Data Replication Service

FAQs

Issue 01

Date 2022-09-30

HUAWEI TECHNOLOGIES CO., LTD.

Huawei Technologies Co., Ltd.

Address:	Huawei Industrial Base Bantian, Longgang Shenzhen 518129 People's Republic of China
Website:	https://www.huawei.com
Email:	support@huawei.com

Contents

1 Product Consulting

1.1 What Are Regions and AZs?

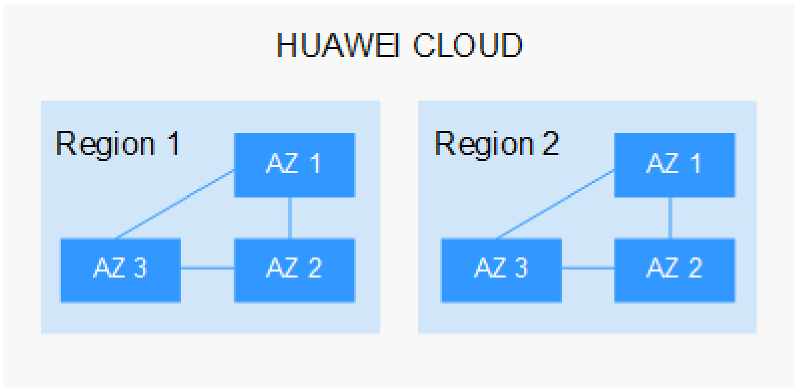
Concept

A region and availability zone (AZ) identify the location of a data center. You can create resources in a specific region and AZ.

- Regions are divided from the dimensions of geographical location and network latency. Public services, such as Elastic Cloud Server (ECS), Elastic Volume Service (EVS), Object Storage Service (OBS), Virtual Private Cloud (VPC), Elastic IP (EIP), and Image Management Service (IMS), are shared within the same region. Regions are classified as universal regions and dedicated regions. A universal region provides universal cloud services for common tenants. A dedicated region provides services of the same type only or for specific tenants.
- An AZ contains one or multiple physical data centers. Each AZ has independent cooling, fire extinguishing, moisture-proof, and electricity facilities. Within an AZ, computing, network, storage, and other resources are logically divided into multiple clusters. AZs within a region are interconnected using high-speed optical fibers to allow you to build cross-AZ high-availability systems.

Figure 1-1 shows the relationship between regions and AZs.

Figure 1-1 Regions and AZs



Selecting an AZ

When determining whether to deploy resources in the same AZ, consider your applications' requirements on disaster recovery (DR) and network latency.

- For high DR capability, deploy resources in different AZs in the same region.
- For low network latency, deploy resources in the same AZ.

1.2 What Is DRS?

DRS is a stable, efficient, and easy-to-use cloud service for database migration and synchronization in real time.

It simplifies data migration processes and reduces migration costs.

You can use DRS to quickly transmit data between different DB engines.

DRS provides multiple functions, including real-time migration, backup migration, real-time synchronization, data subscription, and disaster recovery.

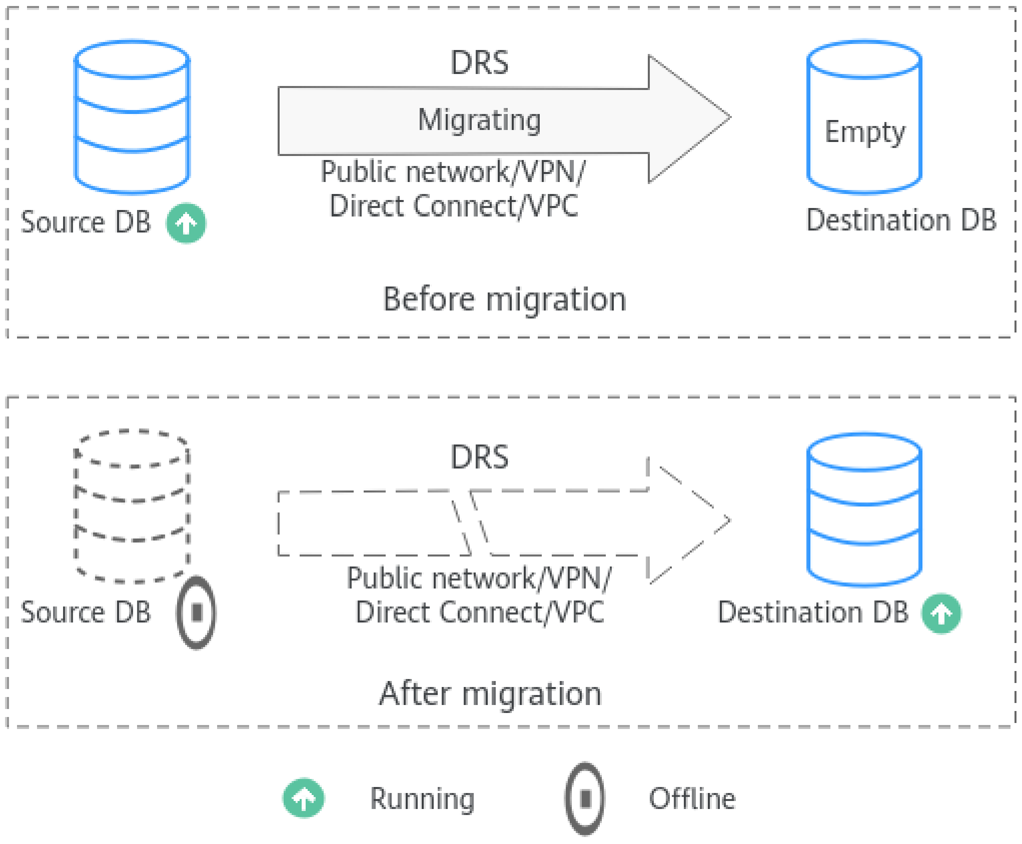
Real-Time Migration

With DRS, you can migrate data from sources to destinations in real time. You create a replication instance to connect to both the source and destination and configure objects to be migrated. DRS will help you compare metrics and data between source and destination, so you can determine the best time to switch to the destination database while minimizing service downtime.

You can perform a migration task over multiple types of networks, such as public networks, VPCs, VPNs, and direct connections. With these network connections, you can migrate between different cloud platforms, from on-premises databases to cloud databases, or between cloud databases across regions.

DRS supports incremental migration, so you can replicate ongoing changes to keep sources and destinations in sync while minimizing the impact of service downtime and migration.

Figure 1-2 Real-time migration process



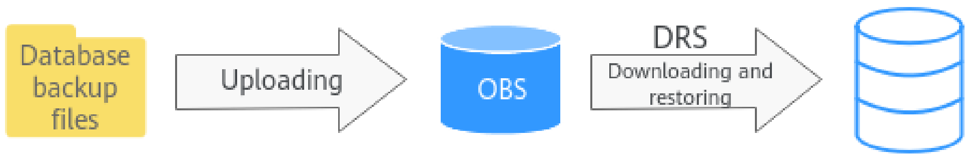
Backup Migration

It often becomes necessary to hide the real IP address of your database for the sake of security. Migrating data through direct connections is an option, but costly. DRS supports backup migration, which allows you to export data from your source database for backup and upload the backup files to OBS. Then, you can restore the backup files to the destination database to complete the migration. Using this method, data migration can be realized without exposing your source databases.

You can use backup migration when you want to migrate on-premises databases to the cloud.

Without connecting to your sources, DRS can help you complete data migration.

Figure 1-3 Backup migration process



Real-Time Synchronization

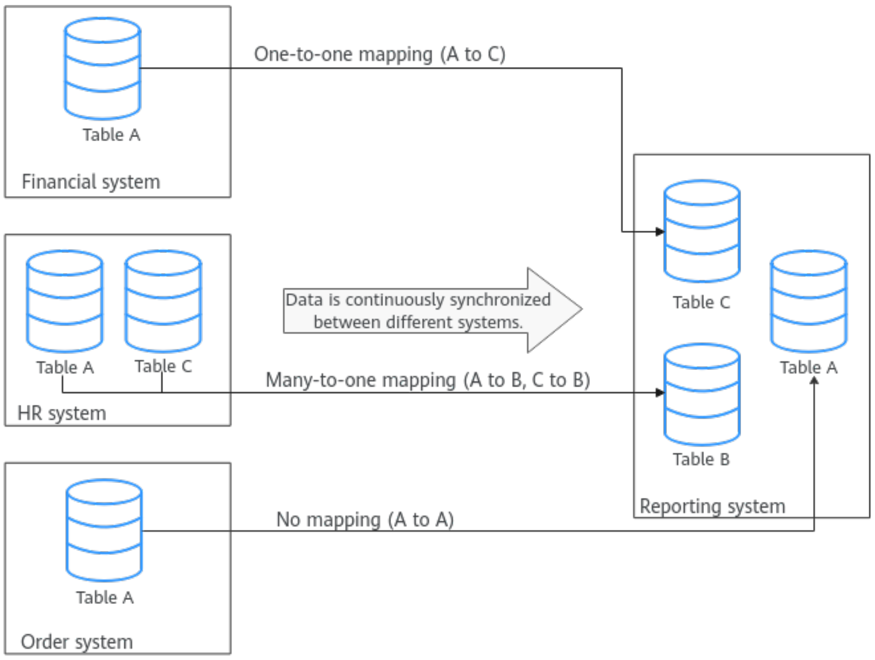
Real-time synchronization refers to the real-time flow of key service data from sources to destinations through a synchronization instance while consistency of data can be ensured.

It is different from migration. Migration means moving your overall database from one platform to another. Synchronization refers to the continuous flow of data between different services.

You can use real-time synchronization in many scenarios such as real-time analysis, report system, and data warehouse environment.

Real-time synchronization is mainly used for synchronizing tables and data. It can meet various requirements, such as many-to-one, one-to-many synchronization, dynamic addition and deletion of tables, and synchronization between tables with different names.

Figure 1-4 Many-to-one real-time synchronization process

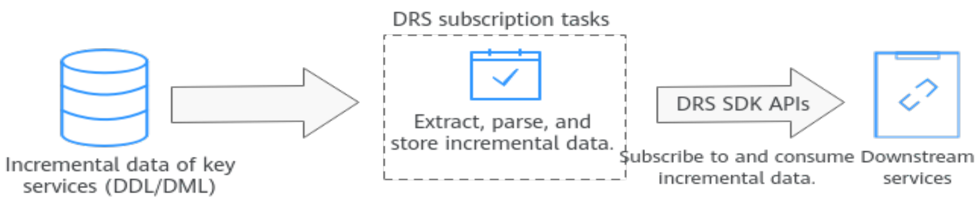

One-to-one mapping (A to C)
Table A
Financial system
Table A
Table C
HR system
Data is continuously synchronized between different systems.
Many-to-one mapping (A to B, C to B)
Table C
Table A
Table B
Reporting system
No mapping (A to A)
Table A
Order system

Data Subscription

With DRS, you can subscribe changes made to key services in a database for downstream services to consume. DRS caches the changes and uses a unified SDK API to facilitate downstream services to subscribe to, obtain, and consume the changes, decoupling databases from downstream systems.

Data subscription can be used by Kafka to subscribe to MySQL incremental data.

Figure 1-5 Data subscription

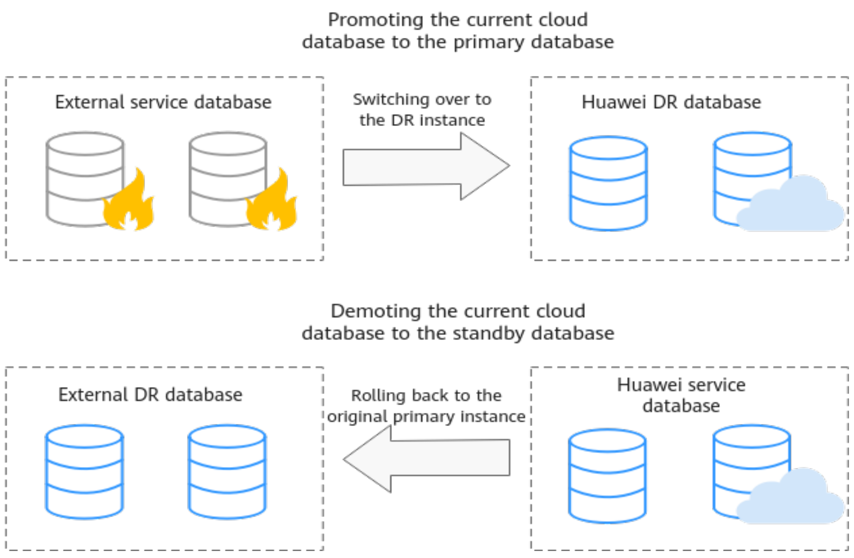

Incremental data of key services (DDL/DML)
DRS subscription tasks
Extract, parse, and store incremental data.
DRS SDK APIs
Subscribe to and consume incremental data.
Downstream services

Real-Time Disaster Recovery

To prevent service unavailability caused by regional faults, DRS provides disaster recovery to ensure service continuity. You can easily implement disaster recovery between on-premises and cloud, without the need to invest a lot in infrastructure in advance.

The disaster recovery architectures, such as two-site three-data-center and two-site four-data center, are supported. A primary/standby switchover can be implemented by promoting a standby node or demoting a primary node in the disaster recovery scenario.

Figure 1-6 Real-time DR switchover


Promoting the current cloud database to the primary database
External service database
Switching over to the DR instance
Huawei DR database
Demoting the current cloud database to the standby database
External DR database
Rolling back to the original primary instance
Huawei service database

1.3 Can DRS Migrate RDS Primary/Standby Instances?

Yes. DRS provides high availability and can migrate a single RDS instance or RDS primary/standby instances. DRS can automatically rebuild the databases connection after a short interruption and resumes data transfer from the point when the connection was lost to ensure the continuity and consistency of data synchronization.

If the HA design of the source database meets the requirements of floating IP address connections and RPO is 0 during a switchover, DRS supports migration of primary/standby instances without manual intervention.

If the HA design does not meet the requirements of floating IP address connections and RPO is 0 during a switchover, the following situations may occur:

- The floating IP address is used and RPO may be 0 during a switchover. In this situation, the database can be connected, but DRS will identify data interruption (if data loss occurs during the switchover) and display a message indicating that the task fails. You can only reset the migration task.
- A fixed IP address is used and RPO is 0 during the switchover. In this situation, the migration is supported only when the instance is running properly.

- The floating IP address is used and zero RPO cannot be ensured during a switchover. In this situation, the database can be connected, but DRS will identify data interruption (if data loss occurs during the switchover) and display a message indicating that the task fails. You can only reset the migration task.

If the destination is primary/standby instances, DRS can ensure that the source data is completely migrated to the destination database. However, the switchover of the destination database cannot ensure zero RPO. As a result, data in the destination database may be incomplete.

1.4 What Constraints Does DRS Have for a Source Database?

Different data flow types require different databases and permissions. For details, refer to the following sections:

- **Supported Databases**
- **Real-Time Migration Overview**
- **Backup Migration Overview**
- **Real-Time Synchronization Overview**
- **Notes on Data Subscription**
- **Real-Time DR Overview**

1.5 What Requirements Does DRS Have for a Destination Database?

When you create a task, the destination database user must have certain permissions. Different data flow types require different permissions. For details, refer to the following sections:

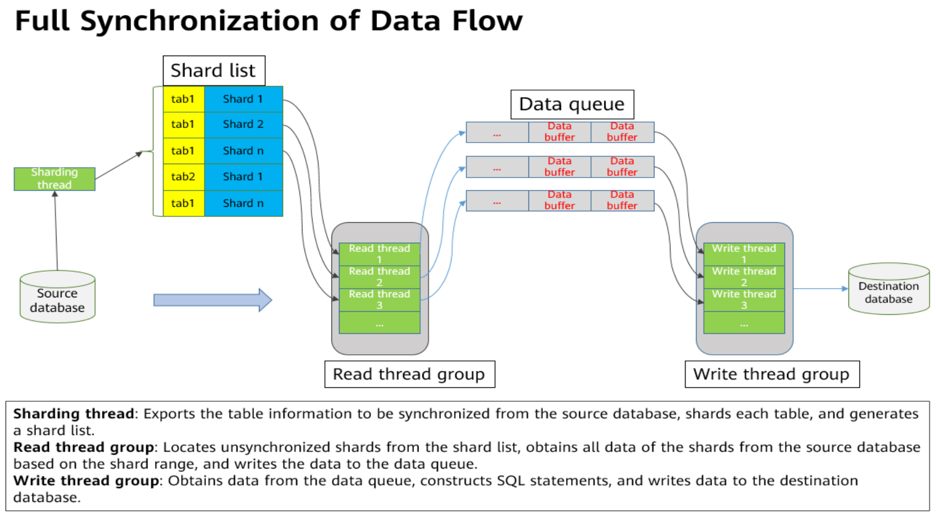
1.6 Does DRS Use Concurrency?

Concurrency is key to performance improvement. DRS uses concurrency in multiple phases. There are two types of concurrency: read/write concurrency and thread concurrency.

Read and Write Concurrency

- To migrate data quickly during full migration and ensure a stable connection, DRS extracts data at row-level with concurrency. As a result, the migration speed is maximized and repeated migration of large amounts of data will not occur.
- To speed up data writes, DRS supports table-level concurrent writes, greatly improving data transmission speed and achieving millisecond synchronization delay in the same city.

Figure 1-7 Read and write concurrency

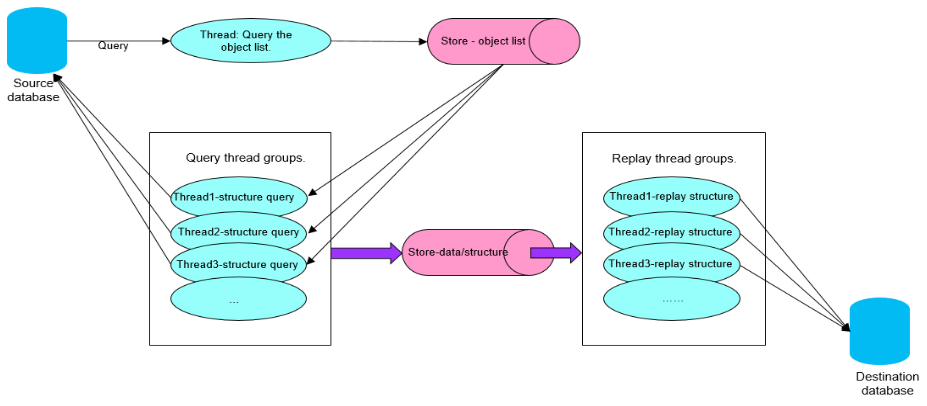


Sharding thread: Exports the table information to be synchronized from the source database, shards each table, and generates a shard list.
Read thread group: Locates unsynchronized shards from the shard list, obtains all data of the shards from the source database based on the shard range, and writes the data to the data queue.
Write thread group: Obtains data from the data queue, constructs SQL statements, and writes data to the destination database.

Thread Concurrency

When you migrate a large number of database objects (for example, more than 10,000 tables), the structure migration will become a bottleneck. DRS optimizes concurrency for such scenarios. It uses multiple threads to concurrently query and replay structures, greatly improving structure migration performance.

Figure 1-8 Thread concurrency



1.7 Dose DRS Use Data Compression?

DRS uses compression in data transmission and incremental data storage to improve migration performance and resolve data expansion.

- Data transmission: If bandwidth is insufficient, you can enable data compression to reduce the bandwidth occupied by data transmission.
- Incremental data storage: After incremental data is stored into logical files, the amount of data will increase sharply. You can enable data compression to reduce the storage capacity occupied by local cached logical files.

1.8 Does DRS Support Migration Between the Public Cloud and the Dedicated Cloud?

DRS supports database migration between the DeC and public cloud. The data flow from the public cloud to the dedicated cloud is outbound, and from the dedicated cloud to the public cloud is inbound. For details about the supported migration scenarios, see **Real-Time Migration**. For details about specific methods, see **Real-Time Migration Overview**.

1.9 What Is the Quota?

Resource quotas are defined on the platform for each service to prevent resource exhaustion. For example, the maximum number of database connections that can be created for a DAS account is 30.

If the existing resource quota cannot meet your requirements, you can apply for a higher quota.

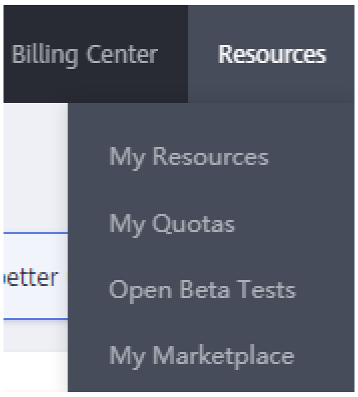
Viewing My Quotas

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 In the upper right corner, choose **Resources** > **My Quotas**.

Figure 1-9 My quota



Step 4 View the used and total quota of each type of resources on the displayed page.

If a quota cannot meet your needs, apply for a higher quota.

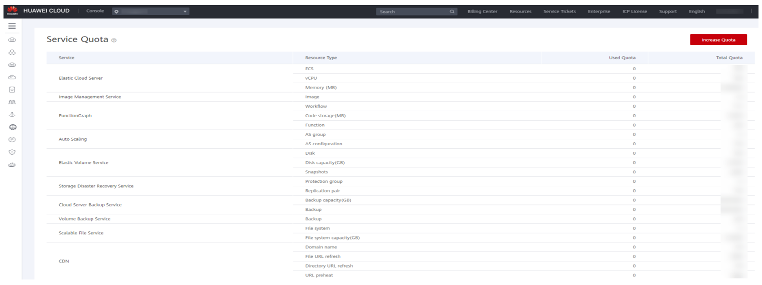
----End

Increasing Quotas

Step 1 Log in to the management console.

Step 2 In the upper right corner of the page, choose **Resources** > **My Quota**. On the **Service Quota** page, click **Increase Quota**.

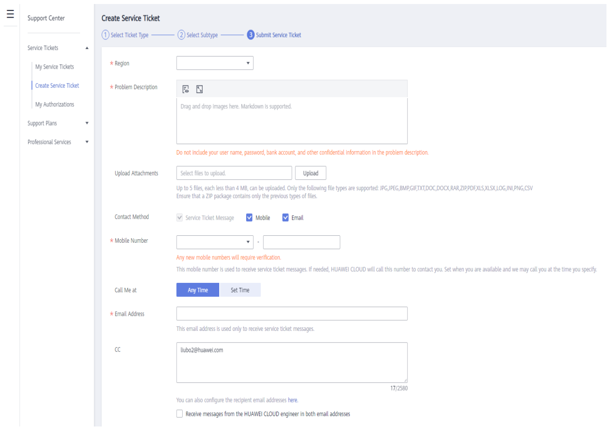
Figure 1-10 Increasing quotas



Service	Resource Type	Used Quota	Total Quota
Elastic Cloud Server	ECS	0	[redacted]
	vCPU	0	[redacted]
	Memory (MB)	0	[redacted]
Image Management Service	Image	0	[redacted]
FunctionGraph	Workflow	0	[redacted]
	Code storage(MB)	0	[redacted]
	Function	0	[redacted]
Auto Scaling	AS group	0	[redacted]
	AS configuration	0	[redacted]
Elastic Volume Service	Disk	0	[redacted]
	Disk capacity(GB)	0	[redacted]
	Snapshots	0	[redacted]
Storage Disaster Recovery Service	Protection group	0	[redacted]
	Replication pair	0	[redacted]
Cloud Server Backup Service	Backup capacity(GB)	0	[redacted]
	Backup	0	[redacted]
Volume Backup Service	Backup	0	[redacted]
Scalable File Service	File system	0	[redacted]
	File system capacity(GB)	0	[redacted]
CDN	Domain name	0	[redacted]
	File URL refresh	0	[redacted]
	Directory URL refresh	0	[redacted]
	URL preheat	0	[redacted]

Step 3 On the **Create Service Ticket** page, configure parameters as required. In the **Problem Description** area, fill in the content and reason for adjustment.

Figure 1-11 Creating a service ticket



* Region

* Problem Description

Drag and drop images here. Markdown is supported.

Do not include your user name, password, bank account, and other confidential information in the problem description.

Upload Attachments: Select files to upload.

Up to 5 files, each less than 4 MB, can be uploaded. Only the following file types are supported: JPG,JPEG,BMP,GIF,TXT,DOC,DOCX,RAR,ZIP,PDF,XLS,XLSX,LOG,INI,PNG,CSV
Ensure that a ZIP package contains only the previous types of files.

Contact Method: [✓ Service Ticket Message] [✓ Mobile] [✓ Email]

* Mobile Number

Any new mobile numbers will require verification.

This mobile number is used to receive service ticket messages. If needed, HUAWEI CLOUD will call this number to contact you. Set when you are available and we may call you at the time you specify.

Call Me at: [✓ Any Time] Set Time

* Email Address

This email address is used only to receive service ticket messages.

CC: liubo2@huawei.com

17/2580

You can also configure the recipient email addresses here.

Receive messages from the HUAWEI CLOUD engineer in both email addresses

Step 4 After all necessary parameters are configured, select **I have read and agree to the Tenant Authorization Letter and Privacy Statement** and click **Submit**.

----End

1.10 Does DRS Support Migration from DB2 for LUW to PostgreSQL?

DRS does not support the migration from DB2 for LUW to PostgreSQL. Currently, only real-time synchronization from **DB2 for LUW to GaussDB** is supported. For details, see **Supported Databases**.

1.11 Can Microsoft SQL Server Database Synchronize Data with Local Databases in Real Time?

The Microsoft SQL Server database does not support real-time data synchronization with the local database. If data is migrated to the cloud, you can use the backup migration mode. Backup migration supports incremental migration of Microsoft SQL Server databases, which effectively shortens the service migration time.

If you want to synchronize Microsoft SQL Server database to the cloud, only GaussDB(DWS), RDS for SQL Server, and GaussDB can be the destination.

However, only whitelisted users can perform this task. You can submit a service ticket to apply for the permission. In the upper right corner of the management console, choose **Service Tickets** > **Create Service Ticket**.

1.12 Does DRS Support Data Replication in a Specified Time Period?

Currently, DRS supports only full and full+incremental migration scenarios.

1.13 Does DRS Support Resumable Uploads?

In database migration and synchronization scenarios, if a migration or synchronization task fails due to unavoidable problems (such as network fluctuation), DRS records the current parsing and replay point (which is the basis of database internal consistency) and then resumes data transfer from the point to ensure data integrity.

For incremental migration and synchronization, DRS automatically retries for multiple times. For full migration of MySQL databases, the system automatically resumes the migration for three times by default. After the number of automatic retry failures reaches a specified value, the task becomes abnormal. You need to analyze the cause based on logs and try to rectify the blocking point (for example, the database password is changed). If the environment cannot be restored and the required logs have been eliminated, you can use the reset the task.

1.14 What Is Single-Active/Dual-Active Disaster Recovery?

With the rapid development of information technologies, data and information play an increasingly important role in modern enterprises. Loss and damage of data will cause inestimable losses to enterprises. How to defend against large-scale disasters has drawn increasing attention. Currently, remote disaster recovery (DR) is the only feasible solution. The backup and restoration of key data is an important part of the routine operation and maintenance of the system.

The dual-AZ, HA instances of Huawei Cloud RDS can meet the requirements of intra-city disaster recovery. DRS provides cross-region and cross-cloud DR capabilities, including single-active DR and dual-active DR.

NOTE

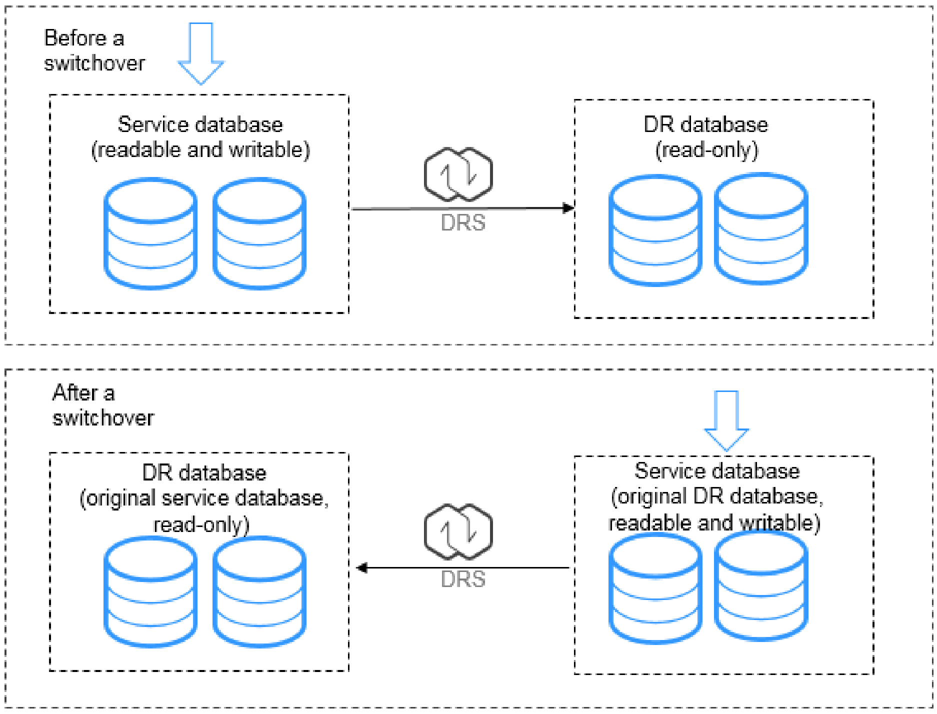
Currently, Huawei Cloud RDS for MySQL instances support the single-active or dual-active DR. If both sides are Huawei Cloud RDS for MySQL instances, cross-region DR can be performed.

Single-Active DR

In single-active DR mode, one active database and one standby database are deployed. When a disaster occurs, the DR database functions as the service database to ensure service continuity. DRS supports active/standby switchover.

Before a switchover, services are running properly in the service database and data is synchronized to the DR database in real time. In this case, data cannot be written into the DR database. After an active/standby switchover, the DR database becomes readable and writable, services can be switched to the DR database, and data cannot be written to the service database.

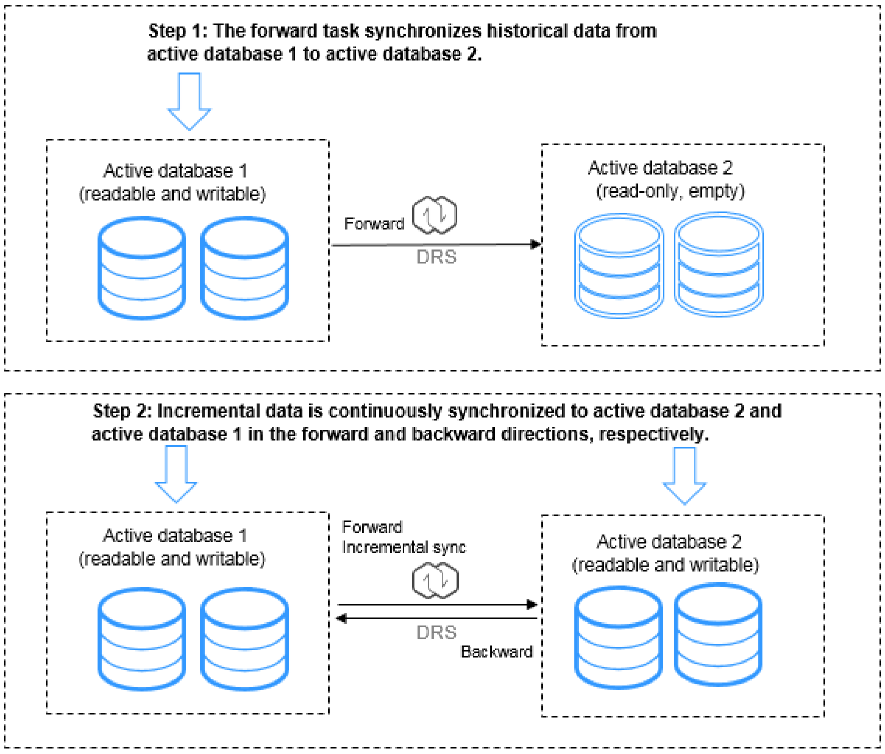
Figure 1-12 Single-active DR


Before a switchover
Service database (readable and writable)
DRS
DR database (read-only)
After a switchover
DR database (original service database, read-only)
DRS
Service database (original DR database, readable and writable)

Dual-Active DR

The dual-active DR mode is used in scenarios where the two databases work in active/standby mode and share services. Dual-active DR contains two roles, active database 1 and active database 2. Before performing dual-active DR, you need to determine the RDS role in the current cloud (region). A complete dual-active DR is performed in two directions, one in the forward direction and the other in the backward direction. The two directions must be created in sequence. At the beginning, active database 1 is readable and writable, and active database 2 is read-only. The backward DR can be started only after the initial data is fully synchronized from active database 1 to active database 2 in the forward direction. In this case, both active database 1 and active database 2 are readable and writable, and incremental data is continuously synchronized to active database 2 and active database 1 in the forward and backward directions, respectively.

Figure 1-13 Dual-Active DR


Step 1: The forward task synchronizes historical data from active database 1 to active database 2.
Active database 1
(readable and writable)
Forward
DRS
Active database 2
(read-only, empty)
Step 2: Incremental data is continuously synchronized to active database 2 and active database 1 in the forward and backward directions, respectively.
Active database 1
(readable and writable)
Forward
Incremental sync
DRS
Backward
Active database 2
(readable and writable)

Features and constraints:

- The dual-active DR architecture has high requirements on the environment. Before deploying the dual-active DR solution, view **Before You Start**.
- The dual-active DR deployment poses strict requirements on the procedure. Perform the following steps to ensure that the dual-active DR task can be successfully deployed.

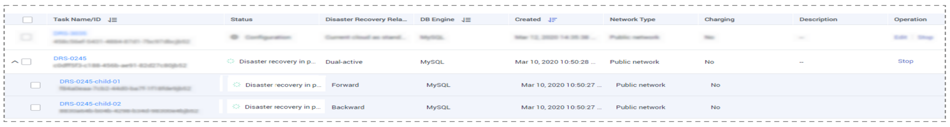
  a. Create a DR task. For details, see **Create a DR Task**. After the creation is complete, two subtasks are generated, that is, the forward DR task and reverse DR task. In this case, the reverse DR task is in the configuration state.

  b. When the forward DR task is in the DR state (the reverse task is displayed in the **Operation** column), configure and start the reverse task.

     On the **Disaster Recovery Management** page, select the backward DR task and click **Edit** in the **Operation** column. The **Create Disaster Recovery Task** page is displayed. Continue to create the backward task.

     You are advised to perform the verification on the active database 2 and start the backward task after the expected result is met.

Figure 1-14 Forward and backward DR task



Task Name/ID	Status	Disaster Recovery Rela...	DB Engine	Created	Network Type	Charging	Description	Operation
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
DRS-0245 [redacted]	Disaster recovery in p...	Dual-active	MySQL	Mar 10, 2020 10:50:28 ...	Public network	No	--	Stop
DRS-0245-child-01 [redacted]	Disaster recovery in p...	Forward	MySQL	Mar 10, 2020 10:50:27 ...	Public network	No		
DRS-0245-child-02 [redacted]	Disaster recovery in p...	Backward	MySQL	Mar 10, 2020 10:50:27 ...	Public network	No		

1.15 What Are the Differences Between Real-Time Migration, Real-Time DR, and Real-Time Synchronization?

Item	Real-Time Migration	Real-Time Synchronization	Real-Time DR
Scenario	Migration can be performed between different cloud platforms, from on-premises databases to cloud databases, or on cloud databases across regions.	Real-time analysis, report system, and data warehouse environment	You can perform disaster recovery between on-premises databases and cloud databases, or between databases across cloud platforms.
Characteristics	Homogeneous databases are migrated as a whole. Tables, data, indexes, views, stored procedures, functions, database accounts, and database parameters can be migrated at the table level, database level, or all dimensions.	Maintains continuous data flow between different services, synchronizes tables and data, and meets various flexibility requirements. Objects can be migrated at the table level or database level. Data synchronization between heterogeneous databases is supported.	The remote primary/standby switchover can be achieved. Instance-level disaster recovery is supported. Object selection is not supported.

Item	Real-Time Migration	Real-Time Synchronization	Real-Time DR
Functions and features	• Users can be migrated. • Parameters can be compared.	• Synchronization objects can be edited in the incremental phase, and added or deleted dynamically. • Object names mapped to the destination database can be changed, so the names of tables and databases in the source and destination databases can be different. • Data processing is supported. You can add rules, such as data filtering and column processing, for selected objects.	• Instance-level DR • Parameter comparison • Primary/Standby switchover
Remarks	Different data flows support different functions and features. For details, see **Precautions**.	Different data flows support different functions and features. For details, see **Precautions**.	Different data flows support different functions and features. For details, see **Precautions**.

1.16 How Do I Solve the Table Bloat Issue?

In the full migration phase, DRS uses the row-level parallel migration mode to ensure migration performance and transmission stability. If the source database data is compact, table bloat may occur after data is migrated to the RDS for MySQL database. As a result, the disk space required is much greater than that of the source database. In this case, you can run the following command in the destination database to free up the space:

```
optimize table table_name
```

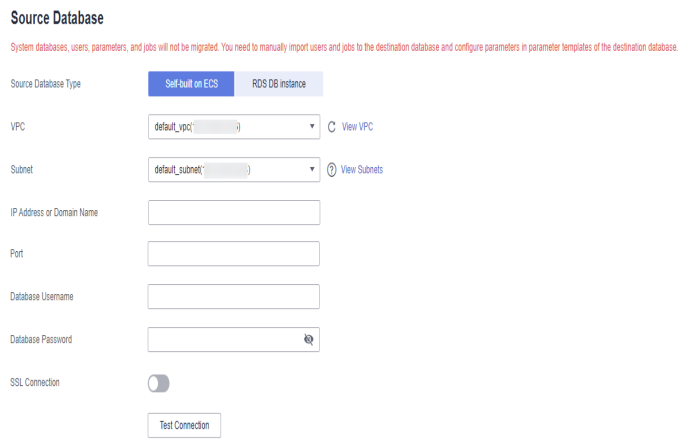
NOTE

The OPTIMIZE TABLE command locks tables. Do not run this command when you operate table data. Otherwise, services may be affected.

1.17 How Do I Select RDS Read Replicas on the DRS Console?

RDS read replicas cannot be selected on the DRS console. You can select **Self-built on ECS** and enter the read replica IP address and port number to connect to the instance. For details, see **Figure 1-15**.

Figure 1-15 Source database information



Source Database

System databases, users, parameters, and jobs will not be migrated. You need to manually import users and jobs to the destination database and configure parameters in parameter templates of the destination database.

Source Database Type	[Self-built on ECS] RDS DB instance
VPC	default_vpc([redacted]) View VPC
Subnet	default_subnet([redacted]) View Subnets
IP Address or Domain Name	
Port	
Database Username	
Database Password	
SSL Connection	
	Test Connection

1.18 How Does DRS Affect the Source and Destination Databases?

Impact on the Source

- During the initialization of a full migration or synchronization task, DRS needs to query all inventory data in the source database. DRS uses simple SQL statements to query data, and the query speed is limited by the I/O performance and network bandwidth of the source database. Generally, if the bandwidth is not limited, the query workload of the source database will be increased by 50 MB/s and 2 to 4 vCPUs will be occupied. If the source database is read concurrently, about 6 to 10 sessions are occupied.
  - Fewer than eight sessions are used to query some system tables, such as tables, views, and columns in the information_schema database, in the source database.
  - Fewer than four sessions are used to query shards in the source database. For example, in the following statement, the conditions following **select** and **where** contain only the primary key or unique key.
    ```
    select id from xxx where id>12345544 and limit 10000,1;
    ```
  - Fewer than four sessions are used to query SQL statements. For example, in the following statement, the information after **select** is all column names in the table, and the condition after **where** contains only the primary key or unique key.
    ```
    select id,name,msg from xxx where id>12345544 and id<=12445544;
    ```
  - The SQL statement for locking a table without a primary key is similar to the following statement. The table is locked to obtain the consistency

point of the table without a primary key. After the table is locked, a connection is obtained to unlock the table.

```
flush table xxx with read lock
lock table xxx read
```

- In the incremental phase, there is no pressure on the source database. Only one dump connection is available to listen to binlog incremental data in real time.

Impact on the Destination Database

- During the initialization of a full migration or synchronization task, DRS needs to write structures, inventory data, and indexes of the source database to the destination database in sequence. Generally, the total number of sessions is less than 16.
  - Fewer than eight sessions are used to create structures.
  - Fewer than eight sessions are writing data. Example:
    ```
    insert into xxx (id,name,msg) values (xxx);
    ```
  - Fewer than eight sessions are used to create indexes. Example:
    ```
    alter table xxx add index xxx;
    ```
- In the incremental phase, DRS parses the incremental data in the binlog file of the source database into SQL statements and executes the SQL statements in the destination database. Generally, the total number of sessions is less than 64.
  - DDL statements are executed in serial mode. When a DDL statement is executed, no other DML statement is executed.
  - There are a maximum of 64 DML connections (short connections with a timeout interval of 30 seconds). The DML statements include insert, update, delete, and replace.

NOTE

To evaluate the impact on the source database, you can create a test task and adjust the migration policy by using rate limiting or run the test during off-peak hours.

1.19 Do I Need to Stop Services Running on the Source Database?

DRS tasks are classified into three modes: full, incremental, and full+incremental. Different data flow types support different modes.

- Full migration: This migration type is suitable for scenarios where service interruption is permitted. It migrates all objects and data in non-system databases to the destination database at one time.
- Incremental: In this mode, incremental data generated on the source database is continuously migrated to the destination database by parsing logs.
- Full+Incremental: This migration type allows you to migrate data without interrupting services. After a full migration initializes the destination database, an incremental migration initiates and parses logs to ensure data consistency between the source and destination databases.

Tasks in incremental or full+incremental mode will not be automatically stopped. Incremental data generated on the source database will be continuously migrated to the destination database. You can determine whether to stop the tasks. For details, see **When Can I Stop a Migration Task?**.

When creating a task, you can select a mode as required. You do not have to stop services on the source database. However, you need to pay attention to the impact of full and incremental backup on the database in different phases. For details, see **How Does DRS Affect the Source and Destination Databases?**.

1.20 What Is an SMN Topic?

- What Is SMN?

  Simple Message Notification (SMN) is a reliable and flexible large-scale message notification service. It enables you to efficiently send messages to email addresses, phone numbers, and HTTP/HTTPS servers.
- For DRS:

  SMN is a related service. You can configure topics on the SMN console. If a topic has been created and subscribed to by other services, you can directly subscribe to the topic for DRS.

  If you have not created or subscribed to an SMN topic, **create a topic** first. A topic serves as a channel for sending messages and subscribing to notifications so that publishers and subscribers can communicate with each other. Then, **add a subscription** and **request subscription confirmation**. After the subscription is confirmed, alarm notifications will be sent to the subscription endpoint through SMN.

1.21 What Are the Differences Between Primary/Standby and Single-Node Instances for DRS Synchronization Tasks?

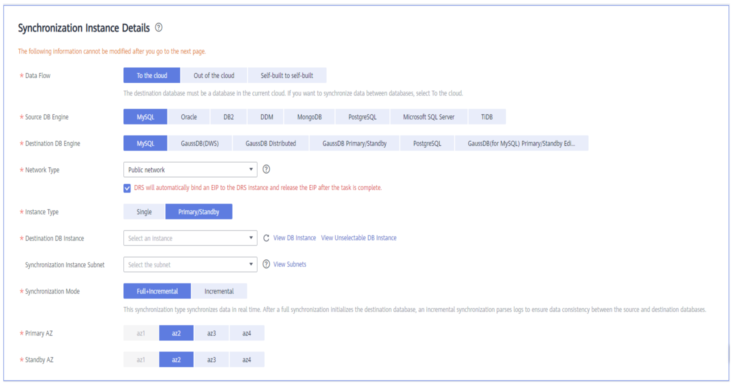
You can set **Instance Type** to **Primary/Standby** or **Single** when creating a DRS real-time synchronization task.

Primary/Standby: After the synchronization task is created, DRS creates two subtasks, each running on the primary and standby nodes. If the subtask on the primary node fails, DRS **automatically starts** the subtask on the standby node to continue the synchronization.

Single: The synchronization task will be created on only one node, which is cost-effective.

This option is available only in specific scenarios. For details, see **Performing a Primary/Standby Switchover**.

Figure 1-16 Synchronization instance details



Synchronization Instance Details

The following information cannot be modified after you go to the next page.

* Data Flow	[✓ To the cloud] Out of the cloud Self-built to self-built The destination database must be a database in the current cloud. If you want to synchronize data between databases, select To the cloud.
* Source DB Engine	[✓ MySQL] Oracle DB2 DDM MongoDB PostgreSQL Microsoft SQL Server TiDB
* Destination DB Engine	[✓ MySQL] GaussDB(DWS) GaussDB Distributed GaussDB Primary/Standby PostgreSQL GaussDB(for MySQL) Primary/Standby Edi...
* Network Type	Public network ✓ DRS will automatically bind an EIP to the DRS instance and release the EIP after the task is complete.
* Instance Type	Single [✓ Primary/Standby]
* Destination DB Instance	Select an instance View DB Instance View Unselectable DB Instance
Synchronization Instance Subnet	Select the subnet View Subnets
* Synchronization Mode	[✓ Full+Incremental] Incremental This synchronization type synchronizes data in real time. After a full synchronization initializes the destination database, an incremental synchronization parses logs to ensure data consistency between the source and destination databases.
* Primary AZ	az1 [✓ az2] az3 az4
* Standby AZ	az1 [✓ az2] az3 az4

1.22 Can DRS Migrates Table Structures Only?

DRS is a cloud service used for real-time data transfer. Currently, DRS cannot migrate table structures only but not data. For details about the objects supported by each data flow, see the following links.

Supported Databases

Real-Time Migration Overview

Real-Time Synchronization Overview

Real-Time DR Overview

1.23 How Do I Migrate Accounts in MySQL Migration, Synchronization, and DR Tasks and Can I Change Passwords?

- MySQL real-time migration: You can choose whether to migrate accounts when creating a migration task. For details, see **Migrating Accounts**. If the source and destination databases are of the same major version and the entire instance is migrated, DCL statements can be migrated in incremental mode, but users cannot be changed by updating the **mysql.user** table.
- MySQL real-time synchronization: Accounts cannot be synchronized.
- MySQL real-time DR: Accounts that have operation permissions on user-defined objects in the system database cannot be used for DR. DR objects cannot be selected. In the DR phase, DCL statements are supported, but accounts cannot be changed by updating the **mysql.user** table.

1.24 What Factors Affect the DRS Task Speed and How Do I Estimate the Time Required?

Factors

- Read throughput of the source database

  The higher the read throughput is, the faster the migration speed can be, and the less the time required can be. Factors that affect the throughput include but are not limited to server specifications, load, disk I/O performance, and database traffic limiting.
- Write throughput of the destination database

  The higher the write throughput is, the faster the migration speed can be, and the less the time required can be. Factors that affect the throughput include but are not limited to server specifications, load, disk I/O performance, and database traffic limiting.
- Available network throughput

  The higher the available network throughput is, the faster the migration speed can be, and the less the time required can be. Factors that affect network throughput include but are not limited to available bandwidth, firewalls, and network device traffic limiting.
- Network quality and delay

  The shorter the network latency is, the faster the migration speed can be, and the less the time required can be. The factors include but are not limited to the distance between the source or destination database and the DRS instance. Poor network quality (for example, high packet loss rate) reduces the migration speed.
- DRS instance specifications

  The larger the DRS instance specifications are, the faster the migration speed can be, and the less the time required can be.
- Model and distribution of source data

  Such factors include whether there is a primary key, whether there is a partition table, whether there is a heap table, average data volume in a single row, number of tables, and number of indexes.
- Whether there is data in the destination database

  Existing data in the destination database may cause data conflicts during migration, resulting in performance deterioration.
- Whether the destination database has a trigger

  If the destination database has a trigger, the write performance may deteriorate during migration.
- Destination database backup and log settings

  If not necessary, disable destination database backup and transaction logs during the migration to improve migration performance.
- Incremental data generation speed of the source database

  The faster the incremental data is generated in the source database, the longer it takes to balance the incremental data.

- Number of DRS tasks

  If performance bottlenecks caused by other factors are not considered, you can split DRS tasks by table to improve the overall migration performance.

Estimated Migration Duration

There are many factors that affect the migration duration. No common calculation method can be used to evaluate the migration duration. You are advised to create a test task in an environment with the same specifications, load, network configuration, and data model as the instance to be migrated to evaluate the migration duration.

1.25 Can I Modify Objects in a DRS Task?

Real-time migration: If a real-time migration task has been created but has not been started, you can modify the migration objects. After the task is started, you cannot modify the migration objects.

Backup migration: You can edit a backup migration task only when creating it. A backup migration task that has been started cannot be modified.

Real-time synchronization: If a real-time synchronization task has been created but has not been started, you can modify the synchronization objects. If a synchronization task is in the incremental phase and is a table-level synchronization, you can edit the synchronization objects. If a synchronization task is a database-level synchronization or in other synchronization phases, you cannot edit the synchronization objects. For details, see **Editing Synchronization Objects**.

Data subscription: If a data subscription task has been created but has not been started, you can modify the selected objects. After the task is started, you cannot modify the selected objects.

Real-time DR: A DR task is a instance-level DR and does not support object selection.

1.26 Does DRS Support Data Synchronization Between Different Databases of the Same DB Instance?

DRS real-time synchronization can use the object name mapping function to change a destination database object name so that the database object names in the source and destination databases are different. In this way, data can be synchronized between different databases of the same DB instance. For details, see **Mapping Object Names**.

1.27 Which Operations on the Source or Destination Database Affect the DRS Task Status?

Take Huawei Cloud RDS for MySQL as an example. The following operations may affect the DRS task status.

- Backing up an instance: Generally, backing up an instance has no impact on DRS tasks.
- Restarting an instance: Restarting an instance will cause a temporary interruption. During this period, the DB instance is unavailable and the DRS connection is interrupted for a short time. In this case, DRS automatically retries. If the failure persists, click **Resume** in the **Operation** column to resume the task after the instance becomes normal.
- Primary/standby switchover: During a primary/standby switchover, services may be intermittently interrupted for several seconds or minutes. In this case, DRS automatically retries. If the failure persists, click **Resume** in the **Operation** column to resume the task after the instance becomes normal.
- Changing specifications: After instance specifications are changed, the instance will be restarted, which will cause temporary interruption. During this period, the DB instance is unavailable and the DRS connection is interrupted for a short time. In this case, DRS automatically retries. If the failure persists, click **Resume** in the **Operation** column to resume the task after the instance becomes normal.
- Upgrading the version of a DB instance: Upgrading the minor version of a database kernel will restart the DB instance. Restarting the DB instance will cause temporary interruption. During this period, the DB instance is unavailable and the DRS connection is interrupted for a short time. In this case, DRS automatically retries. If the failure persists, click **Resume** in the **Operation** column to resume the task after the instance becomes normal.
- Abnormal instances: If a DB instance becomes abnormal, DRS automatically retries. If the failure persists, click **Resume** in the **Operation** column to resume the task after the instance becomes normal.
- Restricting the number of connected sessions: A certain number of sessions are required for a DRS task to connect to the source and destination databases. For details, see **How Does DRS Affect the Source and Destination Databases?**. If the number of connections is insufficient, the DRS task fails. You can adjust the number of database connections and click **Resume** in the **Operation** column to resume the task.
- Network jitter: If the DRS connection fails due to network jitter, DRS automatically retries. If the failure persists, click **Resume** in the **Operation** column to resume the task after the network recovers.
- Changing passwords: Changing a database password may cause DRS connection failures. For details, see **What Do I Do After Changing the Password of the Source or Destination Database?**.
- Changing permissions: Changing database account permissions may cause data migration failures due to insufficient DRS permissions. After assigning permissions to the migration account again, click **Resume** in the **Operation** column to resume the task.
- Clearing source database logs: When source database logs (for example, MySQL binlog) are cleared, DRS cannot obtain logs that connect to the current synchronization position from the source database. As a result, the task may fail (for example, Full or Incremental Phase Error: binlog is not existed). Reset the synchronization task by referring to **Resetting a Synchronization Task**, or create a synchronization task again.
- Changing database parameters: DRS pre-checks the source and destination database parameters before starting a task. Do not modify the database

parameters after the pre-check is complete. Otherwise, the task may fail. If the task fails due to parameter changes, restore the parameters and click **Resume** in the **Operation** column to resume the task.

1.28 What Are Differences Between Data Subscription and Synchronization from MySQL to Kafka?

Item	MySQL-to-Kafka Synchronization	Data Subscription
Supported source databases	● Huawei Cloud RDS for MySQL instances ● On-premises MySQL databases ● MySQL databases on ECSs ● MySQL databases on other clouds	Huawei Cloud RDS for MySQL instances
Supported networks	Public network, VPCs, VPNs, and Direct Connect networks	Only VPCs are supported. As the entire-subnet route is not enabled, container-based networks in a VPC are not supported.
Stability	There are task alarms and monitoring, DRS automatically retries tasks upon exceptions.	There are not task alarms and monitoring, you need to manually retry tasks upon exceptions.
Performance	Multiple specifications are supported and can be selected based on different performance requirements.	Specifications cannot be selected. Only the performance of the minimum specifications is supported.
Commercial state	In commercial use, you will be charged for using this feature.	In open beta testing (OBT), this feature is free of charge.

A subscription task has many subscription objects and operation constraints. You are advised to use the synchronization **from MySQL to Kafka** with higher performance and stability for data subscription.

2 Network and Security

2.1 What Security Protection Policies Does DRS Have?

Network

- Uses security groups to ensure that the sources of access are trusted.
- Uses SSL channels to encrypt data during transmission.

Management

Use the Identity and Access Management (IAM) service to manage DRS permissions.

2.2 What Can I Do If the Network Is Disconnected During the Migration?

If the network is disconnected during the migration, you can view the task status first. If a full or incremental task fails, click **Resume** in the **Operation** column.

Full migration

Incremental migration

Full synchronization

Incremental synchronization

2.3 Which Database Accounts Are Required During Migration?

MySQL

To ensure your database can be successfully migrated to RDS for MySQL DB instances on the current cloud, DRS automatically creates temporary accounts

drsFull and **drsIncremental** for a destination database when you create a migration task. After the task is complete, DRS automatically deletes the temporary account.

NOTICE

Attempting to delete, rename, or change the passwords or permissions for these accounts will cause task errors.

Table 2-1 Temporary accounts created for RDS for MySQL

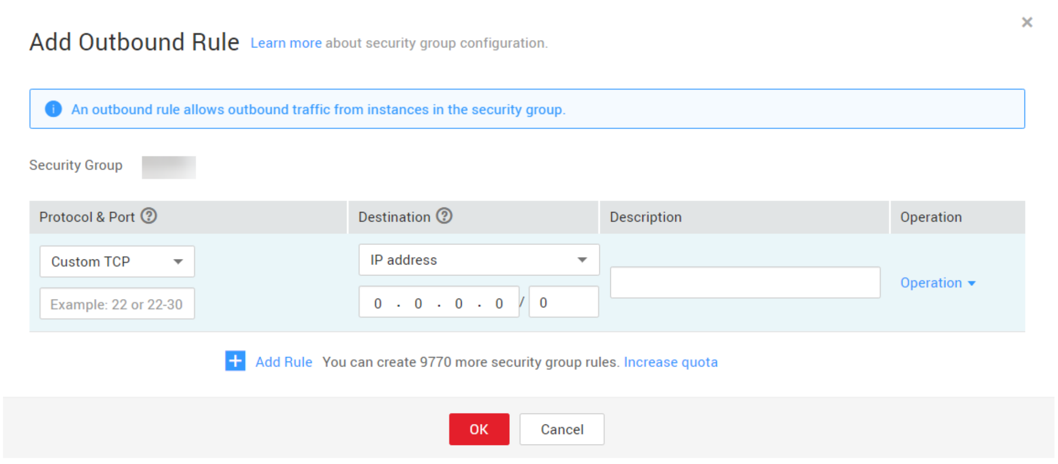
Account	Application Scenario	Host	Description
drsFull	Full migration	RDS for MySQL DB instance	When you start a full migration task, DRS will add this account to the destination RDS database for data migration.
drsIncremental	Incremental migration	RDS for MySQL DB instance	When you start an incremental migration task, DRS will add this account to the destination RDS database for data migration.

2.4 How Can I Configure a VPC Security Group to Allow Traffic from an EIP?

By default, a VPC on the current cloud is isolated from external networks for security reasons. You cannot use an EIP outside a VPC (for example, an EIP of another cloud database or an on-premise database) to access DB instances inside the VPC. However, the replication instance or destination database in a VPC needs to connect to an external EIP to migrate data.

Therefore, you need to add an outbound rule to a security group to allow access from specific external EIPs and ports outside the VPC. The outbound rule allows the replication instance EIP to access the destination database EIP. The following figure shows how to add an outbound rule.

Figure 2-1 Adding an outbound rule


Add Outbound Rule Learn more about security group configuration.

An outbound rule allows outbound traffic from instances in the security group.

Protocol & Port	Destination	Description	Operation
Custom TCP / Example: 22 or 22-30	IP address / 0 . 0 . 0 . 0 / 0		Operation

Add Rule You can create 9770 more security group rules. Increase quota

For details, see **Adding a Security Group Rule**.

2.5 What Can I Do If the Network Connection Between the Replication Instance and Database Is Abnormal?

Before data migration, ensure that network preparations and security rule settings are complete. If the connection is abnormal, check whether the network configuration is correct.

This section uses the migration from MySQL to RDS for MySQL as an example to describe three migration scenarios: cross-cloud online migration, on-premises database migration, and online migration of self-built ECS databases.

Cross-Cloud Real-Time Migration

1. Network settings

   Enable public accessibility for the source database.

   – Source database network settings:

     Enable public accessibility for the source database.

   – Destination database network settings:

     By default, the destination database and the DRS replication instance are in the same VPC and can communicate with each other. No further configuration is required.

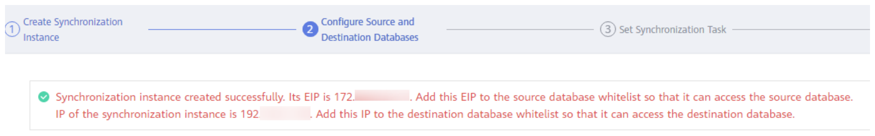
2. Security rules

   – Source database security group settings:

     Add the EIP of the replication instance to the whitelist of the source MySQL DB instance to allow the access from the EIP.

     Before configuring the whitelist for the source database, obtain the EIP of the DRS replication instance. You can find the EIP on the **Configure Source and Destination Databases** page after creating the replication instance on the DRS console.

Figure 2-2 Instance IP address


1 Create Synchronization Instance
2 Configure Source and Destination Databases
3 Set Synchronization Task

Synchronization instance created successfully. Its EIP is 172.[redacted]. Add this EIP to the source database whitelist so that it can access the source database.
IP of the synchronization instance is 192[redacted]. Add this IP to the destination database whitelist so that it can access the destination database.

You can also add 0.0.0.0/0 to the source database whitelist to allow any IP address to access the source database but you must ensure that the above does not pose a risk to your services.

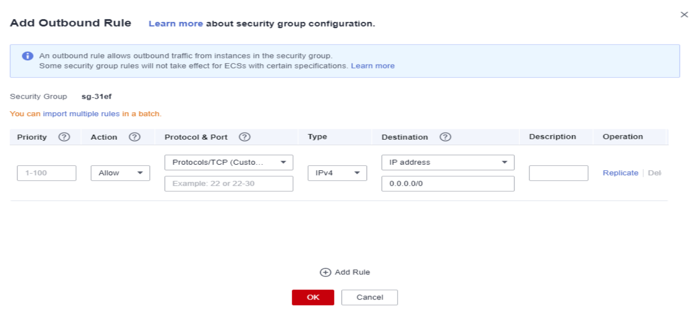
After the migration is complete, you can delete the configuration from the whitelist.

– Destination database security group settings:

  ■ By default, the destination database and the DRS replication instance are in the same VPC and can communicate with each other. DRS can directly write data to the destination database.

  ■ Configure the security group of the VPC where the destination database is located to ensure that the IP addresses and listening ports of the DRS instance are allowed to access the on-premises database. The following figure shows how to add an outbound rule.

Figure 2-3 Adding an outbound rule



Add Outbound Rule   Learn more about security group configuration.

An outbound rule allows outbound traffic from instances in the security group.
Some security group rules will not take effect for ECSs with certain specifications. Learn more

Security Group sg-31ef

You can import multiple rules in a batch.

Priority	Action	Protocol & Port	Type	Destination	Description	Operation
1-100	Allow	Protocols/TCP (Custo... Example: 22 or 22-30	IPv4	IP address 0.0.0.0/0		Replicate | Del

Real-Time Migration of On-Premises Databases

1. Network settings

   – Source database network settings:

   You can migrate on-premises MySQL databases to the RDS for MySQL databases on the current cloud through a VPN or public network. Enable public accessibility or establish a VPN for the on-premises MySQL databases based on the site requirements. You are advised to migrate data through a public network, which is more convenient and cost-effective.

   – Destination database network settings:

     ■ If the source database attempts to access the destination database through a VPN, ensure that the VPN service is enabled and the source database can communicate with the destination RDS for MySQL database.

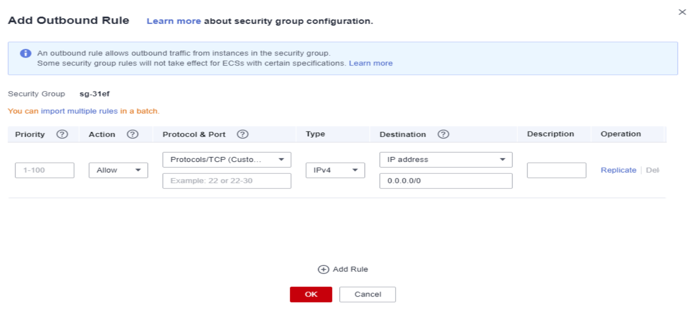
- If the source database attempts to access the destination database through a public network, you do not need to configure the destination RDS for MySQL database.

2. Security rules
   a. Source database security group settings:
      - If the migration is performed over a public network, add the EIP of the DRS replication instance to the network whitelist of the source MySQL database to enable the source MySQL database to communicate with the current cloud. Before setting the network whitelist, obtain the EIP of the replication instance.

        The IP address on the **Configure Source and Destination Databases** page is the EIP of the replication instance.
      - If the migration is performed over a VPN network, add the private IP address of the DRS migration instance to the network whitelist of the source MySQL database to enable the source MySQL database to communicate with the current cloud. The IP address on the **Configure Source and Destination Databases** page is the private IP address of the replication instance.

      After the migration is complete, you can delete the rules.
   b. Destination database security group settings:
      - By default, the destination database and the DRS replication instance are in the same VPC and can communicate with each other. DRS can directly write data to the destination database.
      - Configure the security group of the VPC where the destination database is located to ensure that the IP addresses and listening ports of the DRS instance are allowed to access the on-premises database. The following figure shows how to add an outbound rule.

Figure 2-4 Adding an outbound rule



Add Outbound Rule   Learn more about security group configuration.

An outbound rule allows outbound traffic from instances in the security group.
Some security group rules will not take effect for ECSs with certain specifications. Learn more

Security Group   sg-31ef

You can import multiple rules in a batch.

Priority	Action	Protocol & Port	Type	Destination	Description	Operation
1-100	Allow	Protocols/TCP (Custo... / Example: 22 or 22-30	IPv4	IP address / 0.0.0.0/0		Replicate | Del

Real-Time Migration of Self-Built Databases on the ECS

1. Network settings
   – The source and destination databases must be in the same region.
   – The source and destination databases can be either in the same VPC or different VPCs.

        - If the source and destination databases are in the same VPC, the networks are interconnected by default.
        - If the source and destination databases are in different VPCs, the subnets of the source and destination databases are required to be in different CIDR blocks. You need to create a VPC peering connection between the two VPCs.

2. Security rules
   - In the same VPC, the network is connected by default. You do not need to set a security group.
   - In different VPCs, establish a VPC peering connection between the two VPCs. You do not need to set a security group.

Checking iptables Settings

If the source database is a self-built database on an ECS and cannot be connected after the preceding operations are performed, check the iptables settings. If the DRS frequently initiates connection requests and fails, the HOSTGUARD service adds the requested IP address to the blacklist.

1. Log in to the ECS.
2. Run the following command to check whether any DENY-related project contains the IP address of the DRS instance. The project name is **IN_HIDS_MYSQLD_DENY_DROP**.

   iptables --list
3. If yes, run the following command to query the iptables inbound rule list and obtain the rule ID (line-numbers):

   iptables -L INPUT --line-numbers
4. Run the following command to delete the inbound rules that deny the IP address of the DRS instance: (Note: Delete the rules from the end to the beginning. Otherwise, line-numbers will be updated and you need to query again.)

   iptables -D *Project_name Rule_ID*
5. Delete the iptables rules and test the connection again.

Related Documents

For more information about network settings for deploying the source database in different locations, see **here**.

2.6 How Can the Source and Destination Databases Communicate Across VPCs?

DRS supports migration through a VPC, VPN, Direct Connect, or public network. The VPC network is suitable for migrations between cloud databases in the same region.

- The source and destination databases must be in the same region.

- The source and destination databases can be in either the same VPC or in different VPCs.
- If source and destination databases are in the same VPC, they can communicate with each other by default. Therefore, you do not need to configure a security group.
- If the source and destination databases are not in the same VPC, the CIDR blocks of the source and destination databases cannot be duplicated or overlapped, and the source and destination databases are connected through a VPC peering connection. DRS automatically establishes a route through a single IP address when you test the network connectivity.
- DRS does not support communication between the source database and destination database over a VPC across tenants. If necessary, you can create a VPC peering connection and select **VPN** for **Network Type** to enable communication between the source and destination databases.

Restrictions on VPC Peering Connections

- VPC peering connections created between VPCs that have overlapping subnet CIDR blocks may not take effect.
- You cannot have more than one VPC peering connection between any two VPCs at the same time.
- You cannot create a VPC peering connection between VPCs in different regions.
- If the CIDR blocks of two VPCs overlap, the peering connection can only be created between the subnets of the two VPCs. If two subnets have overlapping CIDR blocks, a VPC peering connection cannot be created between them. When you create a VPC peering connection, ensure that the VPCs involved do not contain overlapping subnets.
- After a VPC peering connection is established, the local and peer tenants must add routes in the local and peer VPCs to enable communication between the two VPCs.
- VPC A is peered with both VPC B and VPC C. If VPC B and VPC C have overlapping CIDR blocks, you cannot configure routes with the same destinations for VPC A.
- To ensure security, do not accept VPC peering connections from unknown accounts.
- Either owner of a VPC in a peering connection can delete the VPC peering connection at any time. If a VPC peering connection is deleted by one of its owners, all information about this connection will also be deleted immediately, including routes added for the VPC peering connection.
- You cannot delete a VPC that has VPC peering connection routes configured.
- A VPC peering connection can be created between VPCs in same region even if one is created on the HUAWEI CLOUD Chinese Mainland console and another on the HUAWEI CLOUD international console.
- Even if VPC 1 and VPC 2 are connected using a VPC peering connection, ECSs in VPC 2 cannot access the Internet through the EIP of VPC 1. If you want to allow the ECSs in VPC 2 to access the Internet through the EIP of VPC 1, you can use a NAT gateway or **configure an SNAT server**. For details, see **Having an ECS Without a Public IP Address Access the Internet**.

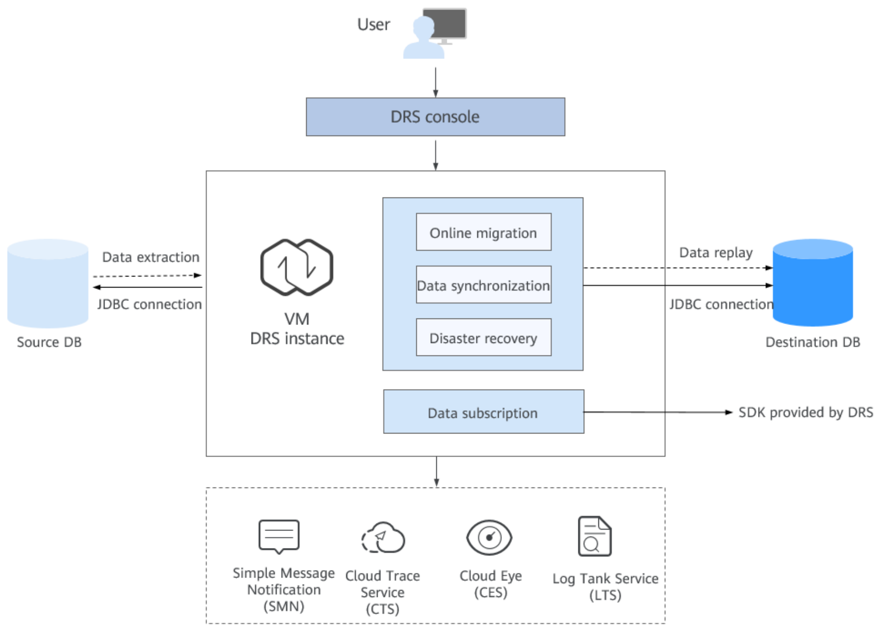
For details about how to create a VPC peering connection, see ***Virtual Private Cloud User Guide***.

2.7 What Is the EIP Bandwidth of DRS?

The EIP bandwidth of DRS is 300 Mbit/s.

2.8 Does DRS Support Cross-Account Cloud Database Migration?

Figure 2-5 DRS product architecture



Currently, DRS supports replication over a public network, VPC, VPN, and Direct Connect. You can select a network type as required.

Theoretically, DRS uses the JDBC connection. You do not need to deploy programs on the source and destination databases. You only need to configure the source and destination databases to accept connections from the DRS instance nodes over the selected network.

For example, if you want to migrate instance RDS-A of account A to instance RDS-B of account B, you can apply for an EIP and bind it to RDS-A, create a DRS (to-the cloud) task using account B, and select the public network for migration.

3 Permissions Management

3.1 How Do I Set an Independent Oracle Account That Has the Least Privilege and Uses DRS?

To perform a full migration for an Oracle database, you must grant the CREATE SESSION, SELECT ANY TRANSACTION, SELECT ANY TABLE, and the SELECT ANY DICTIONARY permissions to the user. If you need to perform an incremental migration, the user must have the log parsing permission. If the destination database is a PostgreSQL database, the SELECT ANY SEQUENCE permission is also required. This section describes how to set an independent Oracle account that has the least privilege and uses DRS.

- Full migration
  a. Create a user for migration. User1 is used as an example.
     Example command: **CREATE USER** *User1* **IDENTIFIED BY** *pwd*

     NOTE

     User1 indicates the username and **pwd** indicates the password.
  b. Run the following statement as user **sys** or as user who has the DBA permission to grant the required permissions to **User1**:
     Example command: **GRANT CREATE SESSION, SELECT ANY TRANSACTION, SELECT ANY TABLE, SELECT ANY DICTIONARY TO** *User1*
- Full+incremental migration
  a. Create a user for migration. User1 is used as an example.
     Example command: **CREATE USER** *User1* **IDENTIFIED BY** *pwd*
  b. Run the following statement as user **sys** or as user who has the DBA permission to grant the required permissions to **User1**:
     Example command: **GRANT CREATE SESSION, SELECT ANY TRANSACTION, SELECT ANY TABLE, SELECT ANY DICTIONARY TO** *User1*
  c. Run the following statement as user **sys** or as user who has the DBA permission to grant the log parsing permission to **User1**:

- If Oracle version is earlier than 12c, run the following statement:

  GRANT EXECUTE_CATALOG_ROLE TO *User1*

- If Oracle version is later than 12c, run the following statement:

  Example command: **GRANT EXECUTE_CATALOG_ROLE TO** *User1*

  Example command: **GRANT LOGMINING TO** *User1*

3.2 Which MySQL Permissions Are Required for DRS?

DRS has certain permission requirements on accounts during migration, synchronization, and DR. This section describes the permission requirements on the MySQL engine.

Permission

- You must have the login permission of the source and destination database connection accounts. If you do not have the account, perform the following operations to create one. user1 is used as an example.

  Reference statement: **CREATE USER** 'user1'@'host' **IDENTIFIED BY** 'password'

- **Table 3-1** lists the required permissions for real-time migration, synchronization, and DR.

Table 3-1 Permission requirements and reference statements

Function Modules	Source/Service Database	Destination/DR Database
Real-time migration	Full migration: SELECT, SHOW VIEW, and EVENT Reference statement: **GRANT** SELECT, SHOW VIEW, EVENT **ON** *.* **TO** 'user1'; Full+incremental migration: SELECT, SHOW VIEW, EVENT, LOCK TABLES, REPLICATION SLAVE, and REPLICATION CLIENT ● REPLICATION SLAVE and REPLICATION CLIENT are global permissions and must be enabled separately. The reference statement is as follows: **GRANT** REPLICATION SLAVE, REPLICATION CLIENT **ON** *.* **TO** 'user1'; ● SELECT, SHOW VIEW, EVENT, and LOCK TABLES are non-global permissions. The reference statement is as follows: **GRANT** SELECT, SHOW VIEW, EVENT, LOCK TABLES, **ON** [Database to be migrated].* **TO** 'user1';	Full migration: SELECT, CREATE, ALTER, DROP, DELETE, INSERT, UPDATE, INDEX, EVENT, CREATE VIEW, CREATE ROUTINE, TRIGGER, REFERENCES, and WITH GRANT OPTION. If the destination database version is in the range 8.0.14 to 8.0.18, the SESSION_VARIABLES_ADMIN permission is required. Reference statement: **GRANT** SELECT, CREATE, ALTER, DROP, DELETE, INSERT, UPDATE, INDEX, EVENT, CREATE VIEW, CREATE ROUTINE, TRIGGER **ON** *.* **TO** 'user1' **WITH GRANT OPTION**; Full+incremental migration: SELECT, CREATE, ALTER, DROP, DELETE, INSERT, UPDATE, INDEX, EVENT, CREATE VIEW, CREATE ROUTINE, TRIGGER, REFERENCES, and WITH GRANT OPTION. If the destination database version is in the range 8.0.14 to 8.0.18, the SESSION_VARIABLES_ADMIN permission is required. Reference statement: **GRANT**SELECT, CREATE, ALTER, DROP, DELETE, INSERT, UPDATE, INDEX, EVENT, CREATE VIEW, CREATE ROUTINE, TRIGGER, REFERENCES **ON** [*Databases to be migrated*].* **TO** 'user1' **WITH GRANT OPTION**;

Function Modules	Source/Service Database	Destination/DR Database
Real-time synchronization	SELECT, SHOW VIEW, EVENT, LOCK TABLES, REPLICATION SLAVE, and REPLICATION CLIENT ● REPLICATION SLAVE and REPLICATION CLIENT are global permissions and must be enabled separately. The reference statement is as follows: **GRANT** REPLICATION SLAVE, REPLICATION CLIENT **ON** *.* **TO** 'user1'; ● SELECT, SHOW VIEW, EVENT, and LOCK TABLES are non-global permissions. The reference statement is as follows: **GRANT** SELECT, SHOW VIEW, EVENT, LOCK TABLES, **ON** [Database to be synchronized].* **TO** 'user1';	The root account of RDS for MySQL has the following permissions by default: SELECT, CREATE, DROP, DELETE, INSERT, UPDATE, ALTER, CREATE VIEW, CREATE ROUTINE, and REFERENCES If the destination database version is in the range 8.0.14 to 8.0.18, the SESSION_VARIABLES_ADMIN permission is required. Reference statement: **GRANT** SELECT, CREATE, DROP, DELETE, INSERT, UPDATE, ALTER, REFERENCES **ON** [*Databases to be migrated*].* TO 'user1';

Function Modules	Source/Service Database	Destination/DR Database
Real-time disaster recovery	The user **root** of the RDS for MySQL instance has the following permissions by default: SELECT, CREATE, ALTER, DROP, DELETE, INSERT, UPDATE, TRIGGER, REFERENCES, SHOW VIEW, EVENT, INDEX, LOCK TABLES, CREATE VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, RELOAD, REPLICATION SLAVE, REPLICATION CLIENT, and WITH GRANT OPTION If the service database version is 8.0.14 to 8.0.18, the SESSION_VARIABLES_ADMIN permission is required. Reference statements: **GRANT** SELECT,CREATE,ALTER,DROP,DELETE,INSERT,UPDATE,TRIGGER,SHOW VIEW,EVENT,INDEX,LOCK TABLES,CREATE VIEW,CREATE ROUTINE,ALTER ROUTINE,CREATE USER,RELOAD,REPLICATION SLAVE,REPLICATION CLIENT **ON** *.* **TO** 'user1';	The user **root** of the RDS for MySQL instance has the following permissions by default: SELECT, CREATE, ALTER, DROP, DELETE, INSERT, UPDATE, TRIGGER, REFERENCES, SHOW VIEW, EVENT, INDEX, LOCK TABLES, CREATE VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, RELOAD, REPLICATION SLAVE, REPLICATION CLIENT, and WITH GRANT OPTION If the DR database version is 8.0.14 to 8.0.18, the SESSION_VARIABLES_ADMIN permission is required. Reference statements: **GRANT** SELECT,CREATE,ALTER,DROP,DELETE,INSERT,UPDATE,TRIGGER,REFERENCES,SHOW VIEW,EVENT,INDEX,LOCK TABLES,CREATE VIEW,CREATE ROUTINE,ALTER ROUTINE,CREATE USER,RELOAD,REPLICATION SLAVE,REPLICATION CLIENT **ON** *.* **TO** 'user1'@'%' **WITH GRANT OPTION**;

NOTE

Run **flush privileges;** after executing the preceding reference statements. Make the authorization take effect.

- Account migration:

  The user must have the SELECT permission of mysql.user if the source database is a non-Alibaba Cloud database. If the source database is an Alibaba Cloud database, the account must have the SELECT permission of mysql.user and mysql.user_view.

  Reference statement:

  GRANT SELECT ON mysql.user **TO** 'user1'@'*host*' ;

  GRANT SELECT ON mysql.user_view **TO** 'user1';

  The destination database users must have the SELECT, INSERT, UPDATE, DELETE, and WITH GRANT OPTION permissions on all databases.

Reference statement: **GRANT SELECT, INSERT, UPDATE, DELETE ON** *.* **TO** 'user1' **WITH GRANT OPTION**

Actions

- Create a user.

  Operation:

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

  · **username**: indicates the account to be created.

  · **host**: indicates the host that allows the account to log in. If the account is allowed to log in to the database from any host, use %.

  · **password**: indicates the password of the account.

  For example, run the following command to grant the drsmigration account with all permissions on all databases and tables and allow the drsmigration account to log in to the database from any host:

  CREATE USER '*drsmigration*'@'%' **IDENTIFIED BY** '*Drs123456*';

- Grant corresponding permissions.

  Operation:

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

  flush privileges;

  · **privileges**: indicates the operation permissions granted to the account, such as SELECT, INSERT, and UPDATE. To grant all permissions to the account, use ALL.

  · **databasename**: indicates the database name. To grant the account with all database operation permissions, use *.

  · **tablename**: indicates table name. To grant the account with all table operation permissions, use *.

  · **username**: indicates the account to be authorized.

  · **host**: indicates the host that allows the account to log in. If the account is allowed to log in from any host, use %.

  · **WITH GRANT OPTION**: indicates that the permission to use the GRANT command is granted to the account. This parameter is optional.

  For example, run the following command to create an account drsmigration with the password Drs123456 and allow the account to log in to the database from any host:

  GRANT ALL ON *.* TO 'drsmigration'@'%';

3.3 How Can I Import Users and Permissions from the Source to the Destination Database?

Step 1 Log in to an ECS that can access the source database.

Step 2 Run the following command, enter the password as prompted, and press **Enter** to export the source database users to the **users.sql** temporary file:

mysql -h *'host'* **-u** *'user'* **-p -N $@ -e** "SELECT CONCAT('SHOW GRANTS FOR ''', user, '''@''', host, ''';') AS query FROM mysql.user" > **/tmp/users.sql**

host indicates the IP address of the source database and ***user*** indicates the username of the source database.

Step 3 Run the following command to export the authorization information of the users from the source database to the **grants.sql** file:

mysql -h *'host'* **-u** *'user'* **-p -N $@ -e "source /tmp/users.sql"** > **/tmp/grants.sql**

sed -i 's/$/;/g' /tmp/grants.sql

host indicates the IP address of the source database and ***user*** indicates the username of the source database.

Step 4 After the preceding command has been executed successfully, open the **grants.sql** file. Information similar to the following is displayed:

```
-- Grants for root@%
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%';

-- Grants for testt@%
GRANT SELECT, INSERT, UPDATE, DELETE ON *.* TO 'testt'@'%';

-- Grants for debian-sys-maint@localhost
GRANT ALL PRIVILEGES ON *.* TO 'debian-sys-maint'@'localhost' WITH GRANT OPTION;

-- Grants for mysql.session@localhost
GRANT SUPER ON *.* TO 'mysql.session'@'localhost';
GRANT SELECT ON `performance_schema`.* TO 'mysql.session'@'localhost';
GRANT SELECT ON `mysql`.`user` TO 'mysql.session'@'localhost';

-- Grants for mysql.sys@localhost
GRANT USAGE ON *.* TO 'mysql.sys'@'localhost';
GRANT TRIGGER ON `sys`.* TO 'mysql.sys'@'localhost';
GRANT SELECT ON `sys`.`sys_config` TO 'mysql.sys'@'localhost';

-- Grants for root@localhost
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION;
GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION;
```

Step 5 The information displayed in **Step 4** shows all users of the source database and their permissions. Add the required users one by one to the RDS for MySQL database on the current cloud. For details, see the **Creating an Account** section in the *Relational Database Service User Guide*.

----End

3.4 Why Cannot Scheduled DRS Tasks Be Started?

If you use a member account to create a DRS task, your scheduled tasks may fail, including automatic startup, completion, resumable transfer, and comparison. To rectify the fault, you can use an agency to create a task.

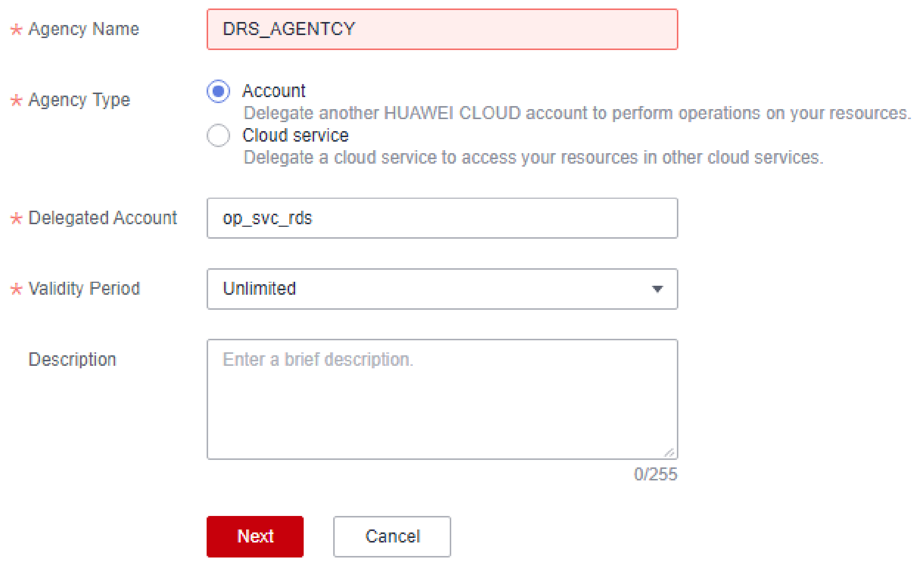
Solution

- Method 1: Use the master account to create a task again because the master account has the Security Administrator permission by default. After the task is created using the master account, an agency is created.
- Method 2: Use the master account to add the Security Administrator permission to the user group to which the member account belongs, and

create a task again. For details about how to add permissions, see **Creating a User Group and Assigning Permissions**.

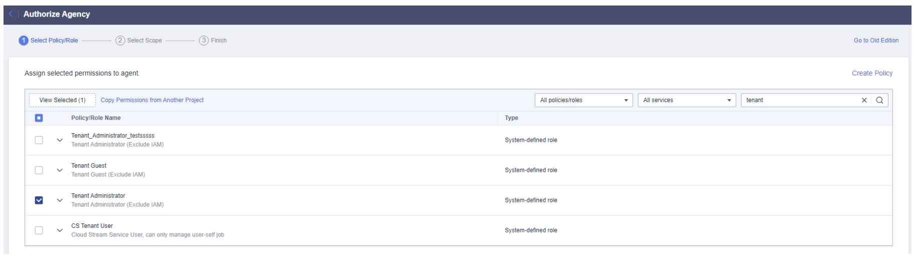
- Method 3: Manually add an agency. The procedure is as follows:
  a. Log in to HUAWEI CLOUD and click **Console** in the upper right corner.
  b. On the management console, hover the mouse pointer over the username in the upper right corner, and choose **Identity and Access Management** from the drop-down list.
  c. In the navigation pane on the left, click **Agencies**.
  d. In the upper right corner, click **+Create Agency**.
  e. Enter **DRS_AGENTCY** in field **Agency Name**. If you select **Account** for **Agency Type**, enter **op_svc_rds** in field **Delegated Account**. If you select **Cloud service** for **Agency Type**, select **MySQL** for **Delegated Account**. Select **Unlimited** for **Validity Period** and then click **Next**.

Figure 3-1 Creating an agency


* Agency Name	DRS_AGENTCY
* Agency Type	[✓ Account] Delegate another HUAWEI CLOUD account to perform operations on your resources. Cloud service Delegate a cloud service to access your resources in other cloud services.
* Delegated Account	op_svc_rds
* Validity Period	Unlimited
Description	Enter a brief description. 0/255

  f. On the **Select Policy/Role** page, select **Tenant Administrator** and click **Next**.

Figure 3-2 Select Policy

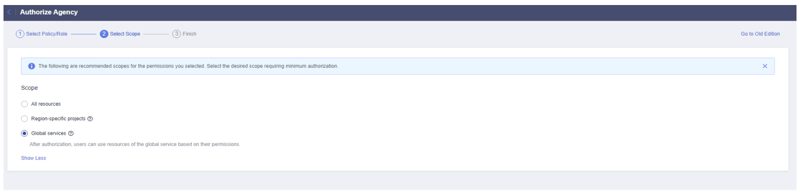


Assign selected permissions to agent.

	Policy/Role Name	Type
	Tenant_Administrator_testsssss Tenant Administrator (Exclude IAM)	System-defined role
	Tenant Guest Tenant Guest (Exclude IAM)	System-defined role
✓	Tenant Administrator Tenant Administrator (Exclude IAM)	System-defined role
	CS Tenant User Cloud Stream Service User, can only manage user-self job	System-defined role

  g. Select the authorization for global services and then region-specific projects, and click **OK**.

Figure 3-3 Authorization for global services



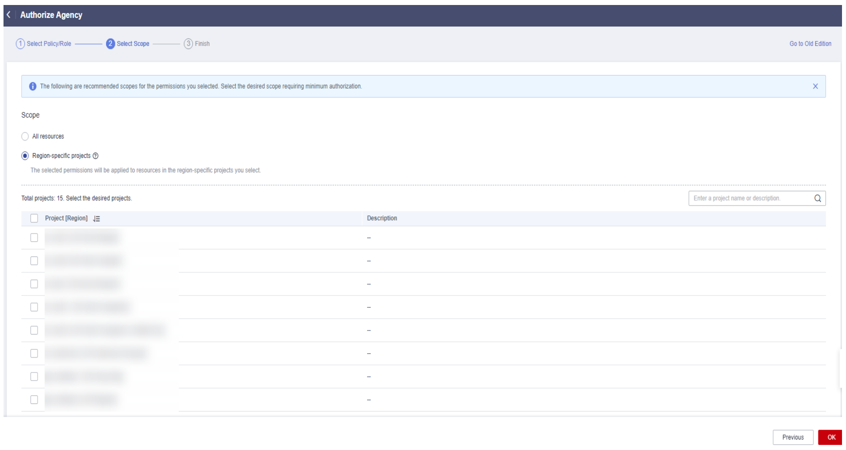
The following are recommended scopes for the permissions you selected. Select the desired scope requiring minimum authorization.

Scope

All resources [ ]
Region-specific projects [ ]
[✓ Global services]

After authorization, users can use resources of the global service based on their permissions.

Figure 3-4 Authorization for region-specific projects



The following are recommended scopes for the permissions you selected. Select the desired scope requiring minimum authorization.

Scope

All resources [ ]
[✓ Region-specific projects]

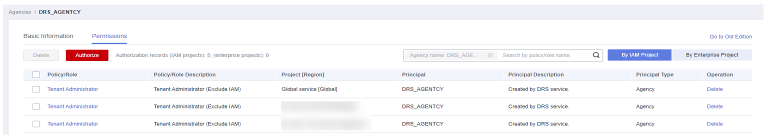
The selected permissions will be applied to resources in the region-specific projects you select.

Total projects: 15. Select the desired projects.

Project [Region]	Description
[redacted]	--
[redacted]	--
[redacted]	--
[redacted]	--
[redacted]	--
[redacted]	--
[redacted]	--
[redacted]	--

h. Click the agency name. On the **Permissions** tab, you can view permissions for global services and region-specific projects.

Figure 3-5 Permissions



Authorization records (IAM projects): 5; (enterprise projects): 0

Policy/Role	Policy/Role Description	Project [Region]	Principal	Principal Description	Principal Type	Operation
Tenant Administrator	Tenant Administrator (Exclude IAM)	Global service [Global]	DRS_AGENTCY	Created by DRS service.	Agency	Delete
Tenant Administrator	Tenant Administrator (Exclude IAM)	[redacted]	DRS_AGENTCY	Created by DRS service.	Agency	Delete
Tenant Administrator	Tenant Administrator (Exclude IAM)	[redacted]	DRS_AGENTCY	Created by DRS service.	Agency	Delete

i. The authorization takes effect after 15 to 30 minutes. After the authorization takes effect, create the task again.

4 Real-Time Migration

4.1 When Can I Stop a Migration Task?

You can refer to the following methods to check whether the task can be stopped. Before stopping the task, ensure that:

1. At least one complete data comparison is performed during off-peak hours.
2. Service cutover is completed.
   a. Interrupt services first. If the workload is not heavy, you may do not need to interrupt the services.
   b. Run the following statement on the source database (MySQL is used as an example) and check whether there are statements executed by new sessions within 1 to 5 minutes. If not, the service is stopped.
      ```
      show processlist;
      ```
      NOTE

      The process list queried by the preceding statement includes the connection of the DRS replication instance. If no additional session executes SQL statements, the service has been stopped.
   c. When the real-time synchronization delay is 0s and remains stable for a period, you can perform a data-level comparison between the source and destination databases. For details about the time required, refer to the comparison results of the previous comparison.
      - If there is enough time, compare all objects.
      - If there is not enough time, use the data-level comparison to compare the tables that are frequently used and that contain key business data or inconsistent data.
   d. Determine a proper time to cut the services over to the destination database. Then, services can be used externally again.
3. Stopping a task only deletes the replication instance, and the migration task is still in the task list. You can choose whether to delete the task.

4.2 How Do I Maintain the Original Service User Permission System After Definer Is Forcibly Converted During MySQL Migration?

Definer is used in views, stored procedures, triggers, and events. Definer does not restrict the permission to invoke objects, instead the permission to access the database. If you select **Yes** for **Migrate Definer to User** during MySQL migration, the Definers of all source database objects will be migrated to the user. The user continues to use the original services without authorization. (Users, permissions, and passwords are migrated). Other users do not have permissions on database objects unless these users are authorized.

The following procedures describe how to use database commands to authorize users.

Step 1 Ensure that the new user (Definer uses the specified account) has sufficient permission to execute view- and stored procedure-related SQL statements.

Step 2 Log in to the destination database using the MySQL official client or other tools.

Step 3 Run the following command to view details about permissions of the user to be authorized:

```
show grants for  'user'@'host';
```

Step 4 To ensure that the original service does not report an error, run the following command to grant the user the operation permissions the involved database objects do not have:

```
grant select,insert,update,delete on db_name.* to 'user'@'host';
```

Generally, the permissions to access the database are as follows: SELECT, CREATE, DROP, DELETE, INSERT, UPDATE, INDEX, EVENT, CREATE VIEW, CREATE ROUTINE, TRIGGER, and EXECUTE. You need to check the permissions that are missing based on the database object, and then perform the authorization operation.

For stored procedures and functions, ensure that the user has the EXECUTE permission. The authorization command is as follows:

```
grant execute on db_name.function_name to 'user'@'host';
```

Step 5 Use the authorized account to access the destination database. If the access is successful, the authorization is successful. Note: If the following information is displayed when a stored procedure or function is invoked in a Java project, the **mysql.proc** database must be authorized: Java.sql.SQLException: User does not have access to metadata required to determine stored procedure parameter types. If rights can not be granted, configure connection with "noAccessToProcedureBodies=true" to have driver generate parameters that represent INOUT strings irregardless of actual parametertypes

```
grant select on mysql.proc to 'user'@'host';
```

----End

4.3 What Can I Do If the Invoking Permission Problem Occurs After the MySQL Stored Procedure Is Migrated to the Cloud?

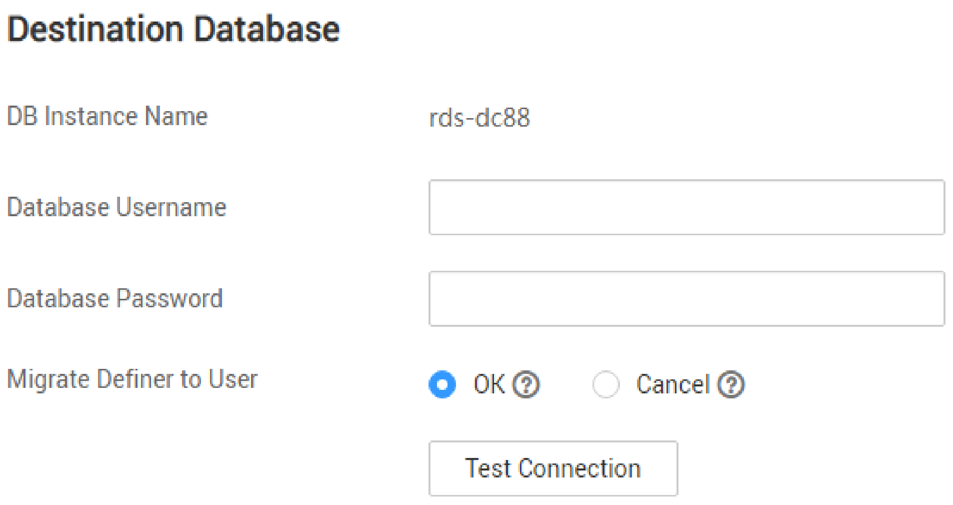
After the MySQL stored procedure is migrated to the cloud, an error may occur when the stored procedure or function is invoked due to permission problems.

The method varies with Definer policies. This section uses user1 as an example to describe how to solve this problem in two Definer policies.

Policy 1

On the **Destination Database** page, enter the database username **user1**, and select **OK** for **Migrate Definer to User**.

Figure 4-1 Policy 1


Destination Database

DB Instance Name	rds-dc88
Database Username	
Database Password	
Migrate Definer to User	[✓ OK] Cancel

In this policy, after the Definers of all stored procedures and methods in the source database are migrated to the destination database, the account is automatically changed to user1, and the value of host is automatically changed to %. If a stored procedure fails to be invoked in the destination database, perform the following operations:

Step 1 Log in to the RDS for MySQL DB instance of the destination database as the user1.

Step 2 Grant the execute permission to the account that you want to use to invoke a stored procedure.

Step 3 Run the following statement to use user1 to grant other accounts the permission to execute stored procedures:

user indicates other accounts that need to invoke the stored procedure.

```
GRANT EXECUTE ON db.* TO user;
```

Step 4 To invoke a stored procedure using Java, run the following statement to use user1 to grant other accounts the permission to query the **mysql.proc** table:

The following is the authorization statement, in which **user** indicates the account that needs to invoke the stored procedure:

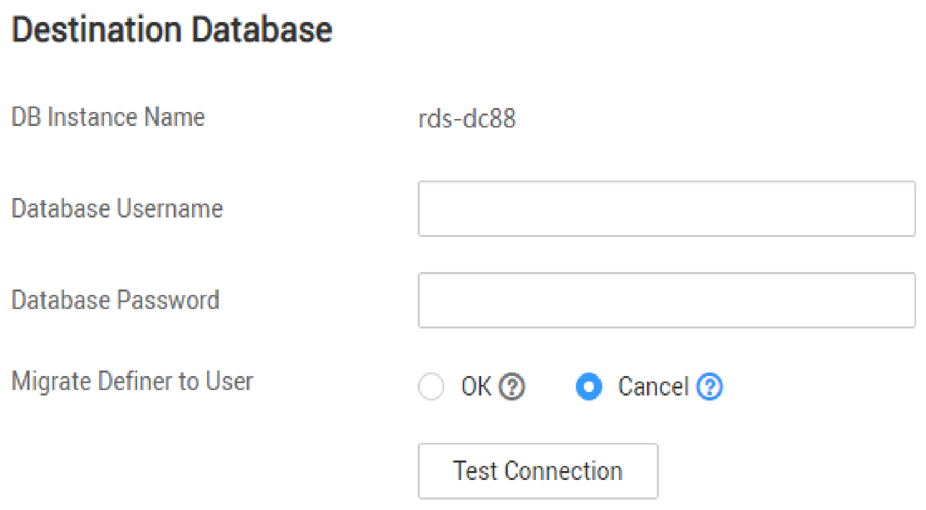
```
GRANT SELECT ON mysql.proc TO 'user'@'%';
```

----End

Policy 2

On the **Destination Database** page, enter the database username **user1**, and select **Cancel** for **Migrate Definer to User**.

Figure 4-2 Policy 2



Destination Database

DB Instance Name	rds-dc88
Database Username	
Database Password	
Migrate Definer to User	OK [✓ Cancel]

In this policy, the account and host in the source database remain unchanged after the Definers of all stored procedures and methods are migrated to the destination database. You need to migrate all users in the source database by referring to **Migrating Accounts**. In this way, the permission system of the source database remains unchanged.

If you do not migrate account permissions or some accounts cannot be migrated, you are advised to use **Policy 1**.

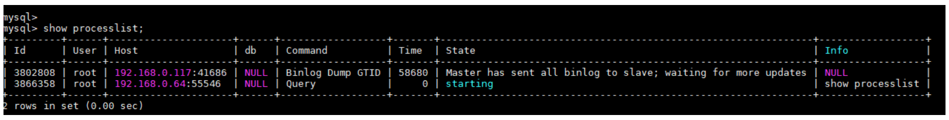
4.4 How Do I Ensure that All Services on the Database Are Stopped?

To ensure that all services on the database are stopped, perform the following steps:

Step 1 Run the following statement on the source database to check whether active connections exist:

```
show processlist;
```

Figure 4-3 Checking active connections



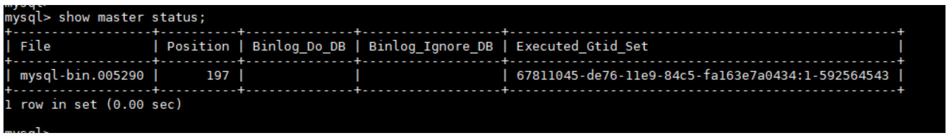
```
mysql>
mysql> show processlist;
+---------+------+---------------------+------+------------------+-------+-----------------------------------------------------------------+-------------------+
| Id      | User | Host                | db   | Command          | Time  | State                                                           | Info              |
+---------+------+---------------------+------+------------------+-------+-----------------------------------------------------------------+-------------------+
| 3802808 | root | 192.168.0.117:41686 | NULL | Binlog Dump GTID | 58680 | Master has sent all binlog to slave; waiting for more updates   | NULL              |
| 3866358 | root | 192.168.0.64:55546  | NULL | Query            |     0 | starting                                                        | show processlist  |
+---------+------+---------------------+------+------------------+-------+-----------------------------------------------------------------+-------------------+
2 rows in set (0.00 sec)
```

Step 2 **Optional:** If there are active connections, locate the service processes based on the values in the **Host** column in the command output and stop the service processes.

Step 3 Run the following statement in the source database to check the binlog position. Then, record the two values in the **file** and **position** columns as **ckpt1**:

```
show master status;
```

Figure 4-4 Viewing the binlog position



```
mysql> show master status;
+------------------+----------+--------------+------------------+---------------------------------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set                           |
+------------------+----------+--------------+------------------+---------------------------------------------+
| mysql-bin.005290 |      197 |              |                  | 67811045-de76-11e9-84c5-fa163e7a0434:1-592564543 |
+------------------+----------+--------------+------------------+---------------------------------------------+
1 row in set (0.00 sec)
```

Step 4 Wait for more than 30s. Run the following statement in the source database to check the binlog position again. Then, record the two values in the **file** and **position** columns as **ckpt2**. If **ckpt1** and **ckpt2** are equal, no data is written to the source database.

```
show master status;
```

----End

4.5 What Can I Do When Message "can not get agency token" Is Displayed in the Migration Log

Possible Causes

The subaccount used for creating a task does not have an agency. As a result, automatic functions such as scheduled task startup fail to be executed. The typical scenarios are as follows:

- When creating a task and setting the task to start as scheduled, you need to use **account entrustment**. Otherwise, the scheduled task fails to be started, leaving message "can not get agency token".
- After a full+incremental task is started and the full migration is complete, you need to use **account entrustment**. Otherwise, the task will not enter the incremental phase, leaving message "can not get agency token".

Solution

Two solutions are provided as follows:

- Method 1: Use your primary account to create a task and select **Start at a specified time** for **Start Time**.

- Method 2: Use the primary account to add the Security Administrator permission to the user group to which the subaccount belongs. Then, create a task again, and select **Start at a specified time** for **Start Time**.
- Method 3: Create a task again and select **Start upon task creation** for **Start Time**.

4.6 What Do I Do If the Maximum Index Length Has Been Reached During Migration from Oracle to MySQL?

Index Length

The maximum length of each MySQL index is limited, which depends on the type of DB engine and character set. For example, each UTF8 character set contains a maximum of three bytes, and each UTF8MB4 character set contains a maximum of 4 bytes.

- For a column index, the length of each column cannot exceed the value given in **Max. Characters of a Column Index** in **Table 4-1**. For example, in MySQL 5.7.6 that uses the InnoDB storage engine, the length of a column index cannot exceed 767 bytes (Max. Characters of a Column Index = 767 / Max. Bytes).
- For a multiple-column index, the length of each column cannot exceed the value given in **Max. Characters of a Column Index** in **Table 4-1**, and the total length of columns cannot exceed the value given in **Max. Characters of a Multiple-Column Index**. For example, in MySQL 5.7.6 that uses the InnoDB storage engine, each column index length cannot exceed 767 bytes (Max. Characters of a Column Index = 767 / Max. Bytes), and the total column index length cannot exceed 3072 bytes (Max. Characters of a Column Index = 3072 / Max. Bytes)

Table 4-1 Index length description

Storage Engine	MySQL Version	Character Set	Max. Bytes	Max. Characters of a Column Index	Max. Characters of a Multiple-Column Index
InnoDB	MySQL 5.7.6 or earlier	UTF8MB4	4	191	768
	MySQL 5.7.7 or later	UTF8MB4	4	768	768

Solution

- Method 1

  Do not migrate tables that contain indexes of which the maximum length has been reached.

- Method 2

  Changes to index length may cause data consistency problems. Exercise caution when performing this operation. For example, if the destination database version is MySQL 5.7.6 or earlier and uses UTF8MB4 character set, run the following command to change the index length:

  ```
  alter table tablename modify columnname varchar2 (768) ;
  ```

Replace **tablename** with the actual table name and **columnname** with the actual column name.

- Method 3

  Delete the index and its constraints from the source database. For example, if the destination database version is MySQL 5.7.6 or earlier and uses UTF8MB4 character set, run the following commands to delete the index and its constraints:

  ```
  drop index indexname;
  alter table tablename drop constraint constraintname;
  ```

  Replace **indexname** with the actual index name, **tablename** with the actual table name, and **constraintname** with the actual constraint name.

4.7 Why Is the Collation of Heterogeneous or Oracle Databases Converted to utf8mb4_bin After Those Databases Are Migrated to MySQL?

Different databases support different types of character sets. After databases such as Oracle databases are migrated to the MySQL database, their encoding will be converted to the UTF8MB4 character set to support more bytes per character. The default collation of UTF8MB4 is utf8_general_ci, which is case insensitive. That is, abc and ABC are the same data. The migration from case-sensitive databases such as Oracle databases to MySQL databases may cause migration failures (primary key conflicts) or have impact on services (incorrect query results). You can refer to the following suggestions:

1. When DRS is used to perform heterogeneous migration or synchronization to MySQL, DRS automatically sorts the character set of the database (only the database to be migrated) with the utf8mb4_bin collation. This ensures that collation settings of new tables and columns in the same database are the same as those of the migrated tables and columns, facilitating associated queries and index queries.

2. You can set **collation_server** to **utf8mb4_bin** at the instance level. This setting ensures that the default character set of all instances is utf8mb4_bin, you can set this parameter based on service requirements.

4.8 What Can I Do If MyISAM Tables Are Not Supported by RDS for MySQL?

Currently, RDS for MySQL does not support the MyISAM engine due to the following reasons.

- MyISAM engine tables do not support transactions and support only table-level locks. As a result, read and write operations conflict with each other.
- MyISAM has a defect in protecting data integrity, which may cause database data damage or even data loss.
- If data is damaged, MyISAM does not support data restoration provided by RDS for MySQL and requires manual restoration.
- Data can be transparently migrated from MyISAM to InnoDB, which does not require code modification for tables.

During migration, DRS automatically converts MyISAM to InnoDB. The MyISAM engine table does not support transactions. To ensure data consistency of the MyISAM table, DRS uses primary keys to ensure final data consistency. If you need to migrate MyISAM tables without primary keys, you are advised to start the migration task when no service is running to ensure data consistency.

4.9 What Are the Precautions for Migrating Data from an Earlier Version MySQL to MySQL 8.0?

Based on MySQL 5.7, some new features have been added to MySQL 8.0. There are performance differences between the two versions. Before migration, you need to analyze compatibility and provide a corresponding solution. The following shows the analysis:

- Compatibility analysis

  MySQL 8.0 and MySQL 5.7 Community Edition are analyzed as follows:

  a. Compatibility does not affect migration, but the solutions are different.

Compatibility	Check Item	Function	Status	Solution
Data types or functions	ENCODE()	Encryption	Deleted	Replaced by AES_ENCRYPT()
	DECODE()	Decryption	Deleted	Replaced by AES_DECRYPT()
	ENCRYPT()	Encryption	Deleted	Replaced by SHA2()

Compatibility	Check Item	Function	Status	Solution
	DES_ENCRYPT()	Encryption	Deleted	Replaced by AES_ENCRYPT()
	DES_DECRYPT()	Decryption	Deleted	Replaced by AES_DECRYPT()
	JSON_APPEND()	Adds JSON elements.	Deleted	Replaced by JSON_ARRAY_APPEND()
	PASSWORD()	Changes a user password.	Deleted	ALTER USER user IDENTIFIED BY 'auth_string';
	JSON_MERGE()	Merges multiple JSONs.	Discarded	Replaced by JSON_MERGE_PERSERVE()
SQL MODE	NO_AUTO_CREATE_USER, DB2, MAXDB, MSSQL, MYSQL323, MYSQL40, ORACLE, POSTGRESQL, NO_FIELD_OPTIONS, NO_KEY_OPTIONS, NO_TABLE_OPTIONS	-	Deleted	-

Compatibility	Check Item	Function	Status	Solution
Foreign key constraint length	The constraint name cannot be greater than 64 characters.	-	-	SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME IN (SELECT LEFT(SUBSTR(ID,INSTR(ID,'/')+1), INSTR(SUBSTR(ID,INSTR(ID,'/')+1),'_ibfk_')-1) FROM INFORMATION_SCHEMA.INNODB_SYS_FOREIGN WHERE LENGTH(SUBSTR(ID,INSTR(ID,'/')+1))>64); Use the ALTER TABLE statement to adjust the length.
Features	Use the GRANT statement to create users.	-	Deleted	CREATE USER
	Use the GRANT statement to modify user information.	-	Deleted	ALTER USER
	IDENTIFIED BY PASSWORD 'auth_string'	Sets new passwords	Deleted	IDENTIFIED WITH auth_plugin AS 'auth_string'
	\N in a SQL statement	NULL	Deleted	Replaced by NULL

Compatibility	Check Item	Function	Status	Solution
	PROCEDURE ANALYSE() syntax	Specifies the recommended field type is provided after the MySQL field value is analyzed.	Deleted	-
	Spatial functions	-	-	-
	mysql_install_db	Initialization	Deleted	mysqld --initialize or --initialize-insecure

b. The following items affect the migration. You need to check in advance.

Compatibility	Check Item	Function	Status	Solution	Original Usage
Reserving keywords	cume_dist, dense_rank, empty, except, first_value, grouping, groups, json_table, lag, last_value, lateral, lead, nth_value, ntile, of, over, percent_rank, rank, recursive,row_number, system, window	-	Added	SET sql_mode = 'ANSI_QUOTES'	Name: database, table, index, column, alias, view, stored procedure, partition, and tablespace

Compatibility	Check Item	Function	Status	Solution	Original Usage
Character set	UTF8MB3	-	Discarded	Replaced by UTF8MB4.	-

Partition table name	Partition tables of storage engines that do not support local partitions are not allowed.	-	Deleted	SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE ENGINE NOT IN ('innodb', 'ndbcluster') AND CREATE_OPTIONS LIKE '%partitioned%';
				You can use either of the following methods:
				(1) ALTER TABLE table_name ENGINE=INNODB;
				(2) ALTER TABLE table_name REMOVE PARTITIONING;
					MyISAM is not supported.

Syntax	group by... asc/desc	Ascending/Descending	Deleted	Replaced by the ORDER By clause.	View and function
Name length	The view name cannot be greater than 64 characters.	-	-	ALTER	The value can contain a maximum of 255 characters.
	The enum or set element contains a maximum of 255 characters.	-	-	Handled by users.	The value can contain a maximum of 64 KB.

Compatibility	Check Item	Function	Status	Solution	Original Usage

Upper and lower case letters	lower_case_table_names	Specifies whether to set the MySQL table name case sensitive.	-	If this parameter is set to **1** during the upgrade, ensure that the schema and table names are in lowercase.
				SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME != LOWER(TABLE_NAME) AND TABLE_TYPE = 'BASE TABLE';
				SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME != LOWER(SCHEMA_NAME);
					-

Triggers	Check whether there is an empty definition or invalid creation context.	-	-	Use the SHOW TRIGGERS statement to check the character_set_client, collation_connection, and Database Collation attributes.	-

- Change the default value of the system variable.

  The analysis of default values of MySQL 5.7 and MySQL 8.0 Community Edition shows that default values do not affect the migration but affect services after the migration.

No.	Parameter/Option	Community		Function	Remarks
		Original Default Value	New Default Value		
Server					

1	character_set_server	latin1	utf8mb4	-	Be consistent with the origin default value.
2	collation_server	latin1_swedish_ci	utf8mb4_0900_ai_ci	-	Be consistent with the origin default value.
3	explicit_defaults_for_timestamp	OFF	ON	Specifies whether to update the timestamp column when a row is updated.	Be consistent with the origin default value.
4	optimizer_trace_max_mem_size	16KB	1MB	-	Be consistent with the origin default value.
5	validate_password_check_user_name	OFF	ON	-	Be consistent with the origin default value.

6	back_log	-1 (autosize) changed from : back_log = 50 + (max_connections / 5)	-1 (autosize) changed to : back_log = max_connections	Specifies the number of requests that can be stored in the stack in a short period before the MySQL database stops responding to new requests.	Be consistent with the origin default value.
7	max_allowed_packet	4194304 (4MB)	67108864 (64MB)	Limits the size of data packets received by the server	Use the default value.
8	max_error_count	64	1024	Controls the number of alarms to be displayed.	Be consistent with the origin default value.
9	event_scheduler	OFF	ON	-	Be consistent with the origin default value.
10	table_open_cache	2000	4000	-	Be consistent with the origin default value.
11	log_error_verbosity	3 (Notes)	2 (Warning)	-	Use the default value.
INNODB					

1	innodb_undo_tablespaces	0	2	-	Use the default value.
2	innodb_undo_log_truncate	OFF	ON	-	Use the default value.
3	innodb_flush_method	NULL	fsync (Unix), unbuffered (Windows)	Controls the enabling and writing modes of InnoDB data files and redo logs.	Use the default value **O_DIRECT** for SQL.
4	innodb_autoinc_lock_mode	1 (consecutive)	2 (interleaved)	Controls the behavior of related locks when data is inserted into a table with the **auto_increment** column.	Be consistent with the origin default value.
5	innodb_flush_neighbors	1 (enable)	0 (disable)	Checks whether other dirty pages in the same range are refreshed when refreshing the page from the buffer pool.	Be consistent with the origin default value.
6	innodb_max_dirty_pages_pct_lwm	0 (%)	10 (%)	Affects the InnoDB dirty page refreshing operation.	Use the default value.

7	innodb_max_dirty_pages_pct	75 (%)	90 (%)	Affects the InnoDB dirty page refreshing operation.	Use the default value.
PERFORMANCE SCHEMA	Enabled globally.	-	-	-	Be consistent with the origin default value.
REPLICATION					
1	log_bin	OFF	ON	-	Enabled by default
2	server_id	0	1	-	If the value is **0**, change it to **1**.
3	log-slave-updates	OFF	ON	-	Enabled by default.
4	expire_log_days	0	30	-	Use the default value.
5	master-info-repository	FILE	TABLE	-	Use the default value **TABLE**.
6	relay-log-info-repository	FILE	TABLE	-	Use the default value **TABLE**.
7	transaction-write-set-extraction	OFF	XXHASH64	-	Use the default value.
8	slave_rows_search_algorithms	INDEX_SCAN, TABLE_SCAN	INDEX_SCAN, HASH_SCAN	-	Use the default value.

- Remove system variables.

  The analysis of MySQL 5.7 and 8.0 Community Edition shows that removing system variables does not affect migration.

System variables
innodb_locks_unsafe_for_binlog
log_builtin_as_identified_by_password
old_passwords
query_cache_limit
query_cache_min_res_unit
query_cache_size
query_cache_type
query_cache_wlock_invalidate
ndb_cache_check_time
ignore_db_dirs
tx_isolation
tx_read_only
sync_frm
secure_auth
multi_range_count
log_error_verbosity
sql_log_bin
metadata_locks_cache_size
metadata_locks_hash_instances
date_format
datetime_format
time_format
max_tmp_tables
ignore_builtin_innodb
innodb_support_xa
innodb_undo_logs
innodb_undo_tablespaces
internal_tmp_disk_storage_engine

4.10 What Can I Do When OOM Occurs During the Migration of MongoDB Databases?

Scenarios

Out of memory (OOM) occurs during the migration of MongoDB databases, causing migration failures.

Possible Cause

The possible causes are as follows:

- If the mongod service of the source database is deployed on a single server, OOM occurs when the migration process consumes large amounts of memory through operations such as creating indexes and sorting queries.
- If the mongod service is deployed on a server with other services and the **cacheSizeGB** value is not specified, OOM occurs when all available memory has been allocated to other services, so the WiredTiger engine does not have sufficient memory.

  NOTE

  By default, the memory used by the WiredTiger engine of mongod is calculated as follows: (Memory in GB-1) x 50% for version 3.2 or (Memory in GB-1) x 50% for version 3.4 and later.

Solution

- If the mongod service is deployed on a single server, do not perform any operations that consume large amounts of memory during the migration.
- If the mongod service and other services are deployed on the same server, set the value of **cacheSizeGB** to the half of the minimal idle memory to ensure that memory used in peak hours will not be allocated to WiredTiger excessively.

4.11 How Do I Disable the Balancer?

Before using the DRS service to migrate collections between sharded clusters, you must disable the balancer of the collections to be migrated.

NOTE

- You can disable the DDS cluster balancer by calling an API. For details, see **Enabling or Disabling Cluster Balancing** or contact DDS technical support. For a self-built MongoDB database, **refer to the following steps**.
- After the migration is complete, enable the balancer. The balancer is disabled during the migration, generating different numbers of chunks on each shard of the source database. After the balancer is enabled, chunks will be distributed between shards in the cluster, which may affect the performance of the source database.

Procedure

Step 1 Log in to a database through mongo shell.

Step 2 Run the following command in the command window of the mongos node to switch to the config database:

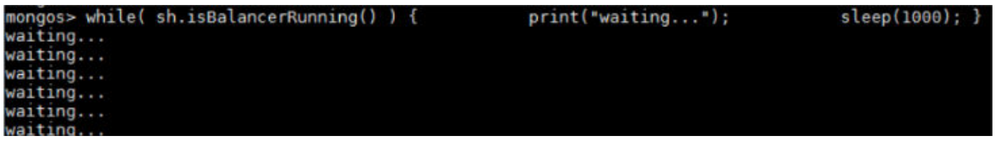
use config

Step 3 Run the following commands to check whether the balancer can be disabled:

```
while( sh.isBalancerRunning() ) {
        print("waiting...");
        sleep(1000);
}
```

- If the command output is **waiting**, the balancer is migrating chunks. In this case, do not disable the balancer. Otherwise, data inconsistency may occur.

Figure 4-5 Viewing the command output



```
mongos> while( sh.isBalancerRunning() ) {          print("waiting...");          sleep(1000); }
waiting...
waiting...
waiting...
waiting...
waiting...
waiting...
```

- If no command output is displayed, the balancer is not migrating any chunks. In this case, you can disable the balancer:

Step 4 Disable the balancer.

- If you migrate the entire DB instance, run the following command to disable the balancer.
```
sh.stopBalancer()
```
- If you need to disable the balancer of the sharded collections to be migrated, run the following command:
```
sh.disableBalancing("database.collection")
```
database.collection indicates the namespace of the collection to be migrated.

----End

4.12 How Do I Export and Import Events and Triggers in Batches?

During the MySQL to MySQL migration, if the migration log indicates that the migration of events and triggers fails after the migration task is complete, you can manually migrate the events and triggers.

This section describes how to export and import events and triggers in batches.

Step 1 Export triggers from the source database in batches.

1. Run the following statement in the source database to obtain values of **TRIGGER_SCHEMA** and **TRIGGER_NAME**:
```
SELECT TRIGGER_SCHEMA,TRIGGER_NAME  FROM INFORMATION_SCHEMA.TRIGGERS
WHERE TRIGGER_SCHEMA in ('DB1','DB2','DB3') order by TRIGGER_NAME;
```
In the preceding statements, **DB1**, **DB2**, and **DB3** indicate the databases to be migrated to the destination database.

2. Run the following statement in the source database to obtain the statement for creating a trigger from the source database from the **SQL Original Statement** field:

```
SHOW CREATE TRIGGER TRIGGER_SCHEMA.TRIGGER_NAME \G;
```

In the preceding statement, replace **TRIGGER_SCHEMA** and **TRIGGER_NAME** with the values obtained in **Step 1.1**.

Step 2 Export events from the source database in batches.

1. Run the following statement in the source database to obtain values of **EVENT_SCHEMA** and **EVENT_NAME**:

```
SELECT EVENT_SCHEMA,EVENT_NAME  FROM INFORMATION_SCHEMA.EVENTS WHERE
EVENT_SCHEMA in ('DB1','DB2','DB3') order by EVENT_NAME;
```

In the preceding statements, **DB1**, **DB2**, and **DB3** indicate the databases to be migrated to the destination database.

2. Run the following statement in the source database to obtain the statement for creating an event from the source database from the **SQL Original Statement** field:

```
SHOW CREATE EVENT EVENT_SCHEMA.EVENT_NAME \G;
```

In the preceding statement, replace **EVENT_SCHEMA** and **EVENT_NAME** with the values obtained in **Step 2.1**.

Step 3 Import triggers and events.

Execute the statements for creating triggers and events exported from the source database in the destination database.

----End

4.13 How Can I Migrate Databases or Tables Whose Names Contain Uppercase Letters?

Scenarios

When the value of source database parameter **lower_case_table_names** is set to **1**, the databases or tables whose names contain uppercase letters cannot be migrated.

Possible Cause

When the value of **lower_case_table_names** in the source database is **1**, the MySQL engine converts the database name or table name into lowercase letters. In this case, the database or table may not be found, resulting in query failure. Simply, if the value of **lower_case_table_names** is **1**, the database or table containing uppercase letters may be inaccessible.

Solutions

Two solutions are provided as follows:

Solution 1

Change the value of **lower_case_table_names** in the source database to **0** (case-sensitive) and ensure that the value of this parameter in the source database is the same as that in the destination database.

Solution 2

If the value of **lower_case_table_names** cannot be changed permanently, change the value to **0**, and then perform the following operations:

- For a table, you can use the following statement to convert the table name to lowercase:
  ```
  alter table `BigTab` rename to `bigtab`
  ```
- For a database, you need to export the database data, change the database name from uppercase to lowercase, and then import the data.

CAUTION

After changing the database name or table name, you need to maintain the permission consistency without affecting application access.

Method 3

Do not migrate the databases or tables that contain uppercase letters.

4.14 How Do I Delete Orphaned Documents in MongoDB Sharded Clusters?

What Is Orphaned Document?

In a sharded cluster, orphaned documents are those documents on a shard that also exist in chunks on other shards as a result of failed migrations or incomplete migration cleanup due to abnormal shutdown.

Migration Impact

During cluster migration, DRS extracts full data from shards. Normal documents and orphaned documents are on different shards and DRS will migrate them all. If the conflict policy of DRS for MongoDB migration is **Ignore**, documents that are first migrated to the destination are stored, resulting in data inconsistency.

Procedure

Step 1 Download **cleanupOrphaned.js**.

Step 2 Modify the **cleanupOrphaned.js** script file and replace **test** with the database name of the orphaned document to be cleared.

Step 3 Run the following command to clear the orphaned documents of all collections in the specified database on the shard node:

```
mongo --host ShardIP --port Primaryport --authenticationDatabase database -u username -p password
cleanupOrphaned.js
```

NOTE

- **ShardIP**: indicates the IP address of the shard node.
- **Primaryport**: indicates the service port of the primary shard node.
- **database**: indicates the database name.
- **username**: indicates the username for logging in to the database.
- **password**: indicates the password for logging in to the database.

NOTE

If you have multiple databases, repeat **Step 2** and **Step 3** to clean up orphaned documents in each database on each shard node.

----End

5 Backup Migration

5.1 What Should I Do If the Last Backup File Is Incorrectly Selected in the Backup Migration Scenario?

During the backup migration, If **Last Backup File** is selected by mistake, perform either of the following operations:

- If you select **Yes** by mistake, the database receives a signal that the restore is complete, and then sets the database to available, making incremental backup migration impossible. In this case, you can only delete the backup database and perform full and incremental backup restoration again.
- SQL Server does not have the last backup file in a strict sense. If you select **No** by mistake, you can perform an incremental backup (even if no data is changed). During the incremental backup, select **Yes** to complete the migration. The related database becomes available.

5.2 Manual Configuration

Scenarios

After data is migrated from the local host or VMs to the RDS SQL Server DB instance on the current cloud through DRS, the Login accounts, DBLink, AgentJobs, and key configurations of the source database also need to be synchronized to the destination database.

Login Account

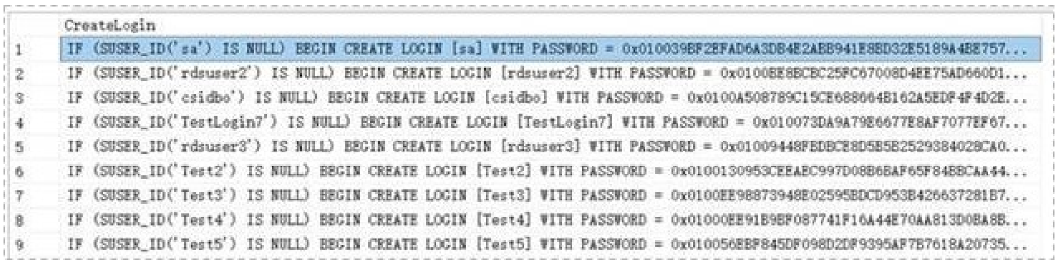
Login account is an instance-level account of Microsoft SQL Server and is used to manage user server and database permissions. Generally, a user has multiple such accounts. After the user is migrated to the RDS SQL Server DB instance, you need to manually create corresponding Login accounts on the DB instance. The following describes how to create a Login account with the same name and password as those of your local Login account on the RDS SQL Server DB instance and grant permissions to the account.

Step 1 Execute the following script to obtain the script for creating a Local account on your local instance. The obtained script can be directly executed on the destination DB instance to create a Login account with the same name and password.

```
SELECT 'IF (SUSER_ID('+QUOTENAME(SP.name,'''')+') IS NULL) BEGIN CREATE LOGIN '
+QUOTENAME(SP.name)+
CASE
WHEN SP.type_desc = 'SQL_LOGIN' THEN ' WITH PASSWORD = '
+CONVERT(NVARCHAR(MAX),SL.password_hash,1)+ ' HASHED,SID='
+CONVERT(NVARCHAR(MAX),SP.SID,1)+',CHECK_EXPIRATION = '
+ CASE WHEN SL.is_expiration_checked = 1 THEN 'ON' ELSE 'OFF' END +', CHECK_POLICY = '
+CASE WHEN SL.is_policy_checked = 1 THEN 'ON,' ELSE 'OFF,' END
ELSE ' FROM WINDOWS WITH'
END
+' DEFAULT_DATABASE=[' +SP.default_database_name+ '], DEFAULT_LANGUAGE=['
+SP.default_language_name+ '] END;' as CreateLogin
FROM sys.server_principals AS SP LEFT JOIN sys.sql_logins AS SL
ON SP.principal_id = SL.principal_id
WHERE SP.type ='S'
AND SP.name NOT LIKE '##%##'
AND SP.name NOT LIKE 'NT AUTHORITY%'
AND SP.name NOT LIKE 'NT SERVICE%'
AND SP.name NOT IN ('rdsadmin','rdsbackup','rdsuser','rdsmirror','public')
```

Step 2 Execute the script in **Step 1**:

Figure 5-1 Obtaining the script



```
   CreateLogin
1  IF (SUSER_ID('sa') IS NULL) BEGIN CREATE LOGIN [sa] WITH PASSWORD = 0x010039BF2BFAD6A3DB4E2ABB941E8BD32E5189A4BE757...
2  IF (SUSER_ID('rdsuser2') IS NULL) BEGIN CREATE LOGIN [rdsuser2] WITH PASSWORD = 0x0100BE8BCBC25FC67008D4BE75AD660D1...
3  IF (SUSER_ID('csidbo') IS NULL) BEGIN CREATE LOGIN [csidbo] WITH PASSWORD = 0x0100A508789C15CE688664B162A5EDF4F4D2E...
4  IF (SUSER_ID('TestLogin7') IS NULL) BEGIN CREATE LOGIN [TestLogin7] WITH PASSWORD = 0x010073DA9A79E6677E8AF7077EF67...
5  IF (SUSER_ID('rdsuser3') IS NULL) BEGIN CREATE LOGIN [rdsuser3] WITH PASSWORD = 0x01009448FBDBCE8D5B5B2529384028CA0...
6  IF (SUSER_ID('Test2') IS NULL) BEGIN CREATE LOGIN [Test2] WITH PASSWORD = 0x0100130953CEEABC997D08B6BAF65F84BBCAA44...
7  IF (SUSER_ID('Test3') IS NULL) BEGIN CREATE LOGIN [Test3] WITH PASSWORD = 0x0100EE98873948E02595BDCD953B42663728187...
8  IF (SUSER_ID('Test4') IS NULL) BEGIN CREATE LOGIN [Test4] WITH PASSWORD = 0x01000EE91B9BF087741F16A44E70AA813D0BA8B...
9  IF (SUSER_ID('Test5') IS NULL) BEGIN CREATE LOGIN [Test5] WITH PASSWORD = 0x010056EBF845DF098D2DF9395AF7B7618A20735...
```

Step 3 Copy and execute the script obtain in **Step 2** on the destination instance. The created Login account is the same as the original one.

Step 4 Map the newly created Login account to the database user permissions that have been migrated to the RDS SQL Server DB instance to ensure permission consistency.

```
declare @DBName nvarchar(200)
declare @Login_name nvarchar(200)
declare @SQL nvarchar(MAX)
set @Login_name = 'TestLogin7' //Enter the login name one by one.
declare DBName_Cursor cursor for
select quotename(name)from sys.databases where  database_id > 4 and state = 0
and name not like '%$%'
and name <> 'rdsadmin'
open DBName_Cursor
fetch next from DBName_Cursor into @DBName
WHILE @@FETCH_STATUS= 0
begin
SET @SQL='    USE '+ (@DBName)+ '
if exists(select top 1 1 from sys.sysusers where name = '''+ @Login_name +''')
begin
ALTER USER '+@Login_name+' with login = '+@Login_name+';
end
'
```

```
print @SQL
EXEC (@SQL)
fetch next from DBName_Cursor into @DBName
end
close DBName_Cursor
deallocate DBName_Cursor
```

NOTE

After the preceding script is executed, you can view the Login account with the same name on the new instance, and the password and permission are the same as those on your local host.

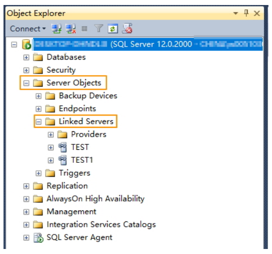
----End

Database Link

SQL Server allows you to create database links to interact with databases on external DB instances. Therefore you can query, synchronize, and compare databases of different types or on different DB instances. However, these links cannot be automatically synchronized to the DB instance on cloud so you need to synchronize them manually.

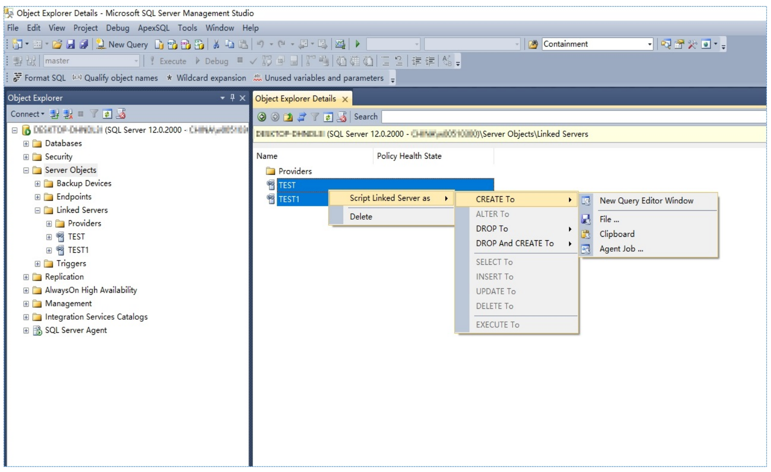
Step 1 Connect the local DB instance and cloud DB instance through Microsoft SQL Server Management Studio. Choose **Server Objects** > **Linked Servers** and locate the DBLink of the current DB instance.

Figure 5-2 Viewing database links



Step 2 Select the linked server and press **F7**. The **Object Explore** page is displayed. On this page, you can quickly create a script.

Figure 5-3 Creating the script



Step 3 In the displayed window, view all the scripts for creating DBLinks of the current DB instance. You only need to copy the scripts to the destination DB instance and change the password on @rmtpassword.

```
USE [master]
GO

/****** Object:  LinkedServer [DRS_TEST_REMOTE]    Script Date: 2019/5/25 17:51:50 ******/
EXEC master.dbo.sp_addlinkedserver @server = N'DRS_TEST_REMOTE', @srvproduct=N'',
@provider=N'SQLNCLI', @datasrc=N'DESKTOP-B18JH5T\SQLSERVER2016EE'
/* For security reasons the linked server remote logins password is changed with ######## */
EXEC master.dbo.sp_addlinkedsrvlogin
@rmtsrvname=N'DRS_TEST_REMOTE',@useself=N'False',@locallogin=NULL,@rmtuser=N'sa',@r
mtpassword='########'
GO
```

NOTE

The preceding script is an example. The created script may contain a large number of default system configuration items. You need to retain only the following two key scripts for each DBLink. In addition, you need to enter the account and password again.

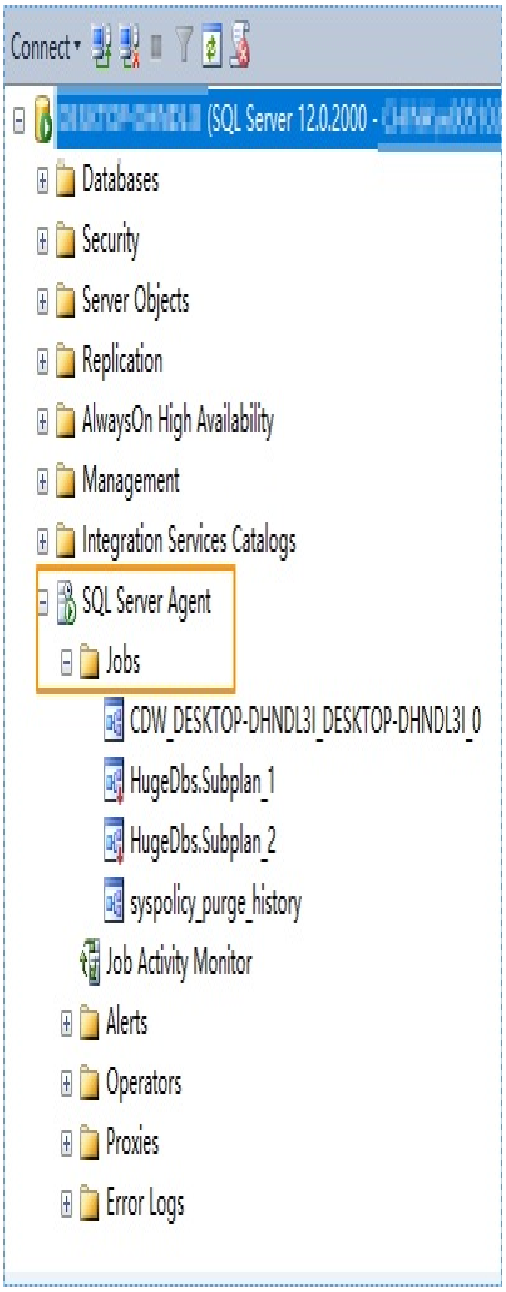
----End

Agent JOB

Agent Job is the agent service of Microsoft SQL Server. It helps you quickly create scheduled tasks on DB instances, perform routine O&M, and process data. You need to manually migrate local Job scripts.

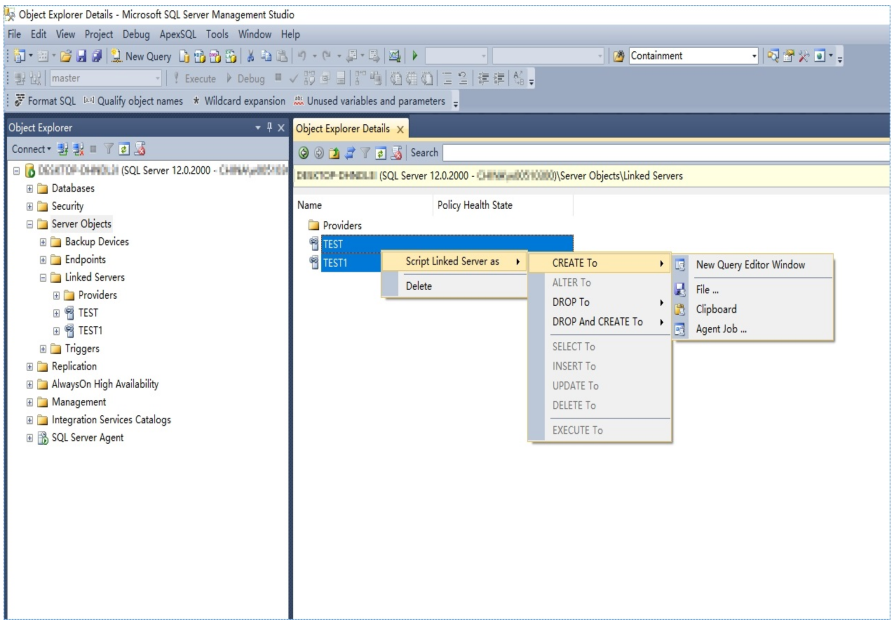
Step 1 Connect the local DB instance and cloud DB instance through Microsoft SQL Server Management Studio. Choose **SQL Server Agent** > **Jobs** and locate all the jobs of the current DB instance.

Figure 5-4 Viewing Jobs



Step 2 Select a job and press **F7**. All jobs are displayed on the **Object Explore** page. Select all jobs and create a script in the new window.

Figure 5-5 Creating a script



Step 3 Copy the T-SQL script in the new window to the new DB instance, and then modify the following key items to ensure that the creation is successful.

- Modify the owner account of each job.

  Example:

  @owner_login_name=N'rdsuser'

- Modify the DB instance name of each job.

  Example:

  @server=N' DB instance IP address'

  @server_name = N'DB instance IP address'

NOTE

The owner account of the new job is very important. On the RDS SQL Server DB instance, only the owner of the job can view the job of the DB instance. Therefore, it is recommended that all job owners use the same account to facilitate job management.

----End

Key Configuration

After the database is restored to the RDS SQL Server DB instance, some local important configuration items need to be synchronized to keep service running properly.

1. tempdb: The file configuration of the temporary database needs to be synchronized.

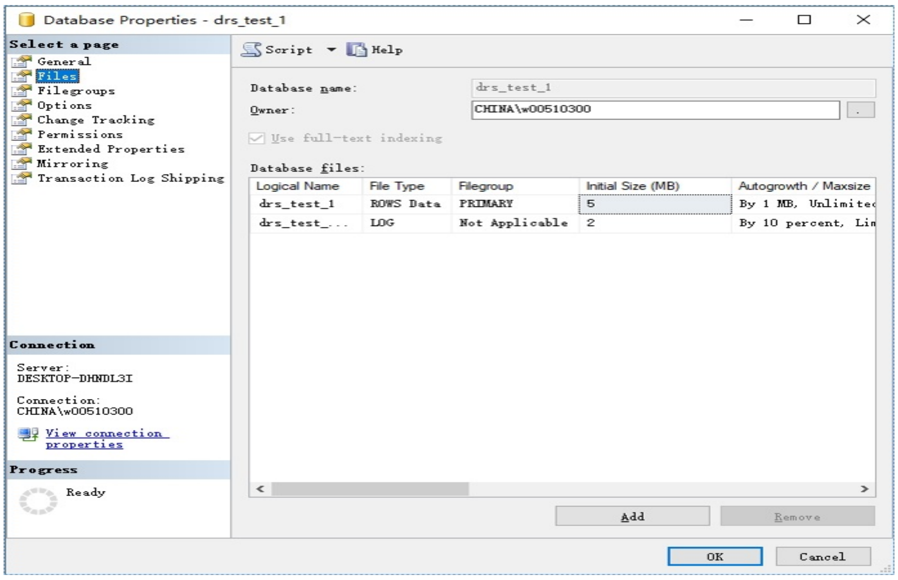
   It is recommended that you set 8 temporary files and ensure that the files are stored in **D:\RDSDBDATA\Temp\**.

   Run the following script on the destination database to add the temporary database file configuration:

```
USE [master]
GO
```

```
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdb1', FILENAME = N'D:
\RDSDBDATA\Temp\tempdb1.ndf' , SIZE = 65536KB , FILEGROWTH = 65536KB )
GO
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdb2', FILENAME = N'D:
\RDSDBDATA\Temp\tempdb2.ndf' , SIZE = 65536KB , FILEGROWTH = 65536KB )
GO
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdb3', FILENAME = N'D:
\RDSDBDATA\Temp\tempdb3.ndf' , SIZE = 65536KB , FILEGROWTH = 65536KB )
GO
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdb4', FILENAME = N'D:
\RDSDBDATA\Temp\tempdb4.ndf' , SIZE = 65536KB , FILEGROWTH = 65536KB )
GO
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdb5', FILENAME = N'D:
\RDSDBDATA\Temp\tempdb5.ndf' , SIZE = 65536KB , FILEGROWTH = 65536KB )
GO
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdb6', FILENAME = N'D:
\RDSDBDATA\Temp\tempdb6.ndf' , SIZE = 65536KB , FILEGROWTH = 65536KB )
GO
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdb7', FILENAME = N'D:
\RDSDBDATA\Temp\tempdb7.ndf' , SIZE = 65536KB , FILEGROWTH = 65536KB )
GO
```

Figure 5-6 Checking temporary files



Database name: drs_test_1
Owner: CHINA\w00510300
Use full-text indexing
Server: DESKTOP-DHNDL3I
Connection: CHINA\w00510300

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth / Maxsize
drs_test_1	ROWS Data	PRIMARY	5	By 1 MB, Unlimited
drs_test_...	LOG	Not Applicable	2	By 10 percent, Lin

2. Database isolation level: Check whether the database isolation level is enabled on the source DB instance and synchronize the isolation level to the RDS SQL Server DB instance. There are two snapshot isolation parameters:
   - Is Read Committed Snapshot On
   - Allow Snapshot Isolation

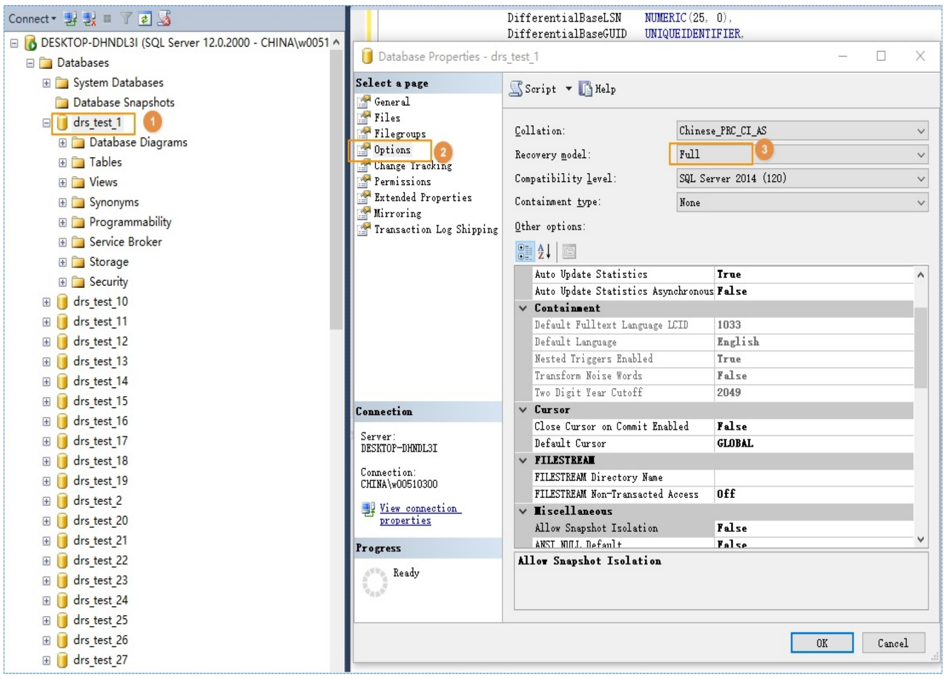
   If the database isolation level of the source DB instance is enabled, you can run the following script on the destination database to enable the database isolation level:

```
USE [DBName]
GO
ALTER DATABASE [DBName] SET READ_COMMITTED_SNAPSHOT ON WITH NO_WAIT
GO
ALTER DATABASE [DBName] SET ALLOW_SNAPSHOT_ISOLATION ON
GO
```

3. Max Degree of Parallelism: The maximum degree of parallelism is set to **0** by default on the RDS SQL Server instance. You can also set the value based on the local settings to avoid exceptions in different service scenarios.

   In **Object Explorer**, right-click a local server and select **Properties**. Click the **Advanced** node. In the **Max Degree of Parallelism** box, view the value of the local instance and change the **max degree of parallelism** value in the parameter group of the destination RDS SQL Server instance to the same.

Figure 5-7 Max Degree of Parallelism



```
DifferentialBaseLSN     NUMERIC(25, 0),
DifferentialBaseGUID    UNIQUEIDENTIFIER,
```

Server: DESKTOP-DHNDL3I

Connection: CHINA\w00510300

Collation: Chinese_PRC_CI_AS

Recovery model: Full

Compatibility level: SQL Server 2014 (120)

Containment type: None

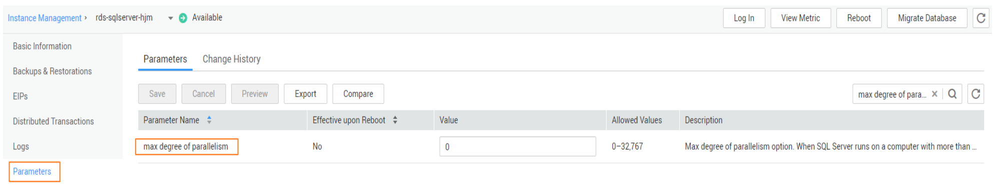
Other options:

Auto Update Statistics	True
Auto Update Statistics Asynchronous	False
Containment	
Default Fulltext Language LCID	1033
Default Language	English
Nested Triggers Enabled	True
Transform Noise Words	False
Two Digit Year Cutoff	2049
Cursor	
Close Cursor on Commit Enabled	False
Default Cursor	GLOBAL
FILESTREAM	
FILESTREAM Directory Name	
FILESTREAM Non-Transacted Access	Off
Miscellaneous	
Allow Snapshot Isolation	False
ANSI NULL Default	False

Allow Snapshot Isolation

   Log in to the RDS console. On the **Instance Management** page, click the target DB instance name. Choose **Parameters**, search for the **max degree of parallelism** parameter, and change its value.

Figure 5-8 max degree of parallelism

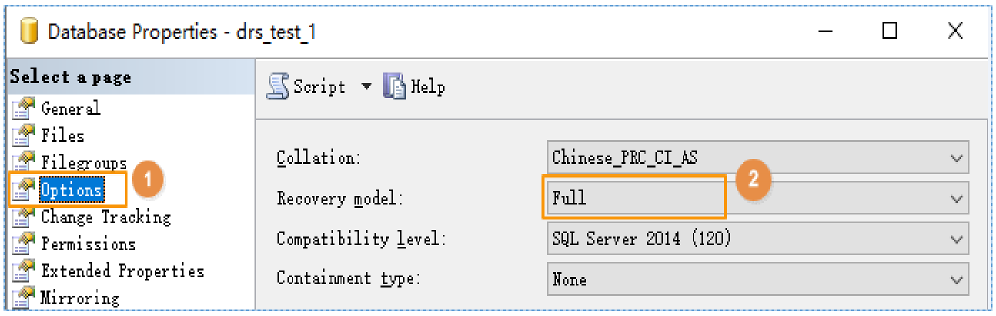


Parameter Name	Effective upon Reboot	Value	Allowed Values	Description
max degree of parallelism	No	0	0–32,767	Max degree of parallelism option. When SQL Server runs on a computer with more than ...

4. Check whether the database recovery model on the cloud is set to **Full**. If not, change the mode.

   Right-click the database and choose **Properties** from the shortcut menu. In the displayed page, select **Options**. Then, verify that **Recovery Model** is set to **Full**. Ensure that the database is highly available and the backup policy is executable.

Figure 5-9 Checking the database recovery model



Database Properties - drs_test_1

Select a page
- General
- Files
- Filegroups
- Options
- Change Tracking
- Permissions
- Extended Properties
- Mirroring

Collation:	Chinese_PRC_CI_AS
Recovery model:	Full
Compatibility level:	SQL Server 2014 (120)
Containment type:	None

6 Real-Time Synchronization

6.1 Can DRS Sync Tables of Different Schemas to the Same Schema?

DRS can directly synchronize tables of different schemas to those of the same schema if the tables do not conflict with each other.

6.2 Can Online DDL Tools Be Used for Real-time Synchronization?

Scenarios

DRS supports migration or synchronization tasks with MySQL serving as the source. In the incremental phase, third-party online DDL tools (such as PT-OSC and GH-OST) are used to execute DDL operations in the source database. An Online DDL tool creates a temporary table and uses the temporary table to perform DDL operations. In this case, there are the following scenarios for DRS migration or synchronization:

- For database- and instance-level migration or synchronization tasks, DRS automatically synchronizes DDL operations because the temporary table used by Online DDL is in the synchronization objects. No special processing is required.
- For table-level migration or synchronization tasks, if the temporary table used by third-party Online DDL has been added to the migration or synchronization objects when you create a DRS task, DRS will automatically synchronize DDL operations. No special processing is required.
- For table-level migration or synchronization tasks, if you select only the table data when creating a DRS task, DRS will not synchronize DDL operations because the temporary table used by Online DDL is not included in the selected objects. You can manually execute DDL operations in the destination database by referring to **Constraints** and **Procedure** to prevent DRS task failures caused by table structure inconsistency between the source and destination databases due to online DDL operations on the source database.

Constraints

- This solution is an alternative solution in scenarios where DRS database- and instance-level migration or synchronization cannot be used. You are advised to preferentially use the database- and instance-level migration or synchronization solution.
- The operation sequence of different DDL statements in the source and destination databases is different. Strictly follow the sequence in **Procedure** to prevent DRS task failures.
- The DDL statements executed in the source database and destination database must have the same semantics, including but not limited to the object name, column type, and length.

Procedure

Step 1 Check the DRS task status. Ensure that the task is in the **Incremental** state and the incremental latency is within 10 seconds.

Step 2 Confirm the DDL operations to be performed. Different operations are performed in different sequences in the source and destination databases.

- Adding columns: Perform the operation in the destination database and then in the source database.
- Deleting columns: Perform the operation in the source database and then in the destination database.
- Adding, modifying, or deleting default values of columns: These operations are irrelevant to the operation sequence.
- Changing column types: Perform the operation in the destination database and then in the source database.
- Changing character sets: Perform the operation in the destination database and then in the source database.
- Changing column names: Perform the operation in the source database, wait until the DRS task fails because the column is not found, and then perform the operation in the destination database to resume the DRS task.
- Adding partitions: Perform the operation in the destination database and then in the source database.
- Deleting partitions: Perform the operation in the source database and then in the destination database.
- Adding indexes: This operation is irrelevant to the operation sequence.
- Deleting indexes: This operation is irrelevant to the operation sequence.
- Adding constraints (such as primary keys, unique keys, and checks): Perform the operation in the source database and then in the destination database.
- Deleting constraints (such as primary keys, unique keys, and checks): Perform the operation in the destination database and then in the source database.
- Increasing field lengths: Perform the operation in the destination database and then in the source database.
- Reducing field lengths: Perform the operation in the source database and then in the destination database.

NOTE

If a DDL contains multiple operations, all operations except those irrelevant to the operation sequence (for example, changing the default value) must be performed in the required sequence. Otherwise, split it into multiple DDL operations. If you change default values when adding a column, perform the operation in the destination database and then in the source database.

Table 6-1 Summary

DDL Operation	Operation Sequence
Adding columns, changing column types, changing character sets, adding partitions, deleting constraints, and increasing field lengths	Perform the corresponding operations in the destination database and then in the source database.
Deleting columns, deleting partitions, adding constraints, and reducing field lengths	Perform the corresponding operations in the source database and then in the destination database.
Adding, modifying, and deleting default values of columns, adding indexes, and deleting indexes	These operations are irrelevant to the operation sequence.
Changing column names	Perform the operation in the source database, wait until the DRS task fails because the column is not found, and then perform the operation in the destination database to resume the DRS task.

Step 3 After the DDL operations are complete in **Step 2**, check whether the DRS task is normal.

----End

6.3 How Do I Delete XStream Outbound Created When XStream Is Enabled?

For Oracle to GaussDB synchronization, if you use XStream for log reading, you need to delete XStream Outbound created by the DRS task from the source database after the task is complete. This prevents the disk space from being used up because the Oracle database does not automatically delete archive logs that are not consumed by XStream Outbound.

Procedure

Step 1 Run the following SQL statements on the source database to check whether XStream Outbound created by the DRS task exists:

```
SELECT SERVER_NAME,
      CONNECT_USER,
      CAPTURE_USER,
      CAPTURE_NAME,
      SOURCE_DATABASE,
      QUEUE_OWNER,
      QUEUE_NAME
FROM DBA_XSTREAM_OUTBOUND;
```

The naming format of **SERVER_NAME** of XStream Outbound is *DRS_+Task ID in uppercase*. The hyphen in the uppercase task ID will be replaced by an underscore (_), for example, DRS_04981F62_84A4_4974_A5D5_772ED2DF63AC.

Step 2 If XStream Outbound exists, run the following SQL statement to delete it: In the following statement, *<xstream SERVER_NAME>* indicates the value of **SERVER_NAME** obtained in **Step 1**.

```
BEGIN DBMS_XSTREAM_ADM.DROP_OUTBOUND(server_name => '<xstream SERVER_NAME>');END;
```

----End

6.4 Why Do I Use the SCAN IP Address to Connect to an Oracle RAC Cluster?

If the source Oracle database is an RAC cluster, you are advised to use SCAN IP +SERVICE_NAMES to create a task because SCAN IP has stronger fault tolerance, better load balancing capability, and faster synchronization.

- If the SCAN IP address is used, ensure that the SCAN IP address can communicate with all virtual IP addresses of the source database. Otherwise, the connection test cannot be passed.
- If SCAN IP is not used, the virtual IP address of a node can be used. If other nodes are abnormal, the synchronization process is not affected.

For details about the SCAN IP address, see the **documents** on the Oracle official website.

6.5 How Do I Check Supplemental Logging of the Source Oracle Database?

In physical standby mode, the Oracle database directly replicates logs from the primary database and does not generate any logs. If the source is an Oracle database, you need to check whether supplemental logging on the primary database meets the requirements to ensure that the task can run properly. The following lists the check and setting methods:

Table level: This setting applies to a specified table.

Database level: This setting applies to the database level.

PK/UI: In addition to the changed columns,the values of the primary key and unique key of each row are recorded.

ALL: Each row of the log records the values of all columns in that row.

NOTE

DRS incremental synchronization requirements can be met if any of the following checks are passed.

Table-level PK/UI Supplemental Logging Check (Minimum Requirement)

Check whether supplemental logging of the table-level objects to be synchronized meets the requirements.

Step 1 Run the following SQL statement in the source database:

```
select * from ALL_LOG_GROUPS where (LOG_GROUP_TYPE='UNIQUE KEY LOGGING' or
LOG_GROUP_TYPE='PRIMARY KEY LOGGING') and OWNER='Schema name in uppercase' and
TABLE_NAME='Table name in uppercase';
```

If the table name corresponds to the records whose **LOG_GROUP_TYPE** is **UNIQUE KEY LOGGING** and **PRIMARY KEY LOGGING** in the query result, the DRS incremental synchronization requirements are met.

Step 2 If the requirements are not met, run the following SQL statement to enable table-level PK/UI logging:

```
alter database add supplemental log data;
alter table Schema_ name.Table_name add supplemental log data(primary key,unique) columns;
```

----End

All Table-Level Supplemental Log Check

Check whether supplemental logging of the table-level objects to be synchronized meets the requirements.

Step 1 Run the following SQL statement in the source database:

```
select * from ALL_LOG_GROUPS where LOG_GROUP_TYPE='ALL COLUMN LOGGING' and
OWNER='Schema_name in uppercase' and TABLE_NAME='Table name in uppercase';
```

If the table name is recorded in the query result, the DRS incremental synchronization requirements can be met.

Step 2 If the requirements are not met, run the following SQL statement to enable all column supplemental logging at the table level:

```
alter database add supplemental log data;
alter table Schema_name.Table_name add supplemental log data(all) columns;
```

----End

Database-level Supplemental Log Check

For the database-level objects to be synchronized, check whether supplemental logging meets the requirements.

Step 1 Run the following SQL statement in the source database:

```
select SUPPLEMENTAL_LOG_DATA_MIN MIN, SUPPLEMENTAL_LOG_DATA_PK PK,
SUPPLEMENTAL_LOG_DATA_UI UI, SUPPLEMENTAL_LOG_DATA_ALL ALL_LOG from v$database;
```

Step 2 Either of the following requirements must be met:

- If both **PK** and **UI** are set to **YES**, DRS incremental synchronization requirements can be met.

If the requirements are not met, run the following SQL statement to enable database-level PK/UI supplemental logging:

```
alter database add supplemental log data(primary key, unique) columns;
```

- If **ALL_LOG** is set to **YES**, DRS incremental synchronization requirements can be met.

  If the requirements are not met, run the following SQL statement to enable all column supplemental logging at the database level:

  ```
  alter database add supplemental log data(all) columns;
  ```

----End

6.6 Which Specifications Should I Select for My DRS Task to Synchronize Data to GaussDB(DWS)?

DRS provides you four types of specifications when you are configuring a synchronization task. To achieve optimal performance, select the specifications while considering your data model, source database workload, destination database workload, bandwidth latency, and DRS maximum transferring speed.

Table 6-2 lists the maximum transferring speeds defined in the DRS specifications.

Table 6-2 Performance upper limit

Specifications	Reference Values of Maximum Performance (Rows/Second)
Micro	300
Small	3,000
Medium	7,500
Large	> 7,500

For details about GaussDB(DWS) synchronization specifications, see **GaussDB(DWS) User Guide**.

6.7 Suggestions on Synchronizing Data to GaussDB(DWS)

DDL Support in Incremental Synchronization

MySQL and Oracle use different syntax from GaussDB(DWS). During a real-time synchronization from MySQL or Oracle to GaussDB(DWS), incremental MySQL or Oracle DDL statements may fail to be synchronized. **Table 6-3** lists DDL statement conversion rules. Before performing a synchronization task, contact GaussDB(DWS) technical support to evaluate DRS support for the DDL statements in your source database.

Table 6-3 DDL statement conversion rules

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
Table and column names contain backquotes.	Replace backquotes with double quotation marks.	`create table `t1`( `c1` int, `c2` varchar(10) );`	`CREATE TABLE "public"."t1"( "c1" INTEGER, "c2" VARCHAR(40)) WITH ( ORIENTATION = ROW, COMPRESSION = NO ) NOCOMPRESS DISTRIBUTE BY HASH ("c1");`	N/A
Key field name	Create indexes.	`Create Table mall_order_dc (id bigint NOT NULL AUTO_INCREMENT,order_id varchar(50) NOT NULL,key order_id(id));`	`CREATE TABLE "public"."mall_order_dc"( "id" BIGSERIAL NOT NULL, "order_id" VARCHAR(200) NOT NULL) WITH ( ORIENTATION = ROW, COMPRESSION = NO ) NOCOMPRESS DISTRIBUTE BY HASH ("id"); CREATE INDEX "order_id" ON "public"."mall_order_dc" USING BTREE ("id");`	N/A
Custom field names are the same as the names of hidden fields.	Add the suffix _new.	`Create Table mall_order_dc ( xc_node_id bigint NOT NULL AUTO_INCREMENT, tableoid varchar(50) NOT NULL, cmax int, xmax int, cmin char, xmin varchar(10), ctid smallint, tid time, tidd int, ctidd int );`	`CREATE TABLE "public"."mall_order_dc"("xc_node_id_new" BIGSERIAL NOT NULL, "tableoid_new" VARCHAR(200) NOT NULL, "cmax_new" INTEGER, "xmax_new" INTEGER, "cmin_new" CHAR(4), "xmin_new" VARCHAR(40), "ctid_new" SMALLINT, "tid_new" TIME WITHOUT TIME ZONE, "tidd" INTEGER, "ctidd" INTEGER) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("xc_node_id_new");`	GaussDB(DWS) has the following system fields: xc_node_id, tableoid, cmax, xmax, cmin, xmin, ctid, and tid. They will be suffixed with _new.

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
GaussDB(DWS) keywords are used as field names.	Add double quotation marks to the field names.	N/A	N/A	If any table or field using keywords like **desc**, **checksum**, **operator**, and **size** as its name, the table and field names will be enclosed in double quotation marks.
GaussDB(DWS) keywords are used as table names.	Add double quotation marks to the table names.	N/A	N/A	If any table or field using keywords like **user** as its name, the table and field names will be enclosed in double quotation marks.

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks

RENAME TABLE	N/A	RENAME TABLE department TO newdept;
		RENAME TABLE employee TO pepole;
		RENAME TABLE newdept TO newdept_02,pepole to pepole_02;
			ALTER TABLE "public"."department" RENAME TO "newdept";
			ALTER TABLE "public"."employee" RENAME TO "pepole";
			ALTER TABLE "public"."newdept" RENAME TO "newdept_02";
			ALTER TABLE "public"."pepole" RENAME TO "pepole_02";
				A parallel rename statement is converted into multiple rename statements.

SET DEFAULT	N/A	ALTER TABLE runoob_alter_test ALTER dataType2 SET DEFAULT 1;
		ALTER TABLE runoob_alter_test ALTER COLUMN dataType2 SET DEFAULT 3;
		
		ALTER TABLE runoob_alter_test ALTER dataType2 SET DEFAULT '1';
		ALTER TABLE runoob_alter_test ALTER COLUMN dataType2 SET DEFAULT '3';
			ALTER TABLE "public"."runoob_alter_test" ALTER COLUMN "datatype2" SET DEFAULT '1';
			ALTER TABLE "public"."runoob_alter_test" ALTER COLUMN "datatype2" SET DEFAULT '3';
			
			ALTER TABLE "public"."runoob_alter_test" ALTER COLUMN "datatype2" SET DEFAULT '1';
			ALTER TABLE "public"."runoob_alter_test" ALTER COLUMN "datatype2" SET DEFAULT '3';
				N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks

CHANGE	N/A	ALTER TABLE runoob_alter_test CHANGE dataType1 dataType1New VARCHAR(50);
		ALTER TABLE runoob_alter_test CHANGE dataType2 dataType2New VARCHAR(50) NOT NULL;
		ALTER TABLE runoob_alter_test CHANGE dataType3 dataType3New VARCHAR(100) FIRST;
		ALTER TABLE runoob_alter_test CHANGE dataType4 dataType4New VARCHAR(50) AFTER dataType1;
			ALTER TABLE "public"."runoob_alter_test" CHANGE COLUMN "datatype1" "datatype1new" VARCHAR(200);
			ALTER TABLE "public"."runoob_alter_test" CHANGE COLUMN "datatype2" "datatype2new" VARCHAR(200) NOT NULL;
			ALTER TABLE "public"."runoob_alter_test" CHANGE COLUMN "datatype3" "datatype3new" VARCHAR(400);
			ALTER TABLE "public"."runoob_alter_test" CHANGE COLUMN "datatype4" "datatype4new" VARCHAR(200);
				N/A

MODIFY	N/A	ALTER TABLE runoob_alter_test modify datatype1 char default 1;
		ALTER TABLE runoob_alter_test modify datatype1 char default '1';
		ALTER TABLE runoob_alter_test modify datatype1 char default "1";
			ALTER TABLE "public"."runoob_alter_test" MODIFY "datatype1" CHAR(4) DEFAULT '1';
			ALTER TABLE "public"."runoob_alter_test" MODIFY "datatype1" CHAR(4) DEFAULT '1';
			ALTER TABLE "public"."runoob_alter_test" MODIFY "datatype1" CHAR(4) DEFAULT '1';
				N/A

ADD PRIMARY KEY	N/A	ALTER TABLE runoob_alter_test ADD PRIMARY KEY (dataType1);	ALTER TABLE "public"."runoob_alter_test" ADD PRIMARY KEY("datatype1");	N/A
DROP PRIMARY KEY	N/A	ALTER TABLE runoob_alter_test DROP PRIMARY KEY;	ALTER TABLE "public"."runoob_alter_test" DROP CONSTRAINT IF EXISTS runoob_alter_test_pkey;	N/A
ADD COLUMN	N/A	ALTER TABLE runoob_alter_test ADD dataType1_1 INT NOT NULL AFTER dataType1;	ALTER TABLE "public"."runoob_alter_test" ADD COLUMN "datatype1_1" INTEGER NOT NULL;	N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
ADD COLUMN	N/A	`ALTER TABLE runoob_alter_test ADD dataType1_1 INT NOT NULL AFTER dataType1;`	`ALTER TABLE "public"."runoob_alter_test" ADD COLUMN "datatype1_1" INTEGER NOT NULL;`	N/A
ADD INDEX	N/A	`ALTER TABLE runoob_tbl ADD INDEX idex_runoob_id(runoob_id) USING BTREE;`	`CREATE INDEX "idex_runoob_id" ON "public"."runoob_tbl" ("runoob_id");`	N/A
ADD UNIQUE INDEX	Generate a common index using the unique key.	`ALTER TABLE runoob_tbl ADD UNIQUE KEY IDEX_runoob_id USING BTREE (runoob_id);`	`CREATE INDEX "idex_runoob_id" ON "public"."runoob_tbl" ("runoob_id");`	N/A
DROP INDEX	N/A	`ALTER TABLE runoob_tbl DROP KEY IDEX_runoob_id;`	`DROP INDEX "public"."idex_runoob_id" RESTRICT;`	N/A
ALGORITHM	Delete	`ALTER TABLE runoob_alter_test ALGORITHM=DEFAULT;` `ALTER TABLE runoob_alter_test ALGORITHM=INPLACE;` `ALTER TABLE runoob_alter_test ALGORITHM=COPY;`	N/A	N/A
DEFAULT CHARACTER SET	Delete	`ALTER TABLE runoob_alter_test CHARACTER SET=utf8;` `ALTER TABLE runoob_alter_test DEFAULT CHARACTER SET=utf8;`	N/A	N/A
COLLATE	Delete	N/A	N/A	**COLLATE** specifies a default database sorting rule.

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
DELAY_KEY_WRITE	Delete	`CREATE TABLE `public`.`runoob_tbl_test`( `runoob_id` VARCHAR(30), `runoob_title` VARCHAR(100) NOT NULL, `runoob_author` VARCHAR(40) NOT NULL, `submission_date` VARCHAR(30) ) ENGINE=MyISAM, DELAY_KEY_WRITE=0; ALTER TABLE `public`.`runoob_tbl_test6` DELAY_KEY_WRITE=1;`	`CREATE TABLE "PUBLIC"."RUNOOB_TBL_TEST"( "RUNOOB_ID" VARCHAR(30), "RUNOOB_TITLE" VARCHAR(100) NOT NULL, "RUNOOB_AUTHOR" VARCHAR(40) NOT NULL, "SUBMISSION_DATE" VARCHAR(30) ) WITH ( ORIENTATION = ROW, COMPRESSION = NO ) NOCOMPRESS DISTRIBUTE BY HASH ("RUNOOB_ID");`	**DELAY_KEY_WRITE** specifies how to use delayed key writes and is valid only for MyISAM tables. If this field is enabled for a table, the key buffer is not flushed for the table on every index update, but only when the table is closed.

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks

DIRECTORY	Delete	CREATE TABLE \`public\`.\`runoob_tbl_test1\` (\`dataType1\` int NOT NULL AUTO_INCREMENT, \`dataType2\` DOUBLE(20,8), PRIMARY KEY(\`dataType1\`)) ENGINE=MYISAM DATA DIRECTORY = 'D:\\\\input' INDEX DIRECTORY= 'D:\\\\input';
		CREATE TABLE \`public\`.\`runoob_tbl_test2\` (\`dataType1\` int NOT NULL AUTO_INCREMENT, \`dataType2\` DOUBLE(20,8), PRIMARY KEY(\`dataType1\`)) ENGINE=INNODB DATA DIRECTORY = 'D:\\\\input'
			CREATE TABLE "public"."runoob_tbl_test1"("datatype1" SERIAL NOT NULL, "datatype2" FLOAT(20), PRIMARY KEY ("datatype1")) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("datatype1");
			CREATE TABLE "public"."runoob_tbl_test2"("datatype1" SERIAL NOT NULL, "datatype2" FLOAT(20), PRIMARY KEY ("datatype1")) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("datatype1");
				N/A

INSERT_METHOD	Delete	CREATE TABLE \`public\`.\`runoob_alter_test\`(\`dataType1\` int NOT NULL AUTO_INCREMENT, \`dataType2\` DOUBLE(20,8), \`dataType3\` TEXT NOT NULL, PRIMARY KEY(\`dataType1\`)) INSERT_METHOD=LAST;
		ALTER TABLE runoob_alter_test INSERT_METHOD NO;
		ALTER TABLE runoob_alter_test INSERT_METHOD=NO;
		ALTER TABLE runoob_alter_test INSERT_METHOD FIRST;
		ALTER TABLE runoob_alter_test INSERT_METHOD=FIRST;
		ALTER TABLE runoob_alter_test INSERT_METHOD LAST;
		ALTER TABLE runoob_alter_test INSERT_METHOD=LAST;
			CREATE TABLE "public"."runoob_alter_test"("datatype1" SERIAL NOT NULL, "datatype2" FLOAT(20), "datatype3" TEXT NOT NULL, PRIMARY KEY ("datatype1")) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("datatype1");	N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
KEY_BLOCK_SIZE	Delete	CREATE TABLE \`public\`.\`runoob_tbl_test\`(\`runoob_id\` VARCHAR(30), \`runoob_title\` VARCHAR(100) NOT NULL, \`runoob_author\` VARCHAR(40) NOT NULL, \`submission_date\` VARCHAR(30)) ENGINE=MyISAM KEY_BLOCK_SIZE=8; ALTER TABLE runoob_tbl_test ENGINE=InnoDB; ALTER TABLE runoob_tbl_test KEY_BLOCK_SIZE=0;	CREATE TABLE "public"."runoob_tbl_test"("runoob_id" VARCHAR(30), "runoob_title" VARCHAR(100) NOT NULL, "runoob_author" VARCHAR(40) NOT NULL, "submission_date" VARCHAR(30)) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("runoob_id");	N/A
MAX_ROWS	Delete	CREATE TABLE \`public\`.\`runoob_alter_test\`(\`dataType1\` int NOT NULL AUTO_INCREMENT, \`dataType2\` DOUBLE(20,8), \`dataType3\` TEXT NOT NULL, PRIMARY KEY(\`dataType1\`)); ALTER TABLE runoob_alter_test MAX_ROWS 100000; ALTER TABLE runoob_alter_test MAX_ROWS=100000;	CREATE TABLE "public"."runoob_alter_test"("datatype1" SERIAL NOT NULL, "datatype2" FLOAT(20), "datatype3" TEXT NOT NULL, PRIMARY KEY ("datatype1")) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("datatype1");	N/A
MIN_ROWS	Delete	CREATE TABLE \`public\`.\`runoob_alter_test\`(\`dataType1\` int NOT NULL AUTO_INCREMENT, \`dataType2\` DOUBLE(20,8), \`dataType3\` TEXT NOT NULL, PRIMARY KEY(\`dataType1\`)); ALTER TABLE runoob_alter_test MIN_ROWS 10000; ALTER TABLE runoob_alter_test MIN_ROWS=10000;	CREATE TABLE "public"."runoob_alter_test"("datatype1" SERIAL NOT NULL, "datatype2" FLOAT(20), "datatype3" TEXT NOT NULL, PRIMARY KEY ("datatype1"))WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("datatype1");	N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks

PACK_KEYS	Delete	CREATE TABLE \`public\`.\`runoob_alter_test\`(
		\`dataType1\` int NOT NULL AUTO_INCREMENT,
		\`dataType2\` DOUBLE(20,8),
		\`dataType3\` TEXT NOT NULL,
		PRIMARY KEY(\`dataType1\`)
		) ENGINE=MyISAM PACK_KEYS=1;
		##A
		ALTER TABLE runoob_alter_test PACK_KEYS 0;
		ALTER TABLE runoob_alter_test PACK_KEYS=0;
		##B
		ALTER TABLE runoob_alter_test PACK_KEYS 1;
		ALTER TABLE runoob_alter_test PACK_KEYS=1;
		##C
		ALTER TABLE runoob_alter_test PACK_KEYS DEFAULT;
		ALTER TABLE runoob_alter_test PACK_KEYS=DEFAULT;
			CREATE TABLE "public"."runoob_alter_test"(
			"datatype1" SERIAL NOT NULL,
			"datatype2" FLOAT(10),
			"datatype3" FLOAT(20),
			"datatype4" TEXT NOT NULL,
			PRIMARY KEY ("datatype1")
			)WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("datatype1");
			--A
			--B
			--C
				N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
PASSWORD	Delete	CREATE TABLE \`public\`.\`runoob_alter_test\`(\`dataType1\` int NOT NULL AUTO_INCREMENT, \`dataType2\` DOUBLE(20,8), \`dataType3\` TEXT NOT NULL, PRIMARY KEY(\`dataType1\`)); ALTER TABLE runoob_alter_test PASSWORD 'HELLO';	CREATE TABLE "public"."runoob_alter_test" ("datatype1" SERIAL NOT NULL, "datatype2" FLOAT(20), "datatype3" TEXT NOT NULL, PRIMARY KEY ("datatype1")) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("datatype1");	N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks

STATS_AUTO_RECALC	Delete	CREATE TABLE \`public\`.\`runoob_alter_test\`(
		\`runoob_id\` VARCHAR(30),
		\`runoob_title\` VARCHAR(100) NOT NULL,
		\`runoob_author\` VARCHAR(40) NOT NULL,
		\`submission_date\` VARCHAR(30)
		) ENGINE=InnoDB, STATS_AUTO_RECALC=DEFAULT;
		## A.
		ALTER TABLE runoob_alter_test STATS_AUTO_RECALC DEFAULT;
		ALTER TABLE runoob_alter_test STATS_AUTO_RECALC=DEFAULT;
		## B.
		ALTER TABLE runoob_alter_test STATS_AUTO_RECALC 0;
		ALTER TABLE runoob_alter_test STATS_AUTO_RECALC=0;
		## C.
		ALTER TABLE runoob_alter_test STATS_AUTO_RECALC 1;
		ALTER TABLE runoob_alter_test STATS_AUTO_RECALC=1;
			CREATE TABLE "public"."runoob_alter_test"
			(
			"runoob_id" VARCHAR(30),
			"runoob_title" VARCHAR(100) NOT NULL,
			"runoob_author" VARCHAR(40) NOT NULL,
			"submission_date" VARCHAR(30)
			)
			WITH (ORIENTATION = ROW, COMPRESSION = NO)
			NOCOMPRESS
			DISTRIBUTE BY HASH ("runoob_id");
			-- A.
			-- B.
			-- C
				N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
STATS_PERSISTENT	Delete	CREATE TABLE \`public\`.\`runoob_alter_test\`(\`dataType1\` int NOT NULL AUTO_INCREMENT, \`dataType2\` DOUBLE(20,8), \`dataType3\` TEXT NOT NULL, PRIMARY KEY(\`dataType1\`)) ENGINE=InnoDB, STATS_PERSISTENT=0; ## A. ALTER TABLE runoob_alter_test STATS_PERSISTENT DEFAULT; ALTER TABLE runoob_alter_test STATS_PERSISTENT=DEFAULT; ## B. ALTER TABLE runoob_alter_test STATS_PERSISTENT 0; ALTER TABLE runoob_alter_test STATS_PERSISTENT=0; ## C. ALTER TABLE runoob_alter_test STATS_PERSISTENT 1; ALTER TABLE runoob_alter_test STATS_PERSISTENT=1	CREATE TABLE "public"."runoob_alter_test" ("datatype1" SERIAL NOT NULL, "datatype2" FLOAT(20), "datatype3" TEXT NOT NULL, PRIMARY KEY ("datatype1")) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("datatype1"); -- A. -- B. -- C.	N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks

STATS_SAMPLE_PAGES	Delete	CREATE TABLE \`public\`.\`runoob_alter_test\`(
		\`dataType1\` int NOT NULL AUTO_INCREMENT,
		\`dataType2\` DOUBLE(20,8),
		\`dataType3\` TEXT NOT NULL,
		PRIMARY KEY(\`dataType1\`)
		) ENGINE=InnoDB,STATS_SAMPLE_PAGES=25;
		ALTER TABLE runoob_alter_test STATS_SAMPLE_PAGES 100;
		ALTER TABLE runoob_alter_test STATS_SAMPLE_PAGES=100;
			CREATE TABLE "public"."runoob_alter_test"
			(
			"datatype1" SERIAL NOT NULL,
			"datatype2" FLOAT(20),
			"datatype3" TEXT NOT NULL,
			PRIMARY KEY ("datatype1")
			)
			WITH (ORIENTATION = ROW, COMPRESSION = NO)
			NOCOMPRESS
			DISTRIBUTE BY HASH ("datatype1");
				N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks

The option **ENGINE** is specified when you create a table.	Delete the option **ENGINE**.	CREATE TABLE \`public\`.\`runoob_alter_test\`(
		\`dataType1\` int NOT NULL,
		\`dataType2\` DOUBLE(20,8),
		PRIMARY KEY(\`dataType1\`)
		)ENGINE=MYISAM;
		## A.
		ALTER TABLE runoob_alter_test ENGINE INNODB;
		ALTER TABLE runoob_alter_test ENGINE=INNODB;
		## B.
		ALTER TABLE runoob_alter_test ENGINE MYISAM;
		ALTER TABLE runoob_alter_test ENGINE=MYISAM;
		## C.
		ALTER TABLE runoob_alter_test ENGINE MEMORY;
		ALTER TABLE runoob_alter_test ENGINE=MEMORY
			CREATE TABLE "public"."runoob_alter_test"
			(
			"datatype1" INTEGER NOT NULL,
			"datatype2" FLOAT(20),
			PRIMARY KEY ("datatype1")
			)
			WITH (ORIENTATION = ROW, COMPRESSION = NO)
			NOCOMPRESS
			DISTRIBUTE BY HASH ("datatype1");
			-- A.
			-- B.
			-- C.
				ENGINE specifies the storage engine for a MySQL table. During synchronization, **ENGINE** is deleted if it is set to **ARCHIVE**, **BLACKHOLE**, **CSV**, **FEDERATED**, **INNODB**, **MYISAM**, **MEMORY**, **MRG_MYISAM**, **NDB**, **NDBCLUSTER**, or **PERFOMANCE_SCHEMA**.

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks

CHECKSUM	Delete	CREATE TABLE \`public\`.\`runoob_alter_test\`(
		\`dataType1\` int NOT NULL AUTO_INCREMENT,
		\`dataType2\` FLOAT(10,2),
		\`dataType3\` DOUBLE(20,8),
		PRIMARY KEY(\`dataType1\`)
		) CHECKSUM=1;
		ALTER TABLE runoob_alter_test CHECKSUM 0;
		ALTER TABLE runoob_alter_test CHECKSUM=0;
		ALTER TABLE runoob_alter_test CHECKSUM 1;
		ALTER TABLE runoob_alter_test CHECKSUM=1
			CREATE TABLE "public"."runoob_alter_test"
			(
			"datatype1" SERIAL NOT NULL,
			"datatype2" REAL,
			"datatype3" DOUBLE PRECISION,
			PRIMARY KEY ("datatype1")
			)
			WITH (ORIENTATION = ROW, COMPRESSION = NO)
			NOCOMPRESS
			DISTRIBUTE BY HASH ("datatype1");
				N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
ON UPDATE CURRENT_TIMESTAMP	Delete	drop table if exists unsupport_parse_test; create table \`unsupport_parse_test\` (\`username\` int, \`update\` timestamp not null default current_timestamp on update current_timestamp);	DROP TABLE IF EXISTS "public"."unsupport_parse_test"; CREATE TABLE "public"."unsupport_parse_test" ("username" INTEGER, "update" TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT CURRENT_TIMESTAMP) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("username");	N/A
SET FOREIGN_KEY_CHECKS=0 or 1	Delete	N/A	N/A	N/A
PRIMARY KEY (\`id\`) USING BTREE	Delete	Create Table mall_order_dc (id int, name varchar(10), primary key(\`id\`) using btree);	CREATE TABLE "public"."mall_order_dc" ("id" INTEGER, "name" VARCHAR(40), PRIMARY KEY ("id")) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("id");	The default value is **btree**. This keyword is not required.

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
Row-column storage conversion	Create tables using row-store by default.	N/A	N/A	N/A
create table if not exists	The keyword **if not exists** is supported.	N/A	N/A	N/A
charset	Delete this keyword during synchronization.	N/A	N/A	**CHARSET** specifies the default character set for a table.

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
Partition table	Range partitioning is supported.	drop table if exists \`runoob_tbl_part_test\`; CREATE TABLE IF NOT EXISTS \`runoob_tbl_part_test\`(\`runoob_id\` INT NOT NULL, \`runoob_title\` VARCHAR(100) NOT NULL, \`runoob_author\` VARCHAR(40) NOT NULL, \`submission_date\` INT)ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY RANGE COLUMNS(runoob_id, submission_date)(PARTITION p0 VALUES LESS THAN(123, MAXVALUE), PARTITION p1 VALUES LESS THAN(200, MAXVALUE), PARTITION p2 VALUES LESS THAN(300, MAXVALUE), PARTITION p3 VALUES LESS THAN(400, MAXVALUE), PARTITION p4 VALUES LESS THAN(500, MAXVALUE), PARTITION p5 VALUES LESS THAN(MAXVALUE, MAXVALUE));	CREATE TABLE IF NOT EXISTS "public"."runoob_tbl_part_test" ("runoob_id" INTEGER NOT NULL, "runoob_title" VARCHAR(400) NOT NULL, "runoob_author" VARCHAR(160) NOT NULL, "submission_date" INTEGER) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("runoob_id") PARTITION BY RANGE ("runoob_id", "submission_date") (PARTITION p0 VALUES LESS THAN (123, MAXVALUE), PARTITION p1 VALUES LESS THAN (200, MAXVALUE), PARTITION p2 VALUES LESS THAN (300, MAXVALUE), PARTITION p3 VALUES LESS THAN (400, MAXVALUE), PARTITION p4 VALUES LESS THAN (500, MAXVALUE),	N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
			PARTITION p5 VALUES LESS THAN (MAXVALUE, MAXVALUE));	

Character encoding specified by a field	Delete	CREATE TABLE \`FCT_TRADE_XXXX_RT\` (
		\`KID\` INT(11) NOT NULL AUTO_INCREMENT,
		\`YM\` VARCHAR(7) CHARSET UTF8MB4 COLLATE UTF8MB4_BIN NOT NULL COMMENT 'YYYY-MM, for example, 2019-04'
		);
			CREATE TABLE "public"."fct_trade_xxxx_rt"
			(
			"kid" SERIAL NOT NULL,
			"ym" VARCHAR(28) NOT NULL COMMENT 'YYYY-MM, for example, 2019-04'
			)
			WITH (ORIENTATION = ROW, COMPRESSION = NO)
			NOCOMPRESS
			DISTRIBUTE BY HASH ("kid");
				N/A

Integer with data width	Delete the data width.	CREATE TABLE \`FCT_TRADE_XXXX_RT\` (\`KID\` INT(11) NOT NULL AUTO_INCREMENT, \`YM\` SMALLINT(15), \`c3\` BIGINT(50));	CREATE TABLE "public"."fct_trade_xxxx_rt" ("kid" SERIAL NOT NULL, "ym" SMALLINT, "c3" BIGINT) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("kid");	The number types **int**, **smallint**, and **bigint** do not support data width and will be deleted.

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
unsigned zerofill	Delete	CREATE TABLE IF NOT EXISTS \`runoob_dataType_test\`(\`dataType_1\` DEC, \`dataType_2\` DEC(10), \`dataType_3\` DEC(10, 2) UNSIGNED ZEROFILL, \`dataType_4\` DEC(10, 2) ZEROFILL)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_general_ci;	CREATE TABLE IF NOT EXISTS "public"."runoob_datatype_test" ("datatype_1" DECIMAL, "datatype_2" DECIMAL(10), "datatype_3" DECIMAL(10,2), "datatype_4" DECIMAL(10,2)) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("datatype_1");	N/A
auto_increment	This attribute is not supported. Convert this attribute to **SERIAL** and delete it during synchronization.	CREATE TABLE \`public\`.\`job_instance\` (\`job_sche_id\` int(11) NOT NULL AUTO_INCREMENT, \`task_name\` varchar(100) NOT NULL DEFAULT '', PRIMARY KEY (\`job_sche_id\`)) ENGINE=InnoDB AUTO_INCREMENT=219 DEFAULT CHARSET=utf8	CREATE TABLE "public"."job_instance" ("job_sche_id" SERIAL NOT NULL, "task_name" VARCHAR(100) NOT NULL DEFAULT '', PRIMARY KEY ("job_sche_id")) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("job_sche_id");	N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
AVG_ROW_LENGTH	This attribute is not supported. Convert this attribute to **SERIAL** and delete it during synchronization.	CREATE TABLE \`public\`.\`runoob_tbl_test\`(\`runoob_id\` VARCHAR(30), \`runoob_title\` VARCHAR(100) NOT NULL, \`runoob_author\` VARCHAR(40) NOT NULL, \`submission_date\` VARCHAR(30))AVG_ROW_LENGTH=10000;	CREATE TABLE "public"."runoob_tbl_test" ("runoob_id" VARCHAR(30), "runoob_title" VARCHAR(100) NOT NULL, "runoob_author" VARCHAR(40) NOT NULL, "submission_date" VARCHAR(30)) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("runoob_id")	N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks

UNION	Convert this function to the CREATE VIEW statement in GaussDB during synchronization.	CREATE TABLE t1 (
		a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
		message CHAR(20)
		) ENGINE=MyISAM;
		CREATE TABLE t2 (
		a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
		message CHAR(20)
		) ENGINE=MyISAM;
		CREATE TABLE total (
		a INT NOT NULL AUTO_INCREMENT,
		message CHAR(20))
		ENGINE=MERGE UNION=(t1,t2) INSERT_METHOD=LAST;
			CREATE TABLE t1 (
			a SERIAL NOT NULL PRIMARY KEY,
			message CHAR(20)
			)
			WITH (ORIENTATION = ROW, COMPRESSION = NO)
			NOCOMPRESS
			DISTRIBUTE BY HASH ("a");
			CREATE TABLE t2 (
			a SERIAL NOT NULL PRIMARY KEY,
			message CHAR(20)
			)
			WITH (ORIENTATION = ROW, COMPRESSION = NO)
			NOCOMPRESS
			DISTRIBUTE BY HASH ("a");
			CREATE VIEW total(a, message) AS
			SELECT * FROM t1
			UNION ALL
			SELECT * FROM t2;
				UNION is an option used in table creation and it works only when you create **MERGE** tables. Creating a table using this option is similar to creating a common view. The created table logically combines the data of multiple tables that are specified by **UNION**.

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
LIKE (table cloning)	Add additional table attribute information.	CREATE TABLE IF NOT EXISTS \`public\`.\`runoob_tbl_old\`(\`dataType_1\` YEAR, \`dataType_2\` YEAR(4), \`dataType_3\` YEAR DEFAULT '2018', \`dataType_4\` TIME DEFAULT NULL); CREATE TABLE \`runoob_tbl\` (like \`runoob_tbl_old\`);	CREATE TABLE "public"."runoob_tbl_old" ("datatype_1" VARCHAR(4), "datatype_2" VARCHAR(4), "datatype_3" VARCHAR(4) DEFAULT '2018', "datatype_4" TIME WITHOUT TIME ZONE DEFAULT NULL) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("datatype_1"); CREATE TABLE "public"."runoob_tbl"(LIKE "public"."runoob_tbl_old" INCLUDING COMMENTS INCLUDING CONSTRAINTS INCLUDING DEFAULTS INCLUDING INDEXES INCLUDING STORAGE);	N/A
DROP TABLE	The default schema is **public**.	DROP TABLE \`test_create_table01\`;	DROP TABLE "public"."test_create_table01";	N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
TRUNCATE (table deletion)	N/A	TRUNCATE TABLE \`test_create_table01\`;	TRUNCATE TABLE "public"."test_create_table01" CONTINUE IDENTITY RESTRICT;	N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
Hash indexes	Replace these indexes with common indexes based on GaussDB features during synchronization.	CREATE TABLE \`public\`.\`test_create_table03\` (\`DEMAND_ID\` INT(11) NOT NULL AUTO_INCREMENT, \`DEMAND_NAME\` CHAR(100) NOT NULL, \`THEME\` VARCHAR(200) NULL DEFAULT NULL, \`SEND_ID\` INT(11) NULL DEFAULT NULL, \`SEND_NAME\` CHAR(20) NULL DEFAULT NULL, \`SEND_TIME\` DATETIME NULL DEFAULT NULL, \`DEMAND_CONTENT\` TEXT NOT NULL, PRIMARY KEY(\`DEMAND_ID\`), INDEX CON_INDEX(DEMAND_CONTENT(100)) USING HASH , INDEX SEND_INFO_INDEX USING HASH (SEND_ID,SEND_NAME(10),SEND_TIME)); ALTER TABLE runoob_alter_test ADD KEY alterTable_addKey_indexType(dataType1) USING HASH;	CREATE TABLE "public"."test_create_table03" ("demand_id" SERIAL NOT NULL, "demand_name" CHAR(100) NOT NULL, "theme" VARCHAR(200) DEFAULT NULL, "send_id" INTEGER(11) DEFAULT NULL, "send_name" CHAR(20) DEFAULT NULL, "send_time" TIMESTAMP WITHOUT TIME ZONE DEFAULT NULL, "demand_content" TEXT NOT NULL, PRIMARY KEY ("demand_id")) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("demand_id"); CREATE INDEX "con_index" ON "public"."test_create_table03" ("demand_content"); CREATE INDEX "send_info_index" ON "public"."test_create_table03" ("send_id","send_name", "send_time"); CREATE INDEX "altertable_addkey_indextype" ON	N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
			"public"."runoob_alter_test" ("datatype1");	
B-tree indexes	DRS automatically adjusts the DDL statements based on GaussDB features during synchronization.	ALTER TABLE runoob_alter_test ADD KEY alterTable_addKey_indexType (dataType1) USING BTREE;	CREATE INDEX "altertable_addkey_indextype" ON "public"."runoob_alter_test" ("datatype1");	N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
Spatial indexes	DRS automatically adjusts the DDL statements based on GaussDB features during synchronization.	CREATE TABLE \`public\`.\`test_create_table04\` (\`ID\` INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY, \`A\` POINT NOT NULL, \`B\` POLYGON NOT NULL, \`C\` GEOMETRYCOLLECTION NOT NULL, \`D\` LINESTRING NOT NULL, \`E\` MULTILINESTRING NOT NULL, \`F\` MULTIPOINT NOT NULL, \`G\` MULTIPOLYGON NOT NULL, SPATIAL INDEX A_INDEX(A), SPATIAL INDEX B_INDEX(B), SPATIAL INDEX C_INDEX(C), SPATIAL KEY D_INDEX(D), SPATIAL KEY E_INDEX(E), SPATIAL KEY F_INDEX(F), SPATIAL INDEX G_INDEX(G));	CREATE TABLE "public"."test_create_table04" ("id" SERIAL NOT NULL PRIMARY KEY, "a" POINT NOT NULL, "b" POLYGON NOT NULL, "c" CIRCLE NOT NULL, "d" POLYGON NOT NULL, "e" BOX NOT NULL, "f" BOX NOT NULL, "g" POLYGON NOT NULL) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("id"); CREATE INDEX "a_index" ON "public"."test_create_table04" USING GIST ("a"); CREATE INDEX "b_index" ON "public"."test_create_table04" USING GIST ("b"); CREATE INDEX "c_index" ON "public"."test_create_table04" USING GIST ("c"); CREATE INDEX "d_index" ON "public"."test_create_table04" USING GIST ("d"); CREATE INDEX "e_index" ON	N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
			"public"."test_create_table04" USING GIST ("e"); CREATE INDEX "f_index" ON "public"."test_create_table04" USING GIST ("f"); CREATE INDEX "g_index" ON "public"."test_create_table04" USING GIST ("g")	

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks
Full-text indexes	DRS automatically adjusts the DDL statements based on GaussDB features during synchronization.	CREATE TABLE \`public\`.\`test_create_table02\` (\`ID\` INT(11) NOT NULL PRIMARY KEY, \`TITLE\` CHAR(255) NOT NULL, \`CONTENT\` TEXT NULL, \`CREATE_TIME\` DATETIME NULL DEFAULT NULL, FULLTEXT (\`CONTENT\`)); CREATE TABLE IF NOT EXISTS \`public\`.\`runoob_dataType_test\` (\`id\` INT PRIMARY KEY AUTO_INCREMENT, \`name\` VARCHAR(128) NOT NULL, FULLTEXT INDEX (name)); CREATE TABLE IF NOT EXISTS \`public\`.\`runoob_dataType_test\` (\`id\` INT PRIMARY KEY AUTO_INCREMENT, \`name\` VARCHAR(128) NOT NULL, FULLTEXT INDEX (name ASC));	CREATE TABLE "public"."test_create_table02" ("id" INTEGER(11) NOT NULL PRIMARY KEY, "title" CHAR(255) NOT NULL, "content" TEXT, "create_time" TIMESTAMP WITHOUT TIME ZONE DEFAULT NULL) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS DISTRIBUTE BY HASH ("id"); CREATE INDEX "idx_test_create_table02_content" ON "public"."test_create_table02" USING GIN(to_tsvector(coalesce("content",''))); CREATE TABLE "public"."runoob_datatype_test" ("id" SERIAL PRIMARY KEY, "name" VARCHAR(128) NOT NULL) WITH (ORIENTATION = ROW, COMPRESSION = NO) NOCOMPRESS	N/A

Conversion Object	Conversion Action	Original Statement Example	New Statement After Conversion	Remarks

			DISTRIBUTE BY HASH ("id");
			CREATE INDEX "idx_runoob_datatype_test_name" ON "public"."runoob_datatype_test" USING
			GIN(to_tsvector(coalesce("name",'')));
			CREATE TABLE "public"."runoob_datatype_test"
			(
			"id" SERIAL PRIMARY KEY,
			"name" VARCHAR(128) NOT NULL
			)
			WITH (ORIENTATION = ROW, COMPRESSION = NO)
			NOCOMPRESS
			DISTRIBUTE BY HASH ("id");
			CREATE INDEX "idx_runoob_datatype_test_name" ON "public"."runoob_datatype_test" USING
			GIN(to_tsvector(coalesce("name",'')));	

Typical DML Failure Causes

- There is a mismatch between the field types of the source and destination databases. For details about data type mapping, see **here**.
- The field precision and width of the source database are inconsistent with those of the destination database.

Precautions for Many-to-One Real-Time Synchronization

In many-to-one data synchronization tasks, it is recommended that only the ADD COLUMN operation is synchronized, or tasks may fail or data may be inconsistent due to changes in destination tables. Typical failure scenarios are as follows: Synchronizing the TRUNCATE operation clears all destination data; Synchronizing the ADD INDEX operation locks the destination table; Synchronizing the RENAME operation causes other task failure because the destination table cannot be found; Synchronizing the MODIFY COLUMN operation causes other task failure due to incompatible data types.

- **High-risk operations for many-to-one synchronization**
  - High-risk DDL statements, such as DROP, TRUNCATE, and RENAME, are not filtered out. As a result, these DDL operations are executed on the destination table.
  - DDL statements are not executed in a correct sequence, so the synchronization task is interrupted and data conflicts are not detected. As a result, data inconsistency occurs after the synchronization.
  - Many-to-one data synchronization for tables does not support Online DDL tools. Online DDL tools use temporary tables and rename temporary tables by executing the RENAME operation. If the RENAME operation is not filtered out in many-to-one data synchronization tasks, the destination data will be lost.
- **DDL operation skills for many-to-one synchronization**
  - ADD COLUMN

    If ADD COLUMN is filtered out when you configure a many-to-one data synchronization task, add columns to the destination table, and then add columns to all source tables. The synchronization task is automatically compatible with the scenario where the number of columns in the source table is less than that in the destination table. Therefore, the task is not affected. If columns are added for source tables prior to the destination table, the synchronization task will be interrupted.

    If ADD COLUMN is not filtered out when you configure a many-to-one data synchronization task, DRS automatically detects the DDL statement and executes it only once, which does not cause the task to fail.
  - MODIFY COLUMN

    If MODIFY COLUMN is filtered out when you configure a many-to-one data synchronization task, ensure that all source tables have been synchronized without latency and no data is written, modify columns in the destination table, and then modify columns in all source tables.

    If MODIFY COLUMN is not filtered out when you configure a many-to-one data synchronization task, DRS automatically detects and executes it only once, which does not cause task failure.
  - DROP COLUMN

    If DROP COLUMN is filtered out when you configure a many-to-one data synchronization task, delete columns from all source tables, and then delete columns from the destination table. The synchronization task is automatically compatible with the scenario where the number of columns in the source table is less than that in the destination table. Therefore, the task is not affected. If columns are deleted from the destination database prior to source databases, the synchronization task will be interrupted.

If DROP COLUMN is not filtered out when you configure a many-to-one data synchronization task, DRS automatically detects and executes it only once, which does not cause task failure.

– DROP and TRUNCATE operations:

  You are advised to filter out these high-risk operations when configuring DRS tasks and manually execute them. If you do not filter out the high-risk operations when you configure a many-to-one data synchronization task, the DROP and TRUNCATE operations on a source table will be synchronized to the destination table. For example, if a table is dropped from a source database, the destination table will be dropped from the destination database. The DROP INDEX and DROP CONSTRAINT statements are similar.

6.8 Character Set Compatibility Between Oracle and GaussDB

During synchronization from Oracle to GaussDB, if the character sets of the source and destination databases are incompatible, some data may include garbled characters or the synchronization task may fail. For details, see **Table 6-4**.

Table 6-4 Character set compatibility

Source/ Destination Database	UTF8	GBK	GB2312	GB18030
US7ASCII	√	√	√	√
UTF8	√	×	×	×
AL32UTF8	√	×	×	×
ZHS16GBK	√	√	×	√
WE8ISO8859P1	√	×	×	×
WE8ISO8859P2	√	×	×	×
WE8ISO8859P4	√	×	×	×
WE8ISO8859P5	√	×	×	×
WE8ISO8859P7	√	×	×	×
WE8ISO8859P9	√	×	×	×
WE8ISO8859P13	√	×	×	×
WE8ISO8859P15	√	×	×	×
WE8MSWIN1252	√	×	×	×

6.9 Garbled Characters or Synchronization Failure Due to Incompatible Character Sets

If the character set of the source database is incompatible with that of the destination database, some data may include garbled characters, data synchronization may be inconsistent, or data may fail to be written into the destination database. In this case, change the character set of the destination database before synchronization.

7 Data Subscription

7.1 How Long Does It Take for SDK to Consume Database Changes?

Users can obtain database changes within 1s if they have consumed the existing subscription information.

7.2 Why Data Cannot Be Obtained Using the Subscribed SDK and the Program Is Abnormal?

Symptom

When a DRS data subscription task is created, data cannot be obtained using the subscribed SDK, and the program is abnormal.

Fault Locating

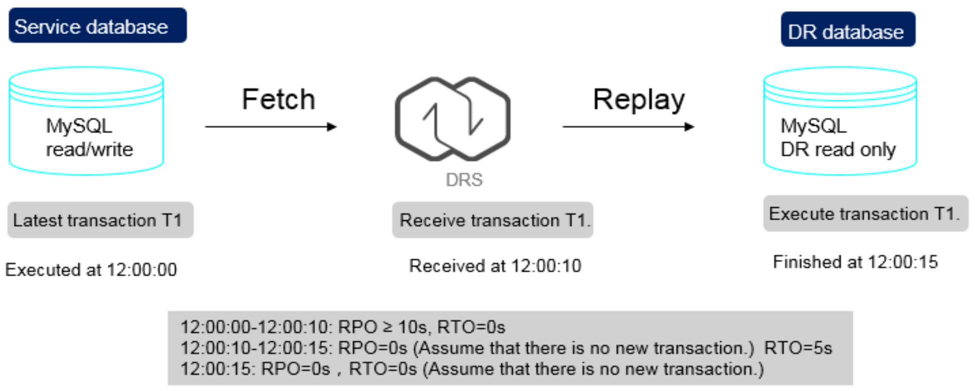
- Check whether the parameters of the SDK APIs are correctly configured. For details, see **Introduction to SDK APIs**.
- Check the network connection. Currently, data subscription supports only VPCs, but not container-based networks in a VPC. If the subscription end runs in a container and the entire-subnet route is not enabled, the network may be disconnected and data cannot be obtained.

8 Real-Time Disaster Recovery

8.1 What Are RPO and RTO of DRS Disaster Recovery?

- Recovery Point Objective (RPO) refers to the difference between the time when a transaction in the current service database is submitted and the time when the transaction is sent to DRS. Generally, the transaction is the latest transaction received by DRS. RPO measures the difference between the data in the service database and the data in the DRS instance. When RPO equals 0, all the data in the service database has been migrated to the DRS instance.
- Recovery Time Objective (RTO) refers to the time difference between the time when a transaction on the current DRS instance is transmitted to the DR instance and the time when the transaction is successfully executed. (This transaction is usually the latest transaction received by DRS.) RTO measures the amount of data being transmitted. When RTO is 0, all transactions on the DRS instance have been completed on the DR database.

Figure 8-1 RPO and RTO



12:00:00-12:00:10: RPO ≥ 10s, RTO=0s
12:00:10-12:00:15: RPO=0s (Assume that there is no new transaction.) RTO=5s
12:00:15: RPO=0s , RTO=0s (Assume that there is no new transaction.)

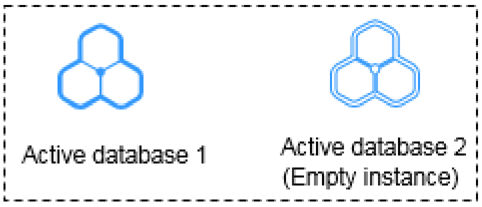
8.2 How Do I Select Active Database 1 and 2 for Dual-Active DR?

In dual-active DR mode, at least one of the two DR databases must be an RDS DB instance on the current cloud, and the other can be an RDS DB instance on the current cloud, other cloud database, self-built database on the ECS, or on-premises database. DRS uses active database 1 and active database 2 to distinguish RDS roles on the current cloud (region). After you determine the role of RDS on the current cloud, the other role is also determined.

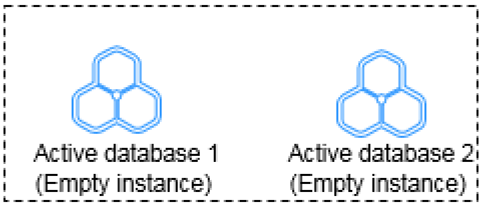
- Active database 1: Generally, service data is on the active database 1. If you select active database 1 when creating a DR task, initial data is stored in the RDS DB instance.
- Active database 2: The database must be empty. If you select active database 2 when creating a DR task, the RDS database on the current cloud is empty and waits for receiving data.

When creating a DR task, comply with the given principles to select active database 1 and 2 in the following scenarios:

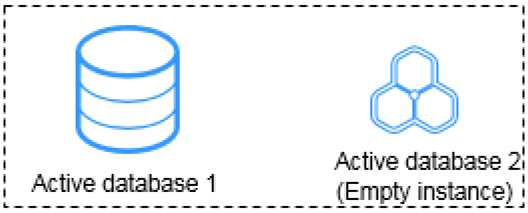
- Both the DR and backup databases are on the RDS DB instances on the current cloud.
  - If one of the instances is empty, the empty instance functions as the active database 2, and the non-empty instance functions as the active database 1.

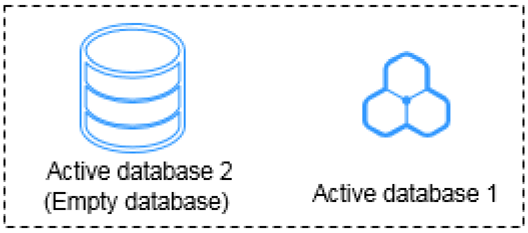
  - Both DB instances are empty. You are advised to select active database 2.

- On database is on the RDS DB instance on the current cloud, and the other is a self-built database on the ECS or on-premises database.
  - One database has initial data, and the other is empty.
    - If the RDS DB instance on the current cloud is empty, select active database 2.



- If the RDS DB instance on current cloud has initial data and the other is empty, select active database 1.



– Both databases are empty. You are advised to select active database 2.

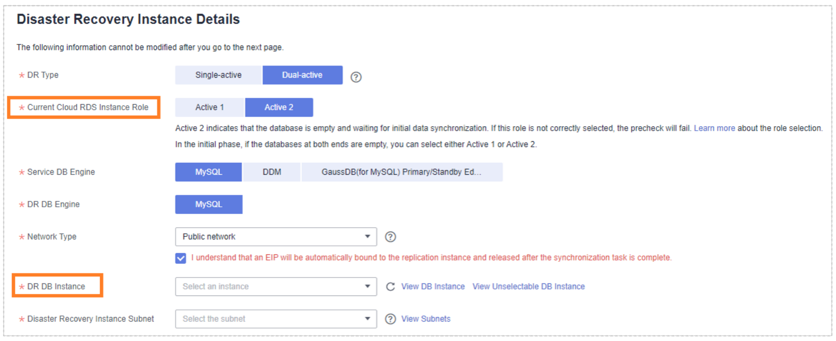
8.3 What Is the Meaning of Forward and Backward Subtasks in Dual-Active Disaster Recovery?

DRS uses active 1 and active 2 to distinguish RDS roles in the current cloud (local region) when you create a dual-active DR task. Active 1 indicates that the selected RDS instance has initial data. Active 2 indicates that the selected RDS instance is empty and waits to receive data.

Figure 8-2 shows the dual-active DR instance information.

- If **Active 2** is selected, the DB instance selected is empty and will receive data. After the task is created, configure a forward task for migrating data to the cloud. After the forward task enters the DR state, configure and start the backward task.
- If you select **Active 1**, the DB instance selected has initial data and data to be synchronized. After the task is created, configure a backward task for migrating data out of the cloud. After the backward task enters the DR state, configure and start the forward task.

Figure 8-2 Dual-active DR instance information



Disaster Recovery Instance Details

The following information cannot be modified after you go to the next page.

* DR Type	Single-active [✓ Dual-active]
* Current Cloud RDS Instance Role	Active 1 [✓ Active 2]
	Active 2 indicates that the database is empty and waiting for initial data synchronization. If this role is not correctly selected, the precheck will fail. Learn more about the role selection. In the initial phase, if the databases at both ends are empty, you can select either Active 1 or Active 2.
* Service DB Engine	[✓ MySQL] DDM GaussDB(for MySQL) Primary/Standby Ed...
* DR DB Engine	[✓ MySQL]
* Network Type	Public network
	✓ I understand that an EIP will be automatically bound to the replication instance and released after the synchronization task is complete.
* DR DB Instance	Select an instance · View DB Instance · View Unselectable DB Instance
* Disaster Recovery Instance Subnet	Select the subnet · View Subnets

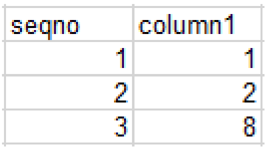
8.4 Common Exceptions in Real-Time Disaster Recovery

Due to certain uncontrollable reasons, data may be inconsistent when data in both the databases is changed at the same time during DR. This section describes common data exceptions. The dr1 and dr2 databases are used as examples in the following scenarios.

Scenario 1: In dual-active DR mode, operations are performed on the same row in the two databases at the same time. As a result, multiple data records are generated.

- The following figure shows the initial data (seqno is the primary key and column1 is the non-primary key).

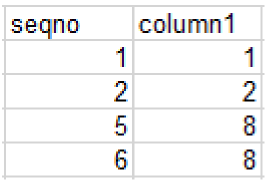
Figure 8-3 Initial data in the dr1 and dr2 databases



seqno	column1
1	1
2	2
3	8

- Perform the following operations on both databases:
  - dr1: update dr1 set seqno=5 where column1=8;
  - dr2: update dr2 set seqno=6 where column1=8;
- After the operations are performed, the data in the databases is consistent but an additional row is generated.

Figure 8-4 Data in the dr1 and dr2 databases

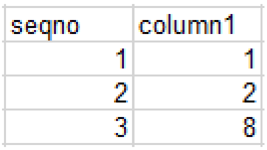


seqno	column1
1	1
2	2
5	8
6	8

Scenario 2: In dual-active DR mode, operations are performed on the same row in both databases at the same time. As a result, data records become inconsistent.

- The following figure shows the initial data (seqno is the primary key and column1 is the non-primary key).

Figure 8-5 Initial data

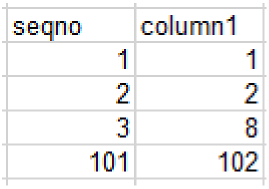


seqno	column1
1	1
2	2
3	8

- Perform the following operations on both databases:

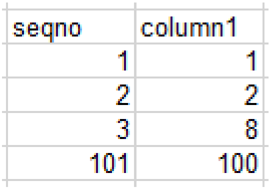
  - dr1: insert into dr1 values(101, 100);
  - dr2: insert into dr2 values(101, 102);
- After the operations are performed, the data in the databases is shown in the following figure.

Figure 8-6 Data in the dr1 database



seqno	column1
1	1
2	2
3	8
101	102

Figure 8-7 Data in the dr2 database

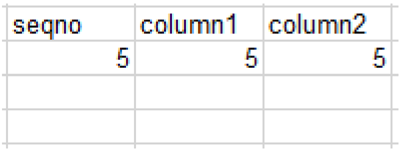


seqno	column1
1	1
2	2
3	8
101	100

Scenario 3: In dual-active DR mode, DDL operations are performed. As a result, data records become inconsistent.

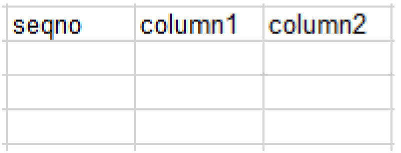
- Perform the following operations on both databases:
  - dr1: truncate table dr1;
  - dr2: insert into dr2 values(5,5,5);
- After the operations are performed, the data in the databases becomes inconsistent.

Figure 8-8 Data in the dr1 database



seqno	column1	column2
5	5	5

Figure 8-9 Data in the dr2 database



seqno	column1	column2

More scenarios are being added.

8.5 Is a Primary/Standby Switchover Triggered Automatically or Manually for DR Tasks?

For real-time DR tasks, if the service database is faulty, manually perform a primary/standby switchover. For details, see **Performing a Primary/Standby Switchover**.

8.6 Can Real-Time DR Be Performed for Specified Databases?

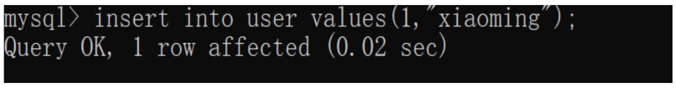
Real-time DR is performed by instance. You cannot select a specified database. You can select a specified table or database for real-time migration and synchronization.

8.7 Why Does a Real-Time DR Task Not Support Triggers and Events?

Database trigger and event operations are recorded in binlogs. DRS parses binlogs to synchronize data. If the service side writes the same data as the trigger and event operations, repeated execution will occur, causing data inconsistency or task failure. Therefore, in DR scenarios, triggers and events are not supported.

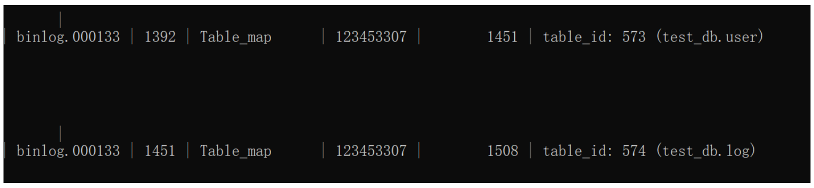
If the user table in the source database has a trigger, when data is written to the user table, the trigger writes a piece of log data to another log table.

The service side on the source database writes a piece of data to the user table.



```
mysql> insert into user values(1,"xiaoming");
Query OK, 1 row affected (0.02 sec)
```

The trigger synchronizes the piece of data to the log table. In this case, there are two pieces of data in binlogs. As shown in the following figure, the first piece of data is the data inserted into the user table by the service side, and the second piece of data is the data written to the log table by the trigger.



```
| binlog.000133 | 1392 | Table_map     | 123453307 |        1451 | table_id: 573 (test_db.user)

| binlog.000133 | 1451 | Table_map     | 123453307 |        1508 | table_id: 574 (test_db.log)
```

The following situations may occur during DRS data synchronization:

- If the inserted data is synchronized to the user table on destination database first, the trigger of the destination database automatically writes data to the log table on the destination database. When the second log table data is synchronized, it cannot be written to the destination database, and a data conflict task reports an error.

- If the log table data is synchronized first, and then the data in the user table, the trigger of the destination database writes data to the log table. As a result, one more data record is added to the log table, causing data inconsistency.

Similarly, the event operations are also recorded in binlogs and executed again in the destination database, which also causes the preceding problem.

DRS real-time migration tasks support triggers and events because DRS migrates triggers and events when a task is stopped to ensure that objects in the destination database are consistent with those in the source database.

9 Data-Level Comparison

9.1 Which of the Following Data Types Are Not Supported By Value Comparison?

DRS's data comparison allows you to check whether the data in the source database is the same as that in the destination database.

DRS does not support value comparison for the data types shown here. During value comparison, these data types are automatically ignored.

Table 9-1 Data types that do not support value comparison

Source DB Type	Data Type
MySQL	TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB, TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT
GaussDB	TEXT, CLOB, BLOB, BYTEA, INTERVAL DAY TO SECOND, INTERVAL
Oracle	BLOB, NCLOB, CLOB, LONG RAW, LONG, INTERVAL DAY TO SECOND, INTERVAL YEAR TO MONTH, UROWID, BFILE, XMLTYPE, SDO_GEOMETRY
MongoDB	_id is of the bindata type.

DRS does not support value comparison for the following primary key types. During value comparison, the following primary key types are grouped into a specified table that does not support comparison.

Table 9-2 Primary key type that does not support value comparison.

Source DB Type	Data Type
MySQL	TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB, TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT, FLOAT
GaussDB	TEXT, CLOB, BLOB, BYTEA, INTERVAL DAY TO SECOND, INTERVAL, REAL, DOUBLE PRECISION, BOOL, TIME, TIMETZ, TIMESTAMP, TIMESTAMPTZ, DATE
Oracle	BLOB, NCLOB, CLOB, LONG RAW, LONG, INTERVAL DAY TO SECOND, INTERVAL YEAR TO MONTH, UROWID, BFILE, XMLTYPE, SDO_GEOMETRY, BINARY_FLOAT, BINARY_DOUBLE, FLOAT, RAW, TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE, DATE
PostgreSQL	REAL, DOUBLE PRECISION, MONEY, TEXT, BYTEA, TIMESTAMP WITHOUT TIME ZONE, TIMESTAMP WITH TIME ZONE, DATE, TIME WITHOUT TIME ZONE, TIME WITH TIME ZONE, INTERVAL, BOOLEAN, ENUMERATED TYPES, POINT, LINE, LSEG, BOX, PATH, POLYGON, CIRCLE, CIDR, INET, MACADDR, MACADDR8, BIT, BIT VARYING, TSVECTOR, TSQUERY, XML, JSON, ARRAY, COMPOSITE TYPES, INT4RANGE, INT8RANGE, NUMRANGE, TSRANGE, TSTZRANGE, DATERANGE

9.2 What Impact Does a DRS Comparison Task Have on Databases?

- Object comparison: System tables of the source and destination databases are queried, occupying about 10 sessions. The database is not affected. However, if there are a large number of objects (for example, hundreds of thousands of tables), the database may be overloaded.
- Row comparison: The number of rows in the source and destination databases is queried, which occupies about 10 sessions. The SELECT COUNT statement does not affect the database. However, if a table contains a large amount of data (hundreds of millions of records), the database will be overloaded and the query results will be returned slowly.
- Value comparison: All data in the source and destination databases is queried, and each field is compared. The query pressure on the database leads to high I/O. The query speed is limited by the I/O and network bandwidth of the source and destination databases. Value comparison occupies one or two CPUs, and about 10 sessions.
- Account comparison: The accounts and permissions of the source and destination databases are queried, which does not affect the database.

9.3 How Long Does a DRS Comparison Task Take?

- Object comparison: Generally, the comparison results are returned within several minutes based on the query performance of the source database. If the amount of data is large, the comparison may take dozens of minutes.
- Row comparison: The SELECT COUNT method is used. The query speed depends on the database performance.
- Value comparison: If the database workload is not heavy and the network is normal, the comparison speed is about 5 MB/s.
- Account comparison: The results are returned with the object-level comparison results. If the number of objects is small, the results are returned in several minutes.

10 General Operations

10.1 What Can I Do When Information Overlaps on the DRS Console?

Information often overlaps when you decrease the size of the page. You are advised to set the page scale at 100%.

10.2 Is the Destination Instance Set to Read-only or Read/Write?

When configuring a migration task, you can set the destination instance to **Read-only** or **Read/Write**.

- **Read-only**: During the migration, the entire destination instance is read-only. After the migration is complete, it restores to the read/write status. This option ensures the integrity and success rate of data migration.
- **Read/Write**: During the migration, the destination instance can be queried or modified. Data being migrated may be modified when operations are performed or applications are connected. It should be noted that background processes can often generate or modify data, which may result in data conflicts, task faults, and upload failures. Do not select this option if you do not fully understand the risks.

Setting the destination instance to read-only can prevent DDL or DML misoperations from being performed on the databases or tables that are being migrated, improving migration integrity and data consistency.

- After a migration task is started, the status of the destination database cannot be changed.
- After all migration tasks in which the destination database status is set to read-only are complete, the destination database can be read and written.

10.3 How Do I Set Global binlog_format=ROW to Take Effect Immediately?

During migration or synchronization for MySQL databases, the source database binlog must be in the ROW format. Otherwise, the task fails. After **binlog_format=ROW** at the global level is set in the source database, all the previous service threads need to be stopped because these threads still connect the binlog in the non-ROW format.

Procedure

Step 1 Log in to the source database using the MySQL official client or other tools.

Step 2 Run the following command for setting global parameters in the source database.

```
set global binlog_format = ROW;
```

Step 3 Run the following command on the source database and check whether the preceding operation is successful:

```
select @@global.binlog_format;
```

Step 4 You can use either of the following methods to ensure that the modified binlog format of the source database takes effect immediately:

Method 1

1. Select a non-service period to disconnect all service connections on the current database.
   a. Run the following command to query all service threads (excluding all binlog dump threads and current threads) in the current database:
      ```
      show processlist;
      ```
   b. Stop all the service threads queried in the previous step.

   NOTE

   Do not create or start a migration task before the preceding operations are complete. Otherwise, data may be inconsistent.
2. To prevent the binlog format of the source database from becoming invalid due to database restart, add or modify the **binlog_format** parameter in the startup configuration file (**my.ini** or **my.cnf**) of the source database and save the modification.
   ```
   binlog_format=ROW
   ```

Method 2

1. To prevent the binlog format of the source database from becoming invalid due to database restart, add or modify the **binlog_format** parameter in the startup configuration file (**my.ini** or **my.cnf**) of the source database and save the modification.
   ```
   binlog_format=ROW
   ```
2. Ensure that the **binlog_format** parameter is successfully added or modified. Then, restart the source database at a non-service period.

----End

10.4 How Do I Set binlog_row_image=FULL to Take Effect Immediately?

When migrating MySQL databases, ensure that the **binlog_row_image** parameter of the source database is set to **FULL**. Otherwise, the migration task will fail. After **binlog_row_image** is set to **FULL** in the source database, the setting takes effect only for new sessions. To close old sessions, restart the source database and reset the task during a non-service period.

Setting binlog_row_image to FULL

- If the source is an RDS instance on the cloud, change **binlog_row_image** to **FULL** on the RDS console, and then restart the source database and reset the task.
- If the source database is an on-premises database, perform the following steps:
  a. Log in to the server where the MySQL source database is located.
  b. Manually change the value of **binlog_row_image** in the **my.cnf** configuration file to **FULL** and save the file.
     ```
     binlog_row_image=full
     ```
  c. To close old sessions, restart the source database and reset the task during a non-service period.

10.5 How Do I Change the Destination Database Password to Meet the Password Policy?

Scenarios

When you set the password for the migration account in the destination database, you need to set the password based on the password strength requirements of the destination database.

Procedure

The following operations apply to the scenario where the target database is an RDS instance.

Step 1 Log in to the RDS console.

Step 2 Locate the target DB instance.

Step 3 Click the DB instance name.

Step 4 On the **Basic Information** page, click the **Parameters** tab.

Step 5 Enter the keyword **password** in the search box in the upper right corner of the page and press **Enter** to view the search result.

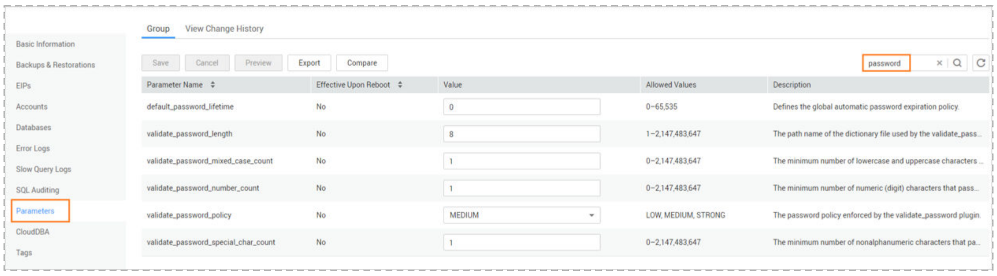


Figure 10-1 Modifying parameters

Basic Information
Backups & Restorations
EIPs
Accounts
Databases
Error Logs
Slow Query Logs
SQL Auditing
Parameters
CloudDBA
Tags

Group | View Change History

Save | Cancel | Preview | Export | Compare | password

Parameter Name	Effective Upon Reboot	Value	Allowed Values	Description
default_password_lifetime	No	0	0~65,535	Defines the global automatic password expiration policy.
validate_password_length	No	8	1~2,147,483,647	The path name of the dictionary file used by the validate_pass...
validate_password_mixed_case_count	No	1	0~2,147,483,647	The minimum number of lowercase and uppercase characters ...
validate_password_number_count	No	1	0~2,147,483,647	The minimum number of numeric (digit) characters that pass...
validate_password_policy	No	MEDIUM	LOW, MEDIUM, STRONG	The password policy enforced by the validate_password plugin.
validate_password_special_char_count	No	1	0~2,147,483,647	The minimum number of nonalphanumeric characters that pa...

Step 6 In the search result in **Step 5**, change the values of the parameters listed in **Table 10-1** based on the password strength requirements. Ensure that the parameter values are within the password complexity range.

Table 10-1 Password description

Parameter	Allowed Value	Description
validate_password_length	0-2,147,483,647	Specifies the minimum password length verified by the validate_password plugin.
validate_password_mixed_case_count	0-2,147,483,647	Specifies the minimum number of lowercase and uppercase letters in a password when the password policy level is **MEDIUM** or higher.
validate_password_number_count	0-2,147,483,647	Specifies the minimum number of digits in a password when the password policy level is **MEDIUM** or higher.
validate_password_policy	LOW, MEDIUM, STRONG	Specifies the password policy executed by the validate_password plugin.
validate_password_special_char_count	0-2,147,483,647	Specifies the minimum number of non-alphanumeric characters in a password when the password policy level is **MEDIUM** or higher.

Step 7 After the parameter values are modified, save the modification.

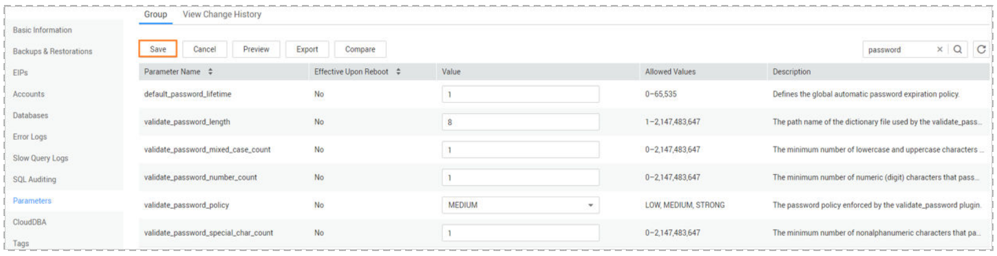


Figure 10-2 Modification result

Basic Information
Backups & Restorations
EIPs
Accounts
Databases
Error Logs
Slow Query Logs
SQL Auditing
Parameters
CloudDBA
Tags

Group | View Change History

Save | Cancel | Preview | Export | Compare | password

Parameter Name	Effective Upon Reboot	Value	Allowed Values	Description
default_password_lifetime	No	1	0~65,535	Defines the global automatic password expiration policy.
validate_password_length	No	8	1~2,147,483,647	The path name of the dictionary file used by the validate_pass...
validate_password_mixed_case_count	No	1	0~2,147,483,647	The minimum number of lowercase and uppercase characters ...
validate_password_number_count	No	1	0~2,147,483,647	The minimum number of numeric (digit) characters that pass...
validate_password_policy	No	MEDIUM	LOW, MEDIUM, STRONG	The password policy enforced by the validate_password plugin.
validate_password_special_char_count	No	1	0~2,147,483,647	The minimum number of nonalphanumeric characters that pa...

Step 8 Back to the **Select Migration Type** page and perform the next step.

----End

10.6 How Do I Configure the Shard Key for a MongoDB Sharded Cluster?

MongoDB shards data at the collection level, distributing the collection data using shard keys.

You choose the shard key when sharding a collection. Each record contains a shard key, and the shard key is either an indexed field or indexed compound fields. MongoDB database distributes data in different chunks according to the shard key, and distributes chunks evenly among the shards. To divide data chunks by shard key, MongoDB database uses two sharding methods: range-based sharding and hashed sharding.

Table 10-2 Shard key classification

Shard Key Type	Description	Application Scenario
Range-based sharding	Ranged-based sharding involves dividing data into contiguous ranges determined by the shard key values. Range-based sharding is the default sharding methodology if no other options are specified. This allows for efficient queries where reads target documents within a contiguous range. The distribution route determines which data chunk stores the data required and forwards the request to the corresponding shard.	It is recommended when the shard key has high cardinality with low frequency, and the shard key value does not change monotonically.
Hashed sharding	Hashed sharding uses a hashed index to partition data across your shared cluster and to create chunks. Hashed sharding provides more even data distribution across the sharded cluster Hash values enable data to be randomly distributed in each chunk, and therefore are randomly distributed in different shards.	If the shard key values that have a high cardinality or change monotonically, or there are large number of different values, hashed sharding is an ideal option.

Once you shard a collection, the shard key and the shard key values are immutable. If you need to modify the shard key of a document, you must delete the document. Then modify the shard key and insert the document again.

NOTE

The shard key does not support array indexes, text indexes, geographical indexes, and spatial indexes.

Range-based Sharding

Step 1 Run the following command to enable database sharding:

sh.enableSharding*(database)*

NOTE

database indicates the database for which the sharded collection is enabled.

Step 2 Configure the collection's shard key.

sh.shardCollection*(namespace, key)*

NOTE

- ***namespace*** consists of a string <database>.<collections> specifying the full namespace of the target collection.
- ***key*** indicates the index for the shard key.

- If the collection is empty, skip this step because the index on the shard key can be created automatically.

  sh.shardCollection()
- If the collection is not empty, create an index key. Then, run the following command to set the shard key:

  sh.shardCollection()

----End

Hashed Sharding

Step 1 Run the following command to enable database sharding:

sh.enableSharding*(database)*

NOTE

database indicates the database for which the sharded collection is enabled.

Step 2 Set hashed shard keys.

sh.shardCollection(*"<database>.<collection>", { <shard key> : "hashed" }* , false, {numInitialChunks: *Number of preconfigured chunks*})

The value of **numInitialChunks** is calculated as follows: db.collection.stats().size / 10*1024*1024*1024.

If the collection contains data, run the following command to create a hashed index for the hashed key:

db.collection.createIndex()

Run the following command to create a hashed shard key:

sh.shardCollection()

----End

10.7 Does Bandwidth Expansion Affect the Running DRS Tasks?

When the cloud connection bandwidth is expanded, the bandwidth link needs to be re-established and the network is disconnected. Whether the network disconnection affects DRS tasks depends on the network disconnection duration and whether the source database IP address changes. For example, for the MySQL DB engine, if the network is disconnected for one day and the binlog of the source database is cleared within this day (the binlog clearing policy of MySQL is configured by the user), the task cannot be resumed. In this scenario, you need to reset the task. If the network is interrupted for a short period of time and the IP address of the source database in the VPN remains unchanged after the bandwidth link is changed, the system can continue to resume the task.

10.8 Why Data in MariaDB and SysDB Cannot Be Migrated or Synchronized?

In some MariaDB versions, the SysDB database is used as a system database (similar to the sys database of MySQL 5.7). Therefore, DRS considers the SysDB database as the system database of all MariaDB databases by default (similar to the MySQL, information_schema, and performance_schema databases). If the SysDB is a service database, you can apply for a service ticket.

10.9 Constraints and Operation Suggestions on Many-to-One Scenario

DRS supports many-to-one scenarios during migration or synchronization of different types of instances and tables to suit your service requirements.

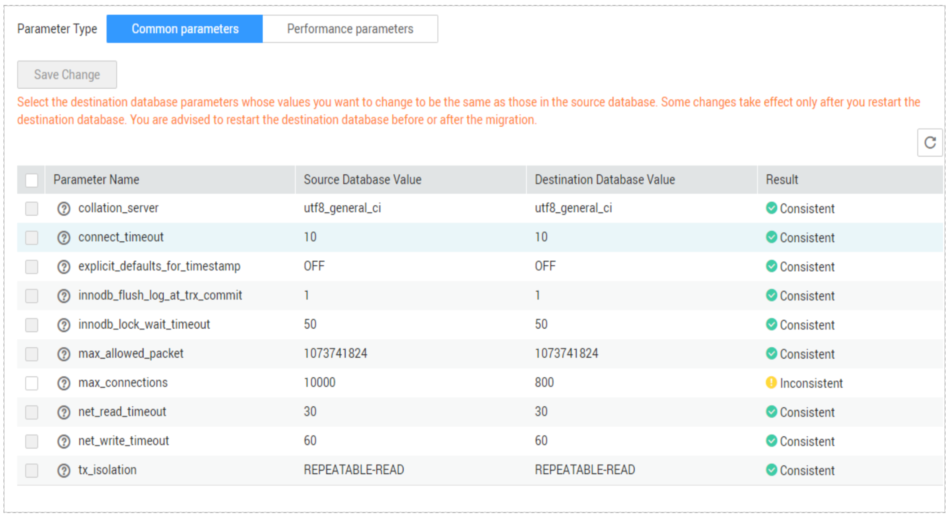
Operation Suggestions

- To ensure that there is sufficient space during task creation, you are advised to calculate the total data volume of the source database and plan how to allocate the disk space of the destination instance. The remaining disk space must be greater than the total data volume of the source database. For example, if the data volume of source system1 is 1 GB, the data volume of source system2 is 3 GB, and the data volume of source system3 is 6 GB, the remaining disk space of the destination instance must be greater than 10 GB.
- To improve the performance of the destination MySQL database, you are advised to use the **Save Change** function to configure common parameters (except **max_connections**). For performance parameters, you need to manually change the parameter values based on the specifications of the destination database.
- When you create a many-to-one synchronization task, the task created later may block the task created earlier. This is because each synchronization task

involves index creation. When an index is created, a schema lock may occur on the destination database, which blocks the synchronization of other tables in the schema. As a result, the previously created tasks cannot be synchronized. To avoid this problem, you are advised to set **Start Time** to **Start at a specified time** to start a task during off-peak hours.

- For many-to-one synchronization tasks that involve the synchronization of the same table, DDL operations cannot be performed on source databases. Otherwise, all synchronization tasks fail.

Figure 10-3 Parameter comparison



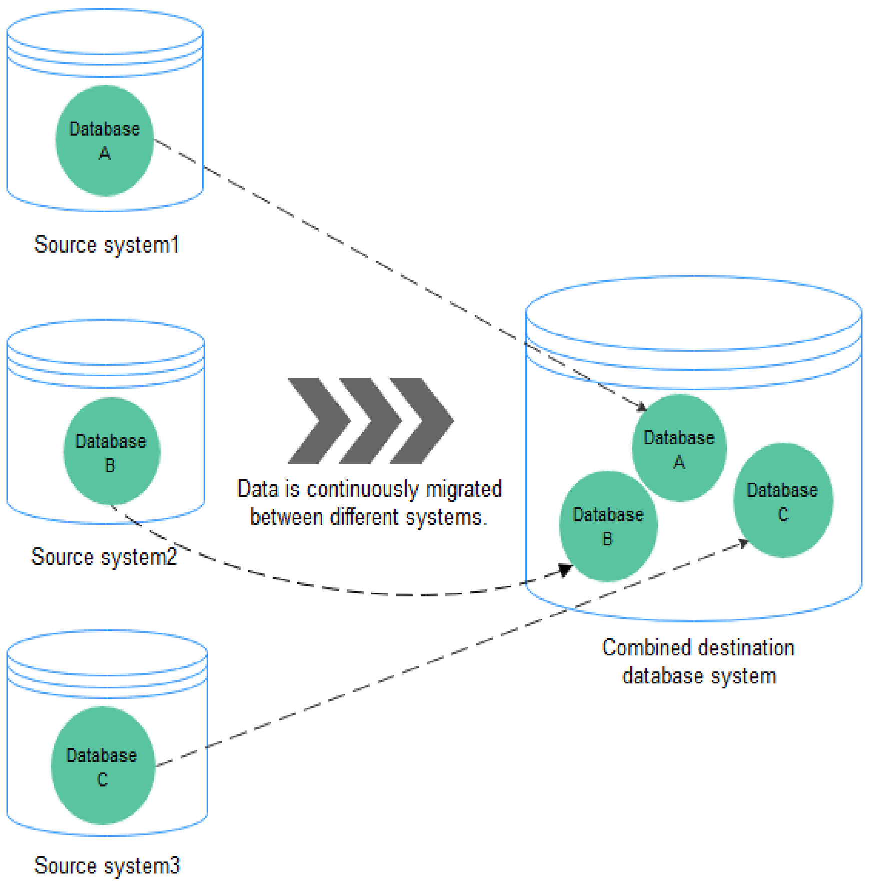
Select the destination database parameters whose values you want to change to be the same as those in the source database. Some changes take effect only after you restart the destination database. You are advised to restart the destination database before or after the migration.

Parameter Name	Source Database Value	Destination Database Value	Result
collation_server	utf8_general_ci	utf8_general_ci	Consistent
connect_timeout	10	10	Consistent
explicit_defaults_for_timestamp	OFF	OFF	Consistent
innodb_flush_log_at_trx_commit	1	1	Consistent
innodb_lock_wait_timeout	50	50	Consistent
max_allowed_packet	1073741824	1073741824	Consistent
max_connections	10000	800	Inconsistent
net_read_timeout	30	30	Consistent
net_write_timeout	60	60	Consistent
tx_isolation	REPEATABLE-READ	REPEATABLE-READ	Consistent

Scenario 1: Many-to-One Data Migration

Data migration aims to migrate the entire database. Multiple databases can be migrated at the instance level. Databases with the same name in the source system cannot be migrated and database name mapping is not supported.

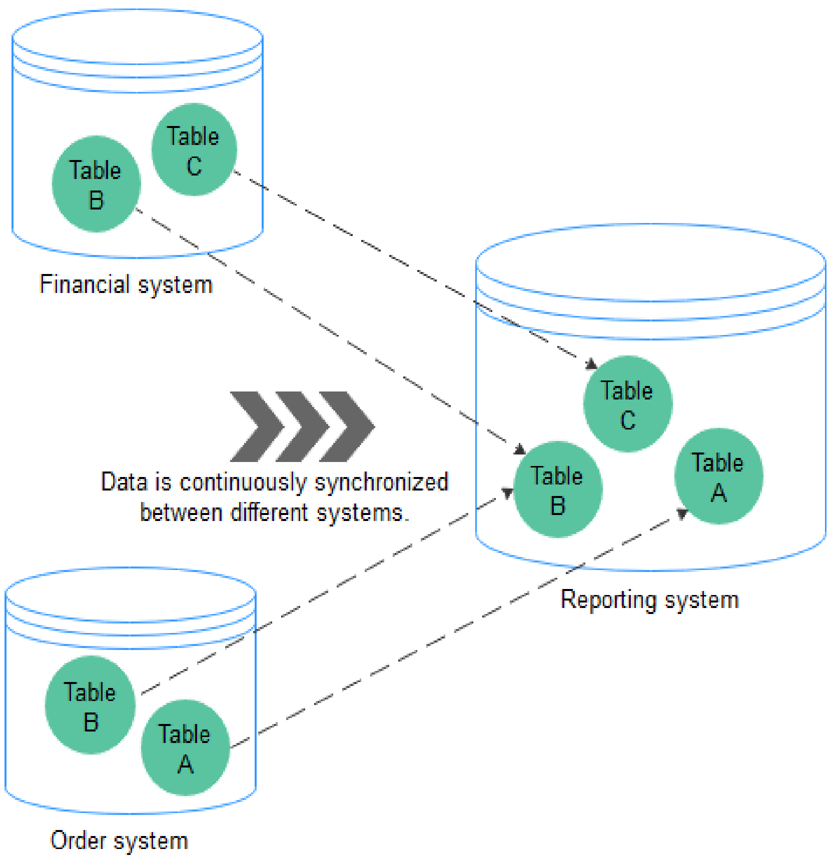
Figure 10-4 Many-to-one data migration



Scenario 2: Many-to-One Real-Time Synchronization

Unlike data migration, real-time synchronization maintains continuous data flow between different services. It supports table-level, many-to-one synchronization and database-level mapping.

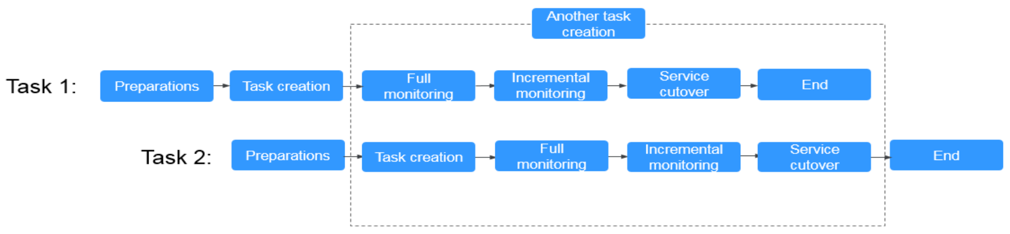
Figure 10-5 Many-to-one synchronization



Flow Chart

When creating a task, ensure that the second task is created after the first task has entered the full migration state. For details, see **Overview**

Figure 10-6 Flow chart



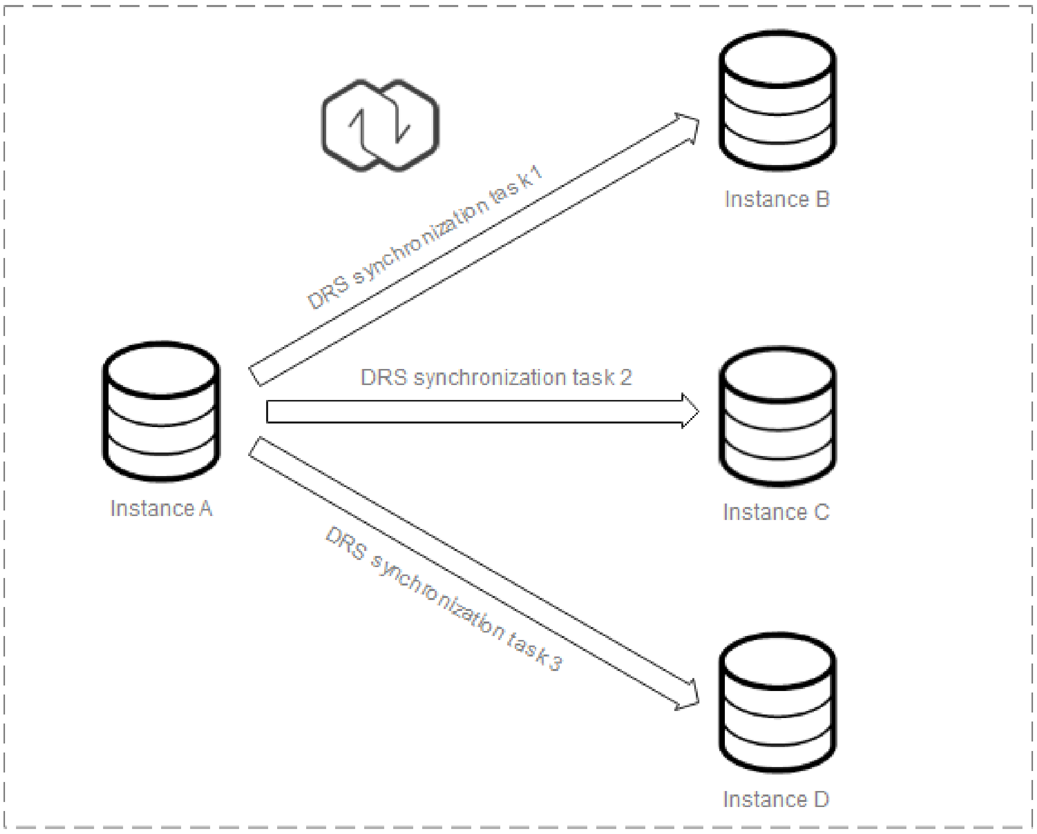
10.10 Constraints and Operation Suggestions on One-to-Many Scenario

DRS supports one-to-many scenarios during migration or synchronization to suit your service requirements.

Operation Suggestions

- In the one-to-many scenario, multiple DRS tasks are required. The workload on the source database is multiplied. To prevent the pressure caused by multiple DRS tasks on the same source database from affecting services on the source database, you need to analyze the load of the source database in advance. If the source database is heavily loaded, you are advised to reduce the number of synchronization tasks.
- You are advised to create tasks one by one in the one-to-many scenario.

One-to-Many Real-Time Synchronization at the Instance Level

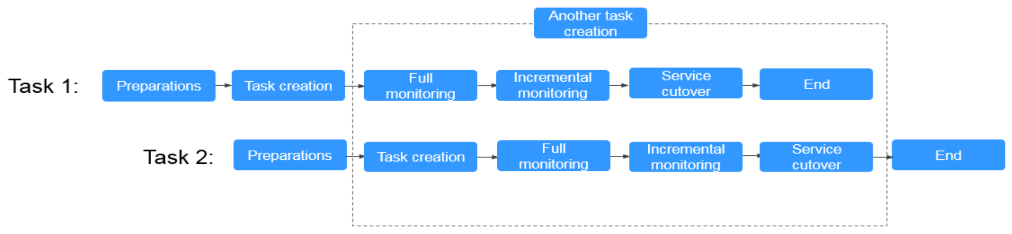

DRS synchronization task 1
Instance B
DRS synchronization task 2
Instance A
Instance C
DRS synchronization task 3
Instance D

You need to create multiple synchronization tasks to implement one-to-many real-time synchronization. For example, to synchronize data from instance A to instances B, C, and D, you need to create three synchronization tasks.

Flowchart

When creating a task, ensure that the second task is created after the first task has entered the full phase. For details, see **Overview**

Figure 10-7 Flowchart



10.11 Where Can I View DRS Operation Logs?

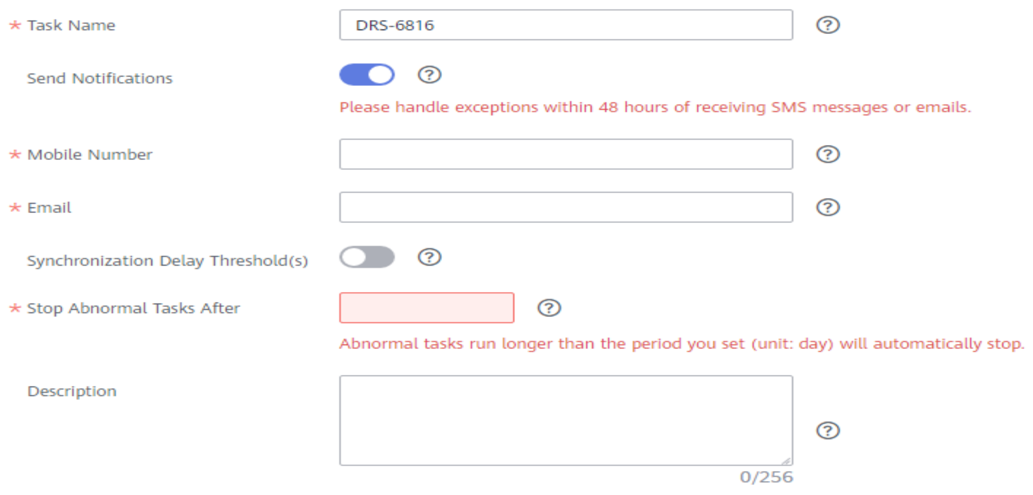
You can view DRS operation logs on the Cloud Trace Service (CTS) console.

Click the username in the upper right corner and select **Operation Log** from the drop-down list.

10.12 Why Is a DRS Task Automatically Stopped?

To avoid unnecessary charges, you can set **Stop Abnormal Tasks After** to a value between 14 to 100 days to automatically stop abnormal tasks. Abnormal tasks run longer than the period you set (unit: day) will automatically stop to avoid unnecessary fees.

Figure 10-8 Stop Abnormal Tasks After

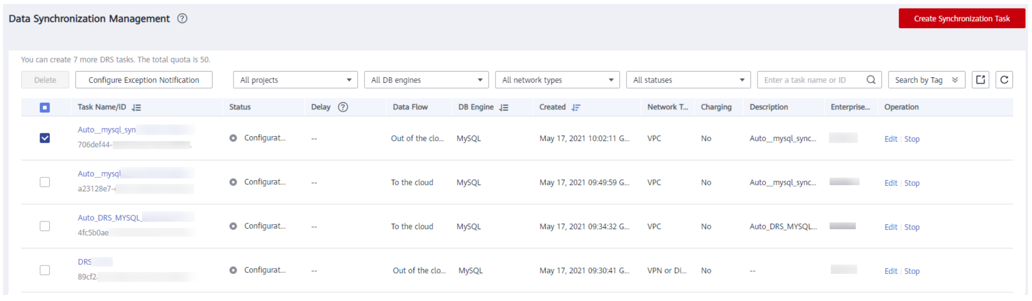


* Task Name	DRS-6816
Send Notifications	On — Please handle exceptions within 48 hours of receiving SMS messages or emails.
* Mobile Number	
* Email	
Synchronization Delay Threshold(s)	Off
* Stop Abnormal Tasks After	Abnormal tasks run longer than the period you set (unit: day) will automatically stop.
Description	0/256

10.13 How Can I Export a DRS Task List?

DRS allows you to query and export detailed task information, including the task name, ID, status, network type, IP addresses or domain names of the source and destination databases, port number, and alarm configuration, so that you can view and manage tasks in different dimensions. In the task list, set the search criteria and click ☐ to export the query result.

Figure 10-9 Exporting task query results


Data Synchronization Management
Create Synchronization Task
You can create 7 more DRS tasks. The total quota is 50.
Delete
Configure Exception Notification
All projects
All DB engines
All network types
All statuses
Enter a task name or ID
Search by Tag

	Task Name/ID	Status	Delay	Data Flow	DB Engine	Created	Network T...	Charging	Description	Enterprise...	Operation
✓	Auto__mysql_syn[redacted] 706def44-[redacted]	Configurat...	--	Out of the clo...	MySQL	May 17, 2021 10:02:11 G...	VPC	No	Auto__mysql_sync...	[redacted]	Edit | Stop
☐	Auto__mysql[redacted] a23128e7-[redacted]	Configurat...	--	To the cloud	MySQL	May 17, 2021 09:49:59 G...	VPC	No	Auto__mysql_sync...	[redacted]	Edit | Stop
☐	Auto_DRS_MYSQL_[redacted] 4fc5b0ae[redacted]	Configurat...	--	To the cloud	MySQL	May 17, 2021 09:34:32 G...	VPC	No	Auto_DRS_MYSQL...	[redacted]	Edit | Stop
☐	DRS[redacted] 89cf2[redacted]	Configurat...	--	Out of the clo...	MySQL	May 17, 2021 09:30:41 G...	VPN or Di...	No	--	[redacted]	Edit | Stop

10.14 Can a Completed Task Be Restarted?

No. DRS cannot restart a completed task.

For a synchronization task that was completed but not deleted, you can **clone** the configuration of the existing task and create a new task.

10.15 What Are the Differences Between Resetting a Task and Recreating a Task?

You can reset a task when the task is suspended or fails. Resetting a task does not clear the destination database. You can determine whether to clear the destination database based on your requirements. After the task is reset, a full synchronization is performed again. You do not need to configure the task again.

For details, see **Resetting a Synchronization Task**.

10.16 Does DRS Support Backward Migration/ Synchronization?

DRS real-time migration and real-time synchronization tasks cannot be reversed.

A single-active DR task can be reversed through a **primary/standby switchover**.

Different from single-active DR, dual-active DR involves forward and backward DR tasks, and primary/standby switchover is not required.

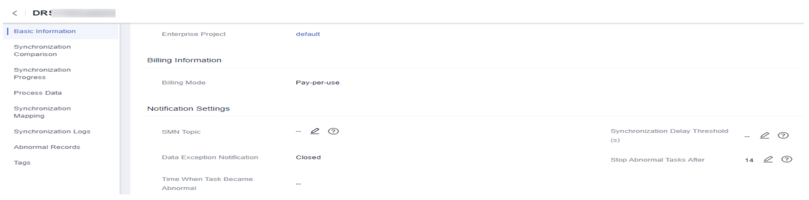
10.17 Why Cannot I Select an Existing SMN Topic?

You may not have confirmed the subscription to a topic. After creating a topic, you need to add the subscription to the topic. SMN then sends subscription confirmation messages to the subscription endpoints, including a confirmation link. The subscription confirmation link is valid within 48 hours. Confirm the subscription on your mobile phone, mailbox, or other endpoints in time. For details, see **Requesting Subscription Confirmation**.

10.18 Can I Change an SMN Topic After a Task Is Created?

You can change the SMN topic. Click the task name to go to the **Basic Information** page. In the **Send Notifications** area, change the SMN topic.

Figure 10-10 Changing an SMN topic


DRS[redacted]

Enterprise Project	default

Billing Information

Billing Mode	Pay-per-use

Notification Settings

SMN Topic	--	Synchronization Delay Threshold (s)	--
Data Exception Notification	Closed	Stop Abnormal Tasks After	14
Time When Task Became Abnormal	--		

10.19 Will Data of DRS Tasks Be Lost After a Primary/ Standby Switchover Occurs on the Source MySQL Database?

If GTID is enabled for the source MySQL database, DRS records the binlog position information when a primary/standby switchover occurs. After the primary/standby switchover, DRS resumes data transfer from the last interrupted position to prevent data loss.

You can modify the database configuration file to enable GTID for the source MySQL database. The modification takes effect after the database is restarted.

The reference commands are as follows:

```
gtid_mode = on
log_slave_updates = true
enforce_gtid_consistency = on
```

10.20 What Are the Differences Between All, Tables, and Databases During DRS Object Selection?

You can select objects for DRS real-time migration and synchronization tasks. Real-time DR is performed by instance and does not support the selection of specified databases or tables.

During real-time migration, you can select **All**, **Tables**, or **Databases** for the migration objects.

During real-time synchronization, you can select **Tables** or **Databases** for the synchronization objects.

- **All**: This option is available only for real-time migration tasks. All objects in the source database are migrated to the destination database. After the migration, the object names will remain the same as those in the source database and cannot be modified.

- **Tables**: The selected table-level objects will be migrated or synchronized. New tables in the source database are not automatically added to the migration or synchronization objects. For a synchronization task, you can edit the synchronization objects to add new tables. Migration objects cannot be edited after the migration task is started.
- **Databases**: The selected database-level objects will be migrated or synchronized. Tables cannot be selected separately. New tables in the selected database will be automatically added to the migration or synchronization objects.

10.21 What Do I Do After Changing the Password of the Source or Destination Database?

A DRS task may fail due to the change of the password of the source or destination database. To continue the task, rectify the information and then retry the task on the DRS console.

Procedure

Step 1 Select a task from the task list and click the task name.

Step 2 On the **Basic Information** tab, click **Modify Connection Details** in the **Connection Information** area.

Step 3 In the displayed dialog box, change the passwords of the source and destination databases and click **OK**.

Step 4 View the change result on the **Basic Information** tab.

----End

11 Billing

11.1 Do I Have to Pay For Failed Tasks?

DRS charges the tasks you created from the time when the tasks are started to the time when the tasks are complete. During this period, billing will not be stopped even if the tasks fail. To avoid unnecessary charges, you can set **Stop Abnormal Tasks After** to a value between 14 to 100 days to automatically stop abnormal tasks. Abnormal tasks run longer than the period you set (unit: day) will automatically stop to avoid unnecessary fees.

11.2 Do I Have to Pay For Paused Tasks?

You will be charged for the configuration and data transfer based on a pay-per-use or yearly/monthly basis.

Table 11-1 Billing items

Item	Description	Rule
Configuration fee	Configuration fees are generated when you use computing and storage resources and process data.	● In pay-per-use mode, you are charged based on the actual usage on an hourly basis. If the usage duration is less than one hour, you are charged a full hour. ● In yearly/monthly mode, you need to make upfront payments on a yearly or monthly basis.

Item	Description	Rule
Data transfer fee	Data transfer fees are generated when you process or transfer data through a public network (excluding VPN, VPC, Direct Connect, and CC networks).	You are billed based on the actually used public network traffic in GB.

You will be charged a configuration fee even if the task is paused.

11.3 Will DRS Tasks That Are Not Started Be Billed?

DRS tasks can be started upon task creation or at a specified time based on your service requirements. The billing of a DRS task starts when the task is started. If you select **Start at a specified time**, you will not be charged before the specified start time.

11.4 What Will Happen to My Tasks After The Yearly/monthly Subscription Expires?

If your bills on yearly/monthly tasks are overdue and you have not topped up your account or renew resources in a timely manner, your tasks will enter a grace period. If you do not top up your account or renew resources after the grace period expires, the tasks enter the retention period. During the retention period, you cannot perform any operations on the task on the DRS console, or call related APIs, and O&M activities such as automatic monitoring and alarm reporting are stopped. If the account is not topped up or the resource package is not renewed before the retention period expires, the DRS instance will become unavailable and data stored on DRS will be deleted and cannot be recovered.

11.5 Will I Be Charged If I Do Not Delete a Task After It Is Completed?

If a task is not deleted after successful completion, you will not be billed. A finished task cannot be restarted.

11.6 Resource Freezing, Release, Deletion, and Unsubscription

Why Are My DRS Resources Released?

If your subscriptions have expired but not been renewed, or you are in arrears due to insufficient balance, your resources enter a grace period. If you still do not complete the payment or renewal after the grace period expires, you will enter a retention period. During the retention period, the resources are not available. If the renewal is still not completed or the outstanding amount is still not paid off when the retention period ends, the stored data will be deleted and the cloud service resources will be released. For details, see **Service Suspension and Resource Release**.

Why Are My DRS Resources Frozen?

Your resources may be frozen for a variety of reasons. The most common reason is that you are in arrears.

How Do I Unfreeze My Resources?

Frozen due to arrears: You can renew your resources or top up your account. DRS instances frozen due to arrears can be renewed, released, or deleted. Yearly/Monthly DRS instances that have expired cannot be unsubscribed from, while those that have not expired can be unsubscribed from.

What Happens When My Resources Are Frozen, Unfrozen, or Released?

- After your resources are frozen:
  - They cannot be accessed, causing downtime. For example, if your DRS instance is frozen, data cannot be migrated.
  - If they are yearly/monthly resources, no changes can be made to them.
  - They can be unsubscribed from or deleted manually.
- After your resources are unfrozen, data can be migrated.
- After your resources are released, your instance will be deleted.

How Do I Renew My Resources?

After a yearly/monthly DRS instance expires, you can renew it on the **Renewals** page. For details, see **Renewal Management**.

Can My Resources Be Recovered After Being Released? /Can I Retrieve an Incorrect Unsubscription?

Deleted instances cannot be recovered.

Before unsubscribing from a resource, confirm the resource information carefully. If you have unsubscribed from a resource by mistake, you are advised to purchase a new one.

How Can I Delete a DRS Instance?

- A pay-per-use DRS instance can be deleted only after it is stopped. For details about how to delete a DRS migration task, see **Deleting a Migration Task**.
- A yearly/monthly DRS instance can be deleted only after it is unsubscribed. For details about how to delete a DRS synchronization task, see **Unsubscribing from a Yearly/Monthly Task**.

12 Delay

12.1 Why Does the Delay of DR Tasks Increase?

Causes for an Increase in RTO

Recovery Time Objective (RTO) is duration of time within which transactions on the DRS instance are transmitted and replayed to the destination database during incremental synchronization. If the RTO value is large, transactions to be replayed on the DRS instance are stacked. The possible causes are as follows:

1. After a DR task is initialized, the incremental data generated from the time when the DR task is started to the current time needs to be replayed.
2. Batch operations are performed on service database tables that do not have primary keys. The DR instance is synchronizing tables that do not have primary keys and have a large amount of changed data. To ensure data consistency in tables without primary keys, all operations are recorded. As a result, the operation execution efficiency is lower than that in tables with primary keys. In addition, if the destination table has no index, the data update efficiency is lower.
3. If the DDL operation is performed on the service database, the DR instance can replay data only after the execution of the DDL operation is complete.
4. Frequently executed operations are performed on hot tables in the service database. The DR instance combines the transactions of the hot table and then replays the transactions, reducing frequent operations on the destination database.
5. The access to the DR database is abnormal. As a result, the incremental data cannot be replayed.

Handling Suggestion

Step 1 On the **Disaster Recovery Management** page, click the target DR task in the **Task Name/ID** column.

Step 2 On the **Basic Information** page, click the **Disaster Recovery Monitoring** tab to view the changes of RTO.

- If RTO decreases gradually or increases only in a short time, no action is required.
- If RTO keeps increasing, run the following statement in the DR database to check whether there are SQL statements that take a long time to execute or DDL statements that are being executed:
  ```
  show processlist
  ```
- If the DR database is abnormal, contact database O&M engineers.

----End

Causes for an Increase in RPO

Recovery Point Objective (RPO) refers to the time that passes from when a transaction in the service database is submitted and the time when the transaction is synchronized to the DRS instance during the incremental synchronization. If RPO is large, the latest changes made to the service database data have not been extracted to the DR instance. The possible causes are as follows:

1. The network between the service database and the DRS DR instance is unstable. Reading changes in logs from the service database is slow.
2. The service database cannot be accessed. As a result, incremental data cannot be extracted.

Handling Suggestion

Step 1 On the **Disaster Recovery Management** page, click the target DR task in the **Task Name/ID** column.

Step 2 On the **Basic Information** page, click the **Disaster Recovery Monitoring** tab to view the changes of RPO.

- If RPO decreases gradually or increases only in a short time, no action is required.
- If the service database is abnormal, contact database O&M engineers.

----End

12.2 Why Is the Delay High In MongoDB Replication Scenarios?

Involved Scenarios

- Migration from MongoDB to DDS
- Migration from DDS to MongoDB
- Synchronization from DDS to MongoDB

Possible Causes

To ensure the performance of migration, synchronization, or disaster recovery, DRS performs concurrent replay at the collection level in the incremental phase. In the following special cases, DRS supports only single-thread write and does not support concurrent replay:

- The collection index contains a unique key.
- The value of **capped** of the collection attribute is **true**.

If the delay increases, check whether the problem is caused by the preceding reasons.

12.3 What Are Possible Causes of Slow Migration or Suspended Progress in Full Phase?

Symptom

During a full migration, the task takes a long time or the migration progress is not updated.

Fault Locating

- Check the size of data to be migrated in the source database.

  The data migration progress depends on the number of tables. If the data migration progress is not updated for a long time, there may be large volumes of data in some tables. In the **Migration Details** area, locate the target migration object and click **View Details** in the **Operation** column to view the migration progress.
- Check the primary keys and indexes of the tables in the source database.

  Large tables in the source database lack primary keys and NOT NULL unique indexes. Take MySQL as an example. Run the **show create table** *<Database name >.<Table name >* command in the source database to check whether a table has a primary key or NOT NULL unique index.
- Check whether the persistent connection of the source database is stopped.

  If the source is a database on other clouds, persistent connections may be automatically terminated. As a result, the full migration takes a long time or the migration progress is not updated.
- Check the index migration of the destination database.

  If the index migration progress is not updated for a long time, the possible cause is that it takes a long time to create indexes in some large tables and the destination database keeps creating indexes. Log in to the destination database and run the **show processlist** command to view the DRS status in the destination database.
- Check whether a deadlock occurs in the destination database.

  If a deadlock occurs in the destination database, full data may fail to be written. Take MySQL as an example. Run the following commands to view and delete deadlocks:
  - Run the **show OPEN TABLES where In_use > 0** command to check whether the table is locked.
  - Run the **show processlist** command to view the table locking process.
  - Run the **KILL [CONNECTION | QUERY] <thread_id>** command to delete the table locking progress.

- Check the network connection between the source database and destination database.

  Check whether the network connectivity is normal and whether the network bandwidth is limited. Run OS commands such as **ping** to test the network connectivity and delay.
- Check whether **Flow Control** is enabled for the DRS task.

  Click the task name and check whether **Flow Control** is enabled in the **Flow Control Information** area on the **Basic Information** tab.

12.4 What Are Possible Causes of High Latency in DRS Incremental Phase?

Symptom

In the incremental migration or synchronization, the task latency is high.

Possible Causes

- Cause 1: The full phase is just complete, and the incremental migration delay is long. During a full migration, incremental data is continuously written to the source database. DRS synchronizes the incremental data to the destination database after the full migration is complete. The latency is high.
- Cause 2: Large volumes of data is imported, updated, or deleted in the source database. It takes a long time to write a large transaction to the source database, and it takes a period of time for DRS to synchronize data to the destination database. The latency increases gradually. If the table of the large transaction does not have a primary key or index, the recovery time is prolonged.
- Cause 3: A lot of DDL operations are performed in the source database. As a result, the latency increases.
- Cause 4: The DRS task specifications are limited. Different DRS specifications correspond to different performance upper limits. When the amount of data written to the source database reaches the bottleneck, tasks will be delayed.
- Cause 5: The class of the destination database is limited, reaching the write bottleneck. For example, if the destination database is RDS for MySQL, you can view database performance metrics on the RDS console.
- Cause 6: There may be hotspot updates. Writing data to a table without a primary key causes hotspot updates. Frequent updates of a single table or row in the source database also cause hotspot updates, increasing the latency. Take RDS for MySQL as an example. You can check the RDS audit logs.
- Cause 7: The network is unstable.

Solution

- Solution 1: In this case, DRS automatically adjusts the latency to a normal value. No action is required. You can check whether the incremental latency decreases.
- Solution 2: If a large transaction is written, wait until the update is complete or avoid writing a large transaction. You can view the execution history of the

source database to check whether large transactions are written. Also, you can view the DRS data replay in the destination database. Take MySQL as an example. Run the **show processlist** command.

- Solution 3: Do not execute DDL statements in batches in the source database. If required, execute them during off-peak hours.
- Solution 4: Create a synchronization task again and select a larger specification to improve the synchronization performance. (Currently, DRS supports specification upgrade only in MySQL-to-MySQL synchronization tasks with single-node DRS instances configured. Task specifications cannot be downgraded.)
- Solution 5: Upgrade the instance class of the destination database to improve the write performance.
- Solution 6: If there are hotspot updates, wait until the hotspot updates are complete or avoid hotspot updates.
- Solution 7: Access the source and destination databases through Direct Connect to reduce latency.

A Change History

Released On	Description
2022-09-30	This issue is the first official release.