or SMS messages to notify you to handle the events in a timely manner. This topic describes how to use the console to view and handle pending events.

Context

PolarDB-X 1.0 pending events are divided into the following types:

- Operations and maintenance (O&M) events: For O&M events such as modifications of network settings, you can schedule the operation time. The O&M system automatically performs the operation at the scheduled time.
- Notification events: For notification events such as version upgrades, you must manually handle the events after you receive the corresponding messages.

Procedure

1. Log on to the PolarDB-X 1.0 console.
2. In the left-side navigation pane, click **Pending Events**.

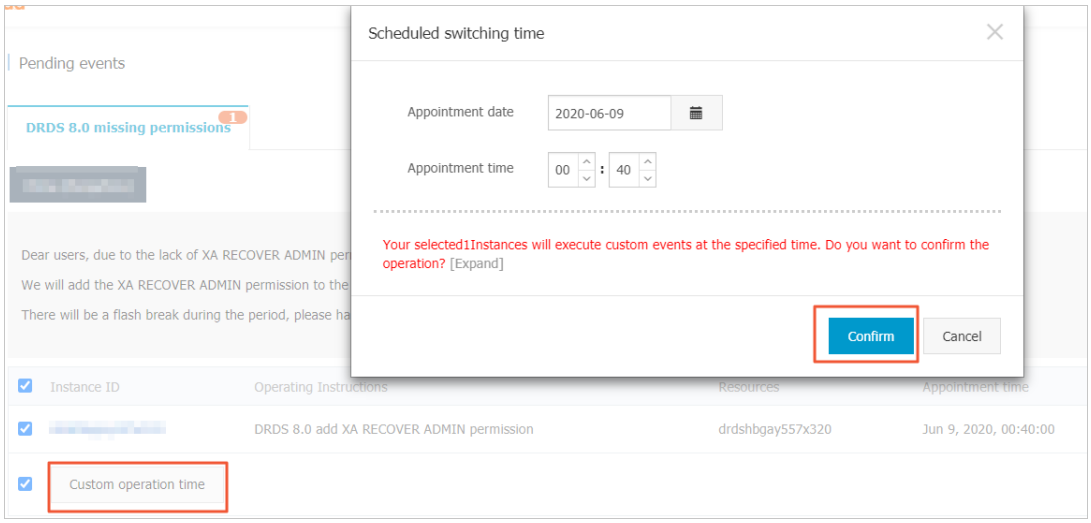
   > **Note** If the **Pending Events** page does not list events, no pending O&M events need to be handled for your instances.

3. On the **Pending Events** page, click the tab based on the event type and select the region where your instances are deployed. In the section that lists the instances, you can view the details of the pending events.
   - If you do not specify the operation time for an O&M event, the system automatically performs the corresponding O&M operation at the default time.

     To change the operation time, perform the following steps:

     a. On the Pending Events page, select the instance for which you want to change the operation time.

     b. Click **Custom Operation Time**. In the dialog box that appears, specify the operation time.

     c. Click **Confirm**.

- You must manually handle notification events. After you handle a notification event, log on to the console to confirm that the event is not displayed on the **Pending Events** page.

  > **Note** To view the events that have been handled, log on to the console. Then, in the left-side navigation pane, click **Historical Events**.

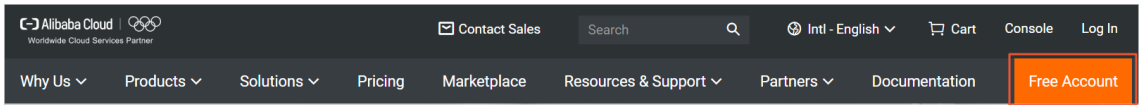
9.Access Control

9.1. Register and log on to an Alibaba Cloud account

This topic describes how to register and log on to an Alibaba Cloud account.

Register an Alibaba Cloud account

1. You can register an Alibaba Cloud account by using one of the following methods:
   - On the Alibaba Cloud International site, click **Free Account** in the upper-right corner.

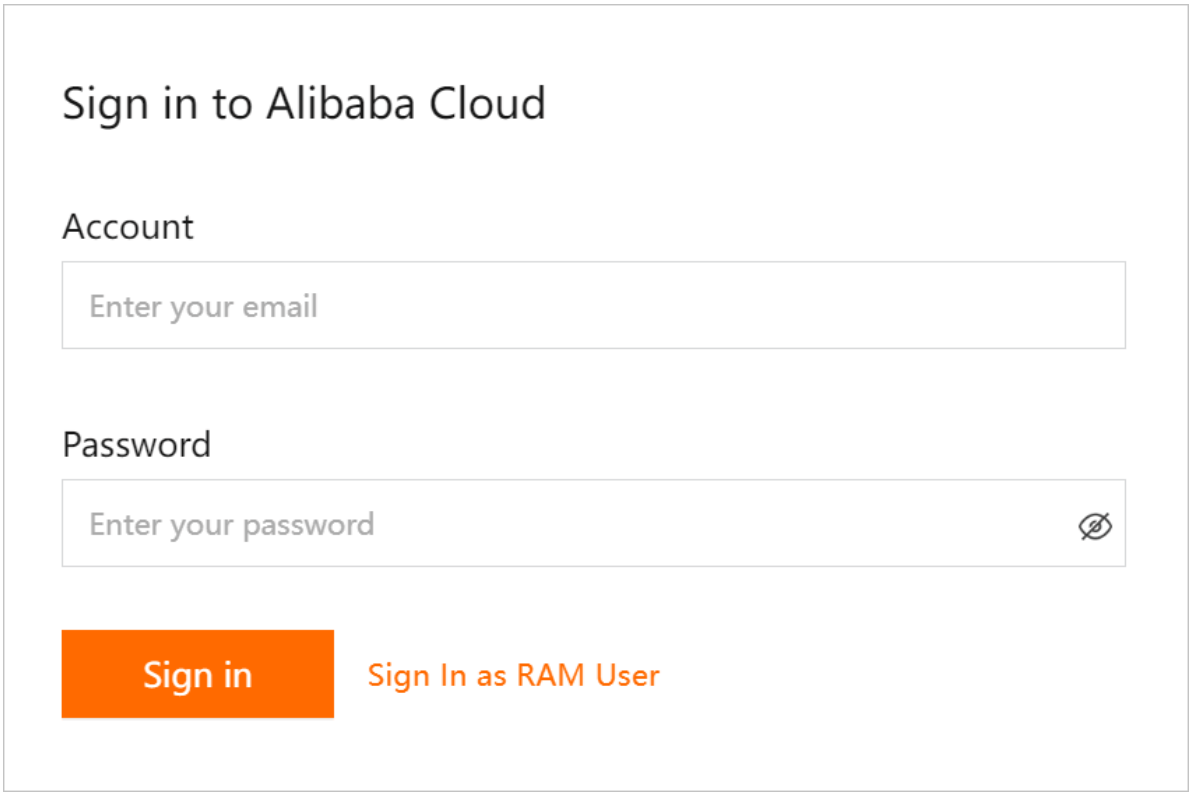


   - Directly go to the Alibaba Cloud account registration page.
2. On the **Create a new Alibaba Cloud account** page, enter your account information, agree to the privacy policy and service agreement, and click Confirm to complete the registration.

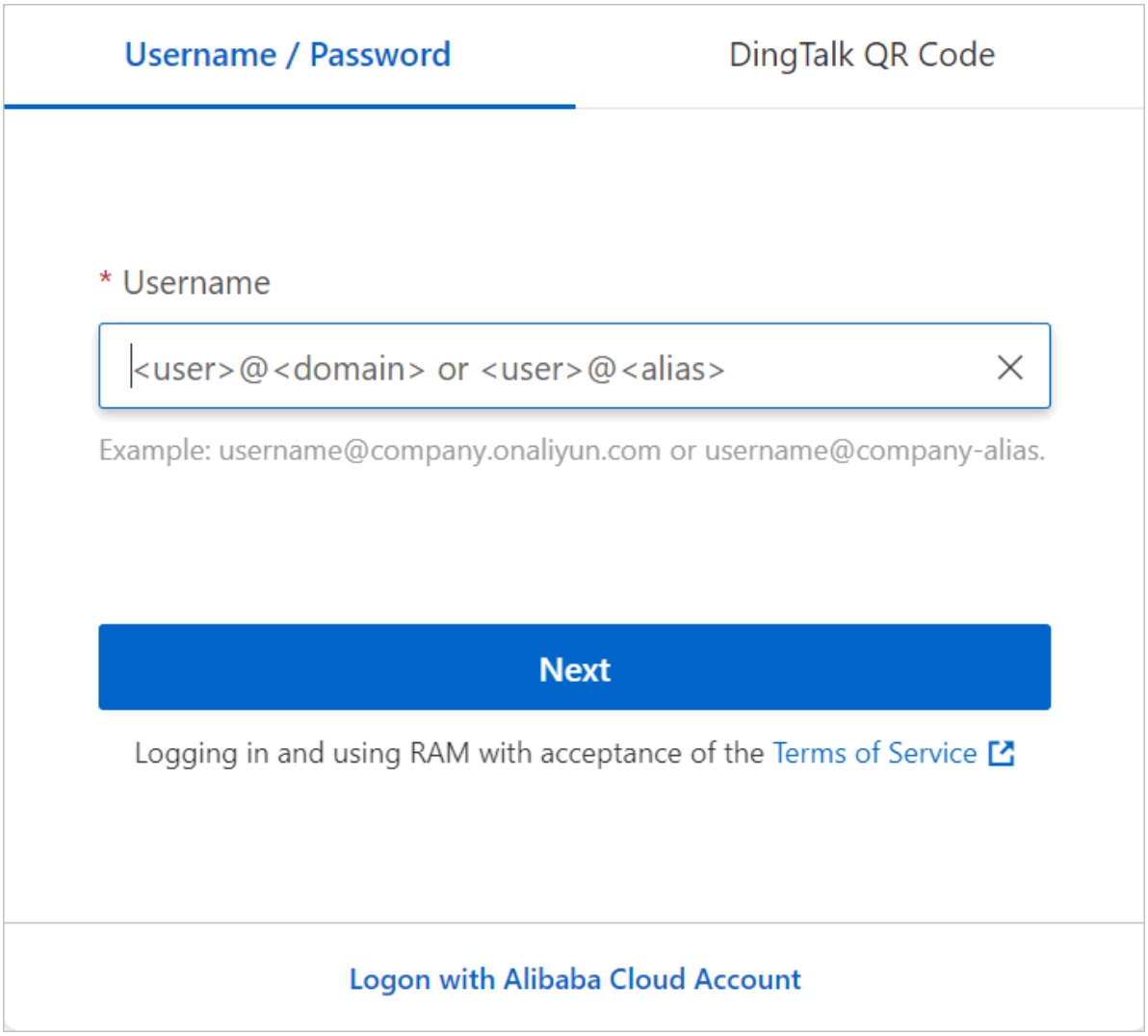
Log on to your Alibaba Cloud account

Your Alibaba Cloud account and Resource Access Management (RAM) user have different logon pages.

- The following figure shows the logon page for an Alibaba Cloud account.



- The following figure shows the logon page for a RAM user.



9.2. Use RAM in

This topic describes the concept of Alibaba Cloud Resource Access Management (RAM) and provides an example of using RAM in PolarDB-X 1.0.

Background

RAM is an Alibaba Cloud service that helps you manage user identities and access to your cloud resources. You can use RAM to create and manage different accounts, such as employee accounts, system accounts, and application accounts. You can also manage the operation permissions that these user accounts have on resources of your Alibaba Cloud account. If multiple users in an enterprise or organization need to use resources in a collaborative manner, RAM can prevent the wide spread of the AccessKey pair of the Alibaba Cloud account and grant users the minimum required permissions. In this way, RAM reduces information security risks. For more information, see What is RAM?.

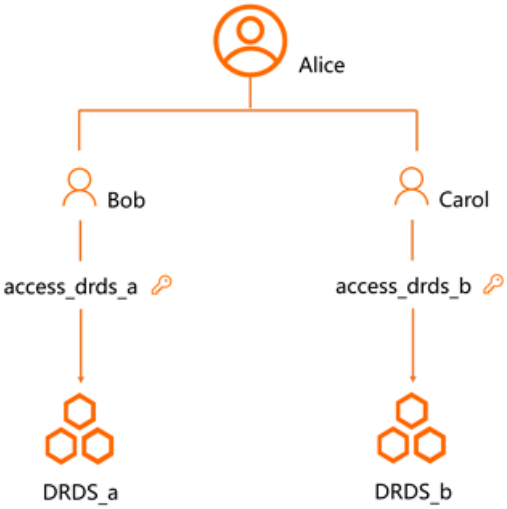
Before you use RAM, you must understand basic concepts, such as the Alibaba Cloud account, RAM user, credential, and RAM role. Understanding these concepts can help you get started with RAM. For more information about RAM, see Terms.

Examples of using RAM in PolarDB-X 1.0

Assume that an Alibaba Cloud user named Alice has two PolarDB-X 1.0 instances: PolarDB-X_a and PolarDB-X_b. Alice has full permissions on both instances.

- To keep the security of the AccessKey pair of the Alibaba Cloud account, Alice uses RAM to create two RAM users: Bob and Carol.
- Alice creates the access_drds_a and access_drds_b permission policies, which represent the read and write permissions on PolarDB-X_a and PolarDB-X_b, respectively.
- Alice separately authorizes Bob and Carol in the console, so that Bob has read and write permissions on PolarDB-X_a and Carol has read and write permissions on PolarDB-X_b.

Bob and Carol have their own AccessKey pairs. If a RAM user's AccessKey pair is leaked, only one of the PolarDB-X 1.0 instances is affected. In the console, Alice can also timely revoke the permission of the RAM user whose AccessKey pair has been leaked.



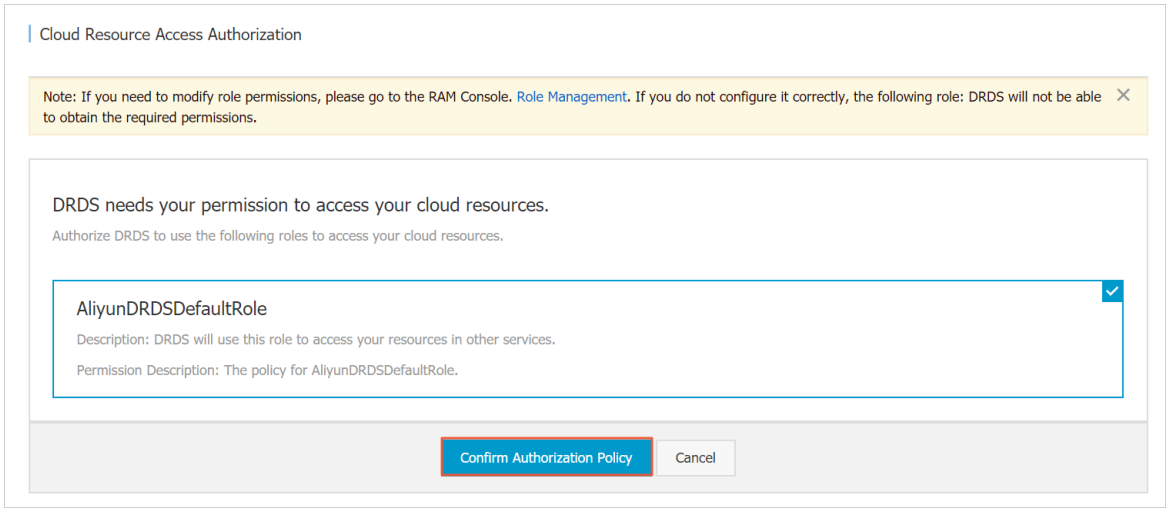
9.3. Activate authorization for to access ApsaraDB for RDS

When you perform specific operations in PolarDB-X 1.0, API operations of ApsaraDB for RDS are called. Therefore, before you use Resource Access Management (RAM), you must activate the authorization for PolarDB-X 1.0 to access ApsaraDB for RDS and create a RAM role that PolarDB-X 1.0 can use to access ApsaraDB for RDS. This topic describes how to activate authorization in the console and by calling API operations of RAM.

Activate RAM authorization in the console

1. Log on to the PolarDB-X 1.0 console.
2. In the left-side navigation pane, click **Resource Authorization**.
3. On the **Cloud Resource Authorization Management** page, click **Activate Authorization**.

4. In the dialog box that appears, click **Confirm Authorization Policy**.



Warning When you activate authorization, a RAM role named **AliyunDRDSDefaultRole** is created by default in the RAM console. PolarDB-X 1.0 uses this RAM role to access the ApsaraDB RDS for MySQL resources of the corresponding account. PolarDB-X 1.0 is also authorized to access API operations of ApsaraDB for RDS. Exercise caution when you delete this RAM role.

Activate RAM authorization by calling API operations of RAM

If you cannot access the console temporarily, you can call the related API operation to create a RAM role. After the RAM role is created, the authorization for PolarDB-X 1.0 to access ApsaraDB RDS for MySQL resources is also activated.

- Create a RAM role named Jack.

- Call the CreateRole operation and specify the following parameters to create a RAM role.

Parameter	Description
Action	CreateRole
RoleName	Jack
AssumeRolePolicyDocument	The trust policy, in the following format:

```
{
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "drds.aliyuncs.com"
        ]
      }
    }
  ],
  "Version": "1"
}
```

- Use the SDK for Java to create a RAM role. The following code is a demo:

```
// Create a role.
String rolePolicyDoc = "{\"Statement\":[{\"Action\": \"sts:AssumeRole\", \"Effect\": \"
Allow\",\"Principal\":{\"Service\":[\"drds.aliyuncs.com\"]}}],\"Version\": \"1\"}";
String roleName = "Jack";
CreateRoleRequest request = new CreateRoleRequest();
request.setAssumeRolePolicyDocument(rolePolicyDoc);
request.setAssumeRolePolicyDocument(rolePolicyDoc);
client.getAcsResponse(request);
// Grant permissions to the role.
AttachPolicyToRoleRequest attachRequest = new AttachPolicyToRoleRequest();
attachRequest.setPolicyType("System");
attachRequest.setPolicyName("AliyunDRDSRolePolicy");
attachRequest.setRoleName("AliyunDRDSDefaultRole");
client.getAcsResponse(attachRequest);
```

- Check whether the RAM role named Jack is created.
  - Call the GetRole operation and specify the following parameters to query the details of the RAM role.

Parameter	Description
Action	CreateRole
RoleName	Jack

- Use the SDK for Java to query the details of the RAM role. The following code is a demo:

```
String roleName = "AliyunDRDSDefaultRole";
GetRoleRequest request = new GetRoleRequest();
request.setRoleName(roleName);
GetRoleResponse resp = client.getAcsResponse(request);
GetRoleResponse.Role role = resp.getRole();
```

9.4. Use RAM in the PolarDB-X 1.0 console

This topic describes how to use the Resource Access Management (RAM) account system and permission policies to control resources and permissions in PolarDB-X 1.0.

Limits

- Multi-factor authentication (MFA) must be enabled for RAM users to delete databases and read-only accounts. For more information, see Policy structure and syntax.
- RAM users do not have the permission to change passwords of PolarDB-X 1.0 databases.

Use RAM in the PolarDB-X 1.0 console

To use RAM in the PolarDB-X 1.0 console, you must complete the following operations in the RAM console:

1. Create a RAM user, see Create a RAM user.
2. Create an authorization policy, see Create a custom policy.
3. Grant permissions to the RAM user, see Grant permissions to a RAM user.

   Before you use RAM in DRDS, make sure you have authorized DRDS to access ApsaraDB for RDS (RDS) and created a RAM role for DRDS. For more information, see Policy structure and syntax.

Samples of creating a custom policy

- Grant a RAM user all the PolarDB-X 1.0 console operation permissions of the corresponding Alibaba Cloud account.

```
{
 "Version": "1",
 "Statement": [
     {
         "//": "1234 indicates the UID of the Alibaba Cloud account of the RAM user."
         "Action": "drds:*",
         "Resource": "acs:drds:*:1234:instance/*",
         "Effect": "Allow"
     },
     {
         "//": "Note: To guarantee normal usage of RAM, ensure that the policy includes t
he following information:"
         "Action": "ram:PassRole",
         "Resource": "*",
         "Effect": "Allow"
     }
 ]
}
```

- Grant users the permission to access all PolarDB-X 1.0 instances only in the zones of China (Hong Kong).

```
{
 "Version": "1",
 "Statement": [
     {
         "//": "1234 indicates the UID of the Alibaba Cloud account of the RAM user."
         "Action": "drds:*",
         "Resource": "acs:drds:cn-hongkong:1234:instance/*",
         "Effect": "Allow"
     },
     {
         "//": "Note: To guarantee normal usage of RAM, ensure that the policy includes t
he following information:"
         "Action": "ram:PassRole",
         "Resource": "*",
         "Effect": "Allow"
     }
 ]
}
```

- Forbid users to access a specific instance. The RAM user with this policy can access all PolarDB-X 1.0 instances excluding `drds******hb4`.

```
{
 "Version": "1",
 "Statement": [
     {
         "//": "1234 indicates the UID of the Alibaba Cloud account of the RAM user."
         "Action": "drds:*",
         "Resource": "acs:drds:*:1234:instance/*",
         "Effect": "Allow"
     },
     {
         "Action": "drds:*",
         "Resource": [
             "acs:drds:*:1234:instance/drds******hb4",
             "acs:drds:*:1234:instance/drds******hb4/*"
         ],
         "Effect": "Deny"
     },
     {
         "//": "Note: To guarantee normal usage of RAM, ensure that the policy includes t
he following information:"
         "Action": "ram:PassRole",
         "Resource": "*",
         "Effect": "Allow"
     }
 ]
}
```

9.5. Use RAM for resource authorization

This topic summarizes the resource authorization rules for Resource Access Management (RAM) supported by PolarDB-X 1.0 and the regions where RAM services are activated for PolarDB-X 1.0.

Supported resource authorization rules

Note

- You must replace `$regionId` in the table with the corresponding regionId value in Regions that support RAM based on the region where the specified instance resides.
- You must replace the parameters that start with `$` in the table with the actual parameters based on your business requirements.

Action	Authorization rule	Description
CreateDrdsInstance	acs:drds:$regionid:$accountid:instance/*	Creates an instance.
DescribeDrdsInstanceList	acs:drds:$regionid:$accountid:instance/*	Queries the instance list.
UpgradeDrdsInstance	acs:drds:$regionid:$accountid:instance/$instanceid	Changes the instance configurations.

Action	Authorization rule	Description
RemoveDRDSInstance	acs:drds:$regionid:$accountid:instance/$instanceid	Releases the instance.
DescribeDrdsInstance	acs:drds:$regionid:$accountid:instance/$instanceid	Queries the instance details.
VersionChanage	acs:drds:$regionid:$accountid:instance/$instanceid	Upgrades or rolls back the version of the PolarDB-X 1.0 instance.
CreateInternetAddress	acs:drds:$regionid:$accountid:instance/$instanceid	Creates a public IP address for the PolarDB-X 1.0 instance.
ReleaseInternetAddress	acs:drds:$regionid:$accountid:instance/$instanceid	Releases the public IP address for the PolarDB-X 1.0 instance.
CreateDrdsDB	acs:drds:$regionid:$accountid:instance/$instanceid/db/*	Creates a PolarDB-X 1.0 database.
DescribeDrdsDbList	acs:drds:$regionid:$accountid:instance/$instanceid/db/*	Queries the database list in the PolarDB-X 1.0 instance.
DescribeDrdsDb	acs:drds:$regionid:$accountid:instance/$instanceid/db/$dbname	Queries the details about the PolarDB-X 1.0 database.
DeleteDrdsDb	acs:drds:$regionid:$accountid:instance/$instanceid/db/$dbname	Deletes the PolarDB-X 1.0 database.
ModifyReadWriteWeight	acs:drds:$regionid:$accountid:instance/$instanceid/db/$dbname	Modifies the read policy.
DescribeLogicTableList	acs:drds:$regionid:$accountid:instance/$instanceid/db/$dbname	Queries the data table list in the PolarDB-X 1.0 database.
ExecuteDDL	acs:drds:$regionid:$accountid:instance/$instanceid/db/$dbname	Executes the data definition language (DDL) statement in the PolarDB-X 1.0 console.
ModifyDrdsIpWhiteList	acs:drds:$regionid:$accountid:instance/$instanceid/db/$dbname	Modifies the IP address whitelist of the PolarDB-X 1.0 database.
DrdsDataImport	acs:drds:$regionid:$accountid:instance/$instanceid/db/$dbname	Imports data.

Action	Authorization rule	Description
DrdsSmoothExpand	acs:drds:$regionid:$accountid:instance/$instanceid/db/$dbname	Performs smooth scale-out.
CreateReadOnlyAccount	acs:drds:$regionid:$accountid:instance/$instanceid/db/$dbname	Creates a read-only account.
ModifyReadOnlyAccountPassword	acs:drds:$regionid:$accountid:instance/$instanceid/db/$dbname	Changes the password of the read-only account.
RemoveReadOnlyAccount	acs:drds:$regionid:$accountid:instance/$instanceid/db/$dbname	Deletes the read-only account.
DescribeAlarmContacts	acs:drds:$regionid:$accountid:contacts/*	Queries the list of alert contacts.
AddAlarmContacts	acs:drds:$regionid:$accountid:contacts/*	Adds an alert contact.
ModifyAlarmContacts	acs:drds:$regionid:$accountid:contacts/*	Modifies the alert contact.
RemoveAlarmContacts	acs:drds:$regionid:$accountid:contacts/*	Deletes the alert contact.
DescribeAlarmGroup	acs:drds:$regionid:$accountid:contacts/*	Queries the list of alert contact groups.
AddAlarmGroup	acs:drds:$regionid:$accountid:contacts/*	Adds an alert contact group.
ModifyAlarmGroup	acs:drds:$regionid:$accountid:contacts/*	Modifies the alert contact group.
RemoveAlarmGroup	acs:drds:$regionid:$accountid:contacts/*	Deletes the alert contact group.
DescribeInstanceMonitor	acs:drds:$regionid:$accountid:instance/$instanceid	Queries the monitoring information about the instance.
DescribeAlarmRule	acs:drds:$regionid:$accountid:instance/$instanceid	Queries the list of alert rules.
CreateAlarmRule	acs:drds:$regionid:$accountid:instance/$instanceid	Creates an alert rule. **Note** The permission to create alert rules depends on the permission to query contact groups.

Action	Authorization rule	Description
ModifyAlarmRule	acs:drds:$regionid:$accountid:instance/$instanceid	Modifies an alert rule. **Note** The permission to modify alert rules depends on the permission to query contact groups.
RemoveAlarmRule	acs:drds:$regionid:$accountid:instance/$instanceid	Deletes the alert rule.
DescribeAlarmHistory	acs:drds:$regionid:$accountid:instance/$instanceid	Queries alert history.
DescribeSlowSql	acs:drds:$regionid:$accountid:instance/$instanceid	Queries the slow SQL statement in PolarDB-X 1.0.
DrdsShardTool	acs:drds:$regionid:$accountid:instance/$instanceid/db/$dbname	Uses the shard change tool.

Regions that support RAM

The following table lists the regions where RAM services are activated for PolarDB-X 1.0 and the corresponding regionId values.

regionId	Region name
cn-hangzhou	China (Hangzhou)
cn-shenzhen	China (Shenzhen)
cn-shanghai	China (Shanghai)
cn-qingdao	China (Qingdao)
cn-beijing	China (Beijing)

10.Distributed transactions

10.1. Distributed transactions based on MySQL 5.7

The distributed transactions that are described in this topic are intended for users who use MySQL 5.7 and PolarDB-X 1.0 V5.3.4 and later.

Overview

If the version of the MySQL engine in your custom ApsaraDB RDS for MySQL instance is 5.7 or later, PolarDB-X 1.0 processes distributed transactions based on the XA protocol by default.

For more information about how to use distributed transactions on other versions of custom ApsaraDB RDS for MySQL instances in PolarDB-X 1.0, see Distributed transactions based on MySQL 5.6.

Usage

The user experience of distributed transactions on PolarDB-X 1.0 is the same as that on standalone databases that run the MySQL engine. Examples:

- You can run `SET AUTOCOMMIT=0` to start a transaction.
- You can run `COMMIT` to commit a transaction.
- You can run `ROLLBACK` to roll back a transaction.

If SQL statements in a transaction involve only a single shard, PolarDB-X 1.0 treats the transaction as a non-distributed transaction. In this case, the system directly routes the transaction to a custom ApsaraDB RDS for MySQL instance. If SQL statements in a transaction are used to modify data in multiple shards, PolarDB-X 1.0 automatically upgrades the transaction to a distributed transaction.

FAQ

Q: Do I need to enable distributed transactions in the console?

A: No. You can use distributed transactions in the same way as you use them on standalone MySQL databases as long as your MySQL version and PolarDB-X 1.0 version meet the requirements.

Q: Which versions of MySQL and PolarDB-X 1.0 are required to use distributed transactions?

A: To use the latest distributed transactions, MySQL 5.7 and later and PolarDB-X 1.0 V5.3.4 and later are required.PolarDB-X 1.0 If the MySQL version and PolarDB-X 1.0 version does not meet this requirement, we recommend you to upgrade your PolarDB-X 1.0 instances and custom ApsaraDB RDS for MySQL instances to the latest versions. For more information about how to use distributed transactions on other versions of custom ApsaraDB RDS for MySQL instances and PolarDB-X 1.0 instances, see Distributed transactions based on MySQL 5.6.

10.2. Distributed transactions based on MySQL 5.6

The distributed transactions described in this topic are intended for MySQL 5.6 or Distributed Relational Database Service (DRDS) of a version earlier than 5.3.4.

Context

The XA protocol for MySQL 5.6 is not mature. Therefore, DRDS independently implements 2-Phase Commit (2PC) transaction policies for distributed transactions. When you use MySQL 5.7 or later versions, we recommend that you use XA transaction policies.

How to use

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. You can process the transaction as a single-database transaction in MySQL. No additional operations are required.

Perform the following steps to enable distributed transactions:

After transactions are enabled, execute `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'; -- Recommended for MySQL 5.6 users.
.... -- Here, you can run your business SQL statement.
COMMIT; -- You can use ROLLBACK instead.
```

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 execute your business SQL statement.
conn.commit(); // You can use rollback() instead.
```

FAQ

Q: How can I use DRDS distributed transactions in the Spring framework?

A: If you use the `@Transactional` annotation of Spring to enable transactions, you can enable DRDS 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 def
inition) throws SQLException {
        try (Statement stmt = con.createStatement()) {
            stmt.executeUpdate("SET drds_transaction_policy = '2PC'"); // In this case, a 2
PC transaction is used as an example.
        }
    }
}
```

After that, instantiate the preceding class in the Spring configuration. You can write the code as the following example:

```
<bean id="drdsTransactionManager" class="my.app.DrdsTransactionManager">
     <property name="dataSource" ref="yourDataSource" />
</bean>
```

For classes that require DRDS distributed transactions to be enabled, you can add the `@Transactional("drdsTransactionManager")` annotation.

11.Import and Export Data

11.1. Overview of data migration or synchronization solutions

This topic summarizes the data migration or synchronization solutions that are supported by PolarDB-X 1.0.

> **Note** In the following table, √ indicates that the solution is supported. × indicates that the solution is not supported.

Solution	Data volume	Full migration	Incremental migration
DTS • Synchronize data between PolarDB-X 1.0 instances in real time • Synchronize data from a PolarDB-X 1.0 instance to an ApsaraDB RDS for MySQL instance **Note** You can use Data Transmission Service (DTS) to synchronize data from or to only the PolarDB-X 1.0 databases that are created based on ApsaraDB RDS for MySQL.	Large	√	√
	Large	×	√
	Large	√	√
	Large	√	√
Assess data import and export. Migrate data from RDS to PolarDB-X 1.0	Large	√	√
	Large	√	√
MySQL native commands • Run the `source` command to import an SQL text file into PolarDB-X 1.0. • Run the `mysqldump` command to export an SQL text file from PolarDB-X 1.0. For more information about the detailed steps, see Import and export data through mysqldump.	Small (less than 20 million records)	√	×

Solution	Data volume	Full migration	Incremental migration
Programming methods • Import data from a database or a text file to PolarDB-X 1.0. • Export data from PolarDB-X 1.0 to a database or a text file. For more information about the detailed steps, see Import big data through a program.	Large	√	×
Data Integration • Import data from MaxCompute (formerly known as ODPS) to PolarDB-X 1.0. • Export data from PolarDB-X 1.0 to MaxCompute (formerly known as ODPS). For more information about the detailed steps, see Import and export data through Data Integration.	Large	√	×

11.2. Evaluate Import

11.2.1. Import data from MyCAT to PolarDB-X 1.0

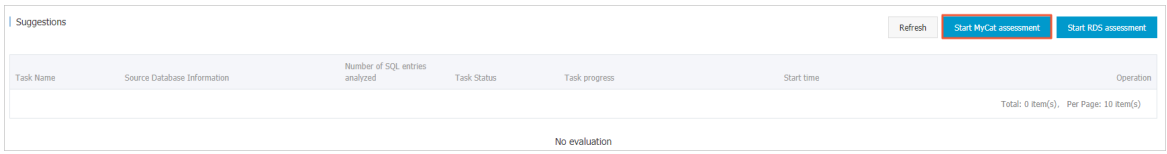
This topic describes how to import data from MyCAT to PolarDB-X 1.0.

Migration process

Step	Description
Step 1: Obtain evaluation suggestions	You can use the evaluation suggestion feature of PolarDB-X 1.0 to analyze your MyCat configuration file. You can use the feature to convert the MyCAT sharding function into a sharding method that matches PolarDB-X 1.0 to the greatest extent possible. You can also use the feature to evaluate the required PolarDB-X 1.0 instance type and the ApsaraDB RDS for MySQL instance type of your database and generate an evaluation suggestion report for your reference based on your submitted business information.
Step 2: Import data	You can use one of the following methods to import data from MyCat to PolarDB-X 1.0: • Evaluation-based import: An integrated migration solution is provided to create databases, create tables, and import data based on the evaluation and recommendations. • Custom import: You can also purchase PolarDB-X 1.0 and ApsaraDB RDS for MySQL instances based on your business requirements, create databases and tables, and then import data from MyCat to the PolarDB-X 1.0 database.

Step 1: Obtain evaluation suggestions

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, choose **Evaluation import > Suggestions**.
4. In the upper-right corner of the **Suggestions** page, click **Start MyCat assessment**.



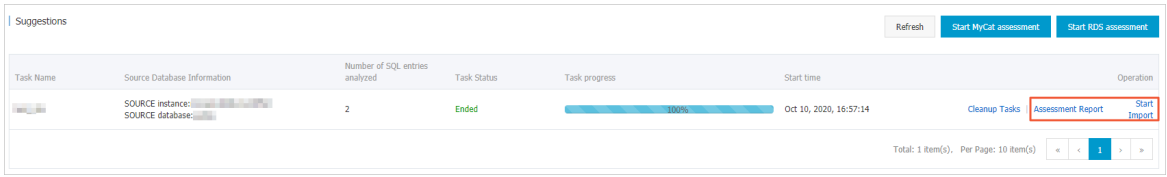
5. On the **MyCat information** page, enter or upload MyCat configuration files, such as schema.xml, rule.xml, and server.xml, and business information, and click **Next**.

> **Note** All MyCat configuration files are used to only generate DDL statements and evaluate the MyCat data that you want to import.

6. On the **Task preview** page, click **Start task** when the **Precheck Result** displays **Success**.

> **Note** Make sure that the database endpoint in your server.xml file can connect to the Internet. Otherwise, the precheck process may fail.

7. After the evaluation task is completed, perform the following steps:

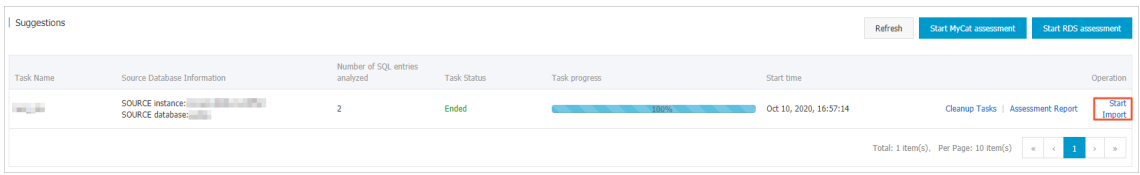


   - You can click **Assessment Report** in the **Operation** column to view evaluation details.
   - You can click **Start Import** in the **Operation** column to import data.

Step 2: Import data

You can use one of the following methods to import data from MyCat to PolarDB-X 1.0:

- Evaluation-based import
  i. Log on to the PolarDB-X console.
  ii. In the top navigation bar, select the region where your instance is deployed.
  iii. In the left-side navigation pane, choose **Evaluation import > Suggestions**.
  iv. On the page that appears, find the task and click **Start Import** in the **Operation** column.



  v. Configure the required parameters to complete data import.
- Custom import

i. Log on to the PolarDB-X console.
ii. In the top navigation bar, select the region where your instance is deployed.
iii. In the left-side navigation pane, choose **Evaluation import > Data import**.
iv. On the **Import tasks** page, click **MyCat custom import** in the upper-right corner.
v. Configure the required parameters to complete custom import.

11.2.2. Migrate data from RDS to PolarDB-X 1.0

This topic describes how to use the evaluation import feature of PolarDB-X 1.0 to migrate data from ApsaraDB for RDS (RDS) to PolarDB-X 1.0.

Background

When you use the RDS service, the data volume, storage, concurrency, and queries per second (QPS) are increased along with the business growth. This may cause performance bottlenecks. You can migrate data from RDS to PolarDB-X 1.0 for database sharding and table sharding. You may need to fix the following issues during the data migration from RDS to PolarDB-X 1.0:

- How do I select the PolarDB-X 1.0 instance type?
- How do I select the type of RDS instance that is attached to the PolarDB-X 1.0 instance?
- How do I shard a RDS table after the table is migrated to PolarDB-X 1.0? How do I determine the number of shards and the shard keys?
- How can I create databases and tables?
- How can I quickly import data from RDS to PolarDB-X 1.0?

Solutions

To fix these issues, PolarDB-X 1.0 provides the evaluation import feature.PolarDB-X 1.0 allows you to simplify data migration from RDS to PolarDB-X 1.0.

Procedure

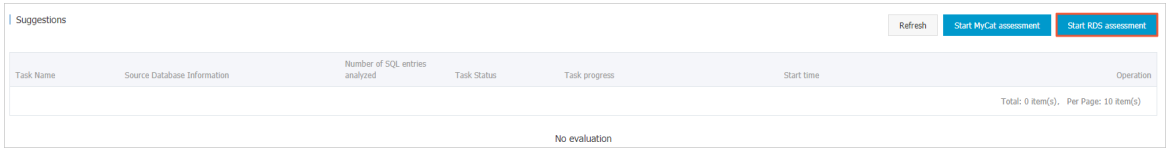
Step	Description
Step 1: Obtain evaluation suggestions	PolarDB-X 1.0 provides the following evaluation suggestions: • Recommends the types of the PolarDB-X 1.0 instance and its attached ApsaraDB RDS for MySQL instances. • Provides sharding solutions for each table in the source RDS. For example, you can check whether to shard each table of the source database, determine the column that is used as the shard key. DRDS also suggests the statements that are used to create tables and import data to PolarDB-X 1.0.

Step	Description
Step 2: Import data	You can use the data import feature provided by PolarDB-X 1.0 in the following ways: • Evaluation-based import: Based on the evaluation and recommendations, an integrated migration solution is provided to create databases, create tables, and import data. Task flow  • Custom import: You can purchase PolarDB-X 1.0 and ApsaraDB RDS for MySQL instances based on your business requirements, and create databases and tables. Then, import data from the source RDS database to the destination PolarDB-X 1.0 database. Task flow 
Step 3: Complete the data migration	After all data is migrated, PolarDB-X 1.0 provides a 14-day synchronization service for incremental data. After all data is imported, you can switch your service from RDS to PolarDB-X 1.0.

Step 1: Obtain evaluation suggestions

PolarDB-X 1.0 evaluates the source RDS and provides suggestions on data migration to PolarDB-X 1.0. All suggestions are included in the evaluation report.

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, choose **Evaluation import > Suggestions**.
4. In the upper-right corner of the **Suggestions** page, click **Start RDS assessment**.

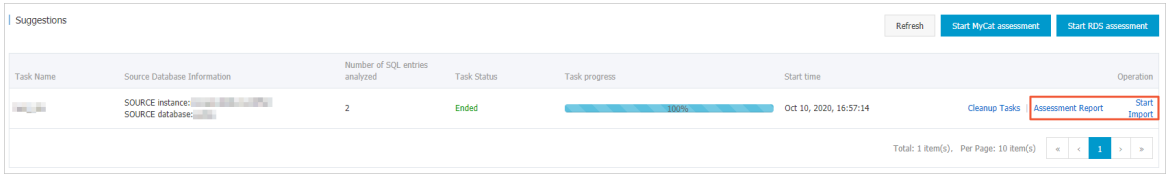


5. On the **SOURCE database information** page, set the following parameters.

Parameter		Description
Task name		The name of a data migration task.
	Instance type	Use the default value of **RDS instance**.
	RDS instance ID	Select the source RDS instance.
	Database name	Select the source database.

Parameter		Description
Source Database Information	Database port	Enter the port number of the source database.
	Database account	Enter the account of the source database.
	Database password	Enter the password of the source database account. **Note** After you enter the account and password of the source database, click **Test connection** to check whether the source RDS can be connected.
Business Information		You can set the **Expected data scale growth percentage** and **Expected average QPS growth percentage** to proper values based on your business requirements to obtain more accurate assessment suggestions.
Evaluation duration		You can set the evaluation duration to 1 hour, 3 hours, 6 hours, 1 day, 3 days, or 7 days. You can also enter a custom evaluation duration from 1 to 336 hours (14 days).

6. Select **Agree to open SQL insight** and click **Next**.
7. On the **Task preview** page, click **Start task** when the **Precheck Result** displays **Success**.
8. After the assessment task is completed, perform the following steps:



- You can click **Assessment Report** in the **Operation** column to view evaluation details.

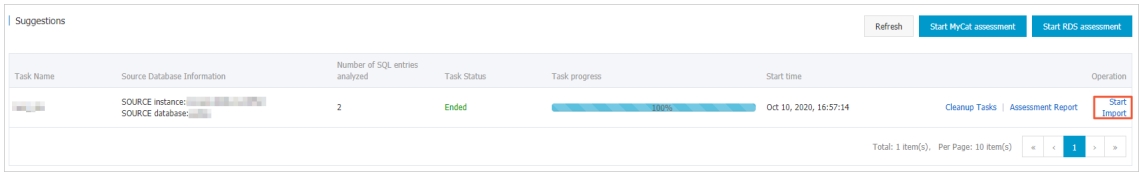
> **Note** You can click **Show** in the **Split Recommendation** section to view details of sharding, such as the sharding type and shard key. You can also click **Export and create table SQL** to download the SQL statements that are used to create tables.

- You can click **Start Import** in the **Operation** column to import data.

Step 2: Import data

You can import data from the source RDS to PolarDB-X 1.0 in the following ways:

- Evaluation-based import:
  i. Log on to the PolarDB-X 1.0 console.
  ii. In the top menu bar, select the region where the destination instance is deployed.
  iii. In the left-side navigation pane, choose **Evaluation import > Suggestions**.
  iv. On the page that appears, find the task and click **Start Import** in the **Operation** column.



  v. On the **Assessment Summary** page, confirm the recommended instance type and number, and click **Next**.
  vi. Select the PolarDB-X 1.0 instance, click > to add the instance to the selected section on the right, and then click **Next**.

> **Note** You can select only one PolarDB-X 1.0 instance.

  vii. Select the ApsaraDB RDS for MySQL instance, click > to add the instance to the selected section on the right, and then click **Next**.

> **Note** You can continue to use the purchased ApsaraDB RDS for MySQL instance. However, we recommend that you purchase a new ApsaraDB RDS for MySQL instance as a separate storage resource for PolarDB-X 1.0.

  viii. On the **Task preview** page, click **Next** when the **Precheck Result** displays **Success**.
  ix. On the **Topology confirmation** page, confirm the source RDS instance, source database, and the destination PolarDB-X 1.0 instance, and click **Create task**.

- Custom import
  i. Log on to the DRDS console.
  ii. In the top menu bar, select the region where the destination instance is deployed.
  iii. In the left-side navigation pane, choose **Evaluation import > Data import**.
  iv. In the upper-right corner of the **Import tasks** page, click **RDS custom import**.
  v. On the **Source database and target database** page, configure the following parameters and click **Next**.

Parameter		Description
Task name		The name of a data migration task.
Source Database Information	**Instance type**	Use the default value of **RDS instance**.
	RDS instance ID	Select the source RDS instance.
	Database name	Select the source database.
	Database port	Enter the port number of the source database.
	Database account	Enter the account of the source database.
	Database password	Enter the password of the source database account. **Note** After you enter the account and password of the source database, click **Test connection** to check whether the source RDS can be connected.
Target database information	**Database port**	Enter the port number of the destination database.
	Database account	Enter the account of the destination database.
	Database password	Enter the password of the destination database account. **Note** After you enter the account and password of the destination database, click **Test connection** to check whether the destination PolarDB-X 1.0 can be connected.

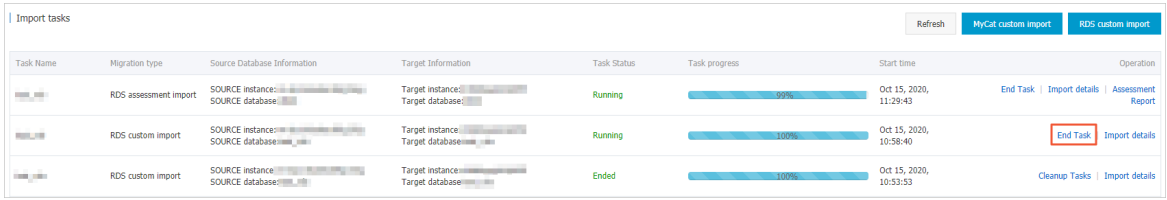
vi. On the **Migration list** page, select the source table from the **SOURCE table name** column on the left, select the destination table from **Target table name** on the right, and then click **Next**.

> **Notice** Tables that use a composite primary key cannot be migrated. Otherwise, an error occurs in the **Task preview** step.

vii. On the **Task preview** page, click **Start task** when the **Precheck Result** displays **Success**.

Step 3: Complete the data migration

1. Log on to the PolarDB-X 1.0 console.
2. In the top navigation bar, select the region where the target instance is located.
3. In the left-side navigation pane, choose **Evaluation import > Data import**.
4. On the page that appears, find the destination table and click **End Task** in the **Operation** column.



> **Note**
>
> - After the full data is migrated, PolarDB-X 1.0 provides a 14-day synchronization service for incremental data. After the **Task progress** is changed to 100%, you can click **End Task** to complete data migration. You can also wait until the system stops 14-day synchronization for incremental data and automatically **End Task**.
> - The **Task Status** of the data import is changed from **Running** to **Ended** after **End Task**. You can click **Cleanup Tasks** to delete the task. Deleted tasks are not displayed in the task list.

FAQ

Q: Do I need to stop excuting writing data to the source database before data importing?

A: No. After the full data is migrated, PolarDB-X 1.0 provides a 14-day synchronization service for incremental data. After the **Task progress** is changed to 100%, you can click **End Task** to complete data migration. You can also wait until the system stops 14-day synchronization for incremental data and automatically **End Task**.

11.3. Import and export data through mysqldump

PolarDB-X 1.0 allows you to export data through mysqldump of MySQL. This topic describes the common use cases and related precautions regarding the typical scenarios of importing data to and exporting data from PolarDB-X 1.0 databases.

Note

- For more information about the mysqldump command, see the MySQL official documentation.
- Mysqldump is suitable for importing and exporting data in small volumes (less than 10 million data records) offline.

Use case 1: Import data to a PolarDB-X 1.0 database from a MySQL database

1. Export data from a MySQL database to a text file. Enter the following command to export the table schema and data from the MySQL database. Assume that data is exported to the file dump.sql.

```
mysqldump -h ip -P port -u user -ppassword --default-character-set=char-set --net_buffe
r_length=10240 --no-create-db --skip-add-locks --skip-lock-tables --skip-tz-utc --set-c
harset  [--hex-blob] [--no-data] database [table1 table2 table3...] > dump.sql
```

   The parameters are listed in the following table. Set the parameters according to the actual situation.

Parameter	Description	Required
`IP`	The IP address of the PolarDB-X 1.0 instance.	✔☺
`port`	The port of the PolarDB-X 1.0 instance.	□
`user`	The username of the PolarDB-X 1.0 instance.	✔☺
`password`	The password of the PolarDB-X 1.0 instance. Note that the password is preceded by `-p` and does not contain a space after `-p`.	✔☺
`char-set`	The specified encoding type.	✔☺
`--hex-blob`	Exports binary string fields in hexadecimal format. This option is required if binary data exists. The affected field types include BINARY, VARBINARY, and BLOB.	□
`--no-data`	Do not export data.	□
`table`	Specifies a table to export. By default, all the tables in the database are exported.	□

2. Modify the table creation statement. The data file exported from the MySQL database contains the table creation statement for each table. If you execute these table creation statements directly in the PolarDB-X 1.0 database, you will create a single table in the PolarDB-X 1.0 database. If you want to partition a table to database shards and table shards, you must modify the table creation statement.
3. Import data file to the PolarDB-X 1.0 database in either of the following ways.
   - Execute `mysql -h ip -P port -u user --default-character-set=char-set` to log on to the

target PolarDB-X 1.0 database. Run `source /yourpath/dump.sql` to import data to the target PolarDB-X 1.0 database.

> **Note** If you use the first method, all the steps appear on the screen, and the process is relatively slow, but you can view the import progress.

- Run `mysql -h ip -P port -u user --default-character-set=char-set< /yourpath/dump.sql` to import data to the target PolarDB-X 1.0 database.

> **Note**
>
> - In the above two commands, `default-character-set` must be set to the actual data encoding format. In Windows, the separators in the file path specified by the SOURCE command must be converted to escape characters.
> - During the import process, due to the implementation differences of some PolarDB-X 1.0 and MySQL databases, you might receive errors such as `ERROR 1231 (HY000): [a29ef6461c00000][10.117.207.130:3306][****]Variable @saved_cs_client can't be set to the value of @@character_set_client`. This type of error does not affect the correctness of imported data.

Use case 2: Import data to a PolarDB-X 1.0 database from another PolarDB-X 1.0 database

Assume that you have a PolarDB-X 1.0 database in the test environment. After you complete the testing, you want to import the table schema and data generated during the testing process to a PolarDB-X 1.0 database in the production environment. Perform the following steps:

1. Export data from the source PolarDB-X 1.0 database to a text file. For more information, see Step 1 of Use case 1.
2. Import the data file to the target PolarDB-X 1.0 database. For more information, see Step 3 of Use case 1.
3. Create a Sequence object manually.

   mysqldump does not support exporting Sequence objects from PolarDB-X 1.0 databases. If the source PolarDB-X 1.0 database uses a Sequence object and you want to use the same Sequence object in the target PolarDB-X 1.0 database, you must create a Sequence object under the same name with the Sequence object in the target PolarDB-X 1.0. Perform the following steps:

   i. Execute the SHOW SEQUENCES command in the source PolarDB-X 1.0 database to obtain the status of the Sequence object in the current PolarDB-X 1.0 database.

   ii. Run the CREATE SEQUENCE command in the target PolarDB-X 1.0 database to create a Sequence object.

Use case 3: Export data from a PolarDB-X 1.0 database to a MySQL database

The procedure of exporting data from a PolarDB-X 1.0 database to a MySQL database is similar to the process of importing data between different PolarDB-X 1.0 databases.

1. Export the table schema and data from the source PolarDB-X 1.0 database. For more information, see Step 1 of Use case 1.

2. Modify the DDL statement for shard table manually.

   The Table creation statement of the shard table in PolarDB-X 1.0 is not compatible with MySQL. To import data to MySQL, you must manually modify the exported SQL file and delete the following keywords:

   - DBPARTITION BY hash(partition_key):
   - TBPARTITION BY hash(partition_key):
   - TBPARTITIONS N
   - BROADCAST

   See the following example for the statement of exporting a shard table:

   ```
   CREATE TABLE multi_db_single_tbl
   (id int,
   name varchar(30),
   primary key(id)) dbpartition by hash(id);
   ```

   Modify the statement as follows:

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

3. Import the modified file. For more information, see Step 3 of Use case 1.

11.4. Import big data through a program

This topic describes how to compile code to import a large volume of data to a Distributed Relational Database Service (DRDS) database offline.

Context

Assume that a table in the current database needs to be imported to a DRDS database. The number of data records to be imported is about 8.14 million. See the following for the table schema:

```
CREATE TABLE `post` (
  `postingType` int NOT NULL,
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `acceptedAnswer` bigint(20) DEFAULT NULL,
  `parentId` bigint(20) DEFAULT NULL,
  `score` int DEFAULT NULL
  `tags` varchar(128) DEFAULT NULL,
  PRIMARY KEY (`id`)
);
```

When you migrate a large volume of data between databases, we recommend that you export the source data as a text file, and then import the file to the target database through a program or commands.

For the preceding post table, you can use SELECT INTO to export data from the MySQL database to the file named as `stackoverflow.csv`. Execute the following command on the MySQL client:

```
SELECT postingType,id,acceptedAnswer,parentId,score,tags
INTO OUTFILE '/tmp/stackoverflow.csv'
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM test_table;
```

Create a table in the DRDS database

The exported data file does not include the table schema, therefore you must create a table in the target DRDS database manually and set a shard key according to the actual situation.

For example, the following command partitions the data of the post table to database shards based on ID:

```
CREATE TABLE `post` (
  `postingType` int NOT NULL,
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `acceptedAnswer` bigint(20) DEFAULT NULL,
  `parentId` bigint(20) DEFAULT NULL,
  `score` int DEFAULT NULL,
  `tags` varchar(128) DEFAULT NULL,
  PRIMARY KEY (`id`)
) DBPARTITION BY hash(id) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Import data to the DRDS database

After you export the data file, you can read the file content through code, and then import the content to the DRDS database. To improve efficiency, we recommend that you insert data in batches.

The following is the demo of code compiled in Java.

Test case: 8,143,801 data records are inserted. It takes about 916 seconds. The TPS is about 9,000.

Configuration of the test client: i5, 8 GB, and SSD

Configuration of the test DRDS database: 4-core 4 GB

```
public static void main(String[] args) throws IOException, URISyntaxException, ClassNotFoun
dException,
        SQLException {
        URL url = Main.class.getClassLoader().getResource("stackoverflow.csv");
        File dataFile = new File(url.toURI());
        String sql = "insert into post(postingType,id,acceptedAnswer,parentId,score,tags) v
alues(?,?,?,?,?,?)";
        int batchSize = 10000;
        try (
            Connection connection = getConnection("XXXXX.drds.aliyuncs.com", 3306, "XXXXX",
                "XXXX",
                "XXXX");
            BufferedReader br = new BufferedReader(new FileReader(dataFile))) {
            String line;
            PreparedStatement st = connection.prepareStatement(sql);
            long startTime = System.currentTimeMillis();
            int batchCount = 0;
            while ((line = br.readLine()) != null) {
```

```
                String[] data = line.split(",");
                st.setInt(1, Integer.valueOf(data[0]));
                st.setInt(2, Integer.valueOf(data[1]));
                st.setObject(3, "".equals(data[2]) ? null : Integer.valueOf(data[2]));
                st.setObject(4, "".equals(data[3]) ? null : Integer.valueOf(data[3]));
                st.setObject(5, "".equals(data[4]) ? null : Integer.valueOf(data[4]));
                if (data.length >= 6) {
                    st.setObject(6, data[5]);
                }
                st.addBatch();
                if (++batchCount % batchSize == 0) {
                    st.executeBatch();
                    System.out.println(String.format("insert %d record", batchCount));
                }
            }
            if (batchCount % batchSize != 0) {
                st.executeBatch();
            }
            long cost = System.currentTimeMillis() - startTime;
            System.out.println(String.format("Take %d second,insert %d record, tps %d", cos
t/1000,batchCount, batchCount/(cost/1000) ));
        }
    }
    /**
     * Obtain the database connection
     *
     * @param host     Database address
     * @param port     Port number
     * @param database Database name
     * @param username Username
     * @param password Password
     * @return
     * @throws ClassNotFoundException
     * @throws SQLException
     */
    private static Connection getConnection(String host, int port, String database, String
username, String password)
        throws ClassNotFoundException, SQLException {
        Class.forName("com.mysql.jdbc.Driver");
        String url = String.format(
            "jdbc:mysql://%s:%d/%s?autoReconnect=true&socketTimeout=600000&rewriteBatchedSt
atements=true", host, port,
            database);
        Connection con = DriverManager.getConnection(url, username, password);
        return con;
    }
```

12.Global Secondary Index

12.1. Use global secondary indexes

PolarDB-X 1.0 supports global secondary indexes (GSIs). This topic describes how to create GSIs and use the GSI feature.

Prerequisites

The MySQL version must be 5.7 or later, and the minor version of the kernel must be 5.4.1 or later.

Create a GSI

PolarDB-X 1.0 has extended MySQL data definition language (DDL) syntax by adding syntax that is used to define GSIs. The syntax usage is the same as the usage of syntax that is used to create indexes on MySQL.

- Define a GSI when you create a table

- Add a GSI after you create a table

> **Note**
>
> - Index name: used as the name of an index table to create the index table.
> - Index column: the shard key of the index table, that is, all the columns that are used in the sharding clause of the index.
> - Covering column: other columns in the index table. By default, the primary key and all the shard keys of the primary table are included.
> - Sharding clause of the index: the sharding algorithm of the index table. Its syntax is the same as the syntax of the sharding clause in CREATE TABLE.

Examples

```
# Define a GSI when you create a table.
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`) COVERING (`id`, `order_id`, `buyer_id`, `order_snap
shot`) dbpartition by hash(`seller_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by hash(`order_id`);
# Add a GSI.
CREATE UNIQUE GLOBAL INDEX `g_i_buyer` ON `t_order`(`buyer_id`)
    COVERING(`seller_id`, `order_snapshot`)
    dbpartition by hash(`buyer_id`) tbpartition by hash(`buyer_id`) tbpartitions 3
```

> **Note** For more information about how to create a GSI, see CREATE INDEX.

Use a GSI

After you create a GSI, you can enable a query to use an index table by using the following methods:

- Use HINT to specify indexes

  You can choose one of the following two HINT statements to specify and use the index for the query:

  - Statement:

    ```
    FORCE INDEX({index_name})
    ```

    Example:

    ```
    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";
    ```

  - Syntax

    ```
    /*+TDDL:INDEX({table_name/table_alias}, {index_name})*/
    ```

    Example:

    ```
    /*+TDDL:index(a, g_i_buyer)*/ SELECT * FROM t_order a WHERE a.buyer_id = 123
    ```

    > **Note** If a query requires the columns that are not contained in an index, you can query the index table first to retrieve all the recorded primary keys and the shard keys of the primary table. Then, query the primary table to retrieve the values of the missing columns. For more information, see INDEX HINT.

- Directly query an index table

If the index table contains all the columns that are required for a query, you can directly query the index table to retrieve the result.

- Select an index

  For a query on the primary table that has a GSI, PolarDB-X 1.0 automatically selects the index table that the optimizer considers to have the minimum cost. Only covering indexes can be selected.

  In the following SQL query, the primary table is `t_order`. The `seller_id` equality filter is contained, and the involved columns such as `id`, `order_snapshot`, and `seller_id` are covered by the GSI `g_i_seller`. The use of the `g_i_seller` covering index eliminates the need to query the primary table and significantly reduces the number of scans on the table shardings. `seller_id` is the shard key of `g_i_seller`. The result of EXPLAIN shows that the PolarDB-X 1.0 optimizer selects `g_i_seller`.

  ```
  EXPLAIN SELECT t_order.id,t_order.order_snapshot FROM t_order WHERE t_order.seller_id = '
  s1';
  IndexScan(tables="g_i_seller_sfL1_2", sql="SELECT `id`, `order_snapshot` FROM `g_i_seller
  ` AS `g_i_seller` WHERE (`seller_id` = ?)")
  ```

- IGNORE INDEX and USE INDEX

  You can enable the optimizer to use or not to use some indexes by using the following HINT:

  - Statement:

    ```
    IGNORE INDEX({index_name},...)
    ```

    Example:

    ```
    SELECT t_order.id,t_order.order_snapshot FROM t_order IGNORE INDEX(g_i_seller) WHERE t_
    order.seller_id = 's1';
    ```

  - Statement:

    ```
    USE INDEX({index_name},...)
    ```

    Example:

    ```
    SELECT t_order.id,t_order.order_snapshot FROM t_order USE INDEX(g_i_seller) WHERE t_ord
    er.seller_id = 's1';
    ```

12.2. Notes for using GSIs

This topic describes the notes for using global secondary indexes (GSIs).

Notes for creating a GSI

- You cannot create a GSI on ApsaraDB RDS for MySQL 5.6.
- You cannot create a GSI in a single table or broadcast tables.
- You cannot create a GSI in tables that do not have primary keys.
- You cannot use a prefixed index in a UNIQUE GSI.
- Specify the index name when you create an index table.
- When you create an index table, specify database sharding rules or database sharding and table

sharding rules. You cannot specify only table sharding rules or specify no sharding rules.

- The INDEX column in index tables must contain all shard keys.
- The name of a GSI cannot be the same as that of other local indexes in a primary table.
- Index columns and covering columns cannot be the same in GSI definition clauses.
- By default, an index table contains all the primary keys and shard keys of the primary table. If the primary keys and shard keys are not explicitly contained in the index columns, the keys are added to the covering columns by default.
- For each local index in the primary table, if all the referenced columns are contained in the index table, the local index is added to the index table by default.
- By default, an index is separately created for each index column of a GSI if no index exists.
- By default, a composite local index is created for a GSI that contains multiple index columns. The composite local index contains all the index columns.
- In an index definition, the length parameter of the index column is used only to create local indexes on the shard keys of the index table.
- If you create a GSI after a table is created, data is automatically verified at the end of the creation process. The execution of the data definition language (DDL) statement to create the GSI can be successful only after the data passes the verification.

> **Note** You can also use the CHECK GLOBAL INDEX statement to verify or modify the index data.

Notes for executing the ALTER TABLE statement

- The following table describes whether the ALTER TABLE statement can be executed to change columns.

Statement	Change the shard keys of the primary table	Change the primary keys of the primary table, also referred to as the primary keys of the index table	Change the local unique index column	Change the shard keys of the index table	Change the unique index column	Change the index column	Change the covering column
ADD COLUMN	No such scenario	Not supported	No such scenario	No such scenario	No such scenario	No such scenario	No such scenario

Statement	Change the shard keys of the primary table	Change the primary keys of the primary table, also referred to as the primary keys of the index table	Change the local unique index column	Change the shard keys of the index table	Change the unique index column	Change the index column	Change the covering column
ALTER COLUMN SET DEFAULT and ALTER COLUMN DROP DEFAULT	Not supported	Not supported	Supported	Not supported	Not supported	Not supported	Not supported
CHANGE COLUMN	Not supported	Not supported	Supported	Not supported	Not supported	Not supported	Not supported
DROP COLUMN	Not supported	Not supported	Supported only when the unique key has only one column	Not supported	Not supported	Not supported	Not supported
MODIFY COLUMN	Not supported	Not supported	Supported	Not supported	Not supported	Not supported	Not supported

> **Note**
>
> - To ensure the stability and performance of GSIs, you are not allowed to delete columns from GSIs by executing the DROP COLUMN statement in a direct way. To delete some columns from a GSI,you can execute the DROP INDEX statement to delete the GSI and create another GSI. You can also Submit a ticket to contact technical support.
> - Overlaps exist in the preceding column classification. For example, the index column contains the shard keys of the index table. The covering column contains the shard keys of the primary table, the primary keys, and the specified columns. If a conflict occurs between Not supported and Supported, Not supported takes precedence over Supported.

- The following table describes whether the ALTER TABLE statement can be executed to change indexes.

Statement	Supported or not
ALTER TABLE ADD PRIMARY KEY	Supported
ALTER TABLE ADD [UNIQUE/FULLTEXT/SPATIAL/FOREIGN] KEY	Supported. You can add local indexes to the primary table and the index table at the same time. An index name cannot be the same as the name of the GSI.
ALTER TABLE ALTER INDEX index_name {VISIBLE | INVISIBLE}	Supported. You can execute this statement on only the primary table. You are not allowed to change the GSI status.
ALTER TABLE {DISABLE | ENABLE} KEYS	Supported. You can execute this statement on only the primary table. You are not allowed to change the GSI status.
ALTER TABLE DROP PRIMARY KEY	Not supported
ALTER TABLE DROP INDEX	You can delete only a common index or a GSI.
ALTER TABLE RENAME INDEX	Not supported

Note To ensure the stability and performance of GSIs, you are not allowed to rename GSIs by executing the ALTER TABLE RENAME INDEX statement in a direct way. To change the GSI name, you can execute the DROP INDEX statement to delete the GSI and create another GSI by using a new name. You can also Submit a ticket to contact technical support.

Notes for changing index tables

- You cannot execute DDL or data manipulation language (DML) statements on index tables.
- You cannot execute DML statements that contain NODE HINT to update primary tables or index tables.

Notes for executing other DDL statements

Statement	Supported or not
DROP TABLE	Supported
DROP INDEX	Supported
TRUNCATE TABLE	Not supported
RENAME TABLE	Not supported
ALTER TABLE RENAME	Not supported

> **Note**
>
> - To ensure data consistency between primary tables and index tables, you are not allowed to execute the TRUNCATE TABLE statement. To clear data in the primary tables and the index tables, you can execute the DELETE statement. You can also Submit a ticket to contact technical support.
> - To ensure the stability and performance of GSIs, you are not allowed to rename GSIs by executing the RENAME TABLE or ALTER TABLE RENAME statement in a direct way. To change the GSI name, you can execute the DROP INDEX statement to delete the GSI, rename the table, and then create another GSI. You can also Submit a ticket to contact technical support.

Notes for executing DML statements

- You cannot execute DML statements on index tables.
- For more information about the limits on executing DML statements on primary tables, see Limits of global secondary indexes on DML.