Distributed Database Middleware

Service Overview

Issue 01

Date 2024-09-26

HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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

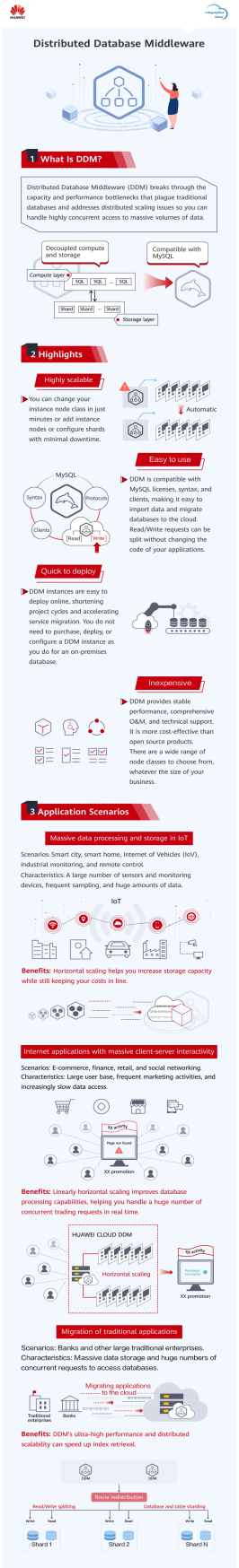
Vulnerability

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 Infographics 1
2 What Is DDM? 3
3 Basic Concepts 5
4 Core Functions 7
5 Product Specifications 10
6 Use Constraints 11
6.1 Network Access 11
6.2 Data Nodes 11
6.3 Unavailable Features and Limitations 12
6.4 High-risk Operations 18
7 Regions and AZs 19
8 Application Scenarios 21
9 Permissions Management 22

1 Infographics



2 What Is DDM?

Definition

Distributed Database Middleware (DDM) is a MySQL-compatible, distributed middleware service designed for relational databases. It can resolve distributed scaling issues to break through capacity and performance bottlenecks of databases, helping handle highly concurrent access to massive volumes of data.

Developed by Huawei Cloud, DDM is cloud-native, reliable, stable, and maintainable. It uses an architecture with decoupled storage and compute and can provide functions such as database and table sharding, read/write splitting, and elastic scaling. Management of instance nodes has no impacts on your workloads. You can perform O&M on your databases and read and write data from and to them on the DDM console, just like as operating a single-node database.

Advantages

- Automatic database and table sharding

  MySQL databases are usually deployed on single nodes. Once a fault occurs, all data may be lost, and your workloads are 100% affected.

  DDM supports automatic database and table sharding to distribute data across multiple data nodes, so impacts on your services are greatly reduced once a fault occurs. It also supports explosive growth of services.

- Read/Write splitting

  DDM can leverage data nodes. If there is still great query pressure after horizontal sharding, you can enable read/write splitting to speed up database processing and access, without the need to reconstruct your service system.

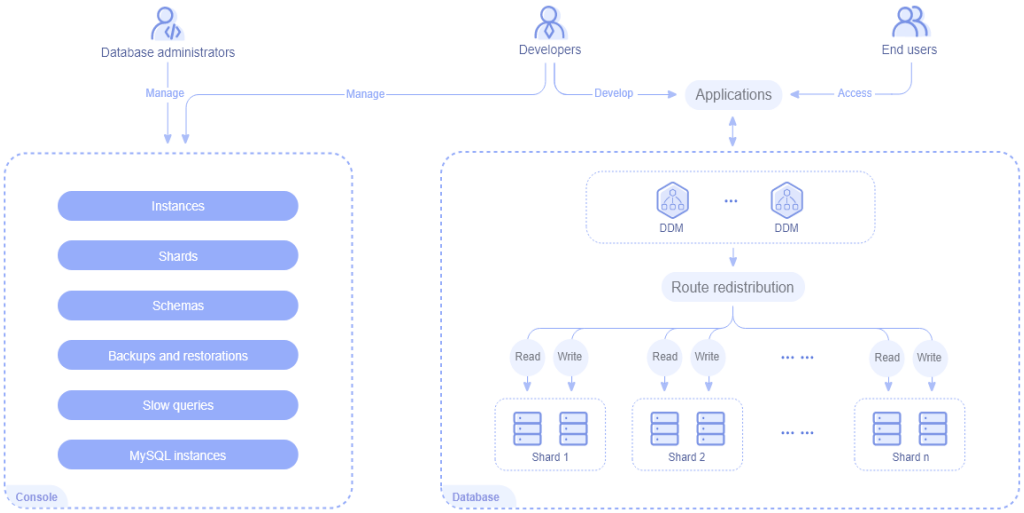
- Elastic scaling

  MySQL databases can support only medium- and small-scale service systems because their CPU, memory, and network processing are limited by server configurations and their storage depends on the size of SSD or EVS disks.

  DDM supports both compute and storage scaling. You can add nodes to a DDM instance or scale up its node class. Alternatively, increase shards or data nodes to distribute data from one large table to multiple tables or scale out storage resources as services grow, without worrying about O&M.

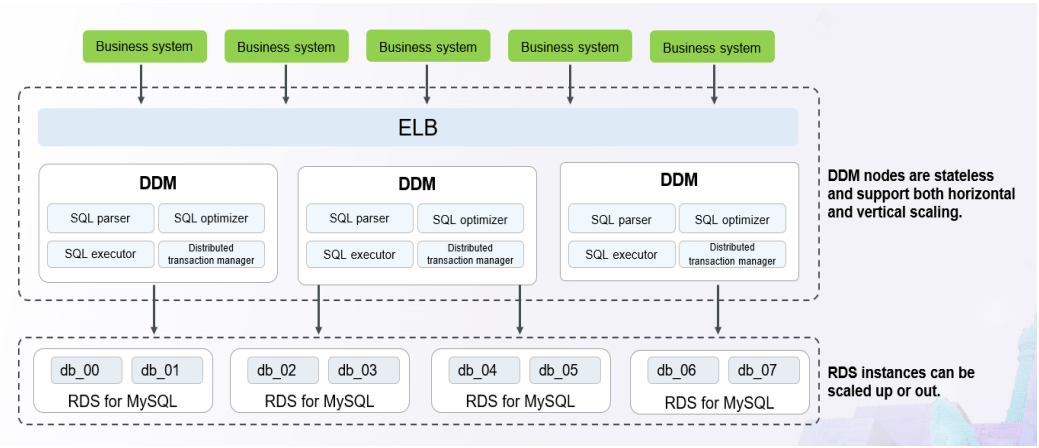
Service Architecture

Figure 2-1 DDM service architecture



How DDM Works

Figure 2-2 DDM working diagram



3 Basic Concepts

Data Node

A data node is the minimum management unit of DDM. Each data node represents an independently running database, and it may an RDS for MySQL instance that is associated with your DDM instance. You can create multiple schemas in a DDM instance to manage data nodes and access each data node independently.

NOTE

DDM instances do not store service-related data, which is stored in shards of data nodes.

VPC

A Virtual Private Cloud (VPC) is a private and isolated virtual network. You can configure IP address ranges, subnets, and security groups, assign EIPs, and allocate bandwidth for DDM instances.

Subnet

A subnet is a range of IP addresses, a logical subdivision of an IP network. Subnets are created for a VPC where you will place your DDM instances. Every subnet is defined by a unique CIDR block which cannot be modified once the subnet is created.

Security Group

A security group is a collection of rules for ECSs that have the same security protection requirements and are mutually trusted. After a security group is created, you can add different access rules to the security group, and these rules will apply to all ECSs added to this security group.

Your account automatically comes with a security group by default. The default security group allows all outbound traffic and denies all inbound traffic. Your ECSs in this security group can communicate with each other without the need to add rules.

Parameter Template

A parameter template acts as a container for configuration values that can be applied to one or more DDM instances. If you want to use your own parameter template, you only need to create a custom parameter template and select it when creating a DDM instance. You can also apply the parameter template to an existing DDM instance.

EIP

The Elastic IP (EIP) service provides independent public IP addresses and bandwidth for Internet access. EIPs can be bound to and unbound from DDM instances.

Region and Endpoint

Before you use an API to call resources, specify its region and endpoint. For more details, see **Regions and Endpoints**.

4 Core Functions

DDM supports a variety of functions, including horizontal sharding, shard configuration, highly compatible SQL syntax, read/write splitting, and global sequence.

Table 4-1 DDM core functions

Function	Description
Horizontal sharding	Select a sharding key when creating a logical table. DDM will generate a sharding rule and horizontally shard data. **NOTE** 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.
Flexible shard configuration	DDM supports both compute and storage scaling. You can add nodes to a DDM instance or scale up its node class. Alternatively, increase shards or data nodes to distribute data from one large table to multiple tables or scale out storage resources. Compute scaling is undetectable to your applications. Storage scaling minimizes service interruption to seconds.
Distributed transactions	DDM processes three types of transactions, including single-shard, FREE, and Extended Architecture (XA). • Single-shard: Transactions cannot be committed across shards. • FREE: Transactions are committed across shards. A transaction is not rolled back when it fails to be executed by any shard, causing data inconsistency. • XA: Transactions are committed in two phases. If a transaction fails to be executed by any shard, all work done will be rolled back to ensure data consistency.

Function	Description
Data import and export	External data can be imported into DDM instances to help you migrate databases to the cloud, and DDM instance data can be exported based on service requirements. For details, see **Data Migration**.
Highly compatible SQL syntax	DDM is compatible with MySQL 5.7 and 8.0 licenses and syntax.
Read and write splitting	Read and write requests can be split without modifying the application code, and this is totally transparent to applications. You only need to create read replicas for a MySQL instance associated with your DDM instance and configure a read policy, and a large number of concurrent requests can read data from those read replicas.
Global sequence	DDM allows you to use globally unique, distributed, and ascending SNs as primary or unique keys or to meet your requirements in specific scenarios.
Online monitoring	DDM monitors reads, writes, and slow query logs of your DDM instance on the console, helping you detect resource and performance bottlenecks as fast as possible.
O&M console	DDM enables you to manage and maintain DDM instances, schemas, and accounts on a visualized console.

Related Services

- VPC

  DDM instances are deployed in an isolated VPC and you can configure IP addresses and bandwidth for accessing these DDM instances and use a security group to control access to them.

- ECS

  You can access your DDM instance through an ECS.

- Relational Database Service (RDS)

  After you buy a DDM instance, you can associate it with RDS for MySQL instances in the same VPC to obtain separated storage resources.

- Cloud Eye

  Cloud Eye monitors DDM metrics and sends notifications in the event of an alarm or event.

- Cloud Trace Service (CTS)

  CTS records operations on your DDM resources for later query, audit, and backtrack.

- Elastic Load Balance (ELB)

  ELB distributes incoming traffic to multiple backend servers based on forwarding policies to balance workloads. So, it can expand external service capabilities of DDM and eliminate single points of failure (SPOFs) to improve service availability.

- Data Admin Service (DAS)

  DAS provides a GUI interface for you to connect to and manage cloud databases.

5 Product Specifications

There are two types of DDM instances: general-enhanced and Kunpeng general computing-plus.

- General-enhanced DDM instances use Intel® Xeon® Scalable processors. Working in high-performance networks, these DDM instances offer high and stable computing performance, meeting enterprise-class application requirements.
- Kunpeng general computing-plus DDM instances use Kunpeng 920 processors and 25GE high-speed intelligent NICs to offer powerful computing and high-performance networks. These instances provide governments and Internet enterprises with cost-effective, secure, and reliable cloud services.

Table 5-1 Product specifications

Specification	Architecture	Specification Code	vCPUs	Memory (GB)
General-enhanced	x86	ddm.c6.2xlarge.2	8	16
		ddm.c6.4xlarge.2	16	32
		ddm.c6.8xlarge.2	32	64
Kunpeng general computing-plus	Arm	ddm.kc1.2xlarge.2	8	16
		ddm.kc1.4xlarge.2	16	32
		ddm.kc1.8xlarge.2	32	64

6 Use Constraints

6.1 Network Access

Restrictions on DDM network access are as follows:

- The data nodes and ECSs running your applications must be in the same VPC as your DDM instance.
- To access DDM from your computer, you need to bind an EIP to your DDM instance and then use the EIP to access the DDM instance.

6.2 Data Nodes

Constraints on data nodes are as follows:

- Data nodes can be only RDS for MySQL instances of versions 5.7 and 8.0. Serverless and container DB instances are not supported.
- GaussDB(for MySQL) and HRDS instances are not supported.
- DDM does not support SSL for MySQL instances.
- Do not enable case sensitivity support for MySQL instances.

  NOTE

  - If you are using MySQL 5.7, select **Case insensitive** for **Table Name** when you create a MySQL instance, or set **lower_case_table_names** to **1** on the **Parameters** page after you complete the creation.
  - If you are using MySQL 8.0, select **Case insensitive** for **Table Name** when you create a MySQL instance.
- When you modify configurations of a MySQL instance associated with DDM, an exception may occur. After the modification, click **Synchronize Data Node Information** on the **Data Nodes** page to synchronize changes from the data node to DDM.
- Do not use character set GBK for data nodes.

6.3 Unavailable Features and Limitations

Unsupported Features

- Stored procedures
- Triggers
- Views
- Events
- User-defined functions
- Foreign key references and associations
- Full-text indexes and SPACE functions
- Temporary tables
- Compound statements such as BEGIN...END, LOOP...END LOOP, REPEAT...UNTIL...END REPEAT, and WHILE...DO...END WHILE
- Process control statements such as IF and WHILE
- RESET and FLUSH statements
- BINLOG statement
- HANDLER statement
- INSTALL and UNINSTALL PLUGIN statements
- Character sets other than ASCII, Latin1, binary, utf8, and utf8mb4
- SYS schema
- MySQL Optimizer Trace
- X-Protocol
- CHECKSUM TABLE syntax
- Table maintenance statements, including CHECK, CHECKSUM, OPTIMIZE, and REPAIR TABLE
- Statements for assigning a value to or querying variable **session**

  Example:

  ```
  set @rowid=0;
  select @rowid:=@rowid+1,id from user;
  ```

- SQL statements that use -- or /.../ to comment out a single line or multiple lines of code
- Incomplete support for system variable queries. The returned values are variable values of RDS instances, instead of DDM kernel variable values. For example, the returned values of SELECT @@autocommit do not indicate the current transaction status.
- Executing SET Syntax to modify global variables
- PARTITION syntax. Partitioned tables are not recommended.
- LOAD XML statement
- Inline comments
- CREATE TABLE AS WITH SELECT syntax
- ZEROFILL CREATE syntax

Unsupported Operators

- Assignment operator **:=**. Executing this operator is allowed. After this operator is executed, it does not take effect even if no errors are reported.
- Arrow operator **->**. This operator can be executing on a single table. Errors will be displayed if the operator is executed on other types of tables.
- Arrow operator **->>**. This operator can be executing on a single table. Errors will be displayed if the operator is executed on other types of tables.
- Expression IS UNKNOWN
- Operators =, <, <=, >, >=, <>, !=, <=> that require comparing JSON fields at the DDM compute layer

Unsupported Functions

The DDM compute layer does not support the following functions. Do not use them if you are not sure that they can be pushed down to RDS for execution.

- XML functions
- Function **ANY_VALUE()**
- Function **ROW_COUNT()**
- Function **COMPRESS()**
- Function **SHA()**
- Function **SHA1()**
- Function **AES_ENCRYPT()**
- Function **AES_DECRYPT()**
- Aggregate function **JSON_OBJECTAGG()**
- Aggregate function **JSON_ARRAYAGG()**
- Aggregate function **STD()**
- Aggregate function **STDDEV()**
- Aggregate function **STDDEV_POP()**
- Aggregate function **STDDEV_SAMP()**
- Aggregate function **VAR_POP()**
- Aggregate function **VAR_SAMP()**
- Aggregate function **VARIANCE()**
- Function **MICROSECOND()**
- Function **TO_DAYS()**
- Function **TO_SECONDS()**
- Function **UNCOMPRESS()**
- Function **UNCOMPRESSED_LENGTH()**
- Function **UNHEX()**
- Function **YEARWEEK()**
- Function **TIME_FORMAT()**

SQL Syntax Limitations

Unsupported SELECT syntax

- DISTINCTROW
- Configuring options [HIGH_PRIORITY], [STRAIGHT_JOIN], [SQL_SMALL_RESULT], [SQL_BIG_RESULT], [SQL_BUFFER_RESULT], and [SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] in SELECT statements on DDM instances
- SELECT ... GROUP BY ... WITH ROLLUP
- SELECT ... ORDER BY ... WITH ROLLUP
- WITH
- JOIN statements with different collations
- Window functions
- SELECT FOR UPDATE supports only simple queries and does not support statements such as JOIN, GROUP BY, ORDER BY, and LIMIT. Option [NOWAIT | SKIP LOCKED] for modifying FOR UPDATE is invalid for DDM.
- DDM does not support multiple columns with the same name for each SELECT statement in UNION.

  For example, duplicate column names are used in the following SELECT statement:

  ```
  SELECT id, id, name FROM t1 UNION SELECT pk, pk, name FROM t2;
  ```

SORT and LIMIT

LIMIT/OFFSET, allowed value range: 0–2147483647

Aggregation

Function **asc** or **desc** cannot be used in the GROUP BY statement to sort out results.

NOTE

- DDM automatically ignores keyword **asc** or **desc** after GROUP BY.
- In MySQL versions earlier than 8.0.13, function **asc** or **desc** can be used in the GROUP BY statement to sort out results. In MySQL 8.0.13 or later, a syntax error is reported if you use function **asc** or **desc** this way. ORDER BY is recommended for sorting.

Subqueries

- Subqueries that join grandparent queries are not supported.
- Subqueries in the HAVING clause and the JOIN ON condition are not supported.
- Each derived table must have an alias.
- A derived table cannot be a correlated subquery.

LOAD DATA

- LOW_PRIORITY is not supported.
- CONCURRENT is not supported.
- PARTITION (partition_name [, partition_name] ...) is not supported.
- LINES STARTING BY 'string' is not supported.
- User-defined variables are not supported.
- ESCAPED BY supports only '\\'.

- If you have not specified a value for your auto-increment key when you insert a data record, DDM will not fill a value for the key. The auto-increment keys of data nodes of a DDM instance all take effect, so the auto-increment key values may be duplicate.
- If the primary key or unique index is not routed to the same physical table, REPLACE does not take effect.
- If the primary key or unique index is not routed to the same physical table, IGNORE does not take effect.
- Executing LOAD DATA statements is not allowed on the tables that contain global secondary indexes.

INSERT and REPLACE

- INSERT DELAYED is not supported.
- Only INSERT statements that contain sharding fields are supported.
- PARTITION syntax is not supported. Partitioned tables are not recommended.
- Setting **YYYY** of **datetime** (in the format of **YYYY-MM-DD HH:MM:SS**) to **1582** or any value smaller in INSERT statements is not supported.
- Nesting a subquery in ON DUPLICATE KEY UPDATE of an INSERT statement is not supported. The following is an example:

  ```
  INSERT INTO t1(a, b)
  SELECT * FROM(SELECT c, d FROM t2 UNION SELECT e, f FROM t3) AS dtest
  ON DUPLICATE KEY UPDATE b = b + c;
  ```

  Subquery c is used in the ON DUPLICATE KEY UPDATE clause.
- The sharding key values in INSERT and REPLACE statements cannot be **DEFAULT**.

UPDATE and DELETE

- Updating a sharding key value to **DEFAULT** is not supported.
- Repeatedly updating the same field in one SQL statement is not supported.
- Updating a sharding key using UPDATE JOIN is not supported. The following is an example:

  ```
  UPDATE tbl_1 a, tbl_2 b set a.name=b.name where a.id=b.id;
  ```

  name indicates the sharding key of table **tbl_1**.
- Updating a sharding key by executing INSERT ON DUPLICATE KEY UPDATE is not supported.
- Updating self-joins is not supported.

  ```
  UPDATE tbl_1 a, tbl_1 b set a.tinyblob_col=concat(b.tinyblob_col, 'aaabbb');
  ```
- Updating sharding keys in subqueries is not allowed for secondary sharded tables that contain JSON fields.
- UPDATE JOIN supports only joins with WHERE conditions.

  The following is an example:

  ```
  UPDATE tbl_3, tbl_4 SET tbl_3.varchar_col='dsgfdg';
  ```
- Referencing other object columns in assignment statements or expressions is not supported when UPDATE JOIN syntax is used. The following is an example:

  ```
  UPDATE tbl_1 a, tbl_2 b SET a.name=concat(b.name, 'aaaa'),b.name=concat(a.name, 'bbbb') ON
  a.id=b.id;
  ```


- You can update a sharding field by two steps: delete the original sharding field and then insert a new field. During this process, the results of querying the sharding fields involved in the target table may be inconsistent.

DDL

- Modifying database names and sharding field names and types is not allowed.
- Creating or deleting schemas by executing SQL statements is not supported.
- Index FULL_TEXT is not supported.
- AS SELECT clause of the CREATE TABLE statement is not supported.
- CREATE TABLE ... LIKE statement is not supported.
- Dropping multiple tables with one SQL statement is not supported.
- Executing multiple DDL statements at the same time is not supported.
- Creating foreign keys for broadcast and sharded tables is not supported.
- Creating tables whose names are prefixed by **_ddm** is not supported.
- Creating temporary sharded or broadcast tables is not supported.
- Specifying globally unique keys in the CREATE TABLE statement is not supported.
- Modifying global secondary indexes is not supported.

Indexes

- Global unique indexes are not supported. Unique keys and primary keys may not be globally unique.
- Values of primary key and sharding key fields cannot exceed the corresponding data range.
- If you want to use a global secondary index, set **sql_mode** to **STRICT_TRANS_TABLES**.

Table Recycle Bins: Unsupported Functions

- Hints
- Deleting tables by schema
- Deleting tables by logical table
- After a table is recovered, its globally unique sequence increases automatically but may not follow the last sequence value.
- Shard configuration
- Retaining copies with no time limit
- Recovering data to a table with any name
- Unlimited copies

Transactions: Unsupported Functions

- Savepoints
- XA syntax. DDM has implemented distributed transactions through XA, so the user layer does not need to process the syntax.

- Customizing the isolation level of a transaction. Currently, DDM supports only the READ COMMITTED isolation level. In consideration of compatibility, DDM does not report errors for any SQL statement (such as SET GLOBAL TRANSACTION ISOLATION LEVEL REPEATABLE READ) to set the database isolation level, but will ignore the modifications to the transaction isolation level.
- Setting a transaction to read-only (START TRANSACTION READ ONLY). DDM can enable read/write of a transaction, instead of enabling read-only, to ensure compatibility.

Permissions

- Column-level permissions
- Subprogram-level permissions

Database Management Statements

- SHOW TRIGGERS statements are not supported.
- Most of SHOW statements such as SHOW PROFILES, SHOW ERRORS, and SHOW WARNINGS are not supported.
- The following SHOW statements are randomly sent to a database shard. If database shards are on different RDS for MySQL instances, the returned variables or table information may be different.
  - SHOW TABLE STATUS
  - SHOW VARIABLES Syntax
  - SHOW WARNINGS Syntax does not support the combination of LIMIT and COUNT.
  - SHOW ERRORS Syntax does not support the combination of LIMIT and COUNT.

INFORMATION_SCHEMA

Only simple queries of SCHEMATA, TABLES, COLUMNS, STATISTICS, and PARTITIONS are supported. No subqueries, JOINs, aggregate functions, ORDER BY, and LIMIT are allowed.

Broadcast Tables

The DDM broadcast table mechanism is that each SQL statement can be executed on all shards of a broadcast table. When you use such a broadcast table, do not use any function that has different results returned each time it is executed. Otherwise, data inconsistency will occur between different broadcast table shards. If such functions are indeed required, calculate their results, write the results to your SQL statements, and then execute the SQL statements on the broadcast table. Functions of this type include but are not limited to the following:

- CONNECTION_ID()
- CURDATE()
- CURRENT_DATE()
- CURRENT_TIME()

- CURRENT_TIMESTAMP()
- CURTIME()
- LAST_INSERT_ID()
- LOCALTIME()
- LOCALTIMESTAMP()
- NOW()
- UNIX_TIMESTAMP()
- UTC_DATE()
- UTC_TIME()
- UTC_TIMESTAMP()
- CURRENT_ROLE()
- CURRENT_USER()
- FOUND_ROWS()
- GET_LOCK()
- IS_FREE_LOCK()
- IS_USED_LOCK()
- JSON_TABLE()
- LOAD_FILE()
- MASTER_POS_WAIT()
- RAND()
- RELEASE_ALL_LOCKS()
- RELEASE_LOCK()
- ROW_COUNT()
- SESSION_USER()
- SLEEP()
- SYSDATE()
- SYSTEM_USER()
- USER()
- UUID()
- UUID_SHORT()

6.4 High-risk Operations

Pay attention to the following when you use DDM:

- Do not connect to any data node for data operations to avoid deleting system catalogs or metadata by mistake.
- Do not clear system tables **TBL_DRDS_TABLE** and **MYCAT_SEQUENCE** to prevent metadata loss.

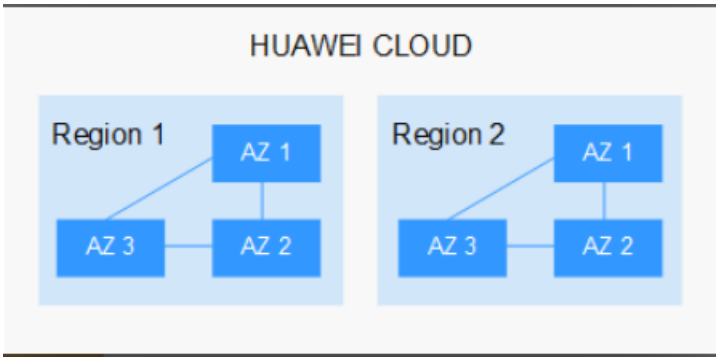
7 Regions and AZs

Concepts

The combination of a region and an availability zone (AZ) identifies the location of a data center. You can create resources in a specific AZ in a region.

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

Figure 7-1 Regions and AZs



Huawei Cloud provides services in many regions around the world. You can select a region and AZ as needed. For more information, see **Huawei Cloud Global Regions**.

Selecting a Region

When selecting a region, consider the following factors:

- Location

  Select the region closest to you or your target users to reduce likelihood of latency issues. Chinese mainland regions provide basically the same infrastructure, BGP network quality, as well as operations and configurations on resources. If you or your target users are in the Chinese mainland, it is not necessary to consider network latency differences when selecting a region.

- Resource price

  Resource prices may vary in different regions. For details, see **Product Pricing Details**.

Selecting an AZ

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

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

8 Application Scenarios

As an OLTP-oriented service, DDM allows you to access distributed relational databases across a variety of industries.

It is especially suitable for applications requiring high-concurrency access to large volumes of data. Typical application scenarios are as follows:

- **Internet**

  E-commerce, finance, O2O, retail, and social networking applications usually face challenges such as large user base, frequent marketing events, and slow response of core transactional systems. DDM can scale compute and storage resources to improve database processing of high-concurrency transactions and ensure fast access to data.

- **IoT**

  In industrial monitoring, remote management, smart city extension, smart home, and Internet of Vehicles (IoV) scenarios, a large number of sensors and monitoring devices frequently collect data and generate huge amounts of data, which may exceed the storage capability of single-node databases. DDM provides horizontal expansion to help you store massive data at low costs.

- **Traditional sectors**

  Government agencies, large-sized enterprises, banks, and the like usually use commercial solutions to support high-concurrency access to large volumes of data. These solutions are expensive because they need to rely on mid-range computers and high-end storage devices. DDM, deployed in clusters with common ECSs, provides cost-efficient database solutions with the same or even higher performance than traditional commercial database solutions.

9 Permissions Management

If your account does not need individual IAM users for permissions management, you may skip over this section.

If you need to assign different permissions to employees in your enterprise to access your DDM resources, IAM is a good choice for fine-grained permissions management. IAM provides functions like identity authentication, permissions management, and access control, helping you secure access to your cloud resources.

You can create IAM users for your employees, and assign permissions to these users to control their access to specific types of resources. For example, you can create IAM users for software developers and assign specific permissions to allow them to use DDM resources but disallow them to delete the resources or perform any high-risk operations.

IAM is a free service. You pay only for the resources in your account.

DDM Permissions

By default, new IAM users do not have any permissions assigned. To assign permissions to these new users, you need to add them to one or more groups, and attach permissions policies or roles to these groups.

DDM is a project-level service deployed in specific physical regions. When you assign DDM permissions to a user group, you need to specify region-specific projects where the permissions will take effect. If you select **All projects**, the permissions will be granted for all region-specific projects. To access DDM, you need to switch to the region where you are authorized.

You can grant users permissions using roles and policies.

- Roles: A type of coarse-grained authorization mechanism that provides only a limited number of service-level roles. When using roles to grant permissions, you also need to assign other dependent roles. Roles are not ideal for fine-grained authorization and secure access control.
- Policies: A fine-grained authorization mechanism that defines permissions required to perform operations on specific cloud resources under certain conditions. This mechanism allows for more flexible policy-based authorization and more secure access control. For example, you can grant

IAM users only the permissions for managing a certain type of DDM resources.

Table 9-1 System-defined policies

Policy Name	Description	Type	Dependency
DDM FullAccess	Full permissions for Distributed Database Middleware	System-defined policy	None
DDM CommonOperations	Common permissions for Distributed Database Middleware, excluding the permissions to create, delete, and add nodes, configure shards, and roll back shard configuration tasks	System-defined policy	None
DDM ReadOnlyAccess	Read-only permissions for Distributed Database Middleware	System-defined policy	None

The following are permission configurations of supported system-defined policies:

- DDM FullAccess

```
{
    "Version": "1.1",
     "Statement": [{
      "Action": ["ddm:*:*",
      "rds:instance:list",
          "rds:instance:modify",
          "rds:instance:modifyParameter",
      "vpc:*:*",
          "ecs:*:get*",
          "ecs:*:list*",
          "ecs:cloudServerNics:update",
          "ecs:serverInterfaces:use"],
       "Effect": "Allow"
      }]
  }
```

- DDM CommonOperations

```
{
   "Version": "1.1",
   "Statement": [{
              "Action": [
                  "vpc:*:list*",
                  "vpc:*:get*",
                  "vpc:ports:update",
                  "ecs:*:get*",
                  "ecs:*:list*",
                  "rds:instance:list",
                  "rds:instance:modify",
                  "rds:instance:modifyParameter"
              ],
              "Effect": "Allow"
              },
              {
                  "Condition": {
                     "StringEqualsIgnoreCase": {
```



```
                    "g:ServiceName": [
                        "ddm"
                    ]
                }
            },
            "NotAction": [
                "ddm:instance:create",
                "ddm:instance:delete",
                "ddm:database:migrate*",
                "ddm:instance:resize",
                "ddm:instance:extendNode"
            ],
            "Effect": "Allow"
        }]
}
```

- DDM ReadOnlyAccess

```
{
    "Version": "1.1",
    "Statement": [{
            "Action": [
                "rds:instance:list",
                "vpc:*:*list*",
                "vpc:*:*get*",
                "ecs:*:get*",
                "ecs:*:list*",
                "ddm:*:list",
                "ddm:*:get",
                "ddm:instance:listParameter",
                "ddm:instance:listRwInfo",
                "ddm:instance:listSlowSqlInfo",
                "ddm:rds:connectivity"
            ],
            "Effect": "Allow"
        }]
}
```

Table 9-2 lists the common operations supported by each DDM system-defined policy. Choose appropriate system-defined policies based on your requirements.

Table 9-2 Common operations supported by each system-defined policy

Operation	DDM FullAccess	DDM CommonOperations	DDM ReadOnlyAccess
Querying DDM instances	Supported	Supported	Supported
Querying details of a DDM instance	Supported	Supported	Supported
Modifying instance information, including the name and security group	Supported	Supported	Not supported
Restarting a DDM instance	Supported	Supported	Not supported
Creating a DDM instance	Supported	Not supported	Not supported
Deleting a DDM Instance	Supported	Not supported	Not supported
Changing node class	Supported	Not supported	Not supported

Operation	DDM FullAccess	DDM CommonOperations	DDM ReadOnlyAccess
Scaling out a DDM instance	Supported	Not supported	Not supported
Creating a schema	Supported	Supported	Not supported
Querying schemas	Supported	Supported	Supported
Querying details of a schema	Supported	Supported	Supported
Performing a rollback if configuring shards fails Deleting source data if configuring shards fails Retrying if configuring shards fails	Supported	Not supported	Not supported
Deleting a schema	Supported	Supported	Not supported
Querying accounts	Supported	Supported	Supported
Creating an account	Supported	Supported	Not supported
Modifying an account	Supported	Supported	Not supported
Resetting a password	Supported	Supported	Not supported
Deleting an account	Supported	Supported	Not supported
Synchronizing data node information	Supported	Supported	Not supported
Querying data nodes	Supported	Supported	Supported
Querying details of a data node	Supported	Supported	Supported
Modifying the read policy of a data node	Supported	Supported	Not supported
Viewing products	Supported	Supported	Supported
Creating a parameter template	Supported	Supported	Not supported
Deleting a parameter template	Supported	Supported	Not supported
Applying a parameter template	Supported	Supported	Not supported
Modifying a parameter template	Supported	Supported	Not supported

Operation	DDM FullAccess	DDM CommonOperations	DDM ReadOnlyAccess
Replicating a parameter template	Supported	Supported	Not supported
Comparing two parameter templates	Supported	Supported	Supported
Querying parameter templates	Supported	Supported	Supported
Viewing all tags	Supported	Supported	Supported
Adding, modifying, or deleting a tag	Supported	Supported	Not supported
Querying a session	Supported	Supported	Supported
Killing a session	Supported	Supported	Not supported

Table 9-3 Common operations and supported actions

Operation Category	Operation	Action
DDM routine operations	Buying a pay-per-use DDM instance Buying a yearly/ monthly DDM instance	ddm:instance:create Before you buy a DDM instance, obtain the following dependent permissions: • ecs:*:get* • ecs:*:list* • vpc:vpcs:list • vpc:securityGroups:get • vpc:subnets:get • ecs:cloudServerNics:update • ecs:serverInterfaces:use • **vpc:ports:*** for a global or regional DDM instance • **BSS Finance** and **BSS Operator** policies This permission is required only when you buy yearly/monthly DDM instances.
	Querying DDM instances	ddm:instance:list

Operation Category	Operation	Action
	Querying details of a DDM instance	ddm:instance:get To view details of a DDM instance, you need to configure the following permissions: ● vpc:*:get* ● vpc:*:list*
	Modifying instance information, including modifying the name, changing the security group, or adding, modifying, or deleting a tag of a DDM instance	ddm:instance:modify To modify a security group, you need to configure the following permissions: ● vpc:*:get* ● vpc:*:list* ● vpc:ports:update
	Restarting a DDM instance	ddm:instance:reboot
	Deleting a DDM instance	ddm:instance:delete vpc:ports:delete
	Changing node class	ddm:instance:resize
	Scaling out a DDM instance	ddm:instance:extendNode
	Monitoring the read/write ratio	ddm:instance:listRwInfo
	Querying slow query logs	ddm:instance:listSlowSqlInfo

Operation Category	Operation	Action
DDM routine operations	Auto-renew (for yearly/ monthly instances)	Configure policies **BSS Finance** and **BSS Operator** as follows: 1. Log in to the IAM console. 2. In the navigation pane, click **User Groups**. 3. Choose **More** > **Assign Permissions**. 4. Click **Attach Policy** in the same row as the project for which you want to edit the permissions. 5. In the **Available Policies** area, select **BSS Finance** and **BSS Operator**.
DDM routine operations	Changing to yearly/ monthly billing	Configure policies **BSS Finance** and **BSS Operator**. The procedure is the same as that for renewing an instance.
Schema operations	Creating a schema	ddm:database:create
	Querying schemas	ddm:database:list
	Querying details of a schema	ddm:database:get
	Performing a rollback if configuring shards fails Deleting source data if configuring shards fails Retrying if configuring shards fails	ddm:database:migrateRollback
	Deleting a schema	ddm:database:delete
DDM account operations	Querying accounts	ddm:user:list
	Creating an account	ddm:user:create
	Modifying an account	ddm:user:modify

Operation Category	Operation	Action
	Resetting a password	ddm:user:modify
	Deleting an account	ddm:user:delete
Data node management (using an RDS for MySQL instance as an example)	Synchronizing data node information	ddm:rds:synchro To synchronize data node information, you need to configure the following permissions: ● rds:instance:list ● rds:instance:modify ● rds:instance:modifyParameter
	Querying data nodes	ddm:rds:list
	Querying details of a data node	ddm:rds:get
	Modifying the read policy of a data node	ddm:rds:modifyReadPolicy
DDM product operations	Viewing products	ddm:product:list
Parameter template operations	Creating a parameter template	ddm:param:create
	Deleting a parameter template	ddm:param:delete
	Applying a parameter template	ddm:param:apply
	Modifying a parameter template	ddm:param:update
	Replicating a parameter template	ddm:param:create

Operation Category	Operation	Action
	Comparing two parameter templates	ddm:param:list
	Querying parameter templates	ddm:param:list
Tag operations	Querying the tag list	ddm:tag:list
	Adding, modifying, or deleting a tag	ddm:tag:modify
Session operations	Querying a session	ddm:instance:queryProcessList
	Killing a session	ddm:instance:killProcessList

