## **Distributed Database Middleware**

# **Best Practices**

**Issue** 01

**Date** 2025-08-01





#### Copyright © Huawei Technologies Co., Ltd. 2025. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

#### **Trademarks and Permissions**

HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd. All other trademarks and trade names mentioned in this document are the property of their respective holders.

#### **Notice**

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

# **Security Declaration**

#### Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process.* For details about this process, visit the following web page:

https://www.huawei.com/en/psirt/vul-response-process

For vulnerability information, enterprise customers can visit the following web page:

https://securitybulletin.huawei.com/enterprise/en/security-advisory

# **Contents**

1 Overview	1
2 Formulating Sharding Rules	3
3 Determining the Number of Shards in a Schema	7
4 Using Broadcast and Unsharded Tables	9
5 Transaction Models	11
6 SQL Standards	14
7 Migrating an Entire RDS Database to DDM	19
8 Migrating an Entire MyCat Database to DDM	22
9 Accessing DDM Using a JDBC Connection Pool	
10 Logging In to a DDM Instance Using Navicat	
11 Migrating Data from RDS for MySQL to DDM Using DRS	
11.1 Overview	
11.2 Resource and Cost Planning	32
11.3 Operation Process	36
11.4 Preparing for the Source RDS for MySQL Instance	37
11.4.1 Creating a VPC and Security Group	37
11.4.2 Creating an RDS for MySQL Instance	39
11.4.3 Creating a VPN for the Source	42
11.5 Preparing for the Destination DDM Instance	43
11.5.1 Creating a VPC and Security Group	43
11.5.2 Creating a DDM Instance	44
11.5.3 Creating an RDS for MySQL Instance	45
11.5.4 Creating a Schema and Associating It with the RDS for MySQL Instance	47
11.5.5 Creating a DDM Account	47
11.5.6 Creating Table Structures in the Destination Database	48
11.5.7 Creating a VPN for the Destination	49
11.6 Creating a VPN Connection	50
11.7 Creating a DRS Migration Task	50
11.8 Checking Migration Results	52

12 Sharding Database and Table Data of an RDS for MySQL Instance	55
12.1 Overview	55
12.2 Preparing an RDS for MySQL Instance	56
12.3 Preparing a DDM Instance	56
12.3.1 Creating a DDM Instance	57
12.3.2 Creating a Schema and Associating It with the Prepared RDS for MySQL Instance	
12.3.3 Creating a DDM Account	
12.3.4 Creating Table Structures in the DDM Instance	59
12.4 Creating a DRS Migration Task	62
12.4.1 Creating a DRS Migration Task	62
12.4.2 Checking Migration Results	64
13 Selecting a Sharding Key and a Sharding Algorithm	66
14 Cross-Shard Complex SQL Tuning	70

# 1 Overview

This document describes how to buy a DDM instance that meets your service requirements and complete relevant settings based on Huawei Cloud DDM practices.

**Table 1-1** DDM best practices

Service	Reference	Description
DDM	Formulating Sharding Rules	Selecting a sharding key and algorithm when creating a sharded table
	Determining the Number of Shards in a Schema	Determining the number of shards when creating a sharded schema
	Using Broadcast and Unsharded Tables	Broadcast and unsharded tables
	Transaction Models	DDM transaction models
	SQL Standards	SQL Standards
	Migrating an Entire RDS Database to DDM	Migrating an entire RDS database to a DDM database
	Migrating an Entire MyCat Database to DDM	Migrating an entire MyCat database to DDM
	Accessing DDM Using a JDBC Connection Pool	Accessing DDM using the JDBC connection pool to perform data operations
	Logging In to a DDM Instance Using Navicat	Obtaining an EIP and using this EIP to connect to a DDM instance through the Navicat client
	Migrating Data from RDS for MySQL to DDM Using DRS	Migrating data from RDS for MySQL to DDM using DRS

Service	Reference	Description		
	Sharding Database and Table Data of an RDS for MySQL Instance	Sharding database and table data of existing RDS for MySQL instances using DDM		
	Cross-Shard Complex SQL Tuning	Optimizing cross-shard complex SQL statements		

# **2** Formulating Sharding Rules

When creating a logical table and selecting **Sharded Table** for **Table Type**, you need to specify a sharding algorithm and a sharding key.

If a relationship exists between entities on different tables, formulate the same sharding rule for these tables, and select the associated table fields as the sharding keys respectively so that associated data in different tables is stored in the same shard to avoid cross-shard JOIN operations. For example, use the customer ID as the sharding key when creating sharded tables for storing customer information, orders, or order details.

Table 2-1 Sharding keys and algorithms

Sharding Algorithm	Hash		Range	
Sharding Key	Table field  +date function		Table field	Table field +date function
Description	Data is evenly distributed to shards by table field.	Data is evenly distributed to shards by table field and date function. The table field must be date, datetime, or timestamp.	Data is distributed to a specific shard based on the rules defined in algorithm metadata.	Data is distributed to shards by table field and date function based on the rules defined in algorithm metadata. The table field must be date, datetime, or timestamp.

to customers (for example, customer account numbers) as the sharding key.  It ime) to players as the sharding by day, month, or year helps you easily collect and query operation statistics of players for a specified day or month, and helps game vendors conduct big data analysis.  It instance, player activity dates, use the table field corresponding to players as the sharding date function shard date easier to activity dates (for example, activity name and date range) as the sharding key. This helps you collect statistics about the sales volume for a specified cycle.  It instance, player activity dates, use the table field corresponding to players as the sharding key. This helps you collect statistics about the sales volume for a specified cycle.  It instance, entities are activity dates, use the table field corresponding to activity dates (for example, activity name and date range) as the sharding key. This helps you collect statistics about the sales volume for a specified cycle.  It ime) corresponding to players as the sharding key. This helps you collect statistics about the sales volume for a specified cycle.  It ime) corresponding to activity dates, use the table field corresponding to activity and easier to activity and easier to activity and easier to activity and easier to activity activi
--

## **Selecting a Sharding Algorithm**

A sharding algorithm partitions data from logical tables to multiple shards. DDM supports hash and range algorithms.

- Hash
   Hash evenly distributes data across shards.
   Select this algorithm if operators = and IN need to be frequently used in SQL queries.
- Range
   Range stores records in tables based on the range specified in algorithm metadata.

Select this algorithm if operators greater (>), less (<), and BETWEEN ... AND ... need to be frequently used in SQL gueries.



If the sharding algorithm is a range algorithm and a DATE function and the sharding key field indicates the creation time, hotspot issues may occur when data is imported to the database. As a result, the advantages of multiple MySQL databases cannot be fully utilized.

Select an appropriate algorithm based on your service requirements to improve efficiency.

#### Selecting a Sharding Key

A sharding key is a table field used to generate a route during horizontal partitioning of logical tables. After specifying a table field, you can select a date function or manually enter *date function (field name)*. The table field must be **date**, **datetime**, or **timestamp**. Select a date function if data needs to be redistributed by year, month, day, week, or some combinations thereof.

DDM calculates routes based on the sharding key and sharding algorithm, horizontally partitions data in sharded tables, and then redistributes it to shards.

Note that when you select a sharding key and a sharding algorithm:

- Ensure that data is evenly distributed to each shard as much as possible.
- Select the most frequently used field or the most important query condition as the sharding key.
- Prioritize the primary key as the sharding key to keep query the fastest.

### Service Scenarios with a Clear Entity

A sharded table generally contains tens of millions of data records. It is extremely important to select an appropriate sharding key and a sharding algorithm. If a logical entity is identified and most database operations are performed on data of that entity, select the table field corresponding to the entity as a sharding key for horizontal partitioning.

Logical entities depend on actual applications. The following scenarios each include a clear logical entity.

- 1. For customer-related applications of banks, the service logical entities are customers. In this case, use the table field corresponding to customers (for example, customer numbers) as the sharding key. Service scenarios of some systems are based on bank cards or accounts. In such cases, select the bank card or account as the sharding key.
- 2. For e-commerce applications, if service scenarios are based on products, the service logical entity is products. In this case, use the table field corresponding to products (for example, product code) as the sharding key.
- 3. Game applications mainly focus on player data, and the service logical entity is players. In this case, use the table field corresponding to players (for example, player ID) as the sharding key.

The following is an example SQL statement for creating tables for bank services:

```
CREATE TABLE PERSONALACCOUNT(
   ACCOUNT VARCHAR(20) NOT NULL PRIMARY KEY,
   NAME VARCHAR(60) NOT NULL,
   TYPE VARCHAR(10) NOT NULL,
   AVAILABLEBALANCE DECIMAL(18, 2) NOT NULL,
   STATUS CHAR(1) NOT NULL,
   CARDNO VARCHAR(24) NOT NULL,
   CUSTOMID VARCHAR(15) NOT NULL
) ENGINE = INNODB DEFAULT CHARSET = UTF8
dbpartition by hash(ACCOUNT);
```

#### **Service Scenarios Without a Clear Entity**

If you cannot identify a suitable entity for your service scenario, select the table field that can provide even data distribution as the sharding key.

For example, the log system may contain a wide range of data records. In this case, you can select the time field as the sharding key.

When the time field is selected as the sharding key, you can specify a date function to partition data.

To make it easier to clear and dump logs, select the range algorithm and convert the time field value into "month" using the date function so that logs are stored in shards by month.

Example SQL statement for creating a table:

```
CREATE TABLE LOG(
LOGTIME DATETIME NOT NULL,
LOGSOURCESYSTEM VARCHAR(100),
LOGDETAIL VARCHAR(10000)
)
dbpartition by range(month(LOGTIME)) {
1 - 2 = 0,
3 - 4 = 1,
5 - 6 = 2,
7 - 8 = 3,
9 - 10 = 4,
11 - 12 = 5,
default = 0
};
```

# 3 Determining the Number of Shards in a Schema

DDM instances support horizontal sharding of tables. After a DDM instance is created, you can create a schema in this instance and specify the shards per MySQL instance and then select appropriate MySQL instances to associate with the schema during the creation. Before creating a schema, you need to first evaluate the expected data volume and determine the shards in a schema. A shard refers to a database in a MySQL instance. The total number of shards in a schema is equal to that of databases in all MySQL instances associated with the schema. There are two schema types:

- Unsharded, a schema corresponds only to one MySQL instance, and only one shard is created for that instance.
- Sharded, a single schema corresponds to multiple MySQL instances. You can specify 1 to 64 shards for each MySQL instance. If more than 64 shards are required, contact DDM technical support.

Rebalancing means that database shards are rebalanced between original and new instances.

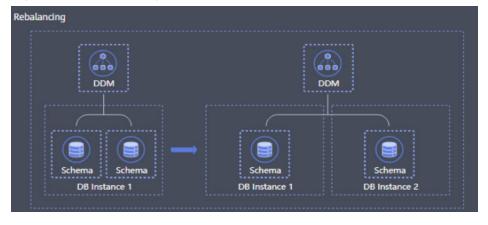
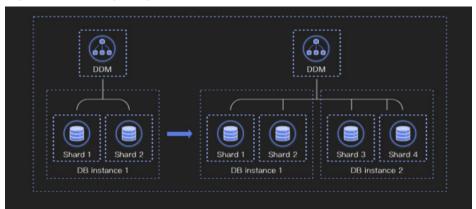


Figure 3-1 Rebalancing shards

Configuring shards: You can increase shards in a schema based on service requirements. A maximum of 64 shards can be distributed to a single data node. If more than 64 shards are required, contact DDM technical support. DDM

distributes shards to all associated data nodes as much as possible. Data will be redistributed as long as the total number of shards changes.

Figure 3-2 Configuring shards



# 4 Using Broadcast and Unsharded Tables

An unsharded table is one whose data is stored in a single default shard. A broadcast table is a table storing the same data in all shards to improve JOIN efficiency.

#### **Unsharded Table**

The DDM console does not allow creation of unsharded tables. To create unsharded tables, you can log in to your DDM instance using a MySQL client or an application.

If a table contains fewer than 10 million data records and has no requests for JOIN operations with other sharded tables for query, set the table type to **Unsharded Table** to store data in the default shard.

#### The following is an example SQL statement for creating an unsharded table:

```
CREATE TABLE single(
id int NOT NULL AUTO_INCREMENT COMMENT 'Primary key ID',
name varchar(128),
PRIMARY KEY(id)
);
```

#### **Broadcast Table**

In business databases, some tables have a small amount of data and are not frequently updated, but they are often used for JOIN operations.

To join such tables and sharded tables, DDM provides broadcast tables, which have the following features:

- The broadcast table stores the same data in each shard. Data insertion, update, and deletion in broadcast tables are executed in each shard in real time.
- Queries of broadcast tables are only executed in one shard.
- JOIN operations can be performed to join any tables with broadcast tables.
- Before using any broadcast hint, ensure that there are available tables.

#### Example:

When the order management system of an e-commerce enterprise needs to query and collect statistics on order data in a region, JOIN query is performed for a

regional table and an order flow table. The order data volume is large and the order flow table needs to be sharded. In this scenario, you can set the regional table as a broadcast table to improve query efficiency.

#### The following is an example SQL statement for creating a broadcast table:

CREATE TABLE broadcast\_tbl (
id int NOT NULL AUTO\_INCREMENT COMMENT 'Primary key ID',
name varchar(128),
PRIMARY KEY(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci
BROADCAST;

# Pay attention to the following when you perform operations on broadcast tables:

- Restrict the execution frequency of SQL statements for full table scanning, or execute them during off-peak hours. Otherwise, an error may occur, indicating that the number of backend RDS connections is insufficient.
- Data is inserted to broadcast tables in each shard. If the table contains more than 1 million data records, you are advised to use sharded tables to avoid occupying too much space.

# 5 Transaction Models

Tables in each DDM instance are usually sharded, so data in the tables may be distributed across different database shards in multiple RDS instances. In DDM, one transaction to add, delete, update, or query a logical table is possibly executed on different database shards in multiple RDS instances, so a series of operations on data tables in one shard of an RDS instance can be deemed as a local transaction. Therefore, one DDM transaction is actually a distributed transaction that consists of local transactions on multiple RDS instances. These local transactions either all succeed, or all fail.

#### Implementation of Distributed Transactions in DDM

DDM executes 2PC distributed transactions based on the MySQL XA protocol, and these transactions can guarantee strong write consistency. For more information about the consistency, see the MySQL official documentation.

In a distributed system, each participant knows whether their operation succeeds or fails, but cannot know the status of the operation on other nodes. If a transaction encompasses multiple nodes, to guarantee atomicity and consistency of a transaction, a coordinator is introduced to control operation results of all participants so that all of the transaction's changes either take effect or do not. The DDM node acts as a coordinator in the distributed transaction, and RDS instances are participants.

The two-phase commit protocol breaks a database commit into two phases:

- 1. **Prepare phase**: Participants notify the coordinator of the operation result. If the participants give a response that they are prepared, they must reserve the required resources before the coordinator makes a decision.
- Commit phase: After receiving a notification from the participants, the
  coordinator sends a notification to the participants again and determines
  whether the participant needs to commit or roll back the operation based on
  their response.

#### For example:

Four people A, B, C, and D want to have a dinner party, so they need to determine the time. Now assume that A is the coordinator and B, C, and D are participants.

#### Prepare phase:

- 1. A sends a text message to B, C, and D, asking them whether they are available on Tuesday noon.
- 2. D replies yes.
- 3. B replies yes.
- 4. C does not reply for a long time. The time is not determined, so A, B, C, and D cannot continue their dinner party.
- C replies yes or no.

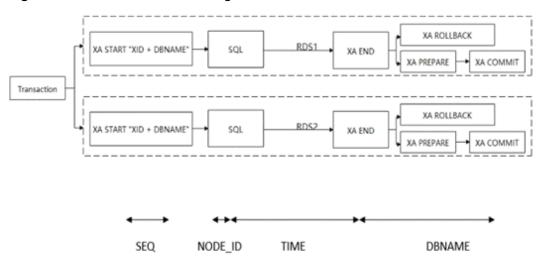
#### Commit phase:

- 1. Coordinator A sends the collected results (Tuesday dinner party) to B, C, and D. What the result is and when does A feed it back to the other three depends on the time when C replies.
- B received.
- C received.
- 4. D received.
- 5. If any of them does not receive the result, A continues to send it until all of them receive the result.

#### 2-Phase Commit

When DDM processes a transaction, applications run BEGIN/COMMIT as they do on a common transaction, without worrying about whether there are distributed transactions at the underlying layer. DDM automatically processes distributed transactions using the two-phase commit (2PC) protocol. If a transaction involves only one data shard, DDM will process it using the one-phase commit protocol. The one-phase commit protocol will not be described in detail here.

Figure 5-1 Flowchart for executing a distributed transaction



XID is the global unique serial number of each distributed transaction. It consists of the transaction serial number, node ID, and timestamp. The XID and the corresponding shard name together constitute the XA\_ID that initiates a distributed transaction on a physical database.

#### **Suggestions for Using Transactions**

- DDM supports distributed transactions. If all SQL statements in a transaction use the same sharding key, DDM can process the transaction as a single-shard transaction to achieve optimal performance.
- DDM uses the 2PC protocol to process distributed transactions. RDS deadlock detection does not take effect for distributed transactions across RDS instances. If a lock wait times out, there may be a cross-shard deadlock. In this case, check your business model.

# 6 SQL Standards

#### **INSERT**

#### Syntax rules

- Do not use INSERT to add data entries one by one. INSERT INTO VALUES (),()..() is recommended.
- The MySQL JDBC driver ignores the executeBatch() statement by default, splits SQL statements that should be executed in batches into single statements, and sends them to the MySQL database one by one. This greatly deteriorates database performance. To execute SQL statements in batches, you have to set parameter rewriteBatchedStatements to true and ensure that the SQL JDBC driver is of version 5.1.13 or later. The driver executes SQL statements in batches only when rewriteBatchedStatements is set to true. This configuration is valid for INSERT, UPDATE, and DELETE operations.

## **A** CAUTION

After you set **rewriteBatchedStatements** to **true**, set a proper value for **batch size** to control the number of INSERT, UPDATE, and DELETE operations. If the **batch size** value is too large, the performance may deteriorate. Unless otherwise specified, the value is not larger than **1000**.

- Do not set the sharding key value to a function, expression, or sub-query.
   A constant is recommended.
- Do not set a common key value to a sub-query. A constant, function, or expression is recommended.

#### • Batch data import

LOAD DATA LOCAL INFILE is recommended for importing a large volume of data in batches.



You only need to enable a session, and DDM will automatically imports data.

#### • Data migration

Use mysqldump to export SQL files and import them by running the MySQL source command.

#### • Field auto increment

- DDM can create a unique global sequence of numbers using the AUTO-INCREMENT attribute.
- If field auto-increment is used, do not assign its value in the VALUES clause. Otherwise, a primary key conflict may occur. If a value has been assigned in the VALUES clause, you can change it using ALTER SEQUENCE.
- Do not set the auto-increment step to 1 to ensure stable performance.
   The default step is 1000.

#### **UPDATE and DELETE**

#### Common updates

- Configure a sharding field in the WHERE condition when you perform an update or delete operation.
- If the sharding field cannot be configured, reduce concurrency and control data entries involved in updates or deletion operations. You can use SELECT to search for the data that you want to update or delete, determine the data scope with DOUBLE CHECK, and finally perform an update or deletion.

#### Sharding field update

- When you update a sharding field, ensure that there is a maximum of 10,000 data entries in the target table. If there are more than 10,000 data entries, create a table with the required sharding field or update the original sharding field by dividing one update operation into multiple small equivalent operations.
- Update sharding fields during off-peak hours.

#### Association

Do not perform an update or delete operation on multiple tables at the same time.

#### Subquery and LIMIT

Do not use subqueries in an UPDATE or DELETE statement. Do not use LIMIT or ORDER BY LIMIT in UPDATE or DELETE statements.

#### CASE WHEN function

Set the maximum number of **CASE WHEN** branches depending on the DDM node class. For example, set it to 2500 for 4 vCPUs | 8 GB instances, 5000 for 8 vCPUs | 16 GB instances, and 20000 for 32 vCPUs | 64 GB instances.

#### **SELECT**

#### • ORDER BY LIMIT function

- When you use statement ORDER BY LIMIT, count or ORDER BY OFFSET, count, do not assign a large value for offset.
- If error Temp table limit exceeded is returned, a temporary table containing intermediate data is generated and its data entries exceed the upper limit. Contact DDM technical support for SQL tuning.

#### • GROUP BY function

- Configure only field group by in statement SELECT\_LIST.
- Do not use clause ORDER BY in aggregate function group\_concat that cannot be pushed.
- Ensure that there are no more than three DISTINCT or GROUP BY fields.
- Do not perform a GROUP BY operation after a JOIN or subquery operation.
- Do not use COUNT(DISTINCT) or SUM(DISTINCT).
- If error Temp table limit exceeded is returned, a temporary table containing intermediate data is generated during aggregation and its data entries exceed the upper limit. Contact DDM technical support for SQL tuning.
- In a GROUP BY statement, if the query results of all groups or a single group exceed 1 million (default value), the error **Temp table limit** exceeded is returned.

#### JOIN function

- In a SELECT statement, specify a sharding field for each table or a broadcast table in the JOIN condition. Alternatively, use INNER/LEFT JOIN or RIGHT JOIN and ensure that the driving table is the smaller one.
- Do not join two large tables directly.
- Do not use OUTER JOIN in the JOIN ON condition.
- If error Temp table limit exceeded is returned, a temporary table containing intermediate data is generated during the JOIN operation and its data entries exceed the upper limit. Contact DDM technical support for SQL tuning.
- Do not join more than five tables directly.
- Do not enable transactions when you perform a JOIN query.
- Do not perform a JOIN query in a transaction. Enabling transactions affects DDM's selection of the most efficient JOIN algorithm.

## **CAUTION**

The size of tables depends on the volume of data selected using the WHERE condition.

#### Subqueries

- Do not use a subquery or its JOIN condition in an OR expression.
- Do not use scalar subqueries containing LIMIT, for example, SELECT (SELECT x FROM t2 WHERE t2.id= t.id LIMIT 1),a,b FROM t.
- If subqueries and primary table are routed to the same shard, add /\*
   +db=xxx\*/ before your SQL statement to improve routing accuracy.
- Do not use JOIN clauses in subqueries.
- Do not use nested subqueries.
- Do not perform an operation by comparing ROW expression with a subquery, for example, SELECT \* FROM t WHERE (a,b,c)=(SELECT x,y,z FROM t2 WHERE ...).

Do not use more than two subqueries in SELECT\_LIST.

#### DDL

#### Execution of DDL statements

Perform DDL operations on existing tables during off-peak hours.

#### Number of shards

Before creating a sharded table, you can estimate the total volume of data to determine table shards. Do not configure more shards than required. The number of table shards is not as large as possible.

#### High-risk DDL statements

Carefully check the SQL statement when you perform a high-risk DDL operation, for example, DROP TABLE and TRUNCATE TABLE.

#### Rectification of DDL failures

If an error occurs during execution of a DDL statement, execute CHECK TABLE *name* to verify the structure of each table shard, locate the failed table shard, and rectify the fault. For example, if ALTER TABLE fails, add /\* +allow\_alter\_rerun=true/ before the statement, enable the POWER operation, and execute the statement again till an output is returned, indicating structures of all tables are consistent.

#### DDL execution error caused by MDL locks

- Background: Before a DDL statement is executed, DDM checks whether there are MDL locks held for tables in the associated RDS database to ensure DDL availability. If there are MDL locks, DDL reports an error and exits.
  - metadata lock exists, one of MDL is [%s],DDL operation can not proceed, please use 'show metadata lock' to check current mdl, and use 'kill physical threadId@host:port' to clean it
- Possible issues: If there are slow SQL statements that have been executed for several minutes, the possible cause may be MDL locks. In this case, DDL statements cannot be executed.
- **Solution 1**: Set a larger value for **ddl\_precheck\_mdl\_threshold\_time** on the DDM console, for example, set it to **30** minutes (1800 seconds).

ddl_precheck_mdl_threshold_time indicates the maximum duration in seconds
for which a DDL statement holds an MDL lock. DDL reports an error only when
the lock duration exceeds the threshold. The default value is 120 seconds

Solution 2: Execute SHOW METADATA LOCK to check whether DDL execution is blocked by the MDL lock held for a slow transaction. If yes, execute kill physical threadId@host:port to disable the underlying slow transaction. Then use hint /\*+allow\_alter\_rerun=true\*/ and CHECK TABLE to complete execution of DDL statements.

$\sim$	$\overline{}$		_	_	_
		NI		П	_
		IV	u		ᆫ

□ NOTE

**threadId** indicates the thread ID of the underlying RDS instance node. **host** and **port** indicate the IP address and port of the RDS instance node.

#### Long execution time of DDL statements

If DDL execution is suspended for a long time during off-peak hours, enable another session and execute XA RECOVER to check whether there are slow transactions. If yes, contact technical support.

# Migrating an Entire RDS Database to DDM

#### Scenario

This section describes how to migrate an entire RDS instance (the old RDS instance) to an unsharded schema of a DDM instance to split read and write requests in DDM.

#### 

- Services may be interrupted during migration. The duration of the interruption depends on the amount of data to be migrated and on network conditions.
- Data migration is complicated and is recommended during off-peak hours. This guide is for reference only. The actual migration scheme you should use depends on your specific service scenarios, the amount of data to be migrated, and how much downtime can be tolerated.
- If a large amount of data is involved, contact DDM technical engineers by submitting a service ticket or through after-sales services. Fully rehearse the migration before migrating data.

## **Preparations before Migration**

- Prepare an ECS that can access the old RDS instance, the target DDM instance, and the RDS instance associated with the target DDM instance.
  - Ensure that the old RDS instance, the target DDM instance, and the RDS instance associated with the target DDM instance are in the same VPC for network connectivity.
  - b. Configure the same security group for the above-mentioned instances. If they belong to different security groups, enable the required ports.
  - c. Install an official MySQL client on the ECS. Version 5.6 or 5.7 is recommended.
    - Red Hat Enterprise Linux: yum install mysql mysql-devel
    - Debian Linux: apt install mysql-client-5.7 mysql-client-core-5.7
  - d. Ensure that there is enough disk space and memory on the ECS to store and compare dump files.

- Prepare a DDM instance that has been associated with an RDS instance, and configure the required DDM account and DDM schema for it.
- If the target schema is sharded, create a logical table with the same structure as data tables in the old RDS instance on the DDM console.

#### **Constraints**

- To ensure data integrity, stop services in the old RDS instance before data migration.
- In this scenario, associating the old RDS instance with DDM does not mean data association. You need to export data from the old RDS instance and then import it into the DDM instance.
- The version of the RDS instance associated with the target DDM instance must be consistent with the MySQL version of the old RDS instance.

#### **Exporting Data from the Old RDS Instance**

- **Step 1** Log in to the target ECS.
- **Step 2** Run the following command to export structure data. Set parameters in italics to actual values. For details about the parameters, see **Table 7-1**.

mysqldump -h  $\{DB\_ADDRESS\}$  -P  $\{DB\_PORT\}$  -u  $\{DB\_USER\}$  -p --skip-lock-tables --add-locks=false --set-gtid-purged=OFF --no-data  $\{DB\_NAME\}$  >  $\{mysql\_schema.sql\}$ 

Table 7	7-1	Parameter	description
---------	-----	-----------	-------------

Parameter	Description	Remarks
DB_ADDRESS	Connection address of the database whose data is to be exported	Mandatory
DB_PORT	Listening port of the database	Mandatory
DB_USER	Database user	Mandatory
DB_NAME	Database name	Mandatory
mysql_schema.sql	Name of the table structure file	The file name varies depending on the table whose structure is exported.
		You are advised to name the file in the format of <i>schema name_schema</i> to prevent data from being overwritten, for example, <i>mysql_schema.sql</i> .
mysql_data.sql	Name of the database data file	N/A

**Step 3** Run the following command to export data of the entire database. Set the parameters in italics. For details, see **Table 7-1**.

mysqldump -h {DB\_ADDRESS} -P {DB\_PORT} -u {DB\_USER} -p --hex-blob --complete-insert --skip-lock-tables --skip-tz-utc --skip-add-locks --set-gtid-purged=OFF --no-create-info {DB\_NAME} > {mysql\_data.sql}

**Step 4** Run the following command on the ECS to view the SQL files exported in **Step 2** and **Step 3**.

ls -l

----End

#### **Importing Data into DDM**

**Step 1** Run the following command on the ECS to import the structure file into DDM:

mysql -f -h {DDM\_ADDRESS} -P {DDM\_PORT} -u {DDM\_USER} -p {DB\_NAME} < {mysql\_schema.sql} Enter password: \*\*\*\*\*\*\*\*\*\*

Table 7-2 Parameter description

Parameter	Description	Remarks	
DDM_ADDRESS	Connection address of the DDM instance into which data is to be imported	View the floating IP address and port number on the <b>Basic Information</b> tab on the DDM console.	
DDM_PORT	Listening port of the DDM instance into which data is to be imported		
DDM_USER	User accessing the DDM instance	Account for creating a schema. The account must have both read and write permissions.	
DB_NAME	Name of the schema into which data is to be imported	N/A	
mysql_schema.sql	Name of the structure file to be imported	Name of the file exported in Step 2	
mysql_data.sql	Name of the entire DB data file to be imported	Name of the file exported in Step 3	

**Step 2** Run the following command on the ECS to import the data file into DDM:

mysql -f -h {DDM\_ADDRESS} -P {DDM\_PORT} -u {DDM\_USER} -p {DB\_NAME} < {mysql\_data.sql} Enter password: \*\*\*\*\*\*\*\*\*

----End

# 8 Migrating an Entire MyCat Database to DDM

#### Scenario

This section describes how to migrate an entire MyCat database to DDM.

#### **◯** NOTE

- Services may be interrupted during migration. The duration of the interruption depends on the amount of data to be migrated and on network conditions.
- Data migration is complicated and is recommended during off-peak hours. This guide is
  for reference only. The actual migration scheme you should use depends on your specific
  service scenarios, the amount of data to be migrated, and how much downtime can be
  tolerated.
- You can only access DDM through an ECS. Therefore, you must export databases as files to the ECS, and then import data in the files into DDM from the ECS.

#### **Preparations for Migration**

- Prepare an ECS that can access the MyCat database, the target DDM instance, and the RDS instance associated with the target DDM instance.
  - Ensure that the MyCat database, the target DDM instance, and the associated RDS instance are in the same VPC for stable network connectivity.
  - b. Configure the same security group for the ECS where the MyCat database is deployed, the target DDM instance, and the associated RDS instance. If they belong to different security groups, configure their security group rules to allow them to access each other.
  - c. Install an official MySQL client on the ECS. Version 5.6 or 5.7 is recommended.
    - Red Hat Linux: yum install mysql mysql-devel
    - Debian Linux: apt install mysql-client-5.7 mysql-client-core-5.7
  - d. Ensure that there is enough disk space and memory on the ECS to store and compare dump files.
- Prepare a DDM instance that has been associated with an RDS instance, and configure the required DDM account and DDM schema for it.

Use MyCat 1.6 as an example in this section.

### **Migration Scheme**

Table types in MyCat and DDM are different from each other and migration schemes vary by table type, as described in **Table 8-1**.

**Table 8-1** Migration schemes

MyCat Table Type	DDM Table Type	Migration Scheme
Unsharded	Unsharded	Export structure data and table data from MyCat.
		2. Connect to the RDS instance associated with the target DDM instance and import data from unsharded tables to the target DDM instance.
Sharded: hash sharding (including by date)	Sharded: hash sharding (including by date function)	Export all table     structure data from     the MyCat database.
Sharded: range sharding (including by date)	Sharded: range sharding (including by date function)	2. Create a table with the same structure as the exported table on the DDM console.
Broadcast	Broadcast	3. Export data from the entire MyCat database to DDM.
		4. Connect to DDM to import all database data.

#### **Constraints**

- MyCat services need to be stopped before data migration to ensure data integrity.
- Associating the MyCat database with a DDM instance for data association is not supported. You must export data from the MyCat database and then import the data into the DDM instance.
- The version of the associated RDS instance associated must be consistent with that of the MyCat database.

#### **Exporting Table Structure Data from the MyCat Database**

**Step 1** Log in to the target ECS.

Step 2 Run the following command to export table structure data in the MyCat database.
 Configure the parameters in italics. For details about the parameters, see Table
 8-2.

Table 8-2 Parameter description

Parameter	Description	Remarks	
DB_ADDRESS	Connection address of the database whose data is to be exported	Mandatory	
DB_PORT	Listening port of the database	Mandatory	
DB_USER	Database user	Mandatory	
DB_NAME	Database name	Mandatory	
mysql_schema.sql	Name of the table structure file	The file varies depending on the table whose structure is exported.  You are advised to name the file in the format of <i>schema name_schema</i> to prevent data from being overwritten, for example, <b>mysql_schema.sql</b> .	
mysql_data.sql	Name of the database data file	N/A	

----End

## **Exporting Data from the Entire MyCat Database**

- **Step 1** Log in to the ECS.
- **Step 2** Run the following command to export data of the MyCat database. Configure the parameters in italics. For details about the parameters, see **Table 8-2**.

**Step 3** Run the following command on the ECS to view the exported SQL file. Is -I

----End

#### Importing Data from Unsharded Tables into the Target DDM Instance

On the ECS, use a MySQL client to connect to the associated RDS instance. Run the following commands to import the table structure and database files.

For unsharded or common tables:

- RDS\_ADDRESS indicates the address of an RDS instance into which data is to be imported.
- RDS\_PORT indicates the port number of an RDS instance.
- RDS\_USER indicates the username of an RDS instance.
- DB\_NAME indicates the name of an RDS database. If data of unsharded tables is to be imported, DB\_NAME indicates the name of the first shard of RDS.
- mysql\_table\_schema.sql indicates the name of a table structure file to be imported.
- mysql\_table\_data.sql indicates the name of the table data file to be imported.

# Importing Data from Sharded or Broadcast Tables into the Target DDM Instance

On the ECS, use a MySQL client to connect to DDM and run the following command to import all database data into DDM.

mysql -f -h {DDM\_ADDRESS} -P {DDM\_PORT} -u {DDM\_USER} -p {DB\_NAME} < {mysql\_data.sql}

Enter password: \*\*\*\*\*\*\*

**Table 8-3** Parameter description

Parameter	Description	Remarks	
DDM_ADDRESS	Connection address of the DDM instance into which data is to be imported	View the connection address and port number on the <b>Basic Information</b> tab on the DDM console.	
DDM_PORT	Listening port of the DDM instance into which the data will be imported		
DDM_USER	User accessing the DDM instance	Account for creating a schema. The account must have both read and write permissions.	

Parameter	Description	Remarks
DB_NAME	Name of the schema that the data will be imported into	N/A
mysql_data.sql	Name of the entire database file to be imported	Name of the file exported in Step 2

# 9 Accessing DDM Using a JDBC Connection Pool

#### Scenario

When a connection pool is used, the system stores database connections in memory during initialization. Then, when you send a request to access a database, the system directly selects an available connection in the connection pool. If the connection is not required, the system returns the connection to the pool for another request to access. Connection establishment and disconnection are both managed by the connection pool itself. In addition, you can set connection pool parameters to control the number of initial connections in the connection pool, upper and lower limits of connections, maximum number of times that each connection can be used, and maximum idle time for each connection. If you prefer, you can also use your own system to monitor the number and usage of database connections.

This section describes how to access DDM using the JDBC connection pool to perform data operations. For Java programs, **HikariCP** is recommended.

- Java 8: Version 3.3.1 is recommended.
- Java 7: Version 2.4.13 is recommended.
- Parameter useCursorFetch cannot be enabled when you connect to a DDM instance using a JDBC connection.

#### **Procedure**

#### Step 1 Configure Maven.

Java 8
 <dependency>
 <groupld>com.zaxxer</groupld>
 <artifactId>HikariCP</artifactId>
 <version>3.3.1</version>
 </dependency>

#### **Step 2** Create a table.

**Table 9-1** Parameters for creating a table

Table Name	Field	Туре	Primary Key
account	account_number	bigint	Yes
	account_type	varchar (45)	No
	account_name	varchar (50)	No

#### **Step 3** Connect to a DDM instance.

- Set the number of connections by configuring parameters in the JDBC URL and HikariCP parameters.
- 2. Insert a data record.

```
Example:
```

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.sql.DataSource;
import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;
// There will be security risks if the username and password used for authentication are directly written into
code. Store the username and password in ciphertext in the configuration file or environment variables.
// In this example, the username and password are stored in the environment variables. Before running this
example, set environment variables EXAMPLE_USERNAME_ENV and EXAMPLE_PASSWORD_ENV as needed.
public class HikariCPDemo {
  private static DataSource datasource;
  private static DataSource getDataSource() {
     if (datasource == null) {
       HikariConfig config = new HikariConfig();
       // Configure parameters in the JDBC URL:
       config.setJdbcUrl("jdbc:mysql:loadbalance://192.168.0.10:5066,192.168.0.11:5066/db name?
loadBalanceAutoCommitStatementThreshold=5&loadBalanceHostRemovalGracePeriod=15000&loadBal
anceBlacklistTimeout=60000&loadBalancePingTimeout=5000&retriesAllDown=10&connectTimeout=10
000");
       // Or configure parameters in the JDBC URL as follows:
       config.addDataSourceProperty("loadBalanceAutoCommitStatementThreshold",5);
       config.addDataSourceProperty("loadBalanceHostRemovalGracePeriod", 15000);
       config.addDataSourceProperty("loadBalanceBlacklistTimeout", 60000);
       config.addDataSourceProperty("loadBalancePingTimeout", 5000); config.addDataSourceProperty("retriesAllDown", 10);
       config.addDataSourceProperty("connectTimeout", 10000);
       config.setUsername("EXAMPLE_USERNAME_ENV");
       config.setPassword("EXAMPLE_PASSWORD_ENV");
       config.setMaximumPoolSize(10);
       config.setAutoCommit(true);
       // Configure HikariCP parameters.
       config.addDataSourceProperty("cachePrepStmts", true);
       config.addDataSourceProperty("prepStmtCacheSize", 250);
       config. add Data Source Property ("prepStmtCacheSqlLimit", 2048);\\
       config.addDataSourceProperty("minimumIdle", 5);
       config.addDataSourceProperty("maximumPoolSize", 10);
```

```
config.addDataSourceProperty("idleTimeout", 30000);
        datasource = new HikariDataSource(config);
     return datasource;
  }
  public static void main(String[] args) {
     Connection connection = null;
     PreparedStatement pstmt = null;
     ResultSet resultSet = null;
        DataSource dataSource = getDataSource();
        connection = dataSource.getConnection();
        System.out.println("The Connection Object is of Class: " + connection.getClass());
        // Start a transaction.
        connection.setAutoCommit(false);
        // Insert test data.
        String insertSql = "insert into account(account_number, account_type, account_name)
values(?, ?, ?);";
        PreparedStatement insertStmt = connection.prepareStatement(insertSql);
        insertStmt.setLong (1, 1L);
        insertStmt.setString (2, "manager");
        insertStmt.setString (3, "demotest01");
        insertStmt.executeUpdate();
        connection.commit ();
        // Query data.
        pstmt = connection.prepareStatement("SELECT * FROM account");
        resultSet = pstmt.executeQuery();
        while (resultSet.next()) {
           String accountNumber = resultSet.getString("account_number");
           String accountType = resultSet.getString("account_type");
           String accountName = resultSet.getString("account_name");
System.out.println(accountNumber + "," + accountType + "," + accountName);
        }
     } catch (Exception e) {
        try {
           if (null != connection) {
              connection.rollback();
        } catch (SQLException e1) {
           e1.printStackTrace();
        e.printStackTrace();
```

----End

# 10 Logging In to a DDM Instance Using Navicat

#### Scenario

This section describes how to obtain an EIP and use this EIP to connect to a DDM instance through a Navicat client.

#### **Using Navicat to Connect to a DDM Instance**

- **Step 1** Log in to the DDM console, locate the target DDM instance, and click its name.
- **Step 2** In the **Instance Information** area, click **Bind**, select an EIP, and bind it with the DDM instance.
- **Step 3** In the left pane, click the VPC icon and choose **Access Control** > **Security Groups**.
- **Step 4** On the **Security Group** page, locate the target security group and click **Manage Rule** in the **Operation** column. On the displayed page, click **Add Rule**. Configure the security group rule and click **OK**.

#### NOTE

- After binding an EIP to your DDM instance, set strict inbound and outbound rules for the security group to enhance database security.
- **Step 5** Open Navicat and click **Connection**. In the displayed dialog box, enter the host IP address (EIP), username, and password (DDM account and password).
- **Step 6** Click **Test Connection**. If a message is returned indicating that the connection is successful, click **OK**. The connection will be succeeded 1 to 2 minutes later. If the connection fails, the failure cause is displayed. Modify the required information and try again.

## ----End

#### **◯** NOTE

Using Navicat to connect to a DDM instance is similar to using other visualized MySQL tools such as MySQL Workbench. The procedure of using other visualized MySQL tools to connect to a DDM instance is omitted here.

# 1 1 Migrating Data from RDS for MySQL to DDM Using DRS

### 11.1 Overview

#### **Scenarios**

This practice describes how to migrate data from RDS for MySQL to DDM in a different region using Data Replication Service (DRS), including how to create an RDS for MySQL instance and a DDM instance on Huawei Cloud, and how to migrate data over the VPN network.

#### **Prerequisites**

- You have a Huawei Cloud account.
- Your account balance is not below zero.

#### **Service List**

- Virtual Private Cloud (VPC)
- Virtual Private Network (VPN)
- RDS
- Distributed Database Middleware (DDM)
- Data Replication Service (DRS)
- Data Admin Service (DAS)

### **Deployment Architecture**

In this example, the source is a Huawei Cloud RDS for MySQL instance and the destination is a DDM instance in a different region. Data is migrated from the source to the destination over a VPN. **Figure 11-1** shows the overall deployment architecture.

Region-A

Region-B

VPC

Subnet

Security group

All the inbound traffic

VPN

DRS

DDM

Figure 11-1 VPN-based deployment architecture

#### **Constraints**

- Before data migration, you need to stop your workloads to ensure data integrity.
- The new RDS for MySQL instance must be of the same version as the source RDS for MySQL instance.

#### **Notes on Usage**

- The resource planning is for demonstration only. Adjust it as needed.
- The end-to-end test data is for reference only.

# 11.2 Resource and Cost Planning

Table 11-1 Resource planning

Categor y	Subcategor y	Plan	Remarks	
Source VPC	VPC name	vpc-DRSsrc	Specify a name that is easy to identify.	
	Region	AP-Singapore	For low network latency and quick resource access, select the region nearest to you.	
	AZ	AZ2	-	
	Subnet CIDR	10.0.0.0/24	Select a subnet with sufficient network resources.	
	Subnet name	subnet-drs01	Specify a name that is easy to identify.	
Source RDS instance	RDS instance name	rds-mysql-src	Specify a name that is easy to identify.	

Categor y	Subcategor y	Plan	Remarks	
	Region	AP-Singapore	For low network latency and quick resource access, select the region nearest to you.	
	DB engine version	MySQL 5.7	-	
	Instance type	Single	A single instance is used in this example.	
			To improve service reliability, select a primary/standby instance.	
	Storage type	Cloud SSD	-	
	AZ	AZ2	A single instance is used in this example.	
			To improve service reliability, create a primary/standby RDS instance and then deploy them in different AZs.	
	Instance class	General-purpose 4 vCPUs   8 GB	-	
Source VPN	Gateway name	vpngw-src	Specify a name that is easy to identify.	
	Region	AP-Singapore	For low network latency and quick resource access, select the region nearest to you.	
	VPC	vpc-DRSsrc	The VPC must be the same as the VPC of the source RDS instance.	
	VPN connection name	vpn-src01	Specify a name that is easy to identify.	
	Local subnet	subnet-drs01	The local subnet is the VPC subnet of the source RDS instance.	
	Remote gateway	123.60.251.207	Peer VPN gateway. The value is the gateway address of the destination VPN. After the destination VPN is created, the gateway address is obtained.	

Categor y	Subcategor y	Plan	Remarks	
	Remote subnet	172.16.0.0/24	Subnet of the peer VPN. The value is the subnet of the destination VPN, which is the subnet of the VPC where the destination DDM instance is deployed.	
Destinati on VPC	VPC name	vpc-DRStar	Specify a name that is easy to identify.	
	Region	CN-Hong Kong	For low network latency and quick resource access, select the region nearest to you.	
	AZ	AZ1	-	
	Subnet CIDR	172.16.0.0/24	Select a subnet with sufficient network resources.	
	Subnet name	subnet-drs02	Specify a name that is easy to identify.	
Destinati on DDM instance	DDM instance name	ddm-drs-tar	Specify a name that is easy to identify.	
	AZ	AZ1	You can select one or more AZs. Ensure that the destination DDM instance is deployed across different AZs to improve service reliability.	
	Node class	General-enhanced 8 vCPUs   16 GB	-	
	Nodes	1	A single node cannot provide high availability. Ensure that the destination DDM instance has at least two nodes.	
RDS instance associat ed with the DDM instance	RDS instance name	rds-ddm01	Specify a name that is easy to identify.	
	Region	CN-Hong Kong	For low network latency and quick resource access, select the region nearest to you.	
	DB engine version	MySQL 5.7	-	

Categor y	Subcategor y	Plan	Remarks	
	Instance type	Single	A single instance is used in this example.  To improve service reliability, select a primary/standby RDS instance.	
	Storage type	Cloud SSD	-	
	AZ	AZ1	A single instance is used in this example.  To improve service reliability, create a primary/standby RDS instance and then deploy them in different AZs.	
	Instance class	General-purpose 4 vCPUs   8 GB	-	
Destinati on VPN	Gateway name	vpngw-tar	Specify a name that is easy to identify.	
	Region	CN-Hong Kong	For low network latency and quick resource access, select the region nearest to you.	
	VPC	vpc-DRStar	The value is the VPC of the destination DDM instance.	
	VPN connection name	vpn-tar01	Specify a name that is easy to identify.	
	Local subnet	subnet-drs02	The value is the VPC subnet of the destination DDM instance.	
	Remote gateway	123.60.236.84	Peer VPN gateway. In this example, this parameter is set to the gateway address of the source VPN. After the source VPN is created, the gateway address is obtained.	
	Remote subnet	10.0.0.0/24	Subnet of the peer VPN. The value is the subnet of the source VPN, which is the subnet of the VPC where the source RDS for MySQL instance is deployed.	
DRS migratio n task	Task name	DRS-MySQLToDDM	Specify a name that is easy to identify.	

Categor y	Subcategor y	Plan	Remarks
	Source DB engine	MySQL	In this example, the source is an RDS for MySQL instance on Huawei Cloud.
	Destination DB engine	DDM	In this example, the destination is a DDM instance.
	Network type	VPN network	In this example, VPNs are used.

# **11.3 Operation Process**

Figure 11-2 shows the process of creating an RDS for MySQL instance and synchronizing data from the RDS instance to DDM.

Using DRS

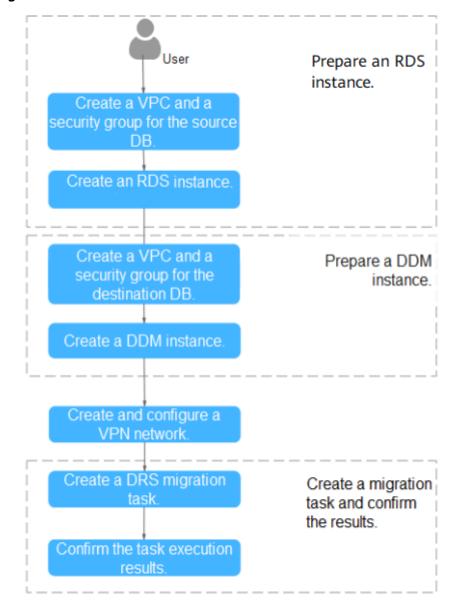


Figure 11-2 Flowchart

# 11.4 Preparing for the Source RDS for MySQL Instance

# 11.4.1 Creating a VPC and Security Group

Create a VPC and security group to prepare for creating an RDS for MySQL instance as the source.

#### Creating a VPC

- **Step 1** Log in to the management console.
- **Step 2** Click in the upper left corner and select AP-Singapore.

**Step 3** Click the service list icon on the left and choose **Networking** > **Virtual Private Cloud**.

The VPC console is displayed.

- Step 4 Click Create VPC. For details, see Creating a VPC.
- **Step 5** Configure parameters as needed and click **Create Now**.
- **Step 6** Return to the VPC list and check whether the VPC is created.

When its status becomes available, the VPC is created.

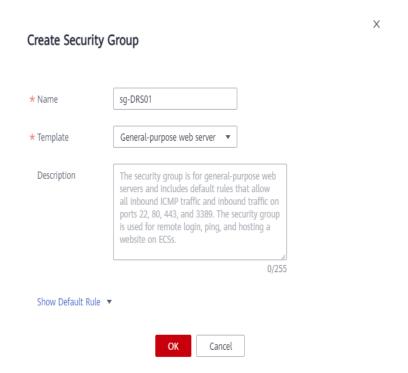
----End

#### **Creating a Security Group**

- **Step 1** Log in to the management console.
- **Step 2** Click on the upper left corner and select AP-Singapore.
- **Step 3** Click the service list icon on the left and choose **Networking** > **Virtual Private Cloud**.

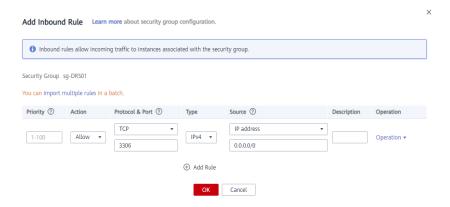
The VPC console is displayed.

- **Step 4** In the navigation pane on the left, choose **Access Control** > **Security Groups**.
- Step 5 Click Create Security Group.
- **Step 6** Specify a security group name and other information.



- Step 7 Click OK.
- **Step 8** Return to the security group list and click the security group name (**sg-DRS01** in this example).

- Step 9 Click the Inbound Rules tab, and then click Add Rule.
- Step 10 Configure an inbound rule to allow access from database port 3306.



----End

# 11.4.2 Creating an RDS for MySQL Instance

Create an RDS for MySQL instance and test data.

#### Creating an RDS for MySQL Instance

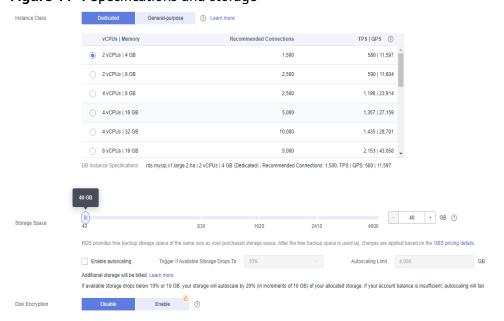
- **Step 1** Log in to the management console.
- **Step 2** Click on the upper left corner and select AP-Singapore.
- Step 3 Click the service list icon on the left and choose Databases > Relational Database Service.
- Step 4 Click Buy DB Instance.
- **Step 5** Configure the instance name and basic information.

Yearly/Monthly Region Regions are geographic areas isolated from each other. For low network latency and guick resource access, select the nearest region DB Instance Name rds-2821 ? If you buy multiple DB instances at a time, they will be named with four digits appended in the format "DB instance name-SN". For example and so on. DB Engine Version (?) DB Instance Type Single Storage Type Cloud SSD Primary AZ az3 Standby AZ Time Zone (UTC+08:00) Beijing, Chongqing, Hong... 🗸

Figure 11-3 Billing mode and basic information

Step 6 Select an instance class.

Figure 11-4 Specifications and storage

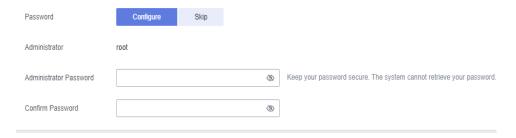


**Step 7** Select a VPC and security group and configure the database port.

The VPC and security group have been created in **Creating a VPC and Security Group**.



**Step 8** Set the instance password.



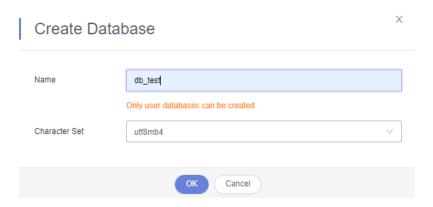
- Step 9 Click Next and complete the whole process.
- **Step 10** Return to the instance list.

If the instance status becomes **Available**, the instance is created.

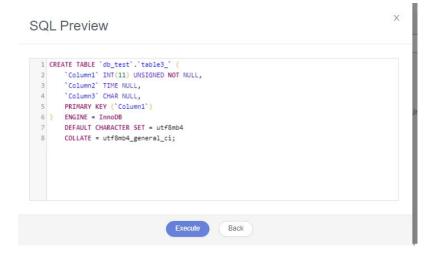
----End

#### **Creating Test Data**

- **Step 1** Log in to the management console.
- **Step 2** Click in the upper left corner and select AP-Singapore.
- Step 3 Click the service list icon on the left and choose Databases > Relational Database Service.
- **Step 4** Locate the created RDS instance and choose **More** > **Log In**.
- **Step 5** In the displayed dialog box, enter the username and password of the instance and click **Test Connection**.
- **Step 6** After the connection is successful, click **Log In**.
- **Step 7** Click **Create Database** to create the **db\_test** database.



#### **Step 8** Run the following SQL statement in **db\_test** to create table **table3**:



#### **Step 9** Run the following statements in table **table3**\_ to insert three lines of data:

```
INSERT INTO `db_test`.`table3_` (`Column1`,`Column2`,`Column3`) VALUES(1,'00:00:11','a');
INSERT INTO `db_test`.`table3_` (`Column1`,`Column2`,`Column3`) VALUES(2,'00:00:22','b');
INSERT INTO `db_test`.`table3_` (`Column1`,`Column2`,`Column3`) VALUES(5,'00:00:55','e');
```

```
I INSERT INTO 'db_test'.'table3_' ('Column1','Column2','Column3') VALUES(1,'00:00:11','a')

2 INSERT INTO 'db_test'.'table3_' ('Column1','Column2','Column3') VALUES(2,'00:00:22','b')

3 INSERT INTO 'db_test'.'table3_' ('Column1','Column2','Column3') VALUES(5,'00:00:55','e')
```

----End

# 11.4.3 Creating a VPN for the Source

- **Step 1** Log in to the management console.
- **Step 2** Click on the upper left corner and select AP-Singapore.
- **Step 3** Click the service list icon on the left and choose **Networking** > **Virtual Private Network**.

- Step 4 In the navigation pane on the left, choose Virtual Private Network > Enterprise VPN Gateways.
- **Step 5** On the **VPN Gateways** page, click **Buy VPN Gateway**. For details, see **Creating a VPN Gateway**.

#### **CAUTION**

- The connected VPN gateway must have the same VPC as the source RDS for MySQL instance, that is, the VPC created in Creating a VPC and Security Group.
- The local subnet must be the same as the VPC subnet where the source RDS for MySQL instance is located, that is, the subnet created in Creating a VPC and Security Group.
- **Step 6** Click **Next**, confirm the information, and click **Submit**.
- **Step 7** After the VPN gateway is created, view its information in the VPN gateway list. Its status is **Not connected**. If this VPN gateway is in use by a VPC connection, the VPN gateway status changes to **Normal**.

----End

# 11.5 Preparing for the Destination DDM Instance

# 11.5.1 Creating a VPC and Security Group

Create a VPC and security group to prepare for the destination DDM instance.

## Creating a VPC

- **Step 1** Log in to the management console.
- **Step 2** Click in the upper left corner and select region CN-Hong Kong.
- **Step 3** Click the service list icon on the left and choose **Networking** > **Virtual Private Cloud**.

The VPC console is displayed.

- Step 4 Click Create VPC. For details, see Creating a VPC.
- **Step 5** Configure parameters as needed and click **Create Now**.
- **Step 6** Return to the VPC list and check whether the VPC is created.

When its status becomes available, the VPC is created.

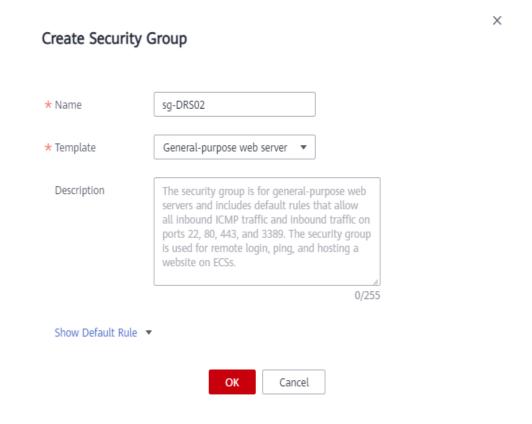
----End

#### **Creating a Security Group**

- **Step 1** Log in to the management console.
- **Step 2** Click on the upper left corner and select region CN-Hong Kong.
- **Step 3** Click the service list icon on the left and choose **Networking** > **Virtual Private Cloud**.

The VPC console is displayed.

- **Step 4** In the navigation pane, choose **Access Control** > **Security Groups**.
- Step 5 Click Create Security Group.
- **Step 6** Specify the security group name and other information, and click **OK**.



----End

# 11.5.2 Creating a DDM Instance

- **Step 1** Log in to the management console.
- **Step 2** Click on the upper left corner and select region CN-Hong Kong.
- Step 3 Click the service list icon on the left and choose Databases > Distributed Database Middleware.
- **Step 4** On the displayed page, in the upper right corner, click **Buy DDM Instance**.

- **Step 5** Specify the instance information and specifications as required. For details, see **Creating a DDM Instance**.
- **Step 6** Select a VPC and security group and configure the database port.
  - The VPC and security group have been created in **Creating a VPC and Security Group**.
- **Step 7** After the configuration is complete, click **Next** at the bottom of the page.
- **Step 8** Go to the **Instances** page to view and manage the instance.

The default database port is **5066** and cab be changed after the instance is created. If the status of the instance is **Running**, the instance has been created.

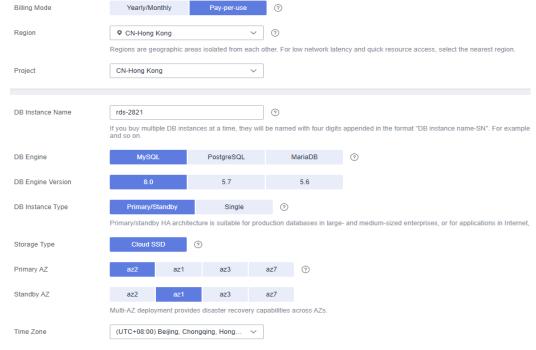
----End

# 11.5.3 Creating an RDS for MySQL Instance

Create an RDS for MySQL instance and associate it with the DDM instance.

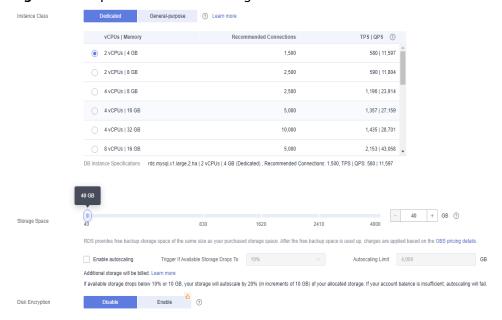
- Step 1 Log in to the management console.
- **Step 2** Click in the upper left corner and select region CN-Hong Kong.
- Step 3 Click the service list icon on the left and choose Databases > Relational Database Service.
- Step 4 Click Buy DB Instance.
- **Step 5** Configure the instance name and basic information.

Figure 11-5 Billing mode and basic information



**Step 6** Select an instance class.

Figure 11-6 Specifications and storage



**Step 7** Select a VPC and security group and configure the database port.

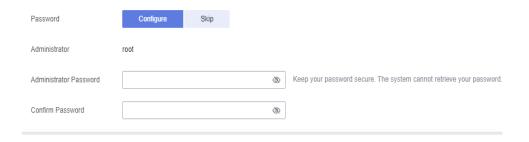
The VPC and security group have been created in **Creating a VPC and Security Group**.



The RDS for MySQL instance must be in the same VPC and subnet as the DDM instance.



**Step 8** Set the instance password.



- **Step 9** Click **Next** and complete the whole process.
- Step 10 Return to the instance list.

If the instance status is **Available**, the instance is created.

----End

# 11.5.4 Creating a Schema and Associating It with the RDS for MySQL Instance

- **Step 1** Log in to the management console.
- **Step 2** Click in the upper left corner and select region CN-Hong Kong.
- Step 3 Click the service list icon on the left and choose Databases > Distributed Database Middleware.
- **Step 4** On the **Instances** page, locate the required DDM instance and click **Create Schema** in the **Operation** column.
- **Step 5** On the displayed page, specify a sharding method, enter a schema name, set the number of shards, select the required DDM accounts, and click **Next**.

In this example, the schema is unsharded, and the schema name is **db\_test**.



Currently, only data can be migrated from the RDS for MySQL to DDM. To migrate table structures and other objects, you need to create schemas in the destination DDM instance based on table structures of the source RDS for MySQL instance.

Figure 11-7 Selecting an RDS for MySQL instance

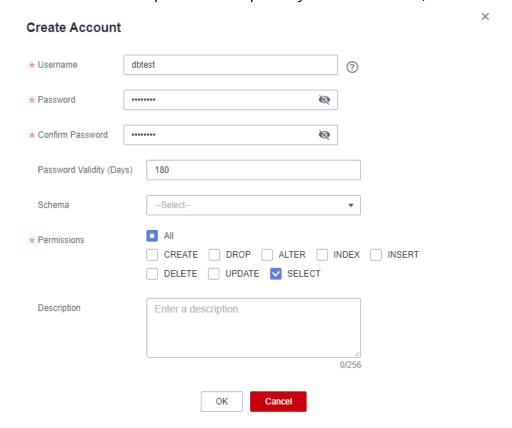


----End

# 11.5.5 Creating a DDM Account

- **Step 1** Log in to the management console.
- **Step 2** Click in the upper left corner and select region CN-Hong Kong.

- Step 3 Click the service list icon on the left and choose Databases > Distributed Database Middleware.
- **Step 4** On the **Instances** page, locate the required DDM instance and click its name.
- **Step 5** In the navigation pane, choose **Accounts**.
- **Step 6** On the displayed page, click **Create Account** and configure parameters as needed. For details about the permissions required by the DDM account, see **Precautions**



Step 7 Click OK.

----End

# 11.5.6 Creating Table Structures in the Destination Database

Currently, only data can be migrated from RDS for MySQL instances to DDM instances. To migrate table structures and other objects, you need to create table structures and indexes in the destination database based on table structures of the source schema. Any source objects that have no corresponding objects created in the destination cannot be migrated. For more constraints, see **Before You Start**.

- Step 1 Log in to the management console.
- **Step 2** Click oin the upper left corner and select region CN-Hong Kong.
- Step 3 Click the service list icon on the left and choose Databases > Distributed Database Middleware.

- **Step 4** On the **Instances** page, locate the required instance and click **Log In** in the **Operation** column.
- **Step 5** In the displayed dialog box, enter the username and password created in **Creating a DDM Account**, and click **Test Connection**.
- **Step 6** After the connection is successful, click **Log In** to log in to the DDM instance.
- Step 7 Click the db\_test schema created in Creating a Schema and Associating It with the RDS for MySQL Instance.
- **Step 8** Run the following SQL statements in database **db\_test** to create table **table3**\_ with the same structure as the source:



----End

# 11.5.7 Creating a VPN for the Destination

- **Step 1** Log in to the management console.
- **Step 2** Click oin the upper left corner and select region CN-Hong Kong.
- **Step 3** Click the service list icon on the left and choose **Networking** > **Virtual Private Network**.
- Step 4 In the navigation pane on the left, choose Virtual Private Network > Enterprise Customer Gateways.
- **Step 5** On the **VPN Gateways** page, click **Create Customer Gateway**. For details, see **Creating a Customer Gateway**.

# **CAUTION**

The IP address must be the same as the VPC subnet where the destination DDM instance is located, that is, the subnet created in **Creating a VPC and Security Group**.

**Step 6** Configure parameters as needed and click **Create Now**.

----End

# 11.6 Creating a VPN Connection

- **Step 1** Log in to the management console.
- **Step 2** Click on the upper left corner and select AP-Singapore.
- **Step 3** Click the service list icon on the left and choose **Networking** > **Virtual Private Network**.
- **Step 4** In the navigation pane on the left, choose **Virtual Private Network > Enterprise VPN Connections**.
- **Step 5** On the **VPN Connections** page, click **Buy VPN Connection**. For details, see **Creating a VPN Connection**.

# **⚠** CAUTION

The customer gateway and subnet are the gateway name and subnet of the destination VPN. Configure parameters based on the VPN created in **Creating a VPN for the Destination**.

- **Step 6** Click **Next**, confirm the information, and click **Submit**.
- **Step 7** After the configuration is complete, view the VPN gateway in the list. The VPN gateway status is **Normal**.

----End

# 11.7 Creating a DRS Migration Task

Create a DRS migration task to migrate data from RDS for MySQL to DDM in a different region.

## **Pre-Migration Check**

Before you create a migration task, check whether migration conditions are met.

This section describes how to migrate data from RDS for MySQL to DDM. For details, see **Before You Start**.

#### **Creating a Migration Task**

- **Step 1** Log in to the management console.
- **Step 2** Click on the upper left corner and select region CN-Hong Kong.
- Step 3 Click the service list icon on the left and choose Databases > Data Replication Service.
- **Step 4** On the displayed page, click **Create Migration Task**.
- **Step 5** On the **Create Replication Instance** page, select a region and project, specify the task name, description, and the replication instance details, and click **Create Now**.
  - For details about parameter description, see From MySQL to DDM.
- **Step 6** On the **Configure Source and Destination Databases** page, wait until the replication instance is created. Then, specify source and destination database information and click **Test Connection** for both the source and destination databases to check whether they have been connected to the replication instance. After the connection tests are successful, select the check box before the agreement and click **Next**.
- **Step 7** On the **Set Task** page, select a migration type.

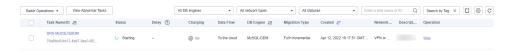
Migrate Object: Select Tables.

- **Step 8** Click **Next**. On the **Check Task** page, check the migration task.
  - If any check item fails, view the cause and rectify the fault. After the fault is rectified, click **Check Again**.
  - If all check items are successful, click Next.

#### Step 9 Click Submit.

Return to the **Online Migration Management** page and check the migration task status

It takes several minutes to complete.



If the status changes to **Full migration**, the migration task has been started.

#### □ NOTE

- For migration from MySQL to DDM, full migration and full+incremental migration are both supported.
- If you create a full migration task, the task automatically stops after full data is migrated to the destination.
- If you create a full+incremental migration task, a full migration is executed first. After the full migration is complete, an incremental migration starts.
- During the incremental migration, data is continuously migrated so the task will not automatically stop.

----End

# 11.8 Checking Migration Results

You can use either of the following methods to check the migration results:

- DRS compares migration objects and data and provides comparison results.
   For details, see Viewing Migration Results on the DRS Console.
- Log in to the destination to check whether databases, tables, and data are migrated. Confirm the data migration status. For details, see Viewing Migration Results on the DDM Console.

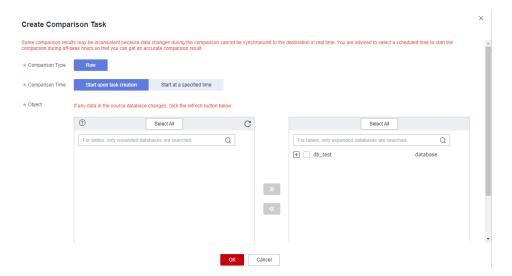
#### Viewing Migration Results on the DRS Console

- **Step 1** Log in to the management console.
- **Step 2** Click in the upper left corner and select region CN-Hong Kong.
- **Step 3** Click the service list icon on the left and choose **Databases** > **Data Replication Service**.
- **Step 4** Locate the required DRS instance and click its name.
- **Step 5** In the navigation pane on the left, choose **Migration Comparison**.
- **Step 6** Click the **Object-Level Comparison** tab and check whether some objects are missing.

Click **Compare**. After the comparison is complete, view the comparison results.



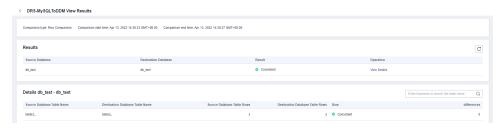
- **Step 7** Choose **Data-Level Comparison** and check whether the number of rows of migrated objects is consistent between the source and destination.
  - 1. Click Create Comparison Task.
  - 2. In the displayed dialog box, select the comparison type, time, and object.



3. After the comparison task is complete, view the data comparison results.



To view the comparison details, click View Results next to the comparison task.



----End

### Viewing Migration Results on the DDM Console

- **Step 1** Log in to the management console.
- **Step 2** Click in the upper left corner and select region CN-Hong Kong.
- Step 3 Click the service list icon on the left and choose Databases > Distributed Database Middleware.
- **Step 4** Locate the target DDM instance and click **Log In** in the **Operation** column.
- **Step 5** In the displayed dialog box, enter the password and click **Test Connection**.
- **Step 6** After the connection is successful, click **Log In**.

**Step 7** View destination database and table names and check whether all data is migrated.

----End

# 12 Sharding Database and Table Data of an RDS for MySQL Instance

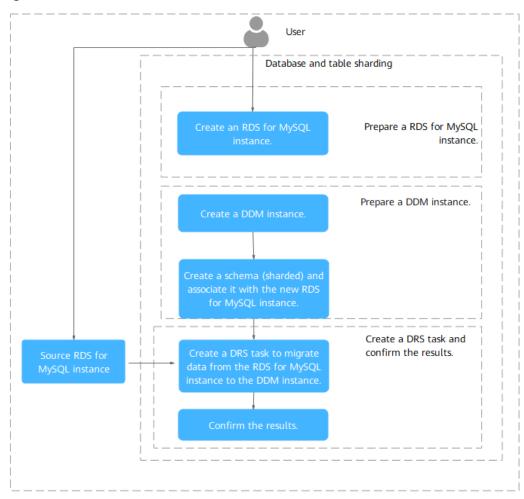
# 12.1 Overview

#### **Scenarios**

This section describes how to shard database and table data of an existing RDS for MySQL instance using DDM.

#### **Operation Process**

Figure 12-1 Flowchart



#### **Prerequisites**

- You have a Huawei Cloud account.
- There is a new RDS for MySQL instance available.
- The existing RDS for MySQL instance, destination DDM instance, and new RDS for MySQL instance are in the same VPC and have the same security group rules.

# 12.2 Preparing an RDS for MySQL Instance

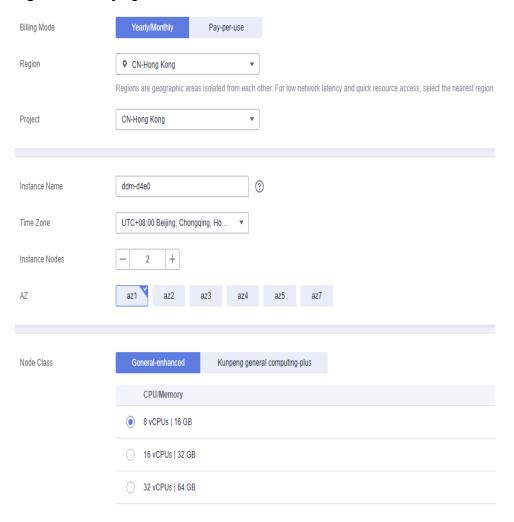
There is a new RDS for MySQL instance available. If there are no RDS for MySQL instances available, create one. For details, see **Buying an Instance**.

# 12.3 Preparing a DDM Instance

# 12.3.1 Creating a DDM Instance

- **Step 1** Log in to the management console.
- **Step 2** Click in the upper left corner and select region CN-Hong Kong.
- Step 3 Click the service list icon on the left and choose Databases > Distributed Database Middleware.
- **Step 4** On the displayed page, in the upper right corner, click **Buy DDM Instance**.
- **Step 5** Specify a node class and other information. For parameter details, see **Buying a DDM Instance**.

Figure 12-2 Buying a DDM instance



- **Step 6** After the configuration is complete, click **Next** at the bottom of the page.
- **Step 7** Confirm the configuration information and perform subsequent operations based on the billing mode you selected.
  - ----End

# 12.3.2 Creating a Schema and Associating It with the Prepared RDS for MySQL Instance

#### **Procedure**

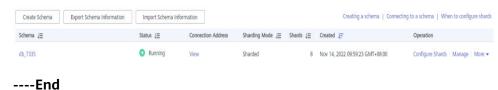
- **Step 1** Log in to the management console.
- **Step 2** Click in the upper left corner and select region CN-Hong Kong.
- Step 3 Click the service list icon on the left and choose Databases > Distributed Database Middleware.
- **Step 4** On the **Instances** page, locate the required DDM instance and click **Create Schema** in the **Operation** column.
- Step 5 Select Sharded for Sharding, specify Shards based on service requirements, select the RDS for MySQL instance created in Preparing an RDS for MySQL Instance for Data Nodes, and click Next.

Figure 12-3 Selecting an RDS for MySQL instance



- **Step 6** Enter the password of the selected RDS for MySQL instance and click **Test Availability**. After the test succeeds, click **Finish**.
- **Step 7** View the created schema in the schema list.

Figure 12-4 Schema created successfully

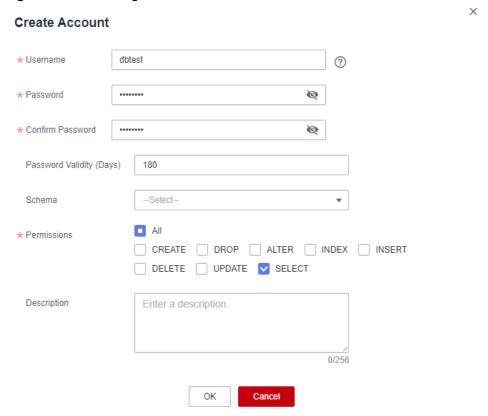


# 12.3.3 Creating a DDM Account

- **Step 1** Log in to the management console.
- **Step 2** Click on the upper left corner and select region CN-Hong Kong.

- Step 3 Click the service list icon on the left and choose Databases > Distributed Database Middleware.
- **Step 4** On the **Instances** page, locate the required DDM instance and click its name.
- **Step 5** In the navigation pane on the left, choose **Accounts**.
- Step 6 In the upper left corner, click Create Account.
- Step 7 Specify a username, password, and permissions, select the schema created in Creating a Schema and Associating It with the Prepared RDS for MySQL Instance, and click OK.

Figure 12-5 Creating a DDM account



**Step 8** View the created account in the account list.

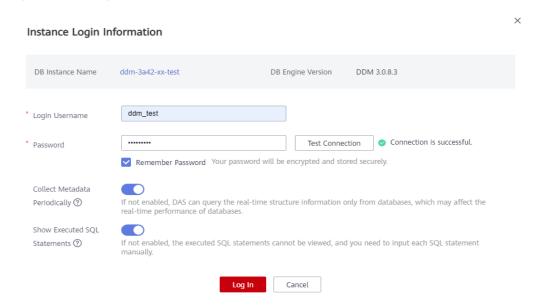
----End

# 12.3.4 Creating Table Structures in the DDM Instance

- **Step 1** Log in to the management console.
- **Step 2** Click  $^{\bigcirc}$  in the upper left corner and select region CN-Hong Kong.
- Step 3 Click the service list icon on the left and choose Databases > Distributed Database Middleware.

- **Step 4** On the **Instances** page, locate the required instance and click **Log In** in the **Operation** column.
- **Step 5** In the displayed dialog box, enter the username and password of the DDM account created in **Creating a DDM Account** and click **Log In**.

Figure 12-6 Login window



Step 6 Create the same tables as those in the existing RDS instance. You can select to create broadcast, unsharded, or sharded tables based on source table properties. For details about how to use a broadcast and unsharded table, see Using Broadcast and Unsharded Tables.

Figure 12-7 Source tables



Figure 12-8 Data in table address\_test



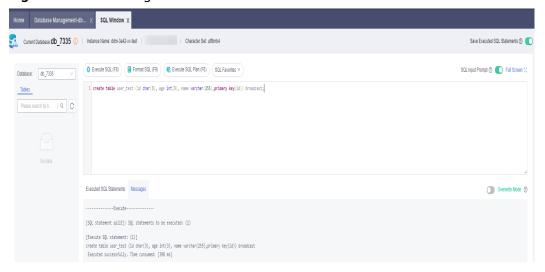
Figure 12-9 Data in table user\_test



**Step 7** Run the following command to create a broadcast table as source table **user\_test** and a sharded table as source table **address\_test**: For SQL statements for creating tables, see **Creating a Table**.

create table user\_test (id char(3), age int(3), name varchar(255),primary key(id)) broadcast;

Figure 12-10 Creating a broadcast table



create table address\_test (id char(3), name varchar(255), address varchar(255),primary key(id)) dbpartition by hash(id);

Figure 12-11 Creating a sharded table



**Step 8** View the table creation results.

Coperts

Metadata Collection

Tables

- Operation

- Operati

Figure 12-12 Viewing table creation results

----End

# 12.4 Creating a DRS Migration Task

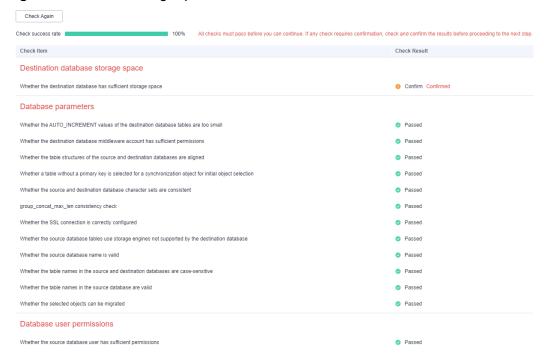
# 12.4.1 Creating a DRS Migration Task

Create a DRS migration task to migrate data from an existing RDS for MySQL instance to a DDM instance in a different region.

- **Step 1** Log in to the management console.
- **Step 2** Click on the upper left corner and select region CN-Hong Kong.
- **Step 3** Click the service list icon on the left and choose **Databases** > **Data Replication Service**.
- **Step 4** Click **Create Migration Task** to migrate data from the existing RDS for MySQL instance to the DDM instance.
- **Step 5** Configure the required parameters. For details about parameter description, see **From MySQL to DDM**.
  - Data Flow: Select To the cloud.
  - Source DB Engine: Select MySQL.
  - **Destination DB Engine**: Select **DDM**.
  - **Network Type**: Set this parameter based on service requirements.
  - Destination DB Instance: Select the DDM instance created in Creating a DDM Instance.
  - Migration Type: Select Full+Incremental or Full.
- Step 6 In the task list, locate the created migration task and click Edit in the Operation column. At the Source Database area, enter the IP address, port, username, and password of the existing RDS for MySQL instance, and click Test Connection. At the Destination Database area, enter the username and password of the created DDM account, and click Test Connection. After both of the connection tests are successful, click Next.
- **Step 7** Select the objects you want to migrate and click **Next**.

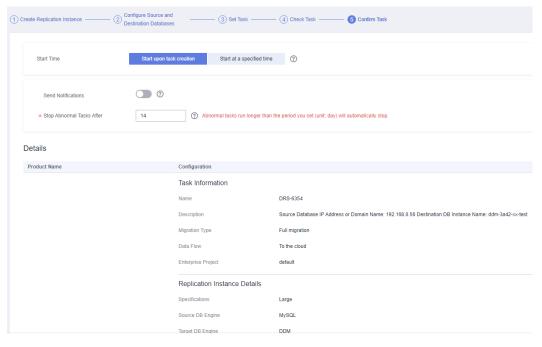
**Step 8** On the **Check Task** page, check the migration task. If any check item fails, view the cause and rectify the fault. After the fault is rectified, click **Check Again**. After all check items are passed, click **Next**.

Figure 12-13 Performing a pre-check



Step 9 Configure the start of the migration task

Figure 12-14 Start configurations



----End

# 12.4.2 Checking Migration Results

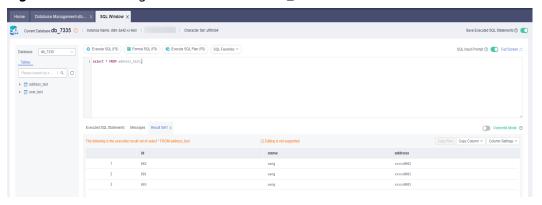
#### **Procedure**

**Step 1** After the migration task is complete, log in to the DDM instance to view migration results. Ensure that the DDM instance has the same data as the existing RDS for MySQL instance.

Figure 12-15 Viewing data in table user\_test



Figure 12-16 Viewing data in table address\_test



**Step 2** Run the following command to view data distribution of tables. The broadcast table stores the same data in each shard. Data in the sharded table is distributed based on the algorithms you specified.

show topology from <table\_name>;

Figure 12-17 Broadcast table user\_test

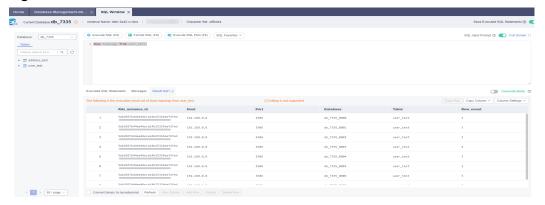
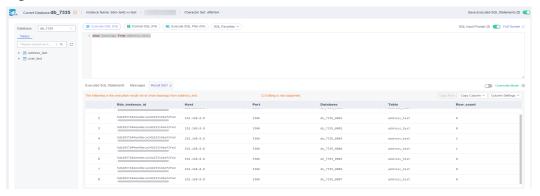


Figure 12-18 Sharded table address\_test



----End

# 13 Selecting a Sharding Key and a Sharding Algorithm

This section describes when to use and how to select a sharding key and algorithm.

For details, see Table 13-1.

Table 13-1 Sharding keys and algorithms

Sharding Algorithm	Hash		Range	
Sharding Key	Table field	Table field+date function	Table field	Table field+date function
Description	Data is evenly distribute d to shards by table field.	Data is evenly distributed to shards by table field and date function. The table field must be date, datetime, or timestamp.	Data is distributed to specific shards based on the rules defined in algorithm metadata.	Data is distributed to shards by table field and date function based on the rules defined in algorithm metadata. The table field must be date, datetime, or timestamp).

Scenario	Scenarios requiring even data distributi on, for example, banking applications where logical entities are customers. In this case, use the table field corresponding to customers (for example, customer account numbers) as the sharding key.	Scenarios requiring data to be split by time (year, month, day, week, or their combinations), for example, gaming applications. For these applications, use the table field (for instance, player registration time) corresponding to players as the sharding key. Sharding by day, month, or year helps you easily collect and query operation statistics of players for a specified day or month, and helps game vendors conduct big data analysis.	Scenarios with a large number of range operations, for instance, e-commerce applications. If a service scenario is focused on promotional activities and logical entities are activity dates, use the table field corresponding to activity dates (for example, activity name and date range) as the sharding key. This helps you collect statistics about the sales volume for a specified cycle.	Scenarios involving many different types of complicated information. For example, for log analysis, you can select the time field as the sharding key and then shard data using the date function.  To make it easier to clear and dump logs, select the range algorithm and convert the time field value into "year" using the date function so that logs are stored in shards by year. For details, see examples in the following passages.

# Selecting a Sharding Algorithm

A sharding algorithm partitions data from logical tables to multiple shards. DDM supports hash and range algorithms.

#### Hash

Hash evenly distributes data across shards.

Select this algorithm if operators = and **IN** need to be frequently used in SQL query.

#### Range

Range stores records in tables based on the range specified in algorithm metadata.

Select this algorithm if operators greater (>), less (<), and BETWEEN ... AND ... need to be frequently used in SQL queries.

Select an appropriate algorithm based on your service requirements to improve efficiency.

#### ■ NOTE

During the scaling out of a schema:

- Data of logical tables sharded by the hash algorithm is migrated and redistributed evenly across all shards.
- Data of logical tables sharded by the range algorithm is not migrated by default. If you modify the sharding rule and redefine the range, connect to the target RDS shards to complete data migration.

#### Selecting a Sharding Key

A sharding key is a table field used to generate a route during horizontal partitioning of logical tables. After specifying a table field, you can select a date function or manually enter *date function (field name)*. The table field must be **date**, **datetime**, or **timestamp**. Select a date function if data needs to be redistributed by year, month, day, week, or some combinations thereof.

DDM calculates routes based on the sharding key and sharding algorithm, horizontally partitions data in sharded tables, and then redistributes it to shards.

Note that when you select a sharding key and a sharding algorithm:

- Ensure that data is evenly distributed to each shard as much as possible.
- Select the most frequently used field or the most important query condition as the sharding key.
- Prioritize the primary key as the sharding key to keep query the fastest.

### Service Scenarios with a Clear Entity

A sharded table generally contains tens of millions of data records. It is extremely important to select an appropriate sharding key and a sharding algorithm. If a logical entity is identified and most database operations are performed on data of that entity, select the table field corresponding to the entity as a sharding key for horizontal partitioning.

Logical entities depend on actual applications. The following scenarios each include a clear logical entity.

1. Banking applications

The logical entities are customers. In this case, use the table field corresponding to customers (for example, customer account numbers) as the sharding key. Service scenarios of some systems are based on bank cards or accounts. In such cases, select the bank card or account as the sharding key.

2. E-commerce applications

If this scenario focuses on products, the logical entities are products. In this case, use the table field corresponding to products (for example, product code) as the sharding key.

3. Gaming applications

These applications mainly focus on player data, and the logical entities are players. In this case, use the table field corresponding to players (for example, player ID) as the sharding key.

The following is an example SQL statement for creating tables for bank services.

CREATE TABLE PERSONALACCOUNT (
ACCOUNT VARCHAR(20) NOT NULL PRIMARY KEY,
NAME VARCHAR(60) NOT NULL,
TYPE VARCHAR(10) NOT NULL,
AVAILABLEBALANCE DECIMAL(18,2) NOT NULL,
STATUS CHAR(1) NOT NULL,
CARDNO VARCHAR(24) NOT NULL,
CUSTOMID VARCHAR(15) NOT NULL
) ENGINE=INNODB DEFAULT CHARSET=UTF8;

#### **Service Scenarios Without a Clear Entity**

If you cannot identify a suitable entity for your service scenario, select the table field that can provide even data distribution as the sharding key.

For example, the log system may contain a wide range of data records. In this case, you can select the time field as the sharding key.

When the time field is selected as the sharding key, you can specify a date function to partition data.

To make it easier to clear and dump logs, use the range algorithm and convert the time field value into "year" using the date function so that logs are stored in shards by year.

Example SQL statement for creating a table:

CREATE TABLE LOG (
LOGTIME DATETIME NOT NULL,
LOGSOURCESYSTEM VARCHAR(100),
LOGDETAIL VARCHAR(10000)
):

# 14 Cross-Shard Complex SQL Tuning

This section describes how to optimize cross-shard complex SQL statements.

### Suggestions on Tuning GROUP BY and ORDER BY

- 1. If the GROUP BY field is not a data column in the table, use Generated Columns to create a column, add an index to it, and use the column as the GROUP BY condition to avoid using temporary tables for query. (Generated Columns is supported in MySQL 5.7 or later. In MySQL 5.6 and earlier versions, you can create common fields and add indexes.)
- 2. Ensure that the GROUP BY and ORDER BY fields are the same. Data queries are faster when the GROUP BY field contains sharding keys.
- 3. Use indexes in GROUP BY and ORDER BY.

# **Suggestions on Tuning JOINs**

- 1. Preferentially use sharding keys to perform EQUI-JOINs on conditional JOIN statements.
- 2. Use the table with a smaller result set as the driving table to reduce loop JOINs. When LEFT JOINs or JOINs are used, the driving table is on the left and the driven table is on the right. When RIGHT JOINs are used, the driving table is on the right and the driven table is on the left.
- 3. Add indexes to JOIN fields of the driven table. Do not join two large tables directly, especially unsharded tables without any constraint, which may use a large number of compute resources.
- 4. Do not use JOINs in explicit transactions. To ensure transaction visibility, the optimal execution plan may not be used.

# **Suggestions on Tuning SQL UNION**

- 1. Do not perform UNION operations (excluding UNION ALL) between large tables, in case that the number of temporary tables exceeds the upper limit during deduplication.
- 2. Do not use ORDER BY or GROUP BY in a UNION statement.

#### **Suggestions on Tuning Temporary Tables**

When complex SQL statements such as JOINs, UNION, GROUP BY, and ORDER BY are executed across shards, DDM may create a temporary table to store data. The maximum size of a temporary table is 1 million rows by default. If the number of rows in the temporary table exceeds this upper limit, an error "Temp table limit exceeded" will be displayed, indicating that the SQL statements need to be calculated again on the DDM instance, and the required temporary table size exceeds the limit allowed by the DDM instance. If such errors on temporary tables are reported, evaluate whether the SQL statements can be optimized by, for example, configuring broadcast tables, adding conditions for batch query, or modifying SQL statements based on sharding architecture. If there is no optimization potential, check data volume and instance resources. If there are no more than 3 million rows of data and instance resources are sufficient, set the instance parameter **temp\_table\_size\_limit** to a large value on the console to remove the restriction on temporary table size.

# **Scenarios Where Temporary Tables Are Generated**

- The GROUP BY and ORDER BY fields are different.
- 2. The DISTINCT keyword is used.
- 3. UNION queries are executed.
- 4. Some cross-shard subqueries are executed.
- 5. In a JOIN query, non-equivalent JOIN conditions are used, or a transaction is enabled on the connection between two subqueries.