分布式关系型数据库
用户使用指南

天翼云科技有限公司

目录

1 产品介绍

1.1 产品概述

产品定义

分布式关系型数据库（Distribute Relational Database Service，简称 DRDS），兼容 MySQL 协议，专注于解决数据库分布式扩展问题，突破传统数据库的容量和性能瓶颈，实现海量数据高并发访问。

DRDS 是一款基于云原生分布式关系型数据库，采用存算分离架构，提供分库分表、读写分离、弹性扩容等能力，具有稳定可靠、高度可扩展、持续可运维的特点。服务器集群管理对用户完全透明，用户通过 DRDS 管理控制台进行数据库运维与数据读写，提供类似传统单机数据库的使用体验。

产品优势

- 自动分库分表

  传统数据库通常是单机部署，一旦出现问题，数据可能全部丢失，故障影响面 100%。

  而 DRDS 支持自动分库分表，将数据分散到多个数据节点存储，分散风险，影响面降低至 1/N，支撑业务爆发式增长。

- 读写分离

  DRDS 充分利用数据节点只读实例能力，当水平拆分后，依然存在较大查询压力，则可以开启读写分离能力，业务系统无需改造，提升数据库处理能力和访问效率，轻松应对高并发场景。

- 弹性扩容

  传统数据库计算能力和存储能力有限，CPU/内存/网络处理能力受限于机器配置，存储受限于 SSD 或者云盘的大小，只能支撑中小规模的业务系统。

  而 DRDS 既支持计算层（DRDS）扩容（增加节点数或提升节点规格），也支持存储层在线扩容，存储层扩容可以通过增加分片数或者数据节点数来解决单表数据量过多和容量瓶颈等问题，确保计算、存储均可线性扩展，解决业务在快速发展的过程中针对数据库扩展性产生的后顾之忧与运维压力。

1.2 常用概念

数据节点

数据节点是分布式关系型数据库服务的最小管理单元，表示 DRDS 下关联的 RDS for MySQL、GaussDB（for MySQL）两种引擎的实例，DRDS 目前仅支持这两种引擎，一个实例代表了一个独立运行的数据库。您可以在一个 DRDS 实例中通过创建多个逻辑库管理多个数据节点，并且可以独立访问数据节点。

虚拟私有云

虚拟私有云（Virtual Private Cloud）是用户在云服务上申请的隔离的、私密的虚拟网络环境。用户可以自由配置 VPC 内的 IP 地址段、子网、安全组等子服务，也可以申请弹性带宽和弹性 IP 搭建业务系统。

子网

子网是虚拟私有云内的 IP 地址块。虚拟私有云中的所有云资源都必须部署在子网内。同一个虚拟私有云下，子网网段不可重复。子网创建成功后，网段无法修改。

安全组

安全组是一个逻辑上的分组，为具有相同安全保护需求并相互信任的云服务器提供访问策略。安全组创建后，用户可以在安全组中定义各种访问规则，当云服务器加入该安全组后，即受到这些访问规则的保护。

系统会为每个用户默认创建一个默认安全组，默认安全组的规则是在出方向上的数据报文全部放行，入方向访问受限，安全组内的云服务器无需添加规则即可互相访问。

参数模板

参数模板就像是参数配置值的容器，这些值可应用于一个或多个 DRDS 实例。如果您想使用您自己的参数模板，只需创建一个新的参数模板，创建实例的时候选择该参数模板。如果是在创建 DRDS 实例后有这个需求，可以重新应用该参数模板。

弹性公网 IP

弹性公网 IP（Elastic IP，简称 EIP）提供独立的公网 IP 资源，包括公网 IP 地址与公网出口带宽服务。可以与 DRDS 实例解绑和绑定。

1.3 产品核心功能

分布式关系型数据库具备水平拆分、分片变更、SQL 语法、读写分离、全局序列等主要功能。

表 1-1 主要功能介绍

功能名称	说明
水平拆分	在创建逻辑库时，只需选择拆分键，DRDS 就可以按照拆分键生成拆分规则，实现数据水平拆分。
分片变更	DRDS 既支持计算层（DRDS）扩容（增加节点数或提升节点规格），也支持存储层在线分片变更，存储层分片变更可以通过增加分片数或者数据节点数来解决单表数据量过多和容量瓶颈等问题。计算层扩容对业务完全透明，存储层扩容对业务秒级影响。
分布式事务	DRDS 当前支持单机、FREE、XA 三种事务模型。 • 单机：不允许跨分片事务。 • FREE：跨分片事务 commit 时部分失败无法回滚，导致数据不一致。 • XA 分布式事务：两阶段提交，跨分片事务 commit 时部分失败会自动回滚，保证事务内数据一致性。
SQL 语法	高度兼容 MySQL 协议和语法。
读写分离	DRDS 的读写分离功能对应用透明，无需修改任何业务代码，将只读实例添加到 DRDS 即可。提升数据库处理能力，提高访问效率，轻松应对高并发的实时交易场景。
全局序列	DRDS 支持分布式全局唯一且有序递增的数字序列。满足业务在使用分布式数据库下对主键或者唯一键以及特定场景的需求。
Console 运维管理界面	DRDS 提供 Console 界面，可在线对 DRDS 实例、逻辑库等进行管理和维护。

1.4 产品规格

通用增强型实例是新推出的一系列性能高、计算能力稳定的实例规格，搭载英特尔® 至强® 可扩展处理器，配套高性能网络，综合性能及稳定性全面提升，满足对业务稳定性及计算性能要求较高的企业级应用诉求。

表 1-2 产品规格

规格类型	架构	CPU（核）	内存（GB）
通用增强型	X86	8	16
		16	32
		32	64

1.5 使用限制

1.5.1 网络访问使用限制

在使用分布式关系型数据库（简称 DRDS）过程中，对于网络访问存在一些使用限制。

- 用户申请的数据节点、部署应用程序的 ECS、DRDS 实例必须属于同一个 VPC。
- 当用户需要在自己电脑访问 DRDS 服务时，需要为 DRDS 开通 EIP 服务，然后通过 EIP 访问 DRDS。

1.5.2 MySQL 实例使用限制

在使用分布式关系型数据库（简称 DRDS）过程中，对于 MySQL 实例存在一些使用限制。

- 目前支持 5.7 及 8.0 系列版本的 MySQL 实例。
- DRDS 暂不支持 MySQL 实例配置 SSL 连接。
- 禁止 MySQL 实例开启区分大小写。
- 对已经被 DRDS 关联的 MySQL 实例进行修改配置等操作时，可能导致使用异常。修改后需要在 DRDS 的 DN 管理页面，单击“同步 DN 信息”，将修改的配置进行同步，保证功能可用性。

1.5.3 不支持的特性和使用限制

不支持的特性

- 不支持存储过程；
- 不支持触发器；
- 不支持视图；
- 不支持事件；
- 不支持自定义函数；
- 不支持外键约束、外键关联；
- 不支持全文索引和空间函数；
- 不支持临时表；
- 不支持 BEGIN…END、LOOP…END LOOP、REPEAT…UNTIL…END REPEAT、WHILE…DO…END WHILE 等复合语句；
- 不支持类似 IF ，WHILE 等流程控制类语句；
- 不支持 RESET、FLUSH 语句；
- 不支持 BINLOG 语句；
- 不支持 HANDLER 语句；
- 不支持 INSTALL/UNINSTALL PLUGIN 语句；

- 不支持非 ascii/latin1/binary/utf8/utf8mb4 的字符集；
- 不支持 SYS schema；
- 不支持 MySQL 追踪优化器；
- 不支持 X-Protocol；
- 不支持 CHECKSUM TABLE 语法；
- 不支持表维护语句，包括 ANALYZE/CHECK/CHECKSUM/OPTIMIZE/REPAIR TABLE；
- 不支持 session 变量赋值与查询，如 set @rowid=0;select @rowid:=@rowid+1,id from user；
- 不支持 SQL 语句中包含单行注释 '--' 或者多行（块）注释 '/.../'；
- 不完整支持系统变量查询，系统变量查询语句返回值为 RDS 实例相关变量值，而非 DRDS 引擎内相关变量值。例如 select @@autocommit 返回的值，并不代表 DRDS 当前事务状态；
- 不支持 SET Syntax 修改全局变量；
- 不支持 PARTITION 语法，建议不要使用 partition 表；
- 不支持 LOAD XML 语句；

不支持的运算符

- 不支持“:=”赋值运算符；
- 不支持“->”运算符；
- 不支持“->>”运算符；
- 暂不支持“<=>”运算符；
- 暂不支持“IS UNKNOWN”表达式；

不支持的函数

DRDS 计算层暂不支持如下函数，如果无法确认函数是否能下推到 RDS，请不要使用该函数。

- 不支持 XML 函数；
- 不支持 ANY_VALUE()函数；
- 不支持 ROW_COUNT()函数；
- 不支持 COMPRESS()函数；
- 不支持 SHA()函数；
- 不支持 SHA1()函数；
- 不支持 MD5()函数；
- 不支持 AES_ENCRYPT()函数；
- 不支持 AES_DECRYPT()函数；
- 不支持 JSON_OBJECTAGG()聚合函数；
- 不支持 JSON_ARRAYAGG()聚合函数；
- 不支持 STD()聚合函数；

- 不支持 STDDEV()聚合函数；
- 不支持 STDDEV_POP()聚合函数；
- 不支持 STDDEV_SAMP()聚合函数；
- 不支持 VAR_POP()聚合函数；
- 不支持 VAR_SAMP()聚合函数；
- 不支持 VARIANCE()聚合函数；

SQL 语法使用限制

SELECT

- 不支持 DISTINCTROW;
- 不支持[HIGH_PRIORITY]、[STRAIGHT_JOIN]、 [SQL_SMALL_RESULT]、[SQL_BIG_RESULT] 、[SQL_BUFFER_RESULT] 、[SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]等选项放在 DRDS 实例下面。
- 不支持 SELECT ... GROUP BY ... WITH ROLLUP 语句；
- 不支持 SELECT ... ORDER BY ... WITH ROLLUP 语句；
- 不支持 WITH 语句；
- 不支持窗口函数；
- SELECT FOR UPDATE 仅支持简单查询，不支持 join、group by、order by、limit 等语句。用于修饰 FOR UPDATE 的[NOWAIT | SKIP LOCKED]选项对于 DRDS 无效；
- 对于 UNION 中的每个 SELECT, DRDS 暂不支持使用多个同名的列。如下 SQL 的 SELECT 中存在重复的列名。

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

排序与 Limit

- LIMIT/OFFSET 参数支持范围为 0-2147483647；

聚合

- 不支持 group by 语句后添加 asc/desc 函数来实现排序语义；

说明

DRDS 自动忽略 group by 后的 asc/desc 关键字。MySQL 8.0.13 以下版本支持 group by 后添加 asc/desc 函数来实现排序语义，8.0.13 及以上版本已废弃该用法，使用时会报语法错误。推荐使用 order by 语句来保证排序语义。

子查询

- 不支持孙子和爷爷存在关联关系的子查询；
- 不支持 HAVING 子句中的子查询，JOIN ON 条件中的子查询；
- Derived Tables 必须拥有一个别名；
- Derived Tables 不可以成为 Correlated Subqueries，即不能包含子查询外部表的引用；

LOAD DATA 语法限制

- ESCAPED BY 只支持'\';
- 不支持 PARTITION (partition_name [, partition_name] ...);
- 不支持 LINES STARTING BY 'string';

INSERT 和 REPLACE

- 不支持 INSERT DELAYED...;
- 不支持不包含拆分字段的 INSERT;
- 暂不支持 PARTITION 语法，建议不要使用 partition 表;
- INSERT 操作不支持“datetime”字段取值 1582 年及之前年份;
- INSERT 不支持 ON DUPLICATE KEY UPDATE 关联子查询列;

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

示例 ON DUPLICATE KEY UPDATE 语句中引用了子查询列 c。

- INSERT 和 REPLACE 不支持拆分键值为 DEFAULT 关键字;

UPDATE 和 DELETE

- 不支持更新拆分键值为 DEFAULT 的关键字;
- 不支持在一个语句中对同一字段重复更新;
- 不支持关联更新拆分键;

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

示例中“name”为 tbl_1 的拆分键。

- 不支持通过 INSERT ON DUPLICATE KEY UPDATE 更新拆分键;
- 不支持自关联更新;

```
UPDATE tbl_1 a,tbl_1 b set a.tinyblob_col=concat(b.tinyblob_col,'aaabbb');
```

- 不支持不带关联条件的关联更新;

```
UPDATE tbl_3,tbl_4 SET tbl_3.varchar_col='dsgfdg';
```

- 关联更新不支持在目标列的赋值语句或表达式中引用其它目标列;

```
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;
```

- 对拆分字段的更新，将转换成 delete+insert 两个阶段操作，操作中间不保证其它涉及到这张表中的拆分字段值的查询语句的一致性;

DDL

- 库名不可修改，拆分字段的名称和类型都不可以变更;
- 不支持通过 SQL 直接创建、删除逻辑库;
- 不支持 FULL_TEXT 索引;
- 不支持 CREATE TABLE tblName AS SELECT stmt 语法;
- 不支持 CREATE TABLE tblName LIKE stmt 语法;
- 不支持单条语句中 DROP 多张表;
- DDL 语句不支持多语句;

- 广播表、拆分表不支持创建外键；
- 不支持创建以“_ddm”为前缀的表；
- 不支持创建 TEMPOPARY 类型的拆分表、广播表；
- create table 中的 unique key 只能保证物理表内唯一，无法保证全局唯一；

索引

- 不支持全局二级索引；
- 不支持全局唯一索引, unique key\primary key 无法保证全局唯一；

表回收站

- 不支持 hint；
- 不支持按逻辑库清除回收表；
- 不支持按逻辑表清除回收表；
- 表恢复后不保证全局唯一序列无缝衔接递增，只确保递增；
- 数据不支持分片变更；
- 不支持无限期保留副本；
- 不支持恢复到任意表名；
- 不支持不限量副本数；

事务

- 不支持 Savepoints；
- 不支持 XA 语法（DRDS 内部已经通过 XA 实现了分布式事务，不需要用户层再处理这个语义）；
- 不支持自定义事务隔离级别，目前 DRDS 只支持 READ COMMITTED 隔离级别。考虑到兼容性因素，对于设置数据库隔离级别的语句（如 SET GLOBAL TRANSACTION ISOLATION LEVEL REPEATABLE READ），DRDS 不会报错，但会忽略对事务隔离级别的修改；
- 不支持设置事务为只读（START TRANSACTION READ ONLY），考虑到兼容性因素，DRDS 会将只读事务的开启自动转换为开启读写事务；

权限

- 不支持列级权限；
- 不支持子程序层级权限；

数据库管理语句

- 不支持 SHOW TRIGGERS 语法；
- 不支持 SHOW PROFILES、SHOW ERRORS、show warnings 等多数运维 SHOW 语句；
- 下列的 SHOW 指令会随机发到某个物理分片，每个物理分片如果在不同的 RDS for MySQL 实例上，查得的变量或者表信息可能不同：

- SHOW TABLE STATUS；
- SHOW VARIABLES Syntax；
- SHOW WARNINGS Syntax 不支持 LIMIT/COUNT 的组合；
- SHOW ERRORS Syntax 不支持 LIMIT/COUNT 的组合；

INFORMATION_SCHEMA

- 仅支持 SCHEMATA、TABLES、COLUMNS、STATISTICS、PARTITIONS 的简单查询（没有子查询、JOIN、聚合函数、ORDER BY、 LIMIT);

1.5.4 高危操作提示

在您使用 DRDS 过程中，请不要进行以下高危操作：

- 为避免系统表、元数据等信息被误清理，用户尽量避免直连 DN 节点进行数据操作。
- 为避免元数据丢失，用户请不要清理 DRDS 系统表（如：TBL_DRDS_TABLE、MYCAT_SEQUENCE）。

1.6 区域和可用区

什么是区域、可用区？

我们用区域和可用区来描述数据中心的位置，您可以在特定的区域、可用区创建资源。

- 区域（Region）指物理的数据中心。每个区域完全独立，这样可以实现最大程度的容错能力和稳定性。资源创建成功后不能更换区域。
- 可用区（AZ，Availability Zone）是同一区域内，电力和网络互相隔离的物理区域，一个可用区不受其他可用区故障的影响。一个区域内可以有多个可用区，不同可用区之间物理隔离，但内网互通，既保障了可用区的独立性，又提供了低价、低时延的网络连接。

区域和可用分区阐明了区域和可用区之间的关系。

图 1-2 区域和可用区

如何选择区域？

建议就近选择靠近您或者您的目标用户的区域，这样可以减少网络时延，提高访问速度。

如何选择可用区？

是否将资源放在同一可用区内，主要取决于您对容灾能力和网络时延的要求。

- 如果您的应用需要较高的容灾能力，建议您将资源部署在同一区域的不同可用区内。
- 如果您的应用要求实例之间的网络延时较低，则建议您将资源创建在同一可用区内。

1.7 应用场景

分布式关系型数据库（简称 DRDS）是一个面向 OLTP 业务的分布式关系型数据库访问服务，适用于各行业数据库应用。

尤其适用于大规模的数据存储与高并发访问的行业应用，如互联网应用、物联网数据、高性价比数据库解决方案等应用场景。

- **互联网应用**

  电商、金融、O2O、零售、社交应用等行业，普遍存在用户基数大、营销活动频繁、核心交易系统数据库响应日益变慢的问题，制约业务发展。DRDS 提供线性水平扩展能力，能够实时提升数据库处理能力，提高访问效率，轻松应对高并发的实时交易场景。

- **物联网数据**

  在工业监控和远程控制、智慧城市的延展、智能家居、车联网等物联网场景下。传感监控设备多，采样频率高，数据规模大，会产生超过单机数据库存储能力极限的数据，造成数据库容量瓶颈。DRDS 提供的容量水平扩展能力，可以有效的帮助用户低成本的存储海量数据。

- **高性价比数据库解决方案**

  政务机构、大型企业、银行等行业为了支持大规模数据存储和高并发数据库访问，传统方案需要强依赖小型机和高端存储等高成本的商业解决方案。DRDS 利用普通服务器进行集群部署，提供与传统商业解决方案相同甚至更高的处理能力。

2 快速入门

2.1 概述

操作场景

本章以 DRDS 实例如何关联数据节点（RDS for MySQL 实例）为例介绍 DRDS 的使用。

使用流程

步骤一：申请分布式关系型数据库实例及 RDS for MySQL 实例

步骤二：创建逻辑库并关联 RDS for MySQL 实例

步骤三：创建 DRDS 帐号

步骤四：连接 DRDS 逻辑库

图 2-1 DRDS 使用流程

2.2 步骤一：申请分布式关系型数据库实例及 RDS for MySQL 实例

操作步骤

步骤 1 在云控制台登录页，输入帐号及密码，登录云控制台。

步骤 2 单击管理左上角的 ，选择对应 Region。

步骤 3 在管理控制页面单击服务列表，选择“数据库 > 分布式关系型数据库 DRDS ”，进入 DRDS 管理控制台。

步骤 4 在实例管理页面，单击页面右上方的申请分布式数据库实例。

步骤 5 在申请分布式关系型数据库实例页面，按需设置实例相关信息。

表 2-1 参数说明

参数	说明
可用区	可选择的可用区。 当前部分区域支持跨可用区部署，以增强 DRDS 实例高可用性。 如果您的实例需要跨可用区部署，可以选择多个可用区，DRDS 实例的节点将部署在不同的可用区中。 **说明** 跨可用区部署会产生一定的网络时延，建议将应用程序和数据库服务（DRDS、RDS）配置在相同可用区，减少网络时延。
实例名称	DRDS 实例的名称。 • 名称不能为空。 • 只能以英文字母开头。 • 长度为 4 到 64 位的字符串。 • 可包含英文字母、数字和中划线（-）。 • 不能包含其他特殊字符。
节点规格	DRDS 实例的规格，“通用增强型”。 **说明** 为了使所申请的 DRDS 实例能更好地满足应用需求，您需要先评估应用所需的计算能力和存储能力，结合业务类型及服务规模，选择合适的实例规格，主要包括：CPU、内存。
节点个数	实例所含有节点个数，最多支持 32 个节点数。默认 2 个节点，保障高可用。
虚拟私有云	DRDS 实例所在的虚拟专用网络，可对不同业务进行网络隔离，方便地管理、配置内部网络，进行安全、快捷的网络变更。 单击“查看虚拟私有云”，系统跳转到虚拟私有云界面，可以查看相应的虚拟私有云，以及安全组的出方向规则和入方向规则。 **说明** 创建 DRDS 实例选择的虚拟私有云要与 RDS for MySQL 实例保持一致。 DRDS 实例必须与应用程序、RDS for MySQL 实例处于相同的 VPC，以保证网络连通。
设置密码	• 现在设置（默认），如果您选择创建实例时设置，请填写帐户对应的密码。 • 创建后设置，系统不会为您设置初始密码。您在使用管理员帐户登录 DRDS 实例前，请先通过新增管理员账户的方式设置帐号和密码，再进行登录。 实例创建成功后重置密码，请参见管理员账户。 实例创建成功后新增管理员账户，请参见管理员账户。

参数	说明
管理员账户名	管理员账户默认为 root，您也可自行设置。设置的用户名长度为 1-32 个字符，必须以字母开头，可以包含字母，数字、下划线，不能包含其它特殊字符。 注意 • 管理员账户创建成功后，不支持修改账户名。 • 此账户拥有最高的 superuser 权限，可以通过 DAL (grant, revoke) 对其他用户的权限进行修改，默认具备所有库表(包括还未创建的逻辑库)的读写权限，一经创建无法删除。
管理员密码	所设置的密码长度为 8~32 个字符，至少包含大写字母、小写字母、数字、特殊字符三种字符的组合，其中允许输入~!@#%^*-_=+?特殊字符。请不要使用简单、强度不够、轻易被猜测的弱口令作为密码。 请妥善保管您的密码，因为系统将无法获取您的密码信息。
确认密码	必须和管理员密码相同。
子网	包括子网名称与子网 IP 地址段。
安全组	已创建的安全组。 建议 DRDS 实例与应用程序、RDS for MySQL 实例选择相同的安全组，三者网络访问不受限制。如果选择了不同的安全组，请注意添加安全组访问规则，开通网络访问。
参数模板	您可以选择已有参数；同时支持单击查看参数模板，在参数模板页面，进行设置参数信息。

步骤 6 实例信息设置完成后，单击页面下方“立即申请”。

步骤 7 确认配置信息，单击“提交”。

步骤 8 实例创建成功后，用户可以在“实例管理”页面对其进行查看和管理。

数据库端口默认为 5066，实例创建成功后可修改。

具体请参见修改数据库端口。

步骤 9 进入 RDS 管理控制台，单击右上角“申请数据库实例”进入详情页面选择相关配置信息后单击立即创建。

⚠ 注意

RDS for MySQL 实例的虚拟私有云（VPC）和子网必须和 DRDS 实例保持一致。

步骤 10 确认配置信息，单击“提交”。等待 1-3 分钟实例即可创建成功。

----结束

2.3 步骤二：创建逻辑库并关联 RDS for MySQL 实例

创建 DRDS 逻辑库有两个入口，以服务列表页为例。

图 2-2 服务列表页进入

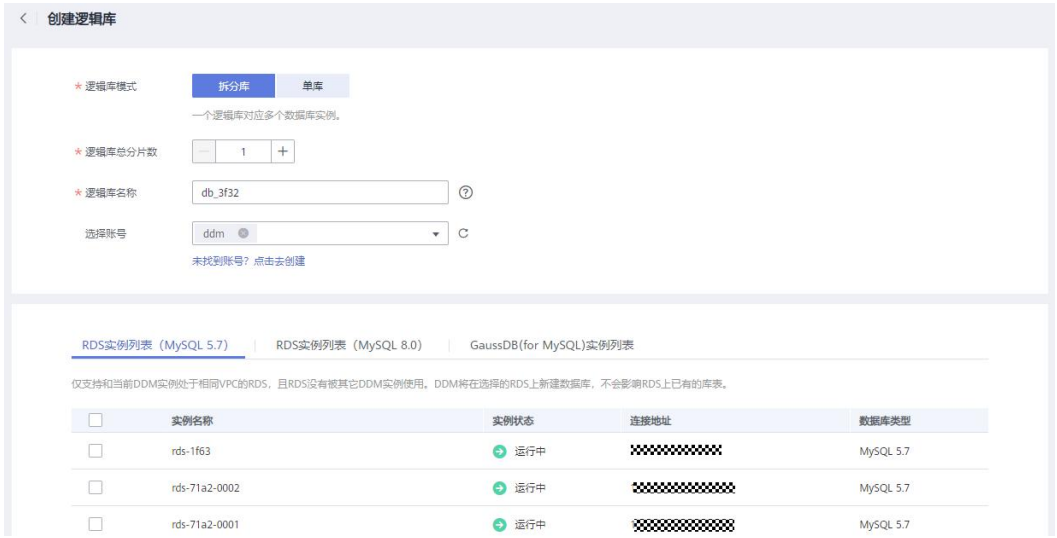
图 2-3 实例详情-逻辑库列表页进入

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，在目标实例操作栏单击“创建逻辑库”。

步骤 2 在创建逻辑库页面，选择“逻辑库模式”、“逻辑库总分片数”，填写“逻辑库名称”，并选择要关联的 DRDS 帐号、要关联的实例，单击“下一步”。

图 2-4 创建逻辑库



说明

- 逻辑库模式：
- 拆分库：一个逻辑库可关联多个数据节点，分片数均分在这些实例上。
- 单库：一个逻辑库仅关联一个数据节点，在该实例上仅创建一个分片。

- 逻辑库名称：长度为 2-48 个字符，以小写字母开头且仅支持小写，可以包含小写字母、数字、下划线，不能包含其它特殊字符。
- 逻辑库总分片数：当前创建的逻辑库的总分片数。单个数据节点物理分片数不超过 64。如果因业务需要分片数超过 64，请联系 DRDS 技术人员。

步骤 3 进行数据库连接验证，输入对应数据库密码，单击“测试连接”。

图 2-5 数据实例连接验证

步骤 4 测试通过后，单击页面下方的“完成”。

----结束

2.4 步骤三：创建 DRDS 帐号

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤 2 在左侧导航栏选择“帐号管理”，进入帐号管理页面。

步骤 3 在帐号管理页面单击“创建 DRDS 帐号”，在弹窗中填选帐号参数信息。

步骤 4

步骤 5 信息填写完成，单击“确定”即可创建成功。

----结束

2.5 步骤四：连接 DRDS 逻辑库

用户申请 DRDS 实例后，可以使用 Navicat 等客户端连接 DRDS 实例，也可以使用 CLI 或 JDBC 驱动连接 DRDS 实例中的目标逻辑库。

本章节指导用户如何连接 DRDS 实例或逻辑库。

准备工作

连接 DRDS 实例或逻辑库前，需要获取 DRDS 实例或逻辑库的连接地址。

获取 DRDS 逻辑库连接地址

步骤 1 登录 DRDS 管理控制台。

步骤 2 在左侧选择“数据库> 分布式关系型数据库”，进入实例管理页面。

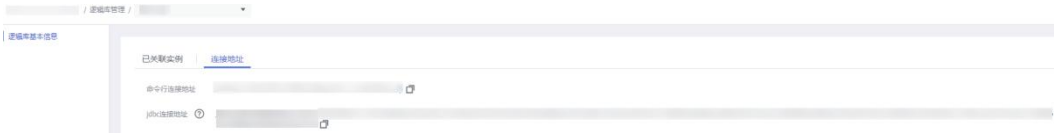
步骤 3 单击左侧菜单栏的“实例管理”，在实例管理界面单击 DRDS 实例名称，进入实例基本信息页。

步骤 4 在左侧导航栏选择“逻辑库管理”。

步骤 5 单击逻辑库名称进入逻辑库基本信息页面。

步骤 6 在“连接地址栏”获取“命令行连接地址”和“jdbc 连接地址”。

图 2-6 逻辑库连接地址



说明

- 如果 DRDS 实例有多个节点，页面内网地址只提供了一个 IP 地址，这是因为 DRDS 集群支持了负载均衡。可通过连接该地址连接到 DRDS 集群，达到负载均衡效果。
- 有部分历史实例的内网地址提供了多个 IP 地址，是因为 DRDS 集群未支持负载均衡，可通过 jdbc 连接串达到负载均衡效果。
- 如果创建了只读组，每个组都会分配一个负载均衡连接地址，进行业务隔离。

----结束

连接方法概述

方法一：Navicat 客户端连接 DRDS 实例。

方法二：MySQL 命令行连接 DRDS 逻辑库。

方法三：JDBC 驱动连接 DRDS 逻辑库。

方法四：Console 界面连接 DRDS 实例。

说明

1. 为保证系统安全，请使用与 DRDS 实例处于同一 VPC 的弹性云服务器。
2. 弹性云服务器已安装 MySQL 客户端或已配置 MySQL 连接驱动。
3. 连接 DRDS 实例前，需要先在工具中配置 DRDS 实例信息。

Navicat 客户端连接 DRDS 实例

步骤 1 登录分布式关系型数据库服务，单击需要连接的 DRDS 实例名称，进入实例基本信息页面。

步骤 2 在“实例信息”模块的弹性公网 IP 单击“绑定”。绑定已申请的公网 IP。

步骤 3 在 DRDS 管理控制台左侧选择虚拟私有云图标。单击“访问控制>安全组”

步骤 4 在安全组界面，单击操作列的“配置规则”，进入安全组详情界面。在安全组详情界面，单击“添加规则”，弹出添加规则窗口。根据界面提示配置安全组规则，设置完成后单击“确定”即可。

说明

绑定弹性公网 IP 后，建议您在内网安全组中设置严格的出入规则，以加强数据库安全性。

步骤 5 打开 Navicat 客户端，单击“连接”。在新建连接窗口中填写主机 IP 地址（弹性公网 IP 地址）、用户名和密码（DRDS 帐号、密码）。

说明

Navicat 客户端推荐使用版本为 Navicat12。

步骤 6 单击“连接测试”，若显示连接成功，单击“确定”，等待 1-2 分钟即可连接成功。连接失败会直接弹出失败原因，请修改后重试。

----结束

说明

通过其他可视化的 MySQL 工具（例如 Workbench）连接 DRDS 实例的操作与此章基本一致，不做详细描述。

MySQL 命令行连接 DRDS 逻辑库

步骤 1 登录弹性云服务器，打开命令行工具，输入以下命令。

```
mysql -h ${DDM_SERVER_ADDRESS} -P${DDM_SERVER_PORT} -u${DDM_USER} -p [-
D${DDM_DBNAME}] [--default-character-set=utf8][--default_auth=mysql_native_password]
```

表 2-2 参数说明

参数示例	参数填写说明	参数举例
DDM_SERVER_ADDRESS	DRDS 实例所在 IP 地址。	192.168.0.200
DDM_SERVER_PORT	DRDS 实例连接端口。	5066
DDM_USER	DRDS 实例帐号。	dbuser01
DDM_DBNAME	DRDS 实例逻辑库名，选填。	-
default-character-set=utf8	指定字符编码为 UTF-8，选填。 当 MySQL 连接编码和实际编码不一致，导致 DRDS 解析出现乱码时请配置该参数。	-
default_auth=mysql_native_password	默认使用密码认证插件。	-

说明

假如您使用了 MySQL 8.0 的客户端，需要增加 default_auth=mysql_native_password 参数。

步骤 2 下面为 Windows 服务器命令行窗口中使用表中举例参数 MySQL 命令连接服务器的回显情况。

```
C:\Users\testDDM>mysql -h192.168.0.200 -P5066 -Ddb_5133 -udbuser01 -p
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ;or \g.
Your MySQL connection id is 5
Server version: 5.6.29

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

----结束

JDBC 驱动连接 DRDS 逻辑库

步骤 1 加载驱动程序。

```
Class.forname(com.mysql.jdbc.Driver);
```

说明

建议 JDBC 驱动版本为 5.1.35~5.1.45。

步骤 2 打开数据库连接。

```
String username = "dbuser01" ;
String password = "xxxxxx" ;
String url = "jdbc:mysql://192.168.0.200:5066/db_5133";
Connection con = DriverManager.getConnection(url , username , password);
```

步骤 3 创建 Statement 对象。

```
Statement stmt = con.createStatement();
```

步骤 4 执行 SQL 语句。

```
ResultSet rs = stmt.executeQuery("select now() as Systemtime");
con.close();
```

步骤 5 （可选）优化代码。

```
loadBalanceAutoCommitStatementThreshold=5&loadBalanceHostRemovalGracePeriod=15000&l
oadBalanceBlacklistTimeout=60000&loadBalancePingTimeout=5000&retriesAllDown=10&conn
ectTimeout=10000";
```

说明

- **loadBalanceAutoCommitStatementThreshold** 和 **retriesAllDown** 参数必须按照以上样例进行配置，否则在连接切换时可能进入死循环，导致栈溢出。
- loadBalanceAutoCommitStatementThreshold：表示执行多少个语句后重新选择连接。
- loadBalanceHostRemovalGracePeriod：设置宽限期，以等待主机从负载均衡连接中移除，当主机当前是活动主机时释放主机。
- loadBalanceBlacklistTimeout：通过控制服务器在全局黑名单中的生存时间，检查不可用服务器之间的时间间隔（以毫秒为单位）。
- loadBalancePingTimeout：使用负载平衡连接时，等待每个负载平衡物理连接 ping 响应的时间（以毫秒为单位）。
- retriesAllDown：当所有的连接地址都无法连接时，轮询重试的最大次数。重试次数达到阈值仍然无法获取有效连接，将会抛出 SQLException
- connectTimeout：套接字连接的超时（毫秒），其中 0 表示没有超时。只能在 JDK-1.4 或更高版本上工作。默认为“0”。

----结束

3 功能总览

分布式关系型数据库（Distribute Relational Database Service，简称 DRDS），兼容 MySQL 协议，专注于解决数据库分布式扩展问题，突破传统数据库的容量和性能瓶颈，实现海量数据高并发访问。

分布式关系型数据库支持的功能如表 3-1 所示。

表 3-1 分布式关系型数据库服务功能列表

功能分类	功能描述
实例管理	包括实例创建、实例规格变更、实例删除、实例重启、实例续费与退订等功能。具体使用方法请参考实例管理。
备份管理	包括数据的恢复到新实例、Metadata 恢复等功能。具体使用方法请参考备份管理。

功能分类	功能描述
参数模板管理	包括参数模板的创建、编辑、比较、复制、应用等功能。具体使用方法请参考参数模板管理。
任务中心	该模块可以查看用户在控制台上提交的异步任务的执行进度和状态。具体使用方法请参考任务中心。
逻辑库管理	包括逻辑库的创建、导出、导入、删除等功能。具体使用方法请参考逻辑库管理。
逻辑库分片变更	可以通过增加分片或者数据节点进行存储层分片变更，具体使用方法请参考分片变更。
帐号管理	包括 DRDS 帐号的创建、修改、删除和重置密码等功能。具体使用方法请参考账号管理。
监控管理	包括监控指标一览表和如何查看监控指标，具体使用方法请参考监控管理。
SQL 语法管理	包括 DDL 语法、DML 语法、全局序列、实用 SQL 语句和多种拆分算法的使用等功能，具体使用方法请参考 SQL 语法。

4 实例管理

4.1 规格变更

前提条件

- 成功登录分布式关系型数据库服务控制台。
- 实例状态为“运行中”。

须知

节点规格变更期间服务会短暂中断，建议在业务低峰时变更。

操作步骤

须知

若开启了只读业务隔离特性，即创建了只读组，规格变更功能入口将移动到组列表的操作列。

步骤 1 在分布式关系型数据库服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。单击“规格变更”。

步骤 2 在分布式关系型数据库变更规格页面，按需选择实例规格。

步骤 3 确认变更信息，根据所选实例的计费模式进行后续操作。

- “按需计费”模式，单击“提交”。
- “包年/包月”模式，单击“去支付”。

步骤 4 确认配置信息，单击“提交”。

步骤 5 返回实例管理列表页面，查看当前实例状态为“规格变更中”，也可在任务中心查看变更任务。

说明

- 一旦执行变更操作后不可撤销。如果需要修改，需要在当前变更操作结束后重新提交变更操作。
- 规格变更支持升高规格和降低规格两种。

----结束

4.2 计算节点扩容

操作场景

随着业务数据的增加，为了提高实例业务稳定性，您可对 DRDS 实例节点进行扩容。

说明

- 请在业务低峰时间段进行节点扩容操作。
- 请确保实例关联的数据节点状态正常并且没有进行其他操作。
- 一个 DRDS 实例最多支持 32 个节点。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。单击“节点扩容”。

步骤 2 在分布式关系型数据库节点扩容页面，您可查看当前规格，在选择节点模块选择可用区，并添加节点。

步骤 3 设置完节点数，单击页面下方的“下一步”。

步骤 4 在规格确认页面，若您需要重新修改节点数，请单击“上一步”，再次确认所选规格无误后，单击页面下方的“提交”，提交节点扩容任务。

----结束

4.3 实例续费

前提条件

- 成功登录分布式关系型数据库服务控制台。
- 当前实例计费模式为“包年/包月”，单击“去支付”。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，“操作”列选择“更多”>“续费”。

步骤 2 在续费页面，按需选择续费时长。

步骤 3 核对信息无误后，单击“确认付款”进行支付。

----结束

4.4 实例重启

前提条件

- 成功登录分布式关系型数据库服务控制台。
- 当前实例状态为“运行中”。

须知

实例重启期间服务不可用且操作无法撤销，请谨慎操作。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，在目标实例操作栏，选择“更多”>“重启实例”。

步骤 2 在弹出确认窗口中，单击“是”。

步骤 3 在实例管理列表页面，等待实例重启成功。

----结束

4.5 实例退订

前提条件

- 成功登录分布式关系型数据库服务控制台。
- 当前实例计费模式为“包年/包月”。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，在目标实例操作栏，选择“更多”>“退订”。

步骤 2 在退订操作提示窗中，单击“是”。

系统自动跳转至“费用中心-退订管理”页面。

步骤 3 在“订单管理/退订管理”页面选择“退订使用中的资源”页签。

步骤 4 勾选需退订实例。在操作栏单击“退订资源”。

步骤 5 再次核实待退订实例信息，填写退订原因。

步骤 6 在退订资源页面，勾选我已确认本次退订金额和相关费用，单击下方“退订”。

说明

资源退订后，数据将立即删除且无法恢复。请确认数据完成备份或不再使用。

步骤 7 在退订弹窗中，单击“是”，即可退订成功。

----结束

4.6 实例删除

前提条件

成功登录分布式关系型数据库服务控制台。

须知

删除操作无法恢复，请谨慎操作。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，在目标实例操作栏，选择“更多”>“删除实例”。

步骤 2 在弹出确认窗口中，单击“是”。

说明

- 如需删除挂载于 DRDS 上的数据节点的数据，请勾选“删除存储层实例数据”。

----结束

4.7 参数管理

操作场景

为了确保 DRDS 服务发挥出最优性能，您可以根据自己的业务情况对 DRDS 实例的运行参数进行配置。

前提条件

已有 DRDS 实例，且实例状态处于“运行中”。

操作步骤

步骤 1 输入账户名和密码登录分布式关系型数据库服务控制台。

步骤 2 单击左侧菜单栏的“实例管理”，进入实例管理页面。

步骤 3 在实例管理页面单击实例名称，进入实例信息详情页。

步骤 4 在该页面单击“参数管理”。您可以根据需要修改对应参数。

图 4-1 参数管理页面

表 4-1 DRDS 实例配置参数说明

参数名称	缺省值	取值范围	参数说明
bind_table	-	格式要求：[{tb.col1,tb2.col2},{tb.col2,tb3.col1},...]该参数要以“表名.列名”的形式出现且可以多对同时出现。 版本要求： DRDS 2.3.2.7 版本及以上。	用于描述多个拆分表的内在数据关联性，用于告知优化器在处理 join 时，把 join 下推到 MySQL 层执行。参数举例请详见表下的说明。
character_set_server	utf8mb4	gbk、utf8、utf8mb4	DRDS 服务端字符集，如果需要存储 emoji 表情符号，请选择 utf8mb4,并设置 RDS 字符集也为 utf8mb4。
collation_server	utf8mb4_unicode_ci	utf8mb4_unicode_ci、utf8mb4_bin、utf8mb4_general_ci	DRDS 服务端字符序。
concurrent_execution_level	DATA_NODE	RDS_INSTANCE、DATA_NODE、PHY_TABLE	逻辑表扫描时的分片并发执行级别: DATA_NODE: 分库间并行扫描，同一分库内各分片串行扫描;RDS_INSTANCE: RDS 实例间并行扫描，同一 RDS 实例内各分片串行扫描;PHY_TABLE: 各物理分片全部并行扫描。

参数名称	缺省值	取值范围	参数说明
connection_idle_timeout	28800	60~86400 (s)	服务器关闭连接之前等待连接活动的秒数，以秒为单位，默认值 28800，表示服务器关闭连接之前等待 28800 秒后，关闭连接。
contains_shard_key	OFF	OFF、ON	是否强制 select，update，delete 语句中过滤条件中包含拆分字段。
ddl_precheck_mdl_threshold_time	120	1~3600	DDL 预检查中 MDL 锁持有时长阈值， 以秒为单位，默认值 120。
enable_table_recycle	OFF	OFF、ON	ON: 开启表回收站。 OFF: 关闭表回收站。
long_query_time	1	0.01~10 (s)	记录慢查询的最小秒数,以秒为单位，默认值为 1，表示如果 sql 执行大于等于 1 秒就定义为慢 sql。
max_allowed_packet	1073741824	1024~1073741824	包或任何生成的中间字符串的最大值。包缓冲区初始化为 net_buffer_length 字节，但需要时可以增长到 max_allowed_packet 字节。该值默认很小，以捕获大的（可能是错误的）数据包。该值必须设置为 1024 的倍数。
max_backend_connections	0	0~10000000	允许每个 DRDS 节点同时连接 RDS 的最大客户端总数。0 为默认值标识符,实际值等于(RDS 的最大连接数-20)/DRDS 节点数。

参数名称	缺省值	取值范围	参数说明
max_connections	20000	10~40000	允许同时连接的客户端总数。 此参数需要结合数据节点的规格及处理能力配置合适的值。连接数过多，可能造成连接等待，影响性能。DRDS 的连接数消耗与分片数量和 SQL 设计等因素相关。 例如：SQL 带拆分键时，1 个 DRDS 连接同时消耗后面 1 个数据节点连接；SQL 不带拆分键时，假设分片个数为 N，那么会消耗 N 个数据节点连接。 因此，SQL 合理设计且 DRDS 和数据节点的处理能力不成为瓶颈的前提，DRDS 最大连接数可以配置成略小于“后端数据节点的数量 * 单个数据节点支持的最大连接数”。 建议根据自己的业务进行实际压测，配置合理的值。
min_backend_connections	10	0~10000000	允许每个 DRDS 节点同时连接 RDS 的最小客户端总数。默认值为 10。
seconds_behind_master	30	0~7200	主从 rds 节点延迟时间阈值，以秒为单位，默认值为 30，表示主 rds 与从 rds 之间的数据同步时间值不能超过 30 秒，如果超过 30s，读数据指令就不走当前读节点。
sql_execute_timeout	28800	100~28800 (s)	SQL 执行超时秒数，以秒为单位，默认值 28800，表示 sql 执行大于等于 28800 秒超时。
temp_table_size_limit	1000000	500000~2000000000	临时表大小。
transaction_policy	XA	XA、FREE、NO_DTX	XA：XA 事务，保证原子性，保证可见性；FREE：允许多写，不保证原子性，无性能损耗；NO_DTX：单分片事务。
transfer_hash_to_mod_hash	OFF	OFF、ON	创表时是否将 hash 算法转为 mod_hash 算法。
ultimate_optimize	ON	OFF、ON	是否需要根据参数值来优化 SQL 执行计划。

参数名称	缺省值	取值范围	参数说明
force_read_master_in_transaction	OFF	OFF、ON	事务中涉及的 SQL 语句是否强制从主节点读取。 **注意** 3.0.9 版本开始支持此参数。如果此开关被打开后，版本降级为 3.0.9 以下，之后再升级为 3.0.9 或以上时，此开关配置保持打开状态。

DRDS 默认支持修改以上实例参数，特殊场景（如数据迁移）下如需修改更多实例参数请联系技术支持人员协助处理。

参数举例：

图 4-2 未使用 bind_table 结果展示

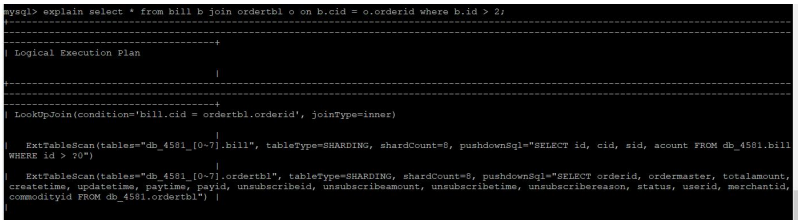
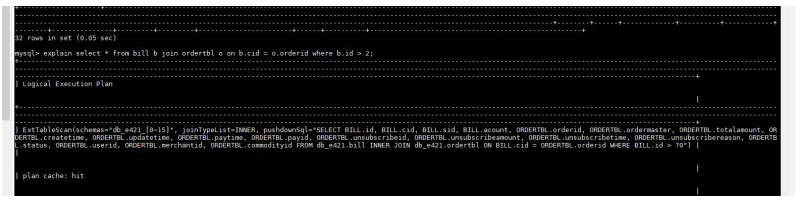


图 4-3 使用 bind_table 结果展示



步骤 5 确认无误后，单击“保存”，并在弹框中单击“是”完成参数修改。

图 4-4 参数确认页面

说明

- 修改配置参数可能影响应用访问 DRDS 实例，请谨慎操作。
- 修改参数命令下发成功后，预计需要 20~60 秒生效，请耐心等待。

----结束

4.8 读写分离

概述

DRDS 读写分离功能可以将只读查询的流量按比例分摊至下挂存储节点的主实例和只读实例，从而减轻主实例的工作负担，保障读写事务的性能。此功能对应用透明，业务代码无需改造，只需要在控制台中设置主实例和只读实例的读权重，即可实现将读流量按照自定义权重分流到主实例和只读实例上，写流量不受影响，默认会分流到主实例上。一般来说该比例的设置需结合业务实际特点以及存储节点实际负载进行设置。

只读实例上的数据是从主实例上异步复制而来，所以存在毫秒级的延迟。如果只读查询对数据实时性要求不高（容忍亚秒级可见性延迟）且只读查询的开销较大并对业务核心读写事务有一定影响，设置主实例和只读实例的权重为 0:100，即所有只读查询均由只读实例承担，最大程度保证主实例性能。对于其他场景，建议结合实际情况酌情调整。

若 select 语句带有 hint 或者在事务中做了数据修改的 select 语句，读请求都会下发主实例执行。

如果存储节点主实例故障，此时只读实例上 Seconds_Behind_Master=NULL，只读查询仍会下发到主实例执行，需要尽快恢复主实例。

前提条件

- 已申请 DRDS 实例和带只读实例的数据节点。
- 已经创建好逻辑库。

操作步骤

步骤 1 登录 DRDS 控制台，选择目标 DRDS 实例，进入实例基本信息页面。

步骤 2 单击选择“DN 管理”模块。

步骤 3 设置实例的读写权重。

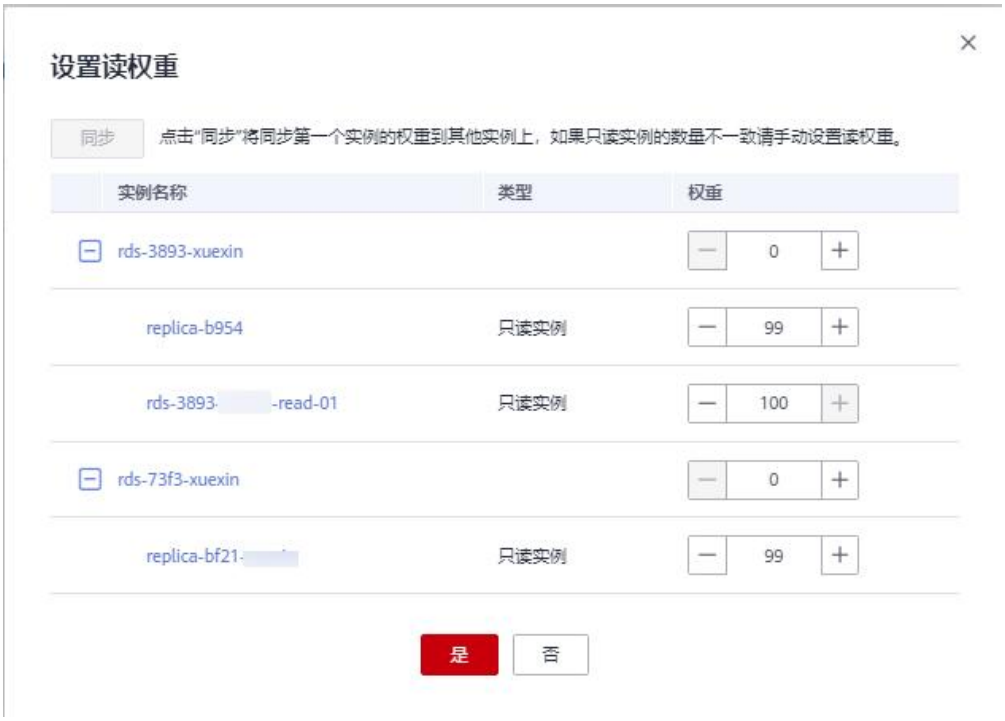
- 如果需要设置多个实例的读写权重，可点击实例上方的“设置读权重”来进行批量设置。如图 4-8 所示。

  在批量设置的弹窗中，“同步”功能可以用来将第一个实例的读权重设置同步到其他的实例上，如图 4-9 所示。此操作需满足所有实例的只读实例数量一致才可以实行。如果有实例的只读实例数量与其他实例不一致，则无法使用“同步”功能，需手动设置各个实例的读权重，如图 4-10 所示。

图 4-5 批量设置读权重

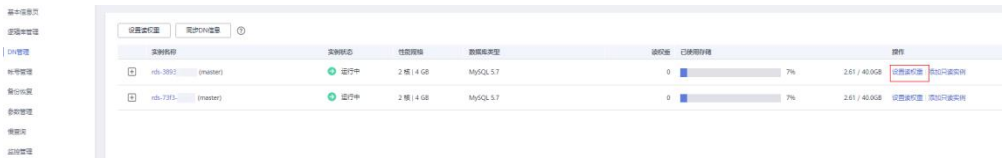
图 4-6 同步第一个实例的读权重

图 4-7 手动设置权重



- 如果需要设置单个实例的读写权重，可点击操作栏的“设置读权重”来进行设置。

图 4-8 单个设置读权重



步骤 4 设置读权重命令下发成功提示。

图 4-9 设置读权重命令下发成功

----结束

4.9 设置参数模板

前提条件

成功登录分布式关系型数据库服务控制台。

选择参数模板

步骤 1 在分布式关系型数据库服务，实例管理列表页面，在目标实例操作栏，选择“更多”>“设置参数模板”。

系统自动弹出设置参数模板窗口。

步骤 2 选择需要设置的参数模板后。单击“确定”。

图 4-10 设置参数模板

----结束

5 连接管理

5.1 修改数据库端口

操作场景

分布式关系型数据库服务支持修改数据库实例的端口，修改后会重启数据库实例。

操作步骤

步骤 1 在分布式关系型数据库服务“实例管理”页面，选择指定的实例，单击实例名称。

步骤 2 在基本信息页面，在“网络信息”模块的“数据库端口”处，单击，修改数据库端口。

DRDS 实例的默认端口为 5066，设置范围为 1025~65534，其中 1033、7009、8888、12017 被 DRDS 系统占用，不可设置。

- 单击，提交修改。
  - 在弹出框中，单击“是”，提交修改。
    修改实例的数据库端口，需要重启实例。
  - 在弹出框中，单击“否”，取消本次修改。
- 单击，取消修改。

步骤 3 在实例的基本信息页面，查看修改结果。

----结束

5.2 修改实例安全组

操作场景

分布式关系型数据库服务支持修改数据库实例的安全组。

操作步骤

步骤 1 在分布式关系型数据库服务“实例管理”页面，选择指定的实例，单击实例名称。

步骤 2 在基本信息页面，在“网络信息”模块的“安全组”处，单击，选择对应的安全组。

- 单击，提交修改。

- 单击，取消修改。

步骤 3 在实例的基本信息页面，查看修改结果。

----结束

6 参数模板管理

6.1 创建参数模板

您可以使用参数模板中的参数来管理 DRDS 实例中的参数配置。参数模板就像是 DRDS 参数配置的容器，这些值可应用于一个或多个 DRDS 实例。

如果您在创建 DRDS 实例时未指定客户创建的参数模板，系统将会为您的实例适配默认的参数模板。该默认参数模板包含多个系统默认值，具体根据计算等级、实例的分配存储空间而定。您无法修改默认参数模板的参数设置，您必须创建自己的参数模板才能更改参数设置的默认值。

如果您想使用您自己的参数模板，只需创建一个新的参数模板，创建实例的时候选择该参数模板，如果是在创建实例后有这个需求，可以重新应用该参数模板，请参见应用参数模板。

若您已成功创建参数模板，并且想在新的参数模板中包含该组中的大部分自定义参数和值时，复制参数模板是一个方便的解决方案，请参见复制参数模板。

以下是您在使用参数模板中的参数时应了解的几个要点：

- 当您修改当前实例的参数模板并保存后，仅应用于当前实例，不会对其他实例造成影响。
- 当您更改参数并保存参数模板时，参数更改将在您应用到实例后，手动重启 DRDS 实例后生效。
- 在参数模板内设置参数不恰当可能会产生意外的不利影响，包括性能降低和系统不稳定。修改参数时应始终保持谨慎，且修改参数模板前要备份数据。将参数模板更改应用于使用 DRDS 实例前，您应当在测试实例上试用这些参数模板设置更改。

操作步骤

步骤 1 登录管理控制台。

步骤 2 单击管理控制台左上角的 ，选择区域和项目。

步骤 3 在页面左上角单击 ，选择“数据库 > 分布式关系型数据库 ”。进入分布式关系型数据库信息页面。

步骤 4 在“参数模板”页面，单击“创建参数模板”。

步骤 5 输入参数模板名称并添加对该参数模板的描述，单击“确定”，创建参数模板。

- 参数模板名称长度在1~64个字符之间，区分大小写，可包含字母、数字、中划线、下划线或句点，不能包含其他特殊字符。
- 参数模板的描述长度不能超过256个字符，且不能包含回车和>!<"&'=特殊字符。

说明

每个用户最多可以创建100个DRDS参数模板。

----结束

6.2 编辑参数模板

为确保分布式关系型数据库服务发挥出最优性能，用户可根据业务需求对用户创建的参数模板中的参数进行调整。

您可以修改用户创建的参数模板中的参数值，但不能更改默认参数模板中的参数值。对用户创建的参数模板中的参数所做的更改，将应用于与此参数模板关联的所有实例。

以下是您在使用参数模板中的参数时应了解的几个要点：

- 当您更改参数并保存参数模板时，参数更改将在您应用到实例后，手动重启与参数模板关联的实例后生效。应用参数模板到DRDS实例，请参见应用参数模板。
- 如果您更改一个参数值，则所做更改的应用时间将由该参数的类型决定。

说明

系统提供的默认参数模板不允许修改，只可单击参数模板名进行查看。当用户参数设置不合理导致实例无法启动时，可参考默认参数模板重新配置。

操作步骤

步骤 1 登录管理控制台。

步骤 2 单击管理控制台左上角的 ，选择区域和项目。

步骤 3 在页面左上角单击 ，选择“数据库 > 分布式关系型数据库 ”。进入分布式关系型数据库信息页面。

步骤 4 在“参数模板”页面的“自定义”页签，选择目标参数模板，单击参数模板名称。

步骤 5 默认在“参数”页签下，根据需要修改相关参数值。

可进行的操作如下：

须知

参数模板修改后，某些参数会立即应用到当前使用实例中，请谨慎操作。

- 单击“保存”，在弹出框中单击“是”，保存修改。
- 单击“取消”，放弃本次设置。

步骤 6 参数修改完成后，您可以单击“模板历史记录”查看参数的修改详情。

须知

参数模板修改后，不会立即应用到当前使用的实例，您需要进行应用操作才可生效，具体操作请参见应用参数模板。

修改某些参数或字符集后需要手动重启，由于变更规格导致的强制重启，不会触发该参数生效。

----结束

6.3 比较参数模板

操作场景

您可以在同一个 DRDS 实例上选择不同的参数模板，以了解参数对实例的影响。

您也可以在不同的 DRDS 实例上选择同样的参数模板，了解参数对不同实例的影响。

操作步骤

步骤 1 登录管理控制台。

步骤 2 单击管理控制台左上角的 ，选择区域和项目。

步骤 3 在页面左上角单击 ，选择“数据库 > 分布式关系型数据库 ”。进入分布式关系型数据库信息页面。

步骤 4 在“参数模板”页面的“自定义”页签，选择一个用户创建的参数模板，单击“比较”。

步骤 5 选择不同参数模板，单击“确定”，比较两个参数模板之间的配置参数差异项。

说明

您也可以在“参数模板”页面的“系统默认”页签，选择一个用户创建的参数模板进行比较。步骤与自定义页签下的一致。

- 有差异项，则会显示差异参数模板的如下信息：参数名称、两个参数模板的参数值。
- 无差异项，则不显示。

----结束

6.4 查看参数修改历史

操作场景

您可以查看当前实例所使用参数模板以及自定义参数模板的修改历史，以满足业务需要。

说明

用户创建或导出的新参数模板，在未进行参数修改前，无修改历史。

操作步骤

步骤 1 登录管理控制台。

步骤 2 单击管理控制台左上角的，选择区域和项目。

步骤 3 在页面左上角单击，选择“数据库 > 分布式关系型数据库 ”。进入分布式关系型数据库信息页面。

步骤 4 在“参数模板”页面的“自定义”页签，单击目标参数模板列表操作栏的“更多 > 历史修改记录”。

您可查看参数对应的参数名称、修改前参数值、修改后参数值、修改状态和修改时间。

----结束

6.5 复制参数模板

操作场景

您可以复制您创建的自定义数据库参数模板。当您已创建一个数据库参数模板，并且想在新的数据库参数模板中包含该组中的大部分自定义参数和值时，复制参数模板是一个方便的解决方案。

复制数据库参数模板之后，新参数模板可能不会立即显示，建议您等待 5 分钟再使用。

您无法复制默认参数模板。不过，您可以创建基于默认参数模板的新参数模板。

操作步骤

步骤 1 登录管理控制台。

步骤 2 单击管理控制台左上角的，选择区域和项目。

步骤 3 在页面左上角单击，选择“数据库 > 分布式关系型数据库 ”。进入分布式关系型数据库信息页面。

步骤 4 在“参数模板”页面的“自定义”页签，选择需要复制的参数模板，单击“复制”。

步骤 5 在弹出框中，填写新参数模板名称和描述，单击“确定”。

- 参数模板名称长度在 1~64 个字符之间，区分大小写，可包含字母、数字、中划线、下划线或句点，不能包含其他特殊字符。
- 参数模板的描述长度不能超过 256 个字符，且不能包含回车和>!<"&'=特殊字符。

创建完成后，会生成一个新的参数模板，您可在参数模板列表中对其进行管理。

----结束

6.6 应用参数模板

操作场景

参数模板创建成功并根据业务需求调整完参数之后，您可以根据业务需要应用到 DRDS 实例中。

操作步骤

步骤 1 登录管理控制台。

步骤 2 单击管理控制台左上角的 ，选择区域和项目。

步骤 3 在页面左上角单击 ，选择“数据库 > 分布式关系型数据库 ”。进入分布式关系型数据库信息页面。

步骤 4 在“参数模板”页面，根据参数模板类型不同进行如下操作。

- 若需要将默认参数模板应用到实例，在“系统默认”页签的目标参数模板单击“应用”。
- 若需要将用户自己创建的参数模板应用到实例，在“自定义”页签的目标参数模板单击“更多 > 应用”。

一个参数模板可被应用到一个或多个实例。

步骤 5 在弹出框中，选择或输入所需应用的实例，单击“确定”。

参数模板应用成功后，您可查看参数模板应用记录。

----结束

6.7 查看参数模板应用记录

操作场景

参数模板应用到 DRDS 实例之后，您可以查看参数模板的应用记录。

操作步骤

步骤 1 登录管理控制台。

步骤 2 单击管理控制台左上角的，选择区域和项目。

步骤 3 在页面左上角单击，选择“数据库 > 分布式关系型数据库 ”。进入分布式关系型数据库信息页面。

步骤 4 单击“参数模板”。

步骤 5 在“系统默认”页签下，选择目标参数模板，单击“应用记录”；或在“自定义”页签下，选择目标参数模板，单击“更多 > 应用记录”，查看应用记录。

您可查看参数模板所应用到的实例名称/ID、应用状态、应用时间、失败原因。

----结束

6.8 修改参数模板描述

操作场景

参数模板创建成功后，用户可根据需要对自己创建的参数模板描述进行修改。

说明

默认参数模板的描述不可修改。

操作步骤

步骤 1 登录管理控制台。

步骤 2 单击管理控制台左上角的，选择区域和项目。

步骤 3 在页面左上角单击，选择“数据库 > 分布式关系型数据库 ”。进入分布式关系型数据库信息页面。

步骤 4 在“参数模板”页面的“自定义”页签，选择一个用户创建的参数模板，单击“描述”列。

步骤 5 输入新的描述信息，单击，提交修改，单击，取消修改。

- 参数模板的描述长度不能超过 256 个字符，且不能包含>!<"&'=特殊字符。
- 修改成功后，可在参数模板列表的“描述”列查看改后的描述信息。

----结束

6.9 删除参数模板

操作场景

您可删除废弃的参数模板。

须知

- 参数模板删除后，不可恢复，请谨慎操作。
- 默认参数模板不可被删除。

操作步骤

步骤 1 登录管理控制台。

步骤 2 单击管理控制台左上角的，选择区域和项目。

步骤 3 在页面左上角单击，选择“数据库 > 分布式关系型数据库 ”。进入分布式关系型数据库信息页面。

步骤 4 在“参数模板”页面的“自定义”页签，选择需要删除的参数模板，单击“更多 > 删除”。

步骤 5 单击“是”，删除参数模板。

----结束

7 任务中心

您可以通过“任务中心”查看用户在控制台上提交的异步任务的执行进度和状态。

说明

分布式关系型数据库服务支持查看以下任务：

- 创建实例
- 删除实例
- 规格变更
- 节点扩容
- 节点缩容
- 重启实例
- 绑定 EIP
- 解绑 EIP
- 恢复数据
- 导入逻辑库信息
- 一致性备份
- 删除一致性备份
- 一致性恢复
- 分片变更

操作步骤

步骤 1 登录管理控制台。

步骤 2 击管理控制台左上角的，选择区域和项目。

步骤 3 在页面左上角单击，选择“数据库 > 分布式关系型数据库 ”。进入分布式关系型数据库信息页面。

步骤 4 在“任务中心”页面，选择目标任务，查看任务信息。

- 通过任务名称/订单 ID、实例名称/ID 确定目标任务，或通过右上角的搜索框输入任务名称来确定目标任务。

- 单击页面右上角的，查看某一时间段内的任务执行进度和状态，默认时长为一周。

  任务保留时长最多为一个月。
- 系统支持查看以下状态的即时任务：
  - 执行中
  - 完成
  - 失败
- 查看任务创建时间和结束时间。

----结束

8 逻辑库管理

8.1 创建逻辑库

前提条件

- 成功登录分布式关系型数据库服务控制台。
- 当前 DRDS 实例状态为“运行中”。
- DRDS 在数据节点上创建的内部帐号（DRDSRW*、DRDSR*、DRDSREP*）请勿修改和删除，否则会影响业务。

说明

- 内部帐号名称组成规则：固定前缀（DRDSRW、DRDSR、DRDSREP）+数据节点 id 取 hash 值。
- 口令规则：口令随机生成，长度最小 16，最长 32。
- 创建逻辑库时，同一个逻辑库，mysql 大版本需要相同，不可以混用。
- 创建逻辑库时，同一个 DRDS 实例可以创建多个逻辑库。多个逻辑库可关联同一个数据节点，但同一个 DRDS 实例不允许同时关联 RDS for MySQL 实例和 GaussDB(for MySQL)实例。
- 一个数据节点无法被不同的 DRDS 实例关联。
- 创建逻辑库时选多个分片的场合，分片名遵循【逻辑库名+xxxx】的命名规则，其中 xxxx 为从“0000”开始递增的数字。如逻辑库名为“db_cbb5”，总分片数为 2，则分片名为“db_cbb5_0000”和“db_cbb5_0001”。

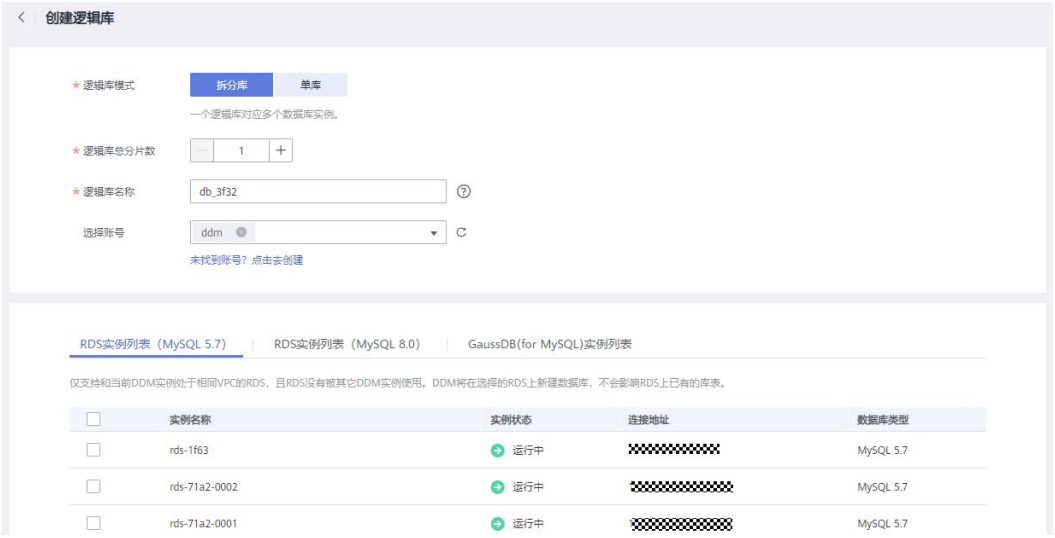
创建 DRDS 逻辑库有两个入口，以服务列表页为例。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，在目标实例操作栏单击“创建逻辑库”。

步骤 2 在创建逻辑库页面，选择“逻辑库模式”、“逻辑库总分片数”，填写“逻辑库名称”，并选择要关联的 DRDS 帐号、要关联的实例，单击“下一步”。

图 8-1 创建逻辑库



步骤 3 下一步，进行数据库连接验证，输入对应数据库密码，单击“测试连接”。

图 8-2 数据实例连接验证

步骤 4 测试通过后，单击页面下方的“完成”。

----结束

8.2 导出逻辑库

操作场景

用于跨 region 容灾或者数据迁移场景。在源实例进行导出，导出的逻辑库信息主要包含逻辑库的基本信息和分片信息，不包含业务数据和索引数据。

前提条件

DRDS 实例中已创建逻辑库。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，选择目标 DRDS 实例，单击实例名称，进入实例基本信息页面。

步骤 2 在实例基本信息页面左侧导航栏，选择“逻辑库管理”选项卡，查看对应实例逻辑库列表。

步骤 3 在逻辑库列表页面，单击“导出逻辑库信息”，系统会自动导出当前实例下的逻辑库的 json 文件。

----结束

8.3 导入逻辑库

操作场景

用于跨 region 容灾或者数据迁移场景。在目标实例进行导入，导入的逻辑库信息主要包含逻辑库的基本信息和分片信息，不包含业务数据和索引数据。

前提条件

DRDS 实例没有逻辑库。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，选择目标 DRDS 实例，单击实例名称，进入实例基本信息页面。

步骤 2 在实例基本信息页面左侧导航栏，选择“逻辑库管理”选项卡，查看对应实例逻辑库列表。

步骤 3 在逻辑库列表页面，单击“导入逻辑库信息”进入导入逻辑库信息页面。

说明

json 可重复导入，但可重复导入的前提是： 目标 DRDS 实例没有同名逻辑库。

步骤 4 在“导入逻辑库信息”页面单击“添加文件”，在本地选择需要导入的.json 文件（导出逻辑库导出的 json 文件），选择需要使用的数据节点，输入正确的数据库密码，单击完成即可。

说明

选择的数据节点的数量与导入 DRDS 关联的数据节点数量一致。

----结束

8.4 删除逻辑库

前提条件

- 成功登录分布式关系型数据库服务控制台。
- 已创建逻辑库。

须知

删除操作无法恢复，请谨慎操作。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤 2 在左侧导航栏选择“逻辑库管理”选项卡，查看对应实例逻辑库列表。

步骤 3 在逻辑库列表页面，选择目标逻辑库，操作列单击“删除”。

步骤 4 在删除确认弹窗中，单击“确定”。

说明

- 请勿直接在数据节点列表删除和 DRDS 逻辑库关联的实例，会直接导致逻辑库故障。
- 如需删除数据节点上的数据，请在删除逻辑库的弹窗中勾选“删除数据节点上的数据”。
- 若您想删除逻辑库，请首先确认数据节点是否存在。若实例已删除，请先单击“同步数据库信息”，再进行删除操作。
- 若您所连接的数据节点有名称、引擎、引擎版本号、最大连接数 max_connections、端口号、ip 等信息的修改，不需要删除逻辑库，只需单击“同步数据库信息”同步最新配置。

----结束

8.5 配置 SQL 黑名单

概述

配置 SQL 黑名单即配置该逻辑库不允许执行的 SQL 语句。

前提条件

- 成功登录分布式关系型数据库服务控制台。
- DRDS 实例逻辑库且 DRDS 实例运行正常。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，选择目标 DRDS 实例，单击实例名称，进入实例基本信息页面。

步骤 2 在实例基本信息页面左侧导航栏，选择“逻辑库列表”选项卡，查看 DRDS 实例逻辑库。

步骤 3 在逻辑库列表页面，单击右侧操作栏的“配置 SQL 黑名单”。

图 8-3 配置 SQL 黑名单

步骤 4 在配置 SQL 黑名单弹窗中，单击“编辑”，按需输入前缀匹配、全量匹配、正则匹配的 SQL 信息，设置完成后单击“确定”即可。

说明

- 前缀匹配：禁止在对应逻辑库执行带有某些关键字的 SQL 语句，例如“DROP XXXX”,"DELETE XXX"。
- 全量匹配：禁止在对应逻辑库执行该 SQL 语句。
- 配置的黑名单 SQL 之间以英文分号隔开，前缀匹配与全量匹配中的 SQL 语句加起来大小不超过 1kb。
- 若在配置 SQL 黑名单弹窗中清除之前编辑的前缀匹配与全量匹配中的 SQL 语句，并单击“确定”，则表示清空之前配置的 SQL 黑名单。

----结束

9 DN 管理

9.1 DN 管理概述

DN 管理提供数据节点管理服务，管理 DRDS 实例关联的 RDS for MySQL 实例或者 GaussDB(for MySQL)实例，展现实例的状态、存储、规格、读权重等信息，提供设置读权重、增加只读实例的快捷操作。

“设置读权重”主要用于批量设置读权重，可同时设置列表中多个数据节点的权重，注意如果数据节点未挂载只读实例，该主实例无法设置权重。

“同步 DN 信息”用于数据节点数量比较多的场景下，可以先设置第一个数据节点的读权重，点击“同步 DN 信息”后，会把第一个数据节点的设置同步到其他只读实例数量相同的数据节点上，如果只读实例数量不同，请手动设置。

9.2 设置读权重

前提条件

成功登录分布式关系型数据库服务控制台。

应用场景

对于拥有较多数据节点的 DRDS 实例来说，增加批量配置数据节点读权重能力，可改善用户体验。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，选择目标实例。

步骤 2 在左侧导航栏，选择“DN 管理”页签，单击“设置读权重”。

图 9-1 DN 管理

步骤 3 在设置读权重窗口中，设置权重数，单击“是”。

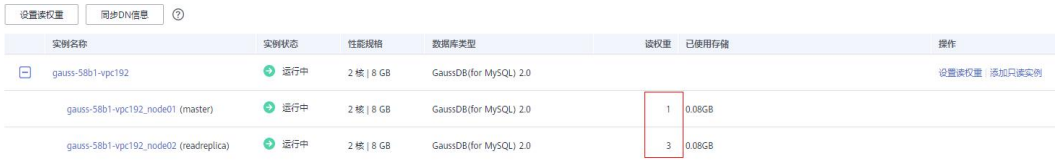
图 9-2 设置读权重

步骤 4 设置读权重命令下发成功提示。

图 9-3 设置读权重成功提示

步骤 5 实例读权重显示最新设置权重数。

图 9-4 实例读权重显示



----结束

9.3 同步 DN 信息

前提条件

成功登录分布式关系型数据库服务控制台。

应用场景

在数据节点相关信息变化时（如增删只读实例，变更连接地址、端口号、安全组、实例规格，删除数据库实例，以及将数据库从一个企业项目迁移到另一个企业项目等变更），需要用户主动下发“同步 DN 信息”，将数据节点变化的信息同步到 DRDS，才能正常使用。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，选择目标实例。

步骤 2 在左侧导航栏，选择“DN 管理”页签，单击“同步 DN 信息”。

图 9-5 同步 DN 信息

步骤 3 系统自动弹出同步 DN 信息命令下发成功。

图 9-6 同步 DN 信息下发成功提示

----结束

9.4 表数据重载

前提条件

成功登录分布式关系型数据库服务控制台。

应用场景

DRDS 跨 region 容灾场景下，通过数据迁移服务（DRS）进行存储层数据迁移，迁移完成之后，目标 DRDS 无法感知逻辑表信息所在位置，所以需要在目标 DRDS 主动下发“表数据重载”，重新加载信息，跟分片建立联系。

操作步骤

步骤 1 在分布式关系型数据库服务“实例管理”页面，选择指定的实例，单击实例名称，进入实例基本信息页面。

步骤 2 在操作栏，选择“更多”>“表数据重载”。

系统自动弹出实例 XXX 表数据重载成功。

图 9-7 表数据重载成功提示

----结束

10 帐号管理

10.1 创建帐号

前提条件

- 成功登录分布式关系型数据库服务控制台。
- 当前 DRDS 实例有逻辑库。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤 2 在左侧导航栏选择“帐号管理”，进入帐号管理页面。

步骤 3 在帐号管理页面单击“创建 DRDS 帐号”，在弹窗中填选帐号参数信息。

表 10-1 创建 DRDS 帐号配置参数

参数	说明
帐号名称	DRDS 帐号的名称，命名规则如下。 长度为 1-32 个字符，必须以字母开头，可以包含字母，数字、下划线，不能包含其它特殊字符。
密码	DRDS 帐号的密码，密码复杂度要求如下。 • 长度为 8~32 个字符 • 至少包含三种字符组合：小写字母、大写字母、数字、特殊字符 ~ ! @ # % ^ * - _ = + ?
确认密码	-
帐号权限	按需选择需要的基础权限，包括 CREATE、DROP、ALTER、INDEX、INSERT、DELETE、UPDATE、SELECT。

参数	说明
关联逻辑库	DRDS 帐号与逻辑库关联绑定，下拉列表中显示可关联的逻辑库。 DRDS 帐号只对已关联的逻辑库有访问权限。
描述	对 DRDS 帐号的详细描述信息，长度不能超过 256 个字符。

步骤 4 确认填写无误后，单击“确定”。

----结束

10.2 修改帐号信息

前提条件

成功登录分布式关系型数据库服务控制台。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤 2 在左侧导航栏选择“帐号管理”，进入帐号管理页面。

步骤 3 帐号管理页面选择目标帐号，在其操作栏单击“修改”。

图 10-2 修改帐号

步骤 4 在修改帐号弹窗中，您可以修改关联逻辑库、帐号权限及描述信息。

步骤 5 编辑完成帐号信息后，单击“确定”，保存修改信息。

----结束

10.3 删除帐号

前提条件

成功登录分布式关系型数据库服务控制台。

说明

删除操作无法恢复，请谨慎操作。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤 2 在左侧导航栏选择“帐号管理”，进入帐号管理页面。

步骤 3 帐号管理页面选择目标帐号，在其操作栏单击“更多 > 删除”。

步骤 4 在删除帐号确认弹窗中，单击“是”，删除帐号信息。

----结束

10.4 重置密码

前提条件

成功登录分布式关系型数据库服务控制台。

操作步骤

步骤 1 在分布式关系型数据库服务，实例管理列表页面，单击目标实例名称，进入实例基本信息页面。

步骤 2 在左侧导航栏选择“帐号管理”，进入帐号管理页面。

步骤 3 帐号管理页面选择目标帐号，在其操作栏单击“更多 > 重置密码”。

步骤 4 在输入新密码并再次输入“确认密码”后，单击“确认”，进行密码重置。

----结束

11 备份管理

11.1 一致性备份说明

DRDS 对一致性备份进行了功能优化调整，将该功能调整成恢复数据中的“Metadata 恢复。升级之后 Console 上的一致性备份列表、创建等相关功能将不可见。若您有任何疑问，请联系客服或者 DRDS 的技术人员。

11.2 恢复到新实例

概述

DRDS 支持基于已有备份集将实例恢复至任意时间点，适用于日常业务常规备份恢复场景。

须知

本章节以 RDS for MySQL 实例举例说明。

限制

- 恢复到目标 DRDS 实例会导致实例数据被覆盖，恢复过程中目标实例数据库不可用。
- 需确保恢复时间点到当前时间之间未进行过逻辑库的创建或者删除操作。
- 恢复的新 RDS for MySQL 实例版本需要大于等于原实例版本，且存储空间大于等于原实例。
- 此恢复功能暂不支持备份恢复到 RDS for MySQL 本地盘实例。
- 此恢复功能暂不支持从 RDS for MySQL 本地盘实例恢复。
- 暂不支持目标 DRDS 实例在主网段、RDS for MySQL 实例在扩展网段的场景。

前提条件

- 成功登录分布式关系型数据库服务控制台。
- 当前实例状态为“运行中”。

操作步骤

步骤 1 在当前 DRDS 实例所在区域创建一个新的 DRDS 实例或者寻找一个满足限制的实例。

步骤 2 创建与当前 DRDS 实例下相同数量的 RDS for MySQL 实例。

说明

- 新建的 DRDS 实例不能挂载 RDS for MySQL 实例，已有的 DRDS 实例不能创建逻辑库和帐号。
- 新创建的 RDS for MySQL 实例版本不得低于源 DRDS 下 RDS for MySQL 实例版本号。
- 每个实例存储空间不得小于当前 DRDS 实例下的 RDS for MySQL 的存储空间。

步骤 3 在 DRDS 实例列表页面单击 1 中的当前实例名称，进入实例基本信息页面。

步骤 4 在左侧导航栏选择“备份恢复”，进入恢复数据页面。单击“恢复新实例”按钮。

步骤 5 在数据恢复页签选择“可恢复时间段”和“可恢复时间点”，并选择 1 中创建的 DRDS 实例作为目标实例。

步骤 6 选择 2 中创建的 RDS 实例作为目标 RDS for MySQL 实例，选中确认复选框。确认无误后，单击“确认”，等待 1-3 分钟数据恢复完成。

----结束

11.3 Metadata 恢复

概述

关于 Metadata 的备份策略，DRDS 将在每日凌晨 2 点至 3 点对实例的 Metadata 数据进行备份，备份保留时长 30 天。删除逻辑库、逻辑库分片变更后清理数据、删除实例等影响 Metadata 的重要操作也会触发元数据备份。

元数据恢复特性用于误删库或者 RDS for MySQL 本身出现异常等业务场景，可根据过去的某个时间点，将 Metadata 数据与已经 PITR 恢复完成的 RDS for MySQL 实例进行匹配，重建 DRDS 和 RDS for MySQL 的关联关系，恢复 DRDS。当前该特性仅支持 RDS for MySQL 引擎。

限制

- 目标 DRDS 实例不能关联 RDS for MySQL 实例，不能创建逻辑库和帐号。
- 确保选择的 RDS for MySQL 实例已经进行 PITR 恢复。
- 暂不支持目标 DRDS 实例在主网段、RDS for MySQL 实例在扩展网段的场景。

前提条件

- 成功登录分布式关系型数据库服务控制台。
- 当前 DRDS 实例状态为“运行中”。

操作步骤

步骤 1 申请新的 DRDS 实例。DRDS 申请请参考实例申请。

步骤 2 在 DRDS 实例列表页面单击当前实例名称，进入实例基本信息页面。

步骤 3 在左侧导航栏选择“备份恢复”，进入恢复数据页面。单击“Metadata 恢复”按钮。

步骤 4 选择恢复到的时间点，DRDS 将在该时间点就近选择合适的 DRDS 元数据备份集。

步骤 5 选择步骤 1 中创建的目标实例，如果没有预置新的的实例，请参照实例申请创建新的实例。

步骤 6 选择已经完成 PITR 的 RDS for MySQL 实例，单击“确认”进行恢复。

步骤 7 Metadata 恢复下发成功提示。

----结束

12 慢查询

操作场景

DRDS 提供“慢查询”功能，将指定时间内的慢 SQL 语句进行统计分类，把结构相同的慢 SQL 语句整理成 SQL 模板，您可以查看指定区间内的所有慢 SQL 类型，针对这些类型进行优化处理。

操作步骤

步骤 1 在分布式关系型数据库服务实例管理列表页面，单击进入目标实例。

步骤 2 单击左侧菜单栏的“慢查询”页签，进入“慢查询”服务页。

步骤 3 您可在“慢查询”服务页面查看超过指定时间的 SQL 语句。

----结束

13 监控管理

13.1 监控指标

功能说明

本节定义了分布式关系型数据库服务上报云监控的监控指标的命名空间，监控指标列表和维度定义，您可以通过云监控提供的 API 接口来检索 DRDS 产生的监控指标信息。

命名空间

SYS.DRDS

监控指标

表 13-1 DRDS 支持的监控指标

指标 ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
ddm_cpu_util	CPU 使用率	该指标用于统计当前 DRDS 节点的 CPU 利用率。	0~100%	DRDS 节点	1 分钟
ddm_mem_util	内存使用率	该指标用于统计当前 DRDS 节点的内存使用率。	0~100%	DRDS 节点	1 分钟
ddm_bytes_in	网络输入吞吐量	该指标用于统计当前 DRDS 节点平均每秒的输入流量。	≥ 0 bytes/s	DRDS 节点	1 分钟
ddm_bytes_out	网络输出吞吐量	该指标用于统计当前 DRDS 节点平均每秒的输出流量。	≥ 0 bytes/s	DRDS 节点	1 分钟

指标 ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
ddm_qps	QPS	该指标用于统计当前DRDS 节点的每秒请求数。	≥ 0 counts	DRDS 节点	1 分钟
ddm_read_count	读次数	该指标用于统计当前DRDS 节点在每个采集周期内新增的读次数。	≥ 0 counts/s	DRDS 节点	1 分钟
ddm_write_count	写次数	该指标用于统计当前DRDS 节点在每个采集周期内新增的写次数。	≥ 0 counts/s	DRDS 节点	1 分钟
ddm_slow_log	慢 SQL 数	该指标用于统计数据面服务 Core 的慢 SQL 条数。	≥ 0 counts	DRDS 节点	1 分钟
ddm_rt_avg	平均响应时延	该指标用于统计数据面服务 Core 的 SQL 平均响应时延。	≥ 0 ms	DRDS 节点	1 分钟
ddm_connections	连接数	该指标用于统计数据面服务 Core 的连接数。	≥ 0 counts	DRDS 节点	1 分钟
ddm_backend_connection_ratio	后端连接池水位	该指标用于统计当前DRDS 节点后端活跃连接数与后端最大连接数的比例。	0-100%	DRDS 节点	1 分钟
active_connections	活跃连接数	该指标用于统计每个DRDS 节点后台正在执行的连接数目。	≥ 0	DRDS 节点	1 分钟
ddm_connections_util	连接数使用率	该指标用于统计每个DRDS 节点已用的连接数占总连接数的百分比。	0~100%	DRDS 节点	1 分钟

维度

Key	Value
node_id	DRDS 节点。

说明

DRDS 只支持节点维度(node_id)，暂不支持实例维度(instance_id)。需通过实例 id 来查找节点 id。

13.2 查看监控指标

操作场景

监控管理控制台提供了对 DRDS 实例的监控管理，您可以根据实时监控监控反馈结果，对数据库进行调优。

前提条件

- 成功登录分布式关系型数据库服务控制台。
- 已创建 DRDS 实例、逻辑库。

操作步骤

步骤 1 在分布式关系型数据库服务实例管理列表页面，单击进入目标实例。

步骤 2 单击左侧菜单栏的“监控管理”页签。

步骤 3 查看“实时监控详情”。

步骤 4 单击“查看监控指标”。进入“云监控服”务页面。

图 13-2 查看监控指标

步骤 5 您可在云监控服务页面查看指标。

1. 在>“云服务监控”导航栏，获取云服务监控列表信息。
2. 选择您所需查看的实例名称，单击 可展开当前实例下节点信息列表，并在目标实例节点右侧操作栏单击“查看监控指标”，进入该节点监控指标详情页面。
3. 您可在监控指标信息页面，根据时间范围查看各项指标详细信息。

----结束

14 SQL 语法

14.1 简介

DRDS 兼容 MySQL 协议及其语法，但因分布式数据库与单机数据库之间存在一定的差异性，导致 SQL 使用存在些限制。

在评估 DRDS 方案之前，请先完成当前应用中的 SQL 语法及与 DRDS 支持语法的兼容性评估。

MySQL EXPLAIN

当您在需要执行的 SQL 语句前加上 EXPLAIN，然后执行 SQL 时，您将会看到其具体的执行计划，以此分析耗时，进而修改 SQL，达到优化的效果。

表 14-1 EXPLAIN 列的解释

列名称	描述
table	显示该行数据所归属的表。
type	显示连接使用了何种类型。从最好到最差的连接类型为 const、eq_reg、ref、range、index 和 ALL。
possible_keys	显示可能应用在该表中的索引。
key	实际使用的索引。如果为 NULL，则没有使用索引。个别情况下，MySQL 会选择优化不足的索引。在 SELECT 语句中使用 USE INDEX（indexname）来强制使用一个索引或者用 IGNORE INDEX（indexname）来强制 MySQL 忽略索引。
key_len	使用的索引的长度。在不损失精确性的情况下，长度越短越好。
ref	显示索引被使用的列，通常为一个常数。
rows	MySQL 用来返回请求数据的行数。
Extra	关于 MySQL 如何解析查询的额外信息。

SQL 大类限制

- 不支持临时表；
- 不支持外键、视图、游标、触发器及存储过程；

- 不支持用户自定义数据类型及自定义函数；
- 不支持“IF”，“WHILE” 等流程控制类语句；
- 不支持 BEGIN…END、LOOP…END LOOP、REPEAT…UNTIL…END REPEAT、WHILE…DO…END WHILE 等复合语句。

DDL 语法

- 拆分表和广播表不支持外键；
- 不支持修改分片字段（拆分键）；
- 不支持 ALTER DATABASE Syntax；
- 不支持从另一张表创建新的拆分表、广播表；
- create table 不支持 generated column 语法；
- 不支持 ALTER 命令修改拆分键和全局序列字段；
- 不支持创建 TEMPORARY 类型的拆分表、广播表；
- 逻辑表名只允许字母(不区分大小写)数字以及下划线
- CREATE TABLE tbl_name LIKE old_tbl_name 不支持；
- CREATE TABLE tbl_name SELECT statement 不支持；
- 不支持 insert into on duplicate key update 更新拆分键值；
- 暂不支持跨 Schema 的 DDL，例如， CREATE TABLE db_name.tbl_name (…)；
- 使用 MySQL 关键字或保留字做表名、列名、索引名等标识符时，需要使用反引号扩起来；

DML 语法

- 不支持 PARTITION 子句；
- 不支持 UPDATE 使用子查询；
- 不支持 INSERT DELAYED Syntax；
- 不支持 STRAIGHT_JOIN 和 NATURAL JOIN；
- 受限支持 跨分片 UPDATE 多表需要 join update 的表需要有 PK；
- 受限支持 跨分片 DELETE 多表中的数据，需要 Join delete 的表需要有 PK；
- 不支持 SQL 中对于变量的引用和操作，比如 SET @c=1, @d=@c+1; SELECT @c, @d；
- INSERT 或者 UPDATE 时，不支持插入或者更新拆分键值为 DEFAULT 关键字；
- UPDATE 不支持在一个语句中对同一字段重复更新；
- UPDATE 不支持关联更新拆分键；
- UPDATE 不支持自关联更新；
- 关联更新中，不支持在目标列的赋值语句或表达式中引用其它目标列，将造成更新结果不符合预期。例如：

  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。
- 当使用文本协议时，BINARY、VARBINARY、TINYBLOB、BLOB、MEDIUMBLOB、LONGBLOB 数据必须转换成 16 进制数据；

- DRDS 对非法数据的处理与后端 MySQL 的 sql_mode 有关；
- 关联更新不支持不带关联条件的 Join；
- SQL 语句中表达式的因子数量请勿超过 1000 个；

函数

- 不支持 XML 函数；
- 不支持 GTID 函数；
- 不支持全文检索函数；
- 不支持企业加密函数；
- 不支持 row_count()函数。

子查询

不支持 HAVING 子句中的子查询，JOIN ON 条件中的子查询；

数据类型

不支持空间数据类型。

14.2 DDL

14.2.1 DDL 概述

DRDS 支持通用的 DDL 操作：建库，建表，修改表结构等，但实现方式与普通的 MySQL 数据库有所区别。

在 MySQL 客户端执行 DDL 操作

- TRUNCATE Syntax；

  举例：
  ```
  TRUNCATE TABLE t1
  ```
  表示清空表格 t1。
  TRUNCATE 会将表完全清空，它需要 DROP 权限。在逻辑上类似于删除所有行的 DELETE 语句。

- ALTER TABLE Syntax；

  举例：
  ```
  ALTER TABLE t2 DROP COLUMN c, DROP COLUMN d;
  ```
  表示更改表 t2 的结构：删除 c 列和 d 列。
  ALTER 可以添加或删除列、创建或销毁索引、更改现有列的类型或重命名列或表本身。还可以更改特性，如用于表或表注释的存储引擎。

- DROP INDEX Syntax；

  举例：
  ```
  DROP INDEX 'PRIMARY' ON t;
  ```
  表示删除表 t 中的主键。
  DROP INDEX 即从表 tbl_name 中删除名为 index_name 的索引。

- CREATE INDEX Syntax；

举例：
CREATE INDEX part_of_name ON customer (name(10));
表示使用 name 列的前 10 个字符创建索引（假设 name 具有非二进制字符串类型）。
CREATE INDEX 用于向现有表添加索引。

14.2.2 创建表

说明

- 禁止创建表名以"_ddm"为前缀的表，系统默认认定此类表为系统内部表。
- 拆分表不支持全局唯一索引，当唯一键和拆分键不一致时，不能保证数据的唯一性。

分库分表

假设使用 HASH 的拆分库算法，拆分表算法为 MOD_HASH，样例如下：

```
CREATE TABLE tbpartition_tbl (
id int NOT NULL AUTO_INCREMENT COMMENT '主键 id',
name varchar(128),
PRIMARY KEY(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
DBPARTITION BY HASH(id)
TBPARTITION BY mod_hash(name) tbpartitions 8;
```

分库不分表

假设使用 HASH 的拆分库算法，样例如下：

```
CREATE TABLE dbpartition_tbl (
id int NOT NULL AUTO_INCREMENT COMMENT '主键 id',
name varchar(128),
PRIMARY KEY(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
DBPARTITION BY HASH(id);
```

广播表

如下为创建广播表的样例：

```
CREATE TABLE broadcast_tbl (
id int NOT NULL AUTO_INCREMENT COMMENT '主键 id',
name varchar(128),
PRIMARY KEY(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
BROADCAST;
```

单表

单表也可以指定全局序列，但是会忽略这个功能。自增功能使用底层存储的自增值进行自增。

创建单表样例如下，不做任何拆分：

```
CREATE TABLE single(
id int NOT NULL AUTO_INCREMENT COMMENT '主键 id',
```

```
name varchar(128),
PRIMARY KEY(id)
);
```

14.2.3 拆分算法概述

支持的拆分算法概览

DRDS 是一个支持既分库又分表的数据库服务，目前 DRDS 分库函数与分表函数的支持情况如下：

表 14-2 拆分算法概览表

拆分函数	描述	能否用于分库	能否用于分表
MOD_HASH	简单取模	是	是
MOD_HASH_CI	简单取模(大小写不敏感)	是	是
HASH	简单取模	是	是
RANGE	按范围	是	否
RIGHT_SHIFT	数值向右移	是	是
YYYYMM	按年月哈希	是	是
YYYYDD	按年日哈希	是	是
YYYYWEEK	按年周哈希	是	是
MM	按月份哈希	否	是
DD	按日期哈希	否	是
MMDD	按月日哈希	否	是
WEEK	按星期哈希	否	是

说明

- 分库的拆分键及分表的拆分键，均不支持为空。
- 在 DRDS 中，一张逻辑表的拆分方式是由拆分函数（包括分片数目与路由算法）与拆分键（包括拆分键的 MySQL 数据类型）共同定义。
- 当一张逻辑表的分库拆分方式与分表拆分方式不一致时，若 SQL 查询没有同时带上分库条件与分表条件，则 DRDS 在查询过程会产生全分库扫描或全分表扫描的操作。

DDL 拆分函数的数据类型

DRDS 的拆分函数对各数据类型对支持情况有所不同，下表显示了 DRDS 拆分函数对各种数据类型的支持情况。

表 14-3 DRDS 拆分函数对各种数据类型的支持情况

拆分算法	TINYINT	SMALLINT	MEDIUMINT	INTEGER	INT	BIGINT	CHAR	VARCHAR	DATE	DATETIME	TIMESTAMP	OTHERS
MOD_HASH	√	√	√	√	√	√	√	√	×	×	×	×
MOD_HASH_CI	√	√	√	√	√	√	√	√	×	×	×	×
HASH	√	√	√	√	√	√	√	√	×	×	×	×
RANGE	√	√	√	√	√	√	×	×	×	×	×	×
RIGHT_SHIFT	√	√	√	√	√	√	×	×	×	×	×	×
YYYYMM	×	×	×	×	×	×	×	×	√	√	√	×
YYYYDD	×	×	×	×	×	×	×	×	√	√	√	×
YYYYWEEK	×	×	×	×	×	×	×	×	√	√	√	×
MM	×	×	×	×	×	×	×	×	√	√	√	×
DD	×	×	×	×	×	×	×	×	√	√	√	×
MMDD	×	×	×	×	×	×	×	×	√	√	√	×
WEEK	×	×	×	×	×	×	×	×	√	√	√	×

DRDS 的拆分函数创表语法

DRDS 兼容 MySQL 的创表语法，并新增加了 partition_options 的分库分表关键字。

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

```
partition_options:
 DBPARTITION BY
     {{RANGE|HASH|MOD_HASH|RIGHT_SHIFT|YYYYMM|YYYYWEEK|YYYYDD}([column])}
    [TBPARTITION BY
{{HASH|MOD_HASH|UNI_HASH|RIGHT_SHIFT|YYYYMM|YYYYWEEK|YYYYDD}(column)}
        [TBPARTITIONS num]
    ]
```

14.2.4 拆分算法使用说明

14.2.4.1 MOD_HASH 算法

适用场景

适用于需要按用户 ID 或订单 ID 进行分库的场景。

使用说明

拆分键的数据类型必须是整数类型（INT, INTEGER, BIGINT, MEDIUMINT, SMALLINT, TINYINT, DECIMAL(支持精度为 0 的情况)）或字符串类型(CHAR,VARCHAR)。

路由方式

根据拆分键的键值直接按分库数/分表数取余。如果键值是字符串，则字符串会被计算成哈希值再进行计算，完成路由计算。

例如，MOD_HASH('8')等价于 8%D（D 是分库数目/分表数）。

算法计算方式

方式一：拆分键是整型

表 14-4 拆分键是整型时的计算方式

条件	算法	举例
分库拆分键 ≠ 分表拆分键	分库路由结果 = 分库拆分键值 % 分库数 分表路由结果 = 分表拆分键值 % 分表数	分库 ：16 % 8 = 0 分表：16 % 3 = 1
分库拆分键 = 分表拆分键(拆分键)	分表路由结果 = 拆分键值 % (分库数 * 分表数) 分库路由结果 = 分表路由结果 / 分表数	分表：16 % (8 * 3) = 16 分库 ：16 / 3 = 5

方式二：拆分键是字符类型

表 14-5 拆分键是字符型时的计算方式

条件	算法	举例
分库拆分键 ≠ 分表拆分键	分库路由结果 = hash(分库拆分键值) % 分库数 分表路由结果 = hash(分表拆分键值) % 分表数	hash（‘abc’)= ‘abc’.hashCode()=96354 分库 :96354 % 8 = 2; 分表 :96354 % 3 = 0；
分库拆分键 = 分表拆分键(拆分键)	分表路由结果 = hash(拆分键值) % (分库数 * 分表数) 分库路由结果 =分表路由结果 / 分表数	hash（‘abc’)= ‘abc’.hashCode()=96354 分表 :96354% (8 * 3) = 18 分库 :18 / 3=6

建表语法

- 如果用户需要对 ID 列按 HASH 函数进行分库不分表：

```
create table mod_hash_tb(
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8 dbpartition by mod_hash(ID);
```

- 如果用户需要对 ID 列按 HASH 函数进行既分库又分表：

```
create table mod_hash_tb(
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8
dbpartition by mod_hash(ID) tbpartition by mod_hash(ID) tbpartitions 4;
```

注意事项

- MOD_HASH 算法是简单取模，要求拆分列的值自身分布均衡才能保证哈希均衡。

14.2.4.2 MOD_HASH_CI 算法

适用场景

适用于需要按用户 ID 或订单 ID 进行分库的场景。

使用说明

拆分键的数据类型必须是整数类型（INT, INTEGER, BIGINT, MEDIUMINT, SMALLINT, TINYINT, DECIMAL(支持精度为 0 的情况)）或字符串类型(CHAR,VARCHAR)。

路由方式

实现原理同 MOD_HASH 算法一致，区别在于 MOD_HASH_CI 算法对大小写不敏感，而 MOD_HASH 算法对大小写敏感。

算法计算方式

方式一：拆分键是整型

表 14-6 拆分键是整型时的计算方式

条件	算法	举例
分库拆分键 ≠ 分表拆分键	分库路由结果 = 分库拆分键值 % 分库数 分表路由结果 = 分表拆分键值 % 分表数	分库 ：16 % 8 = 0 分表：16 % 3 = 1
分库拆分键 = 分表拆分键(拆分键)	分表路由结果 = 拆分键值 % (分库数 * 分表数) 分库路由结果 = 分表路由结果 / 分表数	分表：16 % (8 * 3) = 16 分库 ：16 / 3 = 5

方式二：拆分键是字符类型

表 14-7 拆分键是字符型时的计算方式

条件	算法	举例
分库拆分键 ≠ 分表拆分键	分库路由结果 = hash(分库拆分键值) % 分库数 分表路由结果 = hash(分表拆分键值) % 分表数	hash（‘abc’)= ‘abc’.toUpperCase().hashCode()=64578 分库 :64578 % 8 = 2; 分表 :64578 % 3 = 0；
分库拆分键 = 分表拆分键(拆分键)	分表路由结果 = hash(拆分键值) % (分库数 * 分表数) 分库路由结果 =分表路由结果 / 分表数	hash（‘abc’)= ‘abc’.toUpperCase().hashCode()=64578 分表 :64578% (8 * 3) = 18 分库 :18 / 3=6

建表语法

- 如果用户需要对 ID 列按 MOD_HASH 函数进行分库不分表：

```
create table mod_hash_ci_tb(
    id int,
    name varchar(30) DEFAULT NULL,
```

```
    create_time datetime DEFAULT NULL,
    primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8 dbpartition by mod_hash_ci(ID);
```

- 如果用户需要对 ID 列按 MOD_HASH 函数进行既分库又分表：

```
create table mod_hash_ci_tb(
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8
dbpartition by mod_hash_ci(ID)
tbpartition by mod_hash_ci(ID) tbpartitions 4;
```

注意事项

- MOD_HASH_CI 算法是简单取模，要求拆分列的值自身分布均衡才能保证哈希均衡。

14.2.4.3 RIGHT_SHIFT 算法

适用场景

当拆分键大部分键值的低位部位区分度比较低而高位部分区分度比较高时，则适用于通过此拆分算法提高散列结果的均匀度。

使用说明

拆分键的数据类型必须是整数类型。

路由方式

根据拆分键的键值（键值必须是整数）有符号地向右移二进制移指定的位数（位数由用户通过 DDL 指定），然后将得到的整数值按分库/分表数目取余。

算法计算方式

表 14-8 计算方式

条件	算法	举例
分库拆分键 ≠ 分表拆分键	分库路由结果 = 分库拆分键值 % 分库数 分表路由结果 = 分表拆分键值 % 分表数	分库 ：(123456 >> 4) % 8 = 4 分表：(123456 >> 4) % 3 = 0
分库拆分键 = 分表拆分键(拆分键)	分库路由结果 = 拆分键值 % 分库数 分表路由结果 =(拆分键值%分库数)*分表数+(拆分键值 /分库数)%分表数	分库 ：(123456 >> 4) % 8 = 4 分表：((123456 >> 4) % 8) *3 + ((123456 >> 4)/8)%3= 13

建表语法

```
create table RIGHT_SHIFT(
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8
dbpartition by RIGHT_SHIFT(id, 4)
tbpartition by RIGHT_SHIFT(id, 4) tbpartitions 2;
```

注意事项

- 移位的数目不能超过整数类型所占有的位数目。

14.2.4.4 MM 按月份哈希

适用场景

MM 适用于按月份数进行分表，分表的表名就是月份数。

使用说明

- 拆分键的类型必须是 DATE / DATETIME / TIMESTAMP 其中之一。
- 只能作为分表函数使用，但不能作为分库函数。

路由方式

根据拆分键的时间值的月份数进行取余运算并得到分表下标。

例如：2019-1-15，当根据分库键确定分库后，确定分表的计算方式是：月份 mod 分表数，即：1 mod 12 = 1。

算法计算方式

表 14-9 算法举例

条件	算法	举例
无	分表路由结果 = 分表拆分键值 % 分表数	分表拆分键值 ：2019-1-15 分表：1 % 12 = 1

建表语法

```
create table test_mm_tb (
   id int,
   name varchar(30) DEFAULT NULL,
```

```
   create_time datetime DEFAULT NULL,
   primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by MOD_HASH(id)
tbpartition by MM(create_time) tbpartitions 12;
```

注意事项

按 MM 进行分表，由于一年的月份只有 12 个月，所以各分库的分表数不能超过 12 张分表。

14.2.4.5 DD 按日期哈希

适用场景

DD 适用于按日期的天数进行分表，分表的表数就是日期的天数。

使用说明

- 拆分键的类型必须是 DATE / DATETIME / TIMESTAMP 其中之一。
- 只能作为分表函数使用，但不能作为分库函数。

路由方式

根据拆分键的时间值的日期的天数进行取余运算并得到分表下标。

例如：2019-1-15，当根据分库建确定分库后，确定分表的计算方式是：一个月的第几天 mod 分表数，即：15 mod 31 = 15。

算法计算方式

表 14-10 算法举例

条件	算法	举例
无	分表路由结果 = 分表拆分键值 % 分表数	分表拆分键值 ：2019-1-15 分表：15 % 31 = 15

建表语法

```
create table test_dd_tb (
   id int,
   name varchar(30) DEFAULT NULL,
   create_time datetime DEFAULT NULL,
   primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by MOD_HASH(id)
tbpartition by DD(create_time) tbpartitions 31;
```

注意事项

按 DD 进行分表，由于一个月的中日期（DATE_OF_MONTH）的取值范围是 1~31，所以各分库的分表数不能超过 31 张分表。

14.2.4.6 WEEK 按星期哈希

适用场景

WEEK 适用于按周数的日期目进行分表，分表的表名的下标分别对应一周中的各个日期（星期一到星期天）。

使用说明

- 拆分键的类型必须是 DATE / DATETIME / TIMESTAMP 其中之一。
- 只能作为分表函数使用，但不能作为分库函数。

路由方式

根据拆分键的时间值所对应的一周之中的日期进行取余运算并得到分表下标。

例如：2019-1-15，当根据分库建确定分库后，确定分表的计算方式是：一周的第几天 mod 分表数，即：3 mod 7 = 3; 2019-1-15 是一周的第 3 天。

算法计算方式

表 14-11 算法举例

条件	算法	举例
无	分表路由结果 = 分表拆分键值 % 分表数	分表拆分键值 ：2019-1-15 分表：3 % 7= 3

建表语法

```
create table test_week_tb (
   id int,
   name varchar(30) DEFAULT NULL,
   create_time datetime DEFAULT NULL,
   primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by HASH(name)
tbpartition by WEEK(create_time) tbpartitions 7;
```

注意事项

由于一周共有 7 天，当按 WEEK 进行分表时，所以各分库的分表数不能超过 7 张。

14.2.4.7 MMDD 按月日哈希

适用场景

MMDD 适用于按一年的天数（即一年中日期）进行分表，分表的表名的下标就是一年中的第几天，一年最多 366 天。

使用说明

- 拆分键的类型必须是 DATE / DATETIME / TIMESTAMP 其中之一。
- 只能作为分表函数使用，但不能作为分库函数。

路由方式

根据拆分键的时间值所对应的日期在一年中对应的天数，然后进行取余运算并得到分表下标。

例如：2019-1-15，当根据分库键确定分库后，确定分表的计算方式是：一年的第几天 mod 分表数，即：15 mod 366 = 15; 2019-1-15 是一年的第 15 天。

算法计算方式

表 14-12 算法举例

条件	算法	举例
无	分表路由结果 = 分表拆分键值 % 分表数	分表拆分键值 ：2019-1-15 分表：15 % 366= 15

建表语法

```
create table test_mmdd_tb (
   id int,
   name varchar(30) DEFAULT NULL,
   create_time datetime DEFAULT NULL,
   primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by MOD_HASH(name)
tbpartition by MMDD(create_time) tbpartitions 366;
```

注意事项

由于一年最多只有 366 天，当按 MMDD 进行分表时，所以各个分库的分表数目不能超过 366 张分表。

14.2.4.8 YYYYMM 按年月哈希

适用场景

适用于需要按年份与月份进行分库的场景，建议该函数与 tbpartition YYYYMM(ShardKey) 联合使用。

使用说明

拆分键的数据类型必须是 DATE / DATETIME / TIMESTAMP 其中之一。

路由方式

根据拆分键的时间值的年份与月份计算哈希值，然后再按分库数取余。

例如，YYYYMM(‘2012-12-31 12:12:12’) 等价于 (2012 * 12 + 12) % D（D 是分库数目/分表数）。

算法计算方式

表 14-13 算法计算方式

条件	算法	举例
分库拆分键 ≠ 分表拆分键	拆分键：yyyy-MM-dd 分库路由结果 = (yyyy * 12 + MM) % 分库数 分表路由结果 = (yyyy * 12 + MM) % 分表数	拆分键：2012-11-20 分库 ：(2012 * 12 + 11) % 8 = 3 分表：(2012 * 12 + 11) % 3 = 2
分库拆分键 = 分表拆分键(拆分键)	拆分键：yyyy-MM-dd 分表路由结果 = (yyyy*12+MM) % (分库数 * 分表数) 分库路由结果 = 分表路由结果 / 分表数	拆分键：2012-11-20 分表：(2012 * 12 + 11) % (8 * 3) = 11 分库 ：11 % 3 = 3

建表语法

假设用户的实例里已经分了 8 个物理库，现有一个业务想按年月进行分库。要求同一个月的数据能落在同一张分表内，并且两年以内的每个月都单独对应一张分表，查询时带上分库分表键后能直接将查询落在某个物理分库的某个物理分表。

用户这时就可以使用 YYYYMM 分库函数来解决：业务要求两年以内的每个月都对应一张分表（即一个月一张表），由于一年有 12 个月，所以至少需要创建 24 个物理分表才能满足用户的场景，而用户的 DRDS 有 8 个分库，所以每个分库应该建 3 张物理分表。建表语法如下所示：

```
create table test_yyyymm_tb(
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8
dbpartition by YYYYMM(create_time)
tbpartition by YYYYMM(create_time) tbpartitions 3;
```

只分库场景的建表语法：

```
create table YYYYMM(
    id int,
    name varchar(30) DEFAULT NULL,
    create time datetime DEFAULT NULL,
    primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8
dbpartition by YYYYMM(create_time);
```

注意事项

- YYYYMM 算法不支持对于每一个年月都独立对应一张分库，分库分表时必须固定分表数目。
- 当月份经历一个轮回（如 2013-03 是 2012-03 的一个轮回）后，同一个月份有可能被路由到同一个分库分表，请以实际的分表数目而定。

14.2.4.9 YYYYDD 按年日哈希

适用场景

适用于需要按年份与一年的天数进行分库的场景，建议该函数与 tbpartition YYYYDD(ShardKey) 联合使用。

使用说明

拆分键的数据类型必须是 DATE / DATETIME / TIMESTAMP 其中之一。

路由方式

根据拆分键的时间值的年份与一年的天数计算哈希值，然后再按分库/表数取余。

例如，YYYYDD(‘2012-12-31 12:12:12’) 等价于 (2012 * 366 + 365) % D（D 是分库数目/分表数）。

说明

"2012-12-31"是 2012 年第 366 天，所以"2012 * 366"。

算法计算方式

表 14-14 算法计算方式

条件	算法	举例

条件	算法	举例
分库拆分键 ≠ 分表拆分键	拆分键：yyyy-MM-dd 分库路由结果 = (yyyy * 366 + 一年第几天) % 分库数 分表路由结果 = (yyyy * 366 + 一年第几天) % 分表数	拆分键：2012-12-31 分库 ：(2012 * 366 + 366) % 8 = 6 分表：(2012 * 366 + 366) % 3 = 0
分库拆分键 = 分表拆分键(拆分键)	拆分键：yyyy-MM-dd 分表路由结果 = (yyyy * 366 + 一年第几天) % (分库数 * 分表数) 分库路由结果 = 分表路由结果 / 分表数	拆分键：2012-12-31 分库 ：(2012 * 366 + 366) % (8*3) = 6 分库 ：6 / 3 = 2

建表语法

假设用户的实例里已经分了 8 个物理库，现有一个业务想按年日进行分库。要求同一天的数据都能落在同一张分表，并且两年以内的每一天都能单独对应一张分表，查询时带上分库分表键后能直接将查询落在某个物理分库的某个物理分表。

用户这时就可以使用 YYYYDD 分库函数来解决：业务要求两年以内的每天都对应一张分表（即一天一张表），由于一年最多有 366 天，所以两年至少需要创建 732 个物理分表才能满足用户的场景。用户的 DRDS 有 8 个分库，所以每个分库应该建 92 张物理分表（732 / 8 = 91.5，取整为 92， 分表数最好是分库数的整数倍）。建表语法如下所示：

```
create table test_yyyydd_tb (
       id int,
       name varchar(30) DEFAULT NULL,
       create_time datetime DEFAULT NULL,
       primary key(id)
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8
    dbpartition by YYYYDD(create_time)
    tbpartition by YYYYDD(create_time) tbpartitions 92;
```

只分库的建表语法：

```
create table YYYYDD(
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8
dbpartition by YYYYDD(create_time);
```

注意事项

- YYYYDD 算法不支持对于每一个年日都独立对应一张分库，分库分表时必须固定分表数目。

- 当日期经历一个轮回（如 2013-03 是 2012-03 的一个轮回）后，同一个日期有可能被路由到同一个分库，请以实际的分库数目而定。

14.2.4.10 YYYYWEEK 按年周哈希

适用场景

适用于需要按年份与一年的周数进行分库的场景，建议该函数与 tbpartition YYYYWEEK(ShardKey) 联合使用。

使用说明

拆分键的数据类型必须是 DATE / DATETIME / TIMESTAMP 其中之一。

路由方式

根据拆分键的时间值的年份与一年的周数计算哈希值，然后再按分库/表数取余。

例如，YYYYWEEK(‘2012-12-31 12:12:12’) 等价于 (2013 * 54 + 1) % D（D 是分库数目/分表数）。

说明

此处“2012-12-31”是 2013 年第一周，所以"2013* 54"。

算法计算方式

表 14-15 算法计算方式

条件	算法	举例
分库拆分键 ≠ 分表拆分键	拆分键：yyyy-MM-dd 分库路由结果 = (yyyy * 54 + 一年第几周) % 分库数 分表路由结果 = (yyyy * 54 + 一年第几周) % 分表数	拆分键：2012-12-31 分库 ：(2013 * 54 + 1) % 8 = 7 分表：(2013 * 54 + 1) % 3 = 1
分库拆分键 = 分表拆分键(拆分键)	拆分键：yyyy-MM-dd 分表路由结果 = (yyyy * 54 + 一年第几周) % (分库数 * 分表数) 分库路由结果 = 分表路由结果 / 分表数	拆分键：2012-12-31 分库 ：(2013 * 54 + 1) % (8*3) = 7 分库 ：7 / 3 = 2

建表语法

假设用户的实例里已经分了 8 个物理库，现有一个业务想按年周进行分库。要求同一周的数据都能落在同一张分表，并且两年以内的每个周都单独对应一张分表，查询时带上分库分表键后能直接将查询落在某个物理分库的某个物理分表。

用户这时就可以使用 YYYYWEEK 分库函数来解决：业务要求两年以内的每个周都对应一张分表（就是一个周一张表），由于一年有近 53 个周(四舍五入)，所以两年至少需要创建 106 个物理分表才能满足用户的场景。而用户的 DRDS 有 8 个分库，所以每个分库应该建 14 张物理分表（14 * 8 = 112 > 106， 分表数最好是分库数的整数倍）。建表语法如下所示：

```
create table test_yyyymm_tb(
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8
dbpartition by YYYYWEEK(create_time)
tbpartition by YYYYWEEK(create_time) tbpartitions 14;
```

只分库的建表语法：

```
create table YYYYWEEK(
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
) ENGINE = InnoDB DEFAULT CHARSET = utf8
dbpartition by YYYYWEEK(create_time);
```

注意事项

- YYYYWEEK 算法不支持对于每一个年周都独立对应一张分库，分库分表时必须固定分表数目。
- 当周数经历一个轮回（如 2013 年第一周是 2012 年第一周的一个轮回）后，相同周数有可能被路由到同一个分库，请以实际的分库数目而定。

14.2.4.11 HASH 算法

适用场景

适用于需要将数据均匀分布的场景或需要按时间（年、月、日、周及其组合）对数据进行拆分的场景，在 SQL 查询条件中，使用“=”、“IN”之类运算符相对较多。

使用说明

拆分键的数据类型必须是整数类型（INT, INTEGER, BIGINT, MEDIUMINT, SMALLINT, TINYINT, DECIMAL(支持精度为 0 的情况)）或字符串类型(CHAR,VARCHAR)。若使用日期函数结合 hash 算法，拆分键的数据类型必须是 DATE / DATETIME / TIMESTAMP 其中之一。

路由方式

首先 102400 对分库数/分表数进行分范围。

假如逻辑库分 8 个分片，那么 102400/8=12800，则每一个分片对应的范围是：0=0-12799，1=12800-25599，2=25600-38399，3=38400-51199，4=51200-63999，5=64000-76799，6=76800-89599，7=89600-102399。

当计算路由结果时，计算拆分键值的 CRC32 值然后对 102400 取余，根据计算结果落到那个范围进行路由。

算法计算方式

方式一：拆分键非日期类型

表 14-16 **拆分键非日期类型**

条件	算法	举例
拆分键非日期类型	分库路由结果 = crc32(分库拆分键值) % 102400 分表路由结果 = crc32(分表拆分键值) % 102400	分库/分表：crc32(16) % 102400 = 49364; 49364 属于 3=38400-51199,则路由到分片 3；

方式二：拆分键是日期类型

表 14-17 支持的日期函数

日期函数	算法	举例
year()	year(yyyy-MM-dd)=yyyy	year（‘2019-10-11’)=2019
month()	month(yyyy-MM-dd)=MM	month（‘2019-10-11’)=10
weekofyear()	weekofyear(yyyy-MM-dd)=该日期是今年的第几周	weekofyear （‘2019-10-11’)=41
day()	day(yyyy-MM-dd)=该日期是月份的第几天	day （‘2019-10-11’)=11

表 14-18 **拆分键是日期类型**

条件	算法	举例
拆分键是日期类型	分库路由结果 = crc32(日期函数(分库拆分键值)) % 102400 分表路由结果 = crc32(日期函数(分库拆分键值)) % 102400	分库/分表：crc32(year（‘2019-10-11’)) % 102400 = 5404; 5404 属于 0=0-12799,则路由到分片 0；

建表语法

假设用户需要对 ID 列按 HASH 函数进行分库不分表:

```
create table hash_tb (
   id int,
```

```
   name varchar(30) DEFAULT NULL,
   create_time datetime DEFAULT NULL,
   primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by hash (ID);
```

假设用户需要对 ID 列按 HASH 函数既分库又分表:

```
create table mod_hash_tb (
   id int,
   name varchar(30) DEFAULT NULL,
   create_time datetime DEFAULT NULL,
   primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by hash (ID)
tbpartition by hash (ID) tbpartitions 4;
```

注意事项

无。

14.2.4.12 Range 算法

适用场景

适用于范围类操作较多的场景。在 SQL 查询条件中，使用“>”、“<”、“BETWEEN ... AND ...”之类运算符相对较多。

使用说明

拆分键的类型只支持整型类型、日期类型和日期函数结合，若使用日期函数，拆分键的数据类型必须是 date、datetime、timestamp 其一。

路由方式

根据拆分键，按照算法元数据的规则将数据行存储到相应的分片上。

建表时要设定元数据，假如逻辑库分 8 个分片，则元数据可以设定的范围为：1-2=0,3-4=1,5-6=2,7-8=3,9-10=4,11-12=5,13-14=6,default=7。根据拆分键的值在那个范围路由到对应的分片上。

算法计算方式

方式一：拆分键是整型

表 14-19 拆分键是整型时的计算方式

条件	算法	举例
拆分键是整型	分库路由结果 = 根据分库拆分键值在设定的元数据的范围进行路由 分表路由结果 = 根据分表拆分键值在设定的元数据的范围进行路由	分库 ：拆分值为 3 属于 3-4=1，则路由到 1 分片 分表：拆分值为 5 属于 5-6=2，则路由到 2 分片

方式二：拆分键是日期类型

表 14-20 支持的日期函数

日期函数	算法	举例
year()	year(yyyy-MM-dd)=yyyy	year(‘2019-10-11’)=2019
month()	month(yyyy-MM-dd)=MM	month(‘2019-10-11’)=10
weekofyear()	weekofyear(yyyy-MM-dd)=该日期是今年的第几周	weekofyear (‘2019-10-11’)=41
day()	day(yyyy-MM-dd)=该日期是月份的第几天	day (‘2019-10-11’)=11

表 14-21 拆分键是日期类型时的计算方式

条件	算法	举例
拆分键是日期类型	分库路由结果 =根据日期函数(分库拆分键值)在设定的元数据的范围进行路由 分表路由结果 =根据日期函数(分表拆分键值)在设定的元数据的范围进行路由	分库/分表 ：month(2019-10-11)=10 属于 9-10=4，则路由到 4 分片

建表语法

```
create table range_tb(
    id int,
    name varchar(30) DEFAULT NULL,
    create_time datetime DEFAULT NULL,
    primary key(id)
    )
dbpartition by range(id)
{
    1-2=0,
    3-4=1,
    5-6=2,
    7-8=3,
    9-10=4,
    11-12=5,
    13-14=6,
```

```
    default=7
};
```

注意事项

无。

14.3 DML

14.3.1 INSERT

INSERT 是将数据插入到数据库对象中的指令。

常用语法

```
INSERT [INTO] tbl_name
[(col_name,...)]
{VALUES | VALUE} ({expr },...),(...),...
[ ON DUPLICATE KEY UPDATE
col_name=expr
[, col_name=expr] ... ]
OR
INSERT [INTO] tbl_name
SET col_name={expr | DEFAULT}, ...
[ ON DUPLICATE KEY UPDATE
col_name=expr [, col_name=expr] ... ]
```

语法限制

- 不支持 INSERT DELAYED...；
- 不支持不包含拆分字段的 INSERT；
- 暂不支持 PARTITION 语法，建议不要使用 partition 表；
- INSERT 操作不支持“datetime”字段取值 1582 年及之前年份；
- INSERT 操作不支持插入拆分键值为 DEFAULT 关键字；
- 拆分表执行 INSERT 操作时如果指定了自增值，只影响该插入数据的自增值。后续数据插入时如果不指定自增值，仍以原自增值为基础进行自增；

14.3.2 REPLACE

REPLACE 用于往表中插入行或替换表中的行。

常用语法

```
replace into table(col1,col2,col3)
values(value1,value2,value3)
```

语法限制

- 暂不支持 PARTITION 语法；

- 当自增表格 ID 不存在时，使用 REPLACE 将会插入一条指定 ID 的数据，但不会自动生成 ID。

14.3.3 DELETE

DELETE 指令为用于删除表中符合条件的行。

常用语法

```
DELETE [IGNORE]
FROM tbl_name [WHERE where_condition]
```

语法限制

- WHERE 条件中不支持子查询（相关子查询和非相关子查询）；
- 不支持在多表删除中删除广播表中的数据（目标表列表中不可包含广播表）。

14.3.4 UPDATE

常用语法

```
UPDATE table_reference
SET col_name1={expr1} [, col_name2={expr2}] ...
[WHERE where_condition]
```

语法限制

- 不支持使用子查询（相关子查询和非相关子查询）；
- UPDATE 语句中的 where_condition 不支持计算表达式及其子查询；
- 不支持在多表更新中修改广播表（广播表中的列不可出现在 SET 中赋值语句的左侧）；
- 不支持更新逻辑表的拆分键字段，更新拆分键字段可能导致数据重新分布，DRDS 暂不支持；
- UPDATE 操作不支持“datetime”字段取值 1582 年及之前年份；
- UPDATE 操作不支持更新拆分键值为 DEFAULT 关键字；
- UPDATE 不支持在一个语句中对同一字段重复更新；
- UPDATE 不支持关联更新拆分键；
- UPDATE 不支持自关联更新；
- 关联更新中，不支持在目标列的赋值语句或表达式中引用其它目标列，将造成更新结果不符合预期。例如：

  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。
- 关联更新不支持不带关联条件的 Join；

14.3.5 SELECT

SELECT 通常用于查询一个或多个表中的数据。

常用语法

```
SELECT
[ALL | DISTINCT | DISTINCTROW ]
select_expr
[, select_expr ...]
[FROM table_references [WHERE where_condition]
[GROUP BY {col_name | expr | position} [ASC | DESC], ...]
[HAVING where_condition] [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

表 14-22 说明信息

语法	说明
select_expr	每个 select_expr 都指示一个您想要查询的列。
FROM table_references	指您将从哪个或哪些表中查询。
WHERE	关键词 WHERE 其后跟一个表达式，用于表示被选择的行所须满足的条件。
GROUP BY	语法中被使用的子句将按一定的顺序排列，GROUP BY 表示语句间关系，支持列名。如一个 HAVING 子句必须位于 GROUP BY 子句之后，并在 ORDER BY 子句之前。
ORDER BY	语法顺序排列的一种方式，表示语句间关系，支持列名和指定的排序方式（如 ASC、 DESC）。
LIMIT/OFFSET	对输出结果集的偏移量及大小给予约束，如：LIMIT 接受一个或者两个数字参数。

语法说明

- 暂不支持以空字符串作为别名;
- 不支持 select ... group by ... with rollup;
- 暂不支持 STRAIGHT_JOIN 和 NATURAL JOIN;
- select for update 仅支持简单查询，不支持 join、group by、order by、limit;
- 对于 UNION 中的每个 SELECT, DRDS 暂不支持使用多个同名的列，如下：

  # 如下 SQL 的 SELECT 中存在重复的列名，暂不支持 SELECT id, id, name FROM t1 UNION SELECT pk, pk, name FROM t2。

14.3.6 SELECT JOIN Syntax

常用语法

table_references:

```
table_reference [, table_reference] ...
```

table_reference:

```
table_factor | join_table
```

table_factor:

```
tbl_name [[AS] alias]
| table_subquery [AS] alias
| ( table_references )
```

join_table:

```
table_reference [INNER | CROSS] JOIN table_factor [join_condition]
| table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
| table_reference [{LEFT|RIGHT} [OUTER]] JOIN table_factor
```

join_condition:

```
ON conditional_expr
| USING (column_list)
```

语法限制

不支持 SELECT STRAIGHT_JOIN 和 NATURAL JOIN。

示例

```
select id,name from test1 where id=1;
select distinct id,name from test1 where id>=1;
select id,name from test1 order by id limit 2 offset 2;
select id,name from test1 order by id limit 2,2;
select  1+1,'test',id,id*1.1,now() from test1 limit 3;
select  current_date,current_timestamp;
select abs(sum(id)) from test1;
```

14.3.7 SELECT UNION Syntax

常用语法

```
SELECT ...UNION [ALL | DISTINCT]
SELECT ...[UNION [ALL | DISTINCT] SELECT ...]
```

示例

```
select userid from user union select orderid from ordertbl order by userid;
select userid from user union (select orderid from ordertbl group by orderid) order
by userid;
```

语法限制

对于 UNION 中的每个 SELECT，不支持使用多个同名的列。

14.3.8 SELECT Subquery Syntax

The Subquery as Scalar Operand

示例

```
SELECT (SELECT id FROM test1 where id=1);
SELECT (SELECT id FROM test2 where id=1)FROM test1;
SELECT UPPER((SELECT name FROM test1 limit 1)) FROM test2;
```

Comparisons Using Subqueries

语法

```
non_subquery_operand comparison_operator (subquery)
comparison_operator : = > < >= <= <> != <=> like
```

示例

```
select name from test1 where id > (select id from test2 where id=1);
select name from test1 where id = (select id from test2 where id=1);
select id from test1 where name like (select name from test2 where id=1);
```

Subqueries with ANY, IN, NOT IN, SOME,ALL,Exists,NOT Exists

语法

```
operand comparison_operator SOME (subquery)
operand comparison_operator ALL (subquery)
operand comparison_operator ANY (subquery)
operand IN (subquery)
operand not IN (subquery)
operand exists (subquery)
operand not exists (subquery)
```

示例

```
select id from test1 where id > any (select id from test2);
select id from test1 where id > some (select id from test2);
select id from test1 where id > all (select id from test2);
select id from test1 where id in (select id from test2);
select id from test1 where id not in (select id from test2);
select id from test1 where exists (select id from test2 where id=1);
select id from test1 where not exists (select id from test2 where id=1);
```

Derived Tables (Subqueries in the FROM Clause)

语法

```
SELECT ... FROM (subquery) [AS] tbl_name ...
```

示例

```
select id from (select id,name from test2 where id>1) a  order by a.id;
```

语法限制

- Derived Tables 必须拥有一个别名。
- Derived Tables 不可以成为 Correlated Subqueries，即不能包含子查询外部表的引用。
- 标量子查询在一些场景下当前不能得到正确结果，建议改写为 join，同时可提高性能。
- 不支持 HAVING 子句中的子查询，JOIN ON 条件中的子查询。
- 不支持 Row Subqueries。

14.3.9 不支持的 DML 语法列举

不支持的 DML 语法

表 14-23 DML 的语法限制

DML 语法	限制条件
DELETE 语句	不支持 PARTITION 子句。
UPDATE 语句	• 不支持跨分片子查询。 • 不支持拆分键的更新。
SELECT 语句	不支持类似 ORDER BY FIELD(id,1,2,3)这种自定义排序。
系统库查询	支持以下系统库查询： SELECT version() • information_schema.SCHEMA_PRIVILEGES • information_schema.TABLE_PRIVILEGES • information_schema.USER_PRIVILEGES • information_schema.SCHEMATA • information_schema.tables • information_schema.columns SHOW KEYS FROM `table ` FROM `database **说明** • 仅支持=、in、like 三种操作符，和 and 条件关联。 • 不支持子查询、关联查询、排序、聚合查询、LIMIT 等等复杂查询。 • information_schema.tables 和 information_schema.columns 支持<>操作符。

14.4 函数

支持的函数

表 14-24 操作符函数

函数表达式	示例
IN	SELECT * FROM Products WHERE vendor_id IN ('V000001', 'V000010') ORDER BY product_price
NOT IN	SELECT product_id, product_name FROM Products WHERE NOT vendor_id IN ('V000001', 'V000002') ORDER BY product_id
BETWEEN	SELECT id, product_id, product_name, product_price FROM Products WHERE id BETWEEN 000005 AND 000034 ORDER BY id
NOT… BETWEEN	SELECT product_id, product_name FROM Products WHERE NOT vendor_id BETWEEN 'V000002' and 'V000005' ORDER BY product_id
IS NULL	SELECT product_name FROM Products WHERE product_price IS NULL
IS NOT NULL	SELECT id, product_name FROM Products WHERE product_price IS NOT NULL ORDER BY id
AND	SELECT * FROM Products WHERE vendor_id = 'V000001' AND product_price <= 4000 ORDER BY product_price
OR	SELECT * FROM Products WHERE vendor_id = 'V000001' OR vendor_id = 'V000009'
NOT	SELECT product_id, product_name FROM Products WHERE NOT vendor_id = 'V000002'
LIKE	SELECT * FROM Products WHERE product_name LIKE 'NAME%' ORDER BY product_name
NOT LIKE	SELECT * FROM Products WHERE product_name NOT LIKE 'NAME%' ORDER BY product_name
CONCAT	SELECT product_id, product_name, Concat(product_id , '(', product_name ,')') AS product_test FROM Products ORDER BY product_id
+	SELECT 3 * 2+5-100/50
-	SELECT 3 * 2+5-100/50
*	SELECT order_num, product_id, quantity, item_price, quantity*item_price AS expanded_price FROM OrderItems WHERE order_num BETWEEN 000009 AND 000028 ORDER BY order_num
/	SELECT 3 * 2+5-100/50

函数表达式	示例
UPPER	SELECT id, product_id, UPPER(product_name) FROM Products WHERE id > 10 ORDER BY product_id
LOWER	SELECT id, product_id, LOWER(product_name) FROM Products WHERE id <= 10 ORDER BY product_id
SOUNDEX	SELECT * FROM Vendors WHERE SOUNDEX(vendor_name) = SOUNDEX('Test') ORDER BY vendor_name
IFNULL	SELECT IFNULL(product_id, 0) FROM Products; 说明 • 3 月 20 日之前已经创建的实例，拆分表不支持 IFNULL 与聚合函数的嵌套函数调用，如：select IFNULL(sum(yan),0) from shenhai，结果会和预期不一样。 • 3 月 20 日之后的实例，拆分表只支持 IFNULL 与聚合函数的一层嵌套函数调用。

表 14-25 时间日期函数

函数表达式	示例	支持范围
DAY()	SELECT * FROM TAB_DT WHERE DAY(dt)=21 SELECT * FROM TAB_DT WHERE dt='2018-12-21' INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')	-
MONTH()	SELECT * FROM TAB_DT WHERE MONTH(dt)=12 SELECT * FROM TAB_DT WHERE dt='2018-12-21' INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')	-
YEAR()	SELECT * FROM TAB_DT WHERE YEAR(dt)=2018 SELECT * FROM TAB_DT WHERE dt='2018-12-21' INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')	-
DAYOFYEAR()	SELECT * FROM TAB_DT WHERE DAYOFYEAR(dt)=365 SELECT * FROM TAB_DT WHERE dt='2018-12-31' INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')	-

函数表达式	示例	支持范围
DAYOFWEEK()	SELECT * FROM TAB_DT WHERE DAYOFWEEK(dt)=6 SELECT * FROM TAB_DT WHERE dt='2018-12-21' INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')	-
WEEKOFYEAR()	SELECT * FROM TAB_DT WHERE WEEKOFYEAR(dt)=51 SELECT * FROM TAB_DT WHERE dt='2018-12-21' INSERT INTO TAB_DT(id,dt) VALUES(1,'2018-05-22')	-
TIME()	SELECT * FROM TAB_DT WHERE TIME(dt)='01:02:03' SELECT * FROM TAB_DT WHERE dt='2021-01-01 01:02:03' INSERT INTO TAB_DT(id,dt) VALUES(1,'2021-01-01 01:02:03')	参数需为合法的时间、日期时间表达式或字符串。
TIME_TO_SEC()	SELECT * FROM TAB_DT WHERE TIME_TO_SEC(dt)=3603 SELECT * FROM TAB_DT WHERE dt='2021-01-01 01:00:03' INSERT INTO TAB_DT(id,dt) VALUES(1,'2021-01-01 01:00:03')	参数需为合法的时间、日期时间表达式或字符串。
SEC_TO_TIME()	SELECT * FROM TAB_DT WHERE SEC_TO_TIME(dt)='00:01:00' SELECT * FROM TAB_DT WHERE dt=60 INSERT INTO TAB_DT(id,dt) VALUES(1,60)	参数需为数值或可转化为数值的字符串。
SECOND()	SELECT * FROM TAB_DT WHERE SECOND(dt)=3 SELECT * FROM TAB_DT WHERE dt='2021-01-01 01:02:03' INSERT INTO TAB_DT(id,dt) VALUES(1,'2021-01-01 01:02:03')	参数需为合法的时间、日期时间表达式或字符串。
MINUTE()	SELECT * FROM TAB_DT WHERE MINUTE(dt)=2 SELECT * FROM TAB_DT WHERE dt='2021-01-01 01:02:03' INSERT INTO TAB_DT(id,dt) VALUES(1,'2021-01-01 01:02:03')	参数需为合法的时间、日期时间表达式或字符串。

函数表达式	示例	支持范围
HOUR()	SELECT * FROM TAB_DT WHERE HOUR(dt)=1 SELECT * FROM TAB_DT WHERE dt='2021-01-01 01:02:03' INSERT INTO TAB_DT(id,dt) VALUES(1,'2021-01-01 01:02:03')	参数需为合法的时间、日期时间表达式或字符串。
DAYNAME()	SELECT * FROM TAB_DT WHERE DAYNAME(dt)='Friday' SELECT * FROM TAB_DT WHERE dt='2021-01-01 01:02:03' INSERT INTO TAB_DT(id,dt) VALUES(1,'2021-01-01 01:02:03')	参数需为合法的日期、日期时间表达式或字符串。
MONTHNAME()	SELECT * FROM TAB_DT WHERE MONTHNAME(dt)='January' SELECT * FROM TAB_DT WHERE dt='2021-01-01 01:02:03' INSERT INTO TAB_DT(id,dt) VALUES(1,'2021-01-01 01:02:03')	参数需为合法的日期、日期时间表达式或字符串。
LAST_DAY()	SELECT * FROM TAB_DT WHERE LAST_DAY(dt)='2021-01-31' SELECT * FROM TAB_DT WHERE dt='2021-01-01 01:02:03' INSERT INTO TAB_DT(id,dt) VALUES(1,'2021-01-01 01:02:03')	参数需为合法的日期、日期时间表达式或字符串。
DAYOFWEEK()	SELECT * FROM TAB_DT WHERE DAYOFWEEK(dt)=6 SELECT * FROM TAB_DT WHERE dt='2021-01-01 01:02:03' INSERT INTO TAB_DT(id,dt) VALUES(1,'2021-01-01 01:02:03')	参数需为合法的日期、日期时间表达式或字符串。
DAYOFMONTH()	SELECT * FROM TAB_DT WHERE DAYOFWEEK(dt)=6 SELECT * FROM TAB_DT WHERE dt='2021-01-01 01:02:03' INSERT INTO TAB_DT(id,dt) VALUES(1,'2021-01-01 01:02:03')	参数需为合法的日期、日期时间表达式或字符串。
DAYOFYEAR()	SELECT * FROM TAB_DT WHERE DAYOFYEAR(dt)=365 SELECT * FROM TAB_DT WHERE dt='2021-12-31 01:02:03' INSERT INTO TAB_DT(id,dt) VALUES(1,'2021-12-31 01:02:03')	参数需为合法的日期、日期时间表达式或字符串。

函数表达式	示例	支持范围
WEEKOFYEAR()	SELECT * FROM TAB_DT WHERE WEEKOFYEAR(dt)=53 SELECT * FROM TAB_DT WHERE dt='2021-12-31 01:02:03' INSERT INTO TAB_DT(id,dt) VALUES(1,'2021-12-31 01:02:03')	参数需为合法的日期、日期时间表达式或字符串。
DATE_ADD(date, INTERVAL expr unit)	SELECT * FROM TAB_DT WHERE DATE_ADD(dt，INTERVAL 1 YEAR)='2022-01-01 01:02:03' SELECT * FROM TAB_DT WHERE dt='2021-01-01 01:02:03' INSERT INTO TAB_DT(id,dt) VALUES(1,'2021-01-01 01:02:03')	• date 参数需为合法的时间、日期、日期时间表达式或字符串。 • expr 为从 date 开始加减运算的间隔值，要求为整数或可转化为整数的字符串。 • unit 为单位，暂时可选 SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, YEAR。 • 当 date 为日期类型时，"1000-01-01"是下边界。在运算时如果存在越界行为，运算可能出错。
DATE_SUB(date, INTERVAL expr unit)	SELECT * FROM TAB_DT WHERE DATE_SUB(dt，INTERVAL -1 DAY)='2021-01-02 01:02:03' SELECT * FROM TAB_DT WHERE dt='2021-01-02 01:02:03' INSERT INTO TAB_DT(id,dt) VALUES(1,'2021-01-02 01:02:03')	• date 参数需为合法的时间、日期、日期时间表达式或字符串。 • expr 为从 date 开始加减运算的间隔值，要求为整数或可转化为整数的字符串。 • unit 为单位，暂时可选 SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, YEAR。 • 当 date 为日期类型时，"1000-01-01"是下边界。在运算时如果存在越界行为，运算可能出错。

表 14-26 数学函数

函数表达式	示例	支持范围
SQRT()	SELECT id, product_price, SQRT(product_price) AS price_sqrt FROM Products WHERE product_price < 4000 ORDER BY product_price	-
AVG()	SELECT AVG(product_price) AS avg_product FROM Products	-
COUNT()	SELECT COUNT(*) AS num_product FROM Products	-
MAX()	SELECT id, product_id, product_name, MAX(product_price) AS max_price FROM Products ORDER BY id	-
MIN()	SELECT id, product_id, product_name, MIN(product_price) AS min_price FROM Products ORDER BY id	-
SUM()	SELECT SUM(product_price) AS sum_product FROM Products	-
ROUND()	SELECT ROUND(product_price) AS round_product FROM Products	参数需为数值或可转化为数值的字符串。
SIN()	SELECT SIN(x) AS sin_x FROM math_tbl	参数需为数值或可转化为数值的字符串。
COS()	SELECT COS(x) AS cos_x FROM math_tbl	参数需为数值或可转化为数值的字符串。
TAN()	SELECT TAN(x) AS tan_x FROM math_tbl	参数需为数值或可转化为数值的字符串。
COT()	SELECT COT(x) AS cot_x FROM math_tbl	参数需为数值或可转化为数值的字符串。
FLOOR()	SELECT FLOOR(product_price) AS floor_product FROM Products	参数需为数值或可转化为数值的字符串。
CEILING()	SELECT CEILING(product_price) AS ceiling_product FROM Products	参数需为数值或可转化为数值的字符串。
ABS()	SELECT ABS(x) AS abs_x FROM math_tbl	参数的范围为-9223372036854775807 到9223372036854775807，超过此范围会报错。
LOG()	SELECT LOG(x) AS log_x FROM math_tbl	参数需为大于 0 的数值或可转化为相应数值的字符串。

函数表达式	示例	支持范围
LN()	SELECT LN(x) AS ln_x FROM math_tbl	参数需为大于 0 的数值或可转化为相应数值的字符串。

表 14-27 字符串函数

函数表达式	示例	支持范围
TRIM()	SELECT TRIM(' hello, world ') AS trim_character FROM Character	参数需为字符串相关类型。

说明

有支持范围的函数请在支持范围内使用，支持范围外的用法不保证与 mysql 的行为一致。

不支持的函数

表 14-28 函数的限制

函数	限制条件
ROW_COUNT()	暂不支持 ROW_COUNT()函数。
COMPRESS()	DRDS 暂不支持 COMPRESS()函数。如果无法确认函数是否能下推到 RDS，请不要使用该函数。
SHA()	DRDS 暂不支持 SHA()函数。如果无法确认函数是否能下推到 RDS，请不要使用该函数。
SHA1()	DRDS 暂不支持 SHA1()函数。如果无法确认函数是否能下推到 RDS，请不要使用该函数。
MD5()	DRDS 暂不支持 MD5()函数。如果无法确认函数是否能下推到 RDS，请不要使用该函数。
AES_ENCRYPT()	DRDS 暂不支持 AES_ENCRYPT()函数。如果无法确认函数是否能下推到 RDS，请不要使用该函数。
AES_DECRYPT()	DRDS 暂不支持 AES_DECRYPT()函数。如果无法确认函数是否能下推到 RDS，请不要使用该函数。
YEARWEEK()	DRDS 暂不支持 YEARWEEK()函数。如果无法确认函数是否能下推到 RDS，请不要使用该函数。

14.5 其他不支持语句

- 不支持触发器；
- 不支持临时表；
- 不支持 DO 语句；
- 不支持外键关联；
- 不支持 RESET 语句；
- 不支持 FLUSH 语句；
- 不支持 BINLOG 语句；
- 不支持 HANDLER 语句；
- 不支持 show warnings；
- 不支持 ‘:=’ 赋值运算符；
- 暂不支持<=>运算符；
- 暂不支持'IS UNKNOWN'表达式；
- 不支持用户管理及权限管理语句；
- 不支持 INSTALL/UNINSTALL PLUGIN 语句；
- 不支持分布式级别的存储过程及自定义函数；
- 库名、表名不可修改，拆分字段的名称和类型都不可以变更；
- 不支持 SHOW PROFILES、SHOW ERRORS 等多数运维 SHOW 语句；
- 不支持表维护语句，包括 ANALYZE/CHECK/CHECKSUM/OPTIMIZE/REPAIR TABLE；
- 不支持 session 变量赋值与查询，如 set @rowid=0;select @rowid:=@rowid+1,id from user；
- 不支持 SQL 语句中包含单行注释 '--' 或者多行（块）注释 '/*...*/'；
- 不完整支持系统变量查询，系统变量查询语句返回值为 RDS 实例相关变量值，而非 DRDS 引擎内相关变量值。例如 select @@autocommit 返回的值，并不代表 DRDS 当前事务状态；
- 不支持自定义事务隔离级别，目前 DRDS 只支持 READ COMMITTED 隔离级别。考虑到兼容性因素，对于设置数据库隔离级别的语句（如 SET GLOBAL TRANSACTION ISOLATION LEVEL REPEATABLE READ），DRDS 不会报错，但会忽略对事务隔离级别的修改；
- 不支持设置事务为只读（START TRANSACTION READ ONLY），考虑到兼容性因素，DRDS 会将只读事务的开启自动转换为开启读写事务；
- LIMIT/OFFSET 支持参数范围为 0-2,147,483,647；
- 不支持 group by 语句后添加 asc/desc 函数来实现排序语义。DRDS 自动忽略 group by 后的 asc/desc 关键字。MySQL 8.0.13 以下版本支持 group by 后添加 asc/desc 函数来实现排序语义，8.0.13 及以上版本已废弃该用法，使用时会报语法错误。推荐使用 order by 语句来保证排序语义；

权限级别支持情况：

- 全局层级（暂不支持）

- 数据库层级（支持）
- 表层级（支持）
- 列层级（暂不支持）
- 子程序层级（暂不支持）
- 用户层级（支持）

14.6 实用 SQL 语句

14.6.1 CHECK TABLE

14.6.1.1 检查当前逻辑库下所有逻辑表各分表的 DDL 一致性

用途：用于对某逻辑库所有的逻辑表的一致性情况进行全局概览。

命令格式：

```
check table
```

命令输出：

如果全部逻辑表都一致，输出结果为：

```
mysql> check table;
+------+---------------+------------+------------+-----------------+-------------+--------------------+----------+
| ID   | DATABASE_NAME | TABLE_NAME | TABLE_TYPE | DDL_CONSISTENCY | TOTAL_COUNT | INCONSISTENT_COUNT | DETAILS  |
+------+---------------+------------+------------+-----------------+-------------+--------------------+----------+
| 1    | test          | p1         | SHARDING   | Y               | 8           | 0                  |          |
| 2    | test          | b          | BROADCAST  | Y               | 8           | 0                  |          |
| 3    | test          | p2         | SHARDING   | Y               | 32          | 0                  |          |
| 4    | test          | s1         | SINGLE     | Y               | 1           | 0                  |          |
| 5    | test          | p20        | SHARDING   | Y               | 160         | 0                  |          |
+------+---------------+------------+------------+-----------------+-------------+--------------------+----------+
5 rows in set (0.24 sec)
```

如果存在不一致的逻辑表，输出结果为：

```
mysql> check table;
+------+---------------+------------+------------+-----------------+-------------+--------------------+-----------------------------------------------+
| ID   | DATABASE_NAME | TABLE_NAME | TABLE_TYPE | DDL_CONSISTENCY | TOTAL_COUNT | INCONSISTENT_COUNT | DETAILS                                       |
+------+---------------+------------+------------+-----------------+-------------+--------------------+-----------------------------------------------+
| 1    | test          | p2         | SHARDING   | N               | 32          | 2                  | `test_0004`.`p2_0`, `test_0006`.`p2_2` |
| 2    | test          | p1         | SHARDING   | Y               | 8           | 0                  |                                               |
| 3    | test          | b          | BROADCAST  | Y               | 8           | 0                  |                                               |
| 4    | test          | s1         | SINGLE     | Y               | 1           | 0                  |                                               |
| 5    | test          | p20        | SHARDING   | Y               | 160         | 0                  |                                               |
+------+---------------+------------+------------+-----------------+-------------+--------------------+-----------------------------------------------+
5 rows in set (0.23 sec)
```

输出详解：

每一行表示一个逻辑表的检查结果概况。

- DATABASE_NAME：逻辑库名称。
- TABLE_NAME：逻辑表名称。
- TABLE_TYPE：逻辑表类型。
  - SINGLE：单表。
  - BROADCAST：广播表。
  - SHARDING：拆分表。
- DDL_CONSISTENCY：该逻辑表对应所有物理表 DDL 是否一致。

- TOTAL_COUNT：该逻辑表有几个物理表。
- INCONSISTENT_COUNT：该逻辑表有几个物理表的 DDL 不一致。
- DETAILS：DDL 不一致的物理表名。

14.6.1.2 检查某一张逻辑表各分表的 DDL 一致性

用途：对特定一张逻辑表进行详细检查。

命令格式：

```
check table <table_name>
```

命令输出：

若返回结果集为空，表示该逻辑表各物理分表 DDL 都是一致的：

```
mysql> check table p1;
Empty set (0.02 sec)
```

若返回结果集不为空，表示各个不一致的物理表：

```
mysql> check table p2\G
*************************** 1. row ***************************
                 ID: 1
      DATABASE_NAME: test_0006
         TABLE_NAME: p2_2
         TABLE_TYPE: SHARDING
      EXTRA_COLUMNS:
    MISSING_COLUMNS:
  DIFFERENT_COLUMNS:
           KEY_DIFF:
        ENGINE_DIFF:
       CHARSET_DIFF:
       COLLATE_DIFF:
   EXTRA_PARTITIONS:
 MISSING_PARTITIONS:
DIFFERENT_PARTITIONS:
         EXTRA_INFO: TABLE NOT EXISTS
*************************** 2. row ***************************
                 ID: 2
      DATABASE_NAME: test_0004
         TABLE_NAME: p2_0
         TABLE_TYPE: SHARDING
      EXTRA_COLUMNS:
    MISSING_COLUMNS: `id2` int(11) DEFAULT NULL
  DIFFERENT_COLUMNS:
           KEY_DIFF:
        ENGINE_DIFF:
       CHARSET_DIFF:
       COLLATE_DIFF:
   EXTRA_PARTITIONS:
 MISSING_PARTITIONS:
DIFFERENT_PARTITIONS:
         EXTRA_INFO:
2 rows in set (0.03 sec)
```

输出详解：

每一行表示一个不一致的物理拆分表的详细检查结果。

- DATABASE_NAME：物理表所在的物理分库。
- TABLE_NAME：物理表表的表名。
- TABLE_TYPE：物理表所属逻辑表类型。
- EXTRA_COLUMNS：该物理表多出来的列。
- MISSING_COLUMNS：表示该物理表缺少的列。
- DIFFERENT_COLUMNS: 表示该物理表属性不一致的列(包括名称, 类型)。
- KEY_DIFF：表示该物理表不一致的索引。
- ENGINE_DIFF：表示该物理表不一致的引擎。
- CHARSET_DIFF：表示该物理表不一致的字符集。
- COLLATE_DIFF：表示该物理表不一致的排序规则。
- EXTRA_PARTITIONS：(分区表专用) 表示该物理表多出来的分区。
- MISSING_PARTITIONS：(分区表专用) 表示该物理表缺少的分区。

- DIFFERENT_PARTITIONS：(分区表专用) 表示该物理表属性不一致的分区。
- EXTRA_INFO：其他信息，如物理表缺失，将在这里显示。

14.6.2 SHOW RULE

命令格式：

```
show rule：查看数据库下每一个逻辑表的拆分情况。
show rule from <table_name>：查看数据库下指定逻辑表的拆分情况。
```

命令输出：

show rule 如下截图：

mysql> show rule;

ID	TABLE_NAME	BROADCAST	DB_PARTITION_KEY	DB_PARTITION_POLICY	DB_PARTITION_COUNT	DB_PARTITION_OFFSET	PARTITION_RANGE	TB_PARTITION_KEY	TB_PARTITION_POLICY	TB_PARTITION_COUNT	TB_PARTITION_OFFSET
0	history	0			1					1	
1	tbtest_hash_forerror	0			1					1	
2	single_table_1	0			1					1	
3	carrier	0			1					1	
4	employee3	0			1					1	
5	employee4	0			1					1	
6	supplier	1			8					1	
7	broadcast_table_1	1			8					1	
8	test6	1			8					1	
9	employee1	1			8					1	
10	category	1			8					1	
11	employee2	1			8					1	
12	range_id_dpt_1	0	id_col	range	8		0-10=0, 11-20=1, 21-30=2, 31-40=3, 41-50=4, 51-60=5, 61-70=6, 71-120=7, default=3			1	
13	mhash_bigint_dpt_1	0	bigint_col	mod_hash	8					1	
14	hash_id_dpt_1	0	id_col	hash	8					1	
15	mhash_varchar_dpt_1	0	varchar_col	mod_hash	8						

show rule from <*table_name*> 如下截图：

mysql> show rule from range_id_yd_datetime_3_tpt_1 ;

ID	TABLE_NAME	BROADCAST	DB_PARTITION_KEY	DB_PARTITION_POLICY	DB_PARTITION_COUNT	DB_PARTITION_OFFSET	PARTITION_RANGE	TB_PARTITION_KEY	TB_PARTITION_POLICY	TB_PARTITION_COUNT	TB_PARTITION_OFFSET
1	range_id_yd_datetime_3_tpt_1	0	id_col	range	8		0-10=0, 11-20=1, 21-30=2, 31-40=3, 41-50=4, 51-60=5, 61-70=6, 71-120=7, default=3	datetime_col	yyyydd	3	

1 row in set (0.00 sec)

输出详解：

TABLE_NAME ：表名。

BROADCAST：是否为广播表（0：否，1：是）。

DB_PARTITION_KEY：分库的拆分键，没有分库的话，值为空。

DB_PARTITION_POLICY：分库的拆分策略，取值包括哈希或 YYYYMM、YYYYDD、YYYYWEEK 等日期策略。

DB_PARTITION_COUNT：分库数。

DB_PARTITION_OFFSET ：分库偏移量。

PARTITION_RANGE：分库拆分算法为 range 时的拆分范围设置。

TB_PARTITION_KEY：分表的拆分键，没有分表的话，值为空。

TB_PARTITION_POLICY：分表的拆分策略，取值包括哈希或 MM、DD、MMDD、WEEK 等日期策略。

TB_PARTITION_COUNT：分表数。

TB_PARTITION_OFFSET ：分表偏移量。

14.6.3 SHOW TOPOLOGY

命令格式：

```
show topology from <table_name>：查看数据库下指定逻辑表的物理分布情况。
```

输出详解：

Rds_instance_id：rds 的实例 id。

HOST ：ip 。

PORT ：端口 。

DATABASE ：物理库 。

TABLE ：物理表 。

ROW_COUNT ：表的数据量(大致的值，在 information_schema.TABLES 取值)。

14.6.4 SHOW DATA NODE

命令格式：

```
show data node：查看物理分片的数据
```

输出详解：

RDS_INSTANCE_ID：rds 的实例 id。

PHYSICAL_NODE：物理节点。

HOST：主机号。

PORT：端口号。

14.6.5 TRUNCATE TABLE

14.6.5.1 HINT-DB

命令格式：

```
/*+db=<physical_db_name>*/ TRUNCATE TABLE <table_name>
```

描述：

删除对应的<*physical_db_name*>物理库下对应的所有的<*table_name*>的分表数据，其余分库的表不受影响。

14.6.5.2 HINT-TABLE

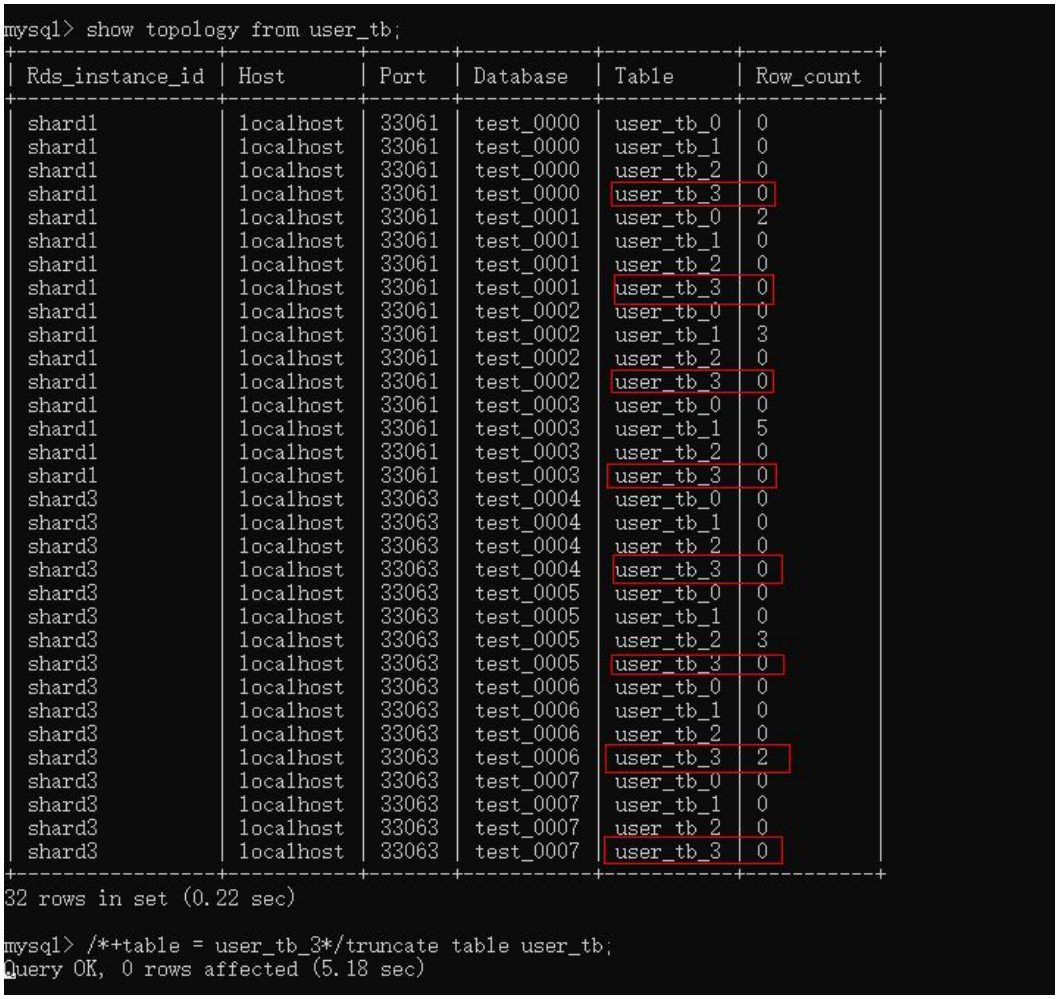
命令格式：

```
/*+table=<physical_table_name>*/ TRUNCATE TABLE <table_name>
```

描述：

删除当前库下表名<*physical_table_name*>的所有物理表的数据，其余分表不受影响。

删除前示例：



```
mysql> show topology from user_tb;
+-----------------+-----------+-------+-----------+-----------+-----------+
| Rds_instance_id | Host      | Port  | Database  | Table     | Row_count |
+-----------------+-----------+-------+-----------+-----------+-----------+
| shard1          | localhost | 33061 | test_0000 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_1 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0000 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_0 | 2         |
| shard1          | localhost | 33061 | test_0001 | user_tb_1 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0001 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_1 | 3         |
| shard1          | localhost | 33061 | test_0002 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0002 | user_tb_3 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_0 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_1 | 5         |
| shard1          | localhost | 33061 | test_0003 | user_tb_2 | 0         |
| shard1          | localhost | 33061 | test_0003 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0004 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0005 | user_tb_2 | 3         |
| shard3          | localhost | 33063 | test_0005 | user_tb_3 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0006 | user_tb_3 | 2         |
| shard3          | localhost | 33063 | test_0007 | user_tb_0 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_1 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_2 | 0         |
| shard3          | localhost | 33063 | test_0007 | user_tb_3 | 0         |
+-----------------+-----------+-------+-----------+-----------+-----------+
32 rows in set (0.22 sec)

mysql> /*+table = user_tb_3*/truncate table user_tb;
Query OK, 0 rows affected (5.18 sec)
```

删除后示例：

```
mysql> show topology from user_tb;
+----------------+-----------+-------+-----------+-----------+-----------+
| Rds_instance_id | Host      | Port  | Database  | Table     | Row_count |
+----------------+-----------+-------+-----------+-----------+-----------+
| shard1         | localhost | 33061 | test_0000 | user_tb_0 | 0         |
| shard1         | localhost | 33061 | test_0000 | user_tb_1 | 0         |
| shard1         | localhost | 33061 | test_0000 | user_tb_2 | 0         |
| shard1         | localhost | 33061 | test_0000 | user_tb_3 | 0         |
| shard1         | localhost | 33061 | test_0001 | user_tb_0 | 2         |
| shard1         | localhost | 33061 | test_0001 | user_tb_1 | 0         |
| shard1         | localhost | 33061 | test_0001 | user_tb_2 | 0         |
| shard1         | localhost | 33061 | test_0001 | user_tb_3 | 0         |
| shard1         | localhost | 33061 | test_0002 | user_tb_0 | 0         |
| shard1         | localhost | 33061 | test_0002 | user_tb_1 | 3         |
| shard1         | localhost | 33061 | test_0002 | user_tb_2 | 0         |
| shard1         | localhost | 33061 | test_0002 | user_tb_3 | 0         |
| shard1         | localhost | 33061 | test_0003 | user_tb_0 | 0         |
| shard1         | localhost | 33061 | test_0003 | user_tb_1 | 5         |
| shard1         | localhost | 33061 | test_0003 | user_tb_2 | 0         |
| shard1         | localhost | 33061 | test_0003 | user_tb_3 | 0         |
| shard3         | localhost | 33063 | test_0004 | user_tb_0 | 0         |
| shard3         | localhost | 33063 | test_0004 | user_tb_1 | 0         |
| shard3         | localhost | 33063 | test_0004 | user_tb_2 | 0         |
| shard3         | localhost | 33063 | test_0004 | user_tb_3 | 0         |
| shard3         | localhost | 33063 | test_0005 | user_tb_0 | 0         |
| shard3         | localhost | 33063 | test_0005 | user_tb_1 | 0         |
| shard3         | localhost | 33063 | test_0005 | user_tb_2 | 3         |
| shard3         | localhost | 33063 | test_0005 | user_tb_3 | 0         |
| shard3         | localhost | 33063 | test_0006 | user_tb_0 | 0         |
| shard3         | localhost | 33063 | test_0006 | user_tb_1 | 0         |
| shard3         | localhost | 33063 | test_0006 | user_tb_2 | 0         |
| shard3         | localhost | 33063 | test_0006 | user_tb_3 | 0         |
| shard3         | localhost | 33063 | test_0007 | user_tb_0 | 0         |
| shard3         | localhost | 33063 | test_0007 | user_tb_1 | 0         |
| shard3         | localhost | 33063 | test_0007 | user_tb_2 | 0         |
| shard3         | localhost | 33063 | test_0007 | user_tb_3 | 0         |
+----------------+-----------+-------+-----------+-----------+-----------+
32 rows in set (0.16 sec)
```

14.6.5.3 HINT-DB/TABLE

命令格式：

```
/*+db=<physical_db_name>,table=<physical_table_name>*/ TRUNCATE TABLE
<table_name>
```

描述：

删除库名为<physical_db_name>下表名为<physical_table_name>的物理分表数据，其余分库下其他分表不受影响。

14.6.5.4 补充说明

hint 对于所有的单表以及全局表失效，只对各种分表起作用。

14.6.6 HINT- ALLOW_ALTER_RERUN

命令格式：

```
/*+ allow_alter_rerun=true*/ALTER TABLE aaa_tb ADD schoolroll varchar(128)
not null comment '学籍'
```

描述：

使用该 hint 可支持命令重复执行不报错，共支持八种 alter table 的命令形式： add column、modify column、drop column、add index、drop index、change column、add partition 和 drop partition。

14.6.7 LOAD DATA

标准示例

LOAD DATA LOCAL INFILE '/data/qq.txt' IGNORE INTO TABLE test CHARACTER SET 'gbk' FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n'(id,sid,asf)

说明

如果数据中可能包含一些特殊字符，比如分割符转义符等，建议用引号扩起来，通过 OPTIONALLY ENCLOSED BY ""指定。

如果上述方法不起作用，可以把字段值中的引号替换成\"。

- 如果指定 **LOCAL** 关键词，则表明从客户端主机读文件。如果 local 没指定，出于安全考虑不支持此功能。
- 可以通过 **FIELDS TERMINATED BY** 指定字符之间的分割符号，默认值为\t 。
- 通过 **OPTIONALLY ENCLOSED BY** 忽略数据源字段中的符号。
- 通过 **LINES TERMINATED BY** 可以指定行之间的换行符，默认为\n。

说明

有些 windows 上的文本文件的换行符可能为\r\n，由于是不可见字符，所以请小心检查。

- 通过 **CHARACTER SET** 指定文件的编码，建议跟 RDS for MySQL 实例上物理库（分片）的编码一致，否则可能乱码。其中字符集编码必须用引号扩起来，否则会解析出错。
- 通过 **IGNORE** 或者 **REPLACE** 指定遇到重复记录是替换还是忽略。
- 目前列名必须指定，且必须包括分片字段，否则没办法确定路由。
- 其他参数参考 MySQL 的 load data infile 官方文档说明。其他参数的先后顺序不能乱，顺序参考官方说明。

须知

1. 数据导入阶段会在一定程度上影响 DRDS 以及 RDS for MySQL 实例性能，请选择在业务低峰时间导入。
2. 建议不要同时发起多个 LOAD DATA 请求。多个 LOAD DATA 同时进行，数据高并发写入，表锁竞争以及系统 IO 抢占会影响总体效率，可能会出现 SQL 事务超时现象，导致 LOAD DATA 全部失败。
3. 由于分布式事务的特性，使用 LOAD DATA 导入数据时，需要设置手动提交事务，以确保数据记录改动的准确无误。

   例如客户端可进行如下设置：

   mysql> set autocommit=0;

mysql> LOAD DATA LOCAL INFILE '/data/qq.txt' IGNORE INTO TABLE test CHARACTER SET 'gbk' FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n'(id,sid,asf);

mysql> commit;

限制

不支持：

- [IGNORE number {LINES | ROWS}]子句
- SET 子句

14.6.8 SHOW PHYSICAL PROCESSLIST

命令格式 1：

show physical processlist：展示后端 rds 的 processlist。

命令格式 2：

show physical processlist with info：在命令 1 的结果集中过滤掉 info 列为空的数据，只展示 info 列不为空的结果。

命令执行效果如下：

图 14-2 命令执行效果

Ip	Port	Instance_id	Type	Physical_thread_id	User	Host	Db	Command	Time	State	Info
localhost	33062	shard2	master	4	DDMRV	localhost:1	world_0007	Sleep	24373		
localhost	33062	shard2	master	5	DDMRV	localhost:1	world_0007	Sleep	24373		
localhost	33062	shard2	master	6	DDMRV	localhost:1	world_0004	Sleep	769		
localhost	33062	shard2	master	7	DDMRV	localhost:1	world_0006	Sleep	769		
localhost	33062	shard2	master	8	DDMRV	localhost:1	world_0007	Sleep	24373		
localhost	33062	shard2	master	9	DDMRV	localhost:1	world_0007	Sleep	24373		
localhost	33062	shard2	master	10	DDMRV	localhost:1	world_0004	Sleep	24373		
localhost	33062	shard2	master	11	DDMRV	localhost:1	world_0005	Sleep	769		
localhost	33062	shard2	master	12	DDMRV	localhost:1	world_0007	Query	0	starting	SHOW FULL
localhost	33062	shard2	master	13	DDMRV	localhost:1	world_0004	Sleep	24373		
localhost	33061	shard1	master	7	DDMRV	localhost:1	world_0000	Sleep	24372		
localhost	33061	shard1	master	8	DDMRV	localhost:1	world_0000	Sleep	19576		

命令输出详解：

ip:后端 rds 的 ip 地址。

port:后端 rds 的端口号。

instance id：后端 rds 的实例 id。

type:master 是读写 rds,readreplica 是只读 rds。

后面的列是对应的后端 rds 的 processlist 信息，与在后端 rds 直接执行 show processlist 获取的信息一致。

命令格式 3:

kill physical physical_thread_id@rds_ip:rds_port：kill 掉对应 rds 上的执行线程

须知

该功能仅支持内核 3.0.1 及以上的版本。

14.6.9 自定义 Hint 读写分离

DRDS 提供 HINT 来指定 SQL 语句是在主实例上执行还是在只读实例上执行。

HINT 支持以下两种格式：

- /*!mycat:db_type=**host***/
- /*+ db_type=**host** */

其中 **host** 可以是 master 或者 slave，master 代表主实例，slave 代表只读实例。

目前只支持 SELECT 语句。

说明

通常情况下，实现了读写分离之后，主实例上只能进行写操作，只读实例上只进行读操作。但是在某些特殊情况，需要在主实例上读取数据时，可以用自定义 Hint 的方式指定在主实例上进行读操作。此方式仅适用于查询功能。

14.6.10 自定义 Hint 跳过执行计划缓存

DRDS 提供 HINT 来控制 SELECT 语句是否跳过缓存的执行计划。

HINT 的格式为：/*!GAUSS:skip_plancache=**flag***/。

其中 **flag** 可以是 true 或者 false，true 代表跳过执行计划缓存，false 代表不跳过。

目前只支持 SELECT 语句。

14.6.11 通过 Hint 指定分片直接执行 SQL

命令格式：

```
/*+db=<physical_db_name>*/ <your query>;
```

描述：

直接在<physical_db_name>这一分片上执行对应 SQL 语句。

示例：

```
/*+db=test_0000*/ select * from t1;
```

使用限制：

- 此 hint 只对 SELECT/DML/TRUNCATE 语句起作用。
- 此 hint 仅在文本协议下工作，Prepare 协议下无法使用。

14.7 全局序列

14.7.1 全局序列概述

全局序列主要指基于 DB 的全局序列。

说明

- 支持修改自增序列初始值。
- 全局序列主要保证 ID 全局唯一，并不能保证一定是连续递增的。

表 14-29 全局序列支持的表类型

表类型	拆分表	广播表	单表
基于 DB 的全局序列	支持	支持	不支持

创建自增序列

步骤 1 使用客户端连接 DRDS 实例。

步骤 2 连接成功后，打开目标逻辑库。

步骤 3 输入命令创建自增序列。

create sequence *xxxxx* ;

说明

- xxxxx 代表序列名。
- 建议使用 bigint 型作为自增键的数据类型。tinyint、smallint、mediumint、integer、int 数据类型不建议作为自增键的类型，容易越界造成值重复。

----结束

删除自增序列

步骤 1 使用客户端连接 DRDS 实例。

步骤 2 连接成功后，打开目标逻辑库。

步骤 3 输入命令“show sequences”查看所有序列。

步骤 4 输入命令删除序列。

drop sequence *xxxxx* ;

drop sequence *DB.xxx*;

说明

- 对大小写不敏感。

- 如果序列属于某张表格（即创建这张表时有一列是自增列），不允许删除。

----结束

修改自增序列初始值

步骤 1 使用客户端连接 DRDS 实例。

步骤 2 连接成功后，打开目标逻辑库。

步骤 3 输入“show sequences”查看所有序列。

步骤 4 输入命令修改序列起始值。

alter sequence *xxxxx* **START WITH** *yyyyy;*

说明

- xxxxx 代表序列名。
- yyyyy 代表目标序列起始值。

----结束

查询自增序列

步骤 1 使用客户端连接 DRDS 实例。

步骤 2 连接成功后，打开目标逻辑库。

步骤 3 输入命令“show sequences”查看所有序列。

show sequences;

```
mysql> show sequences;
+----------------+------------+-----------+
| NAME           | START_WITH | INCREMENT |
+----------------+------------+-----------+
| WORLD.WORLDYML | 1          | 1000      |
| WORLD.YML      | 1          | 1000      |
+----------------+------------+-----------+
```

----结束

修改自增序列 Cache

须知

该功能仅支持内核 3.0.3 以上的版本。

步骤 1 使用客户端连接 DRDS 实例。

步骤 2 连接成功后，打开目标逻辑库。

步骤 3 输入命令“alter sequence test cache 5000”，修改表 test 的全局序列的 cache 值。

步骤 4 输入命令“show sequences”，查看 test 序列的 INCREMENT 值即是 cache 值。

----结束

刷新实例所有表自增序列

须知

该功能仅支持内核 3.0.4.1 以上的版本。

步骤 1 使用客户端连接 DRDS 实例。

步骤 2 输入命令“fresh all sequence start value”，更改所有逻辑库的所有 Sequence。

----结束

14.7.2 nextval、currval 在全局序列的使用

- nextval 返回下一个序列值，currval 返回当前序列值。其中 nextval 可以通过 nextval(n)返回 n 个唯一序列值。
- nextval(n)只能单独用在 select sequence.nextval(n)场景下并且不支持跨库操作。
- currval 不支持 currval(n)的用法。

操作步骤

步骤 1 使用客户端连接 DRDS 实例。

步骤 2 连接成功后，打开目标逻辑库。

步骤 3 输入命令创建全局序列。

create sequence seq_test;

```
mysql> create sequence seq_test;
Query OK, 0 rows affected (0.28 sec)
```

步骤 4 输入命令，返回下一个序列值。

select seq_test.nextval;

```
mysql> select seq_test.nextval;
+------------------+
| seq_test.NEXTVAL |
+------------------+
|                1 |
+------------------+
1 row in set (0.05 sec)

mysql> select seq_test.nextval;
+------------------+
| seq_test.NEXTVAL |
+------------------+
|                2 |
+------------------+
1 row in set (0.04 sec)

mysql> select seq_test.nextval;
+------------------+
| seq_test.NEXTVAL |
+------------------+
|                3 |
+------------------+
1 row in set (0.03 sec)
```

步骤 5 输入命令，获取当前序列值。

select seq_test.currval;

```
mysql> select seq_test.currval;
+------------------+
| seq_test.CURRVAL |
+------------------+
|                3 |
+------------------+
1 row in set (0.31 sec)

mysql> select seq_test.currval;
+------------------+
| seq_test.CURRVAL |
+------------------+
|                3 |
+------------------+
1 row in set (0.03 sec)
```

步骤 6 输入命令，批量获取序列值。

select seq_test.nextval(n);

```
mysql> select seq_test.nextval(5);
+------------------+
| seq_test.NEXTVAL |
+------------------+
|                4 |
|                5 |
|                6 |
|                7 |
|                8 |
+------------------+
5 rows in set (0.04 sec)
```

说明

- 批量获取序列值场景不支持跨库操作。
- 未使用过全局序列时，currval 的返回值是 0。

----结束

14.7.3 全局序列在 INSERT 或 REPLACE 语句中的使用

要想在同一个实例下实现跨逻辑库序列的全局唯一，可以在 insert 语句或者 replace 语句中结合全局序列一起使用。Insert 语句和 replace 语句支持 nextval 和 currval 两个方式序列的获取。其中，nextval 表示返回下一个序列值，currval 表示返回当前序列值。

可以通过 schema.seq.nextval、schema.seq.currval 使用，如果不指定 schema，默认是当前连接 schema 下的全局序列。

支持并发获取全局序列，在多 session 下并发通过 schema.seq.nextval 获取全局序列能够产生唯一值。

用法举例

两个逻辑库 dml_test_1、dml_test_2，里面都有表 test_seq。

表定义

表定义方法：create table test_seq(col1 bigint,col2 bigint) dbpartition by hash(col1)。

操作步骤

步骤 1 使用客户端连接 DRDS 实例。

步骤 2 连接成功后，打开目标逻辑库。

步骤 3 在库级别下，输入命令创建全局序列。

use dml_test_1;

create sequence seq_test;

```
mysql> use dml_test_1;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> create table test_seq(col1 bigint,col2 bigint) dbpartition by hash(col1);
Query OK, 0 rows affected (0.58 sec)

mysql> create sequence seq_test;
Query OK, 0 rows affected (0.73 sec)
```

步骤 4 使用以下语句，实现全局序列在 insert 语句或者 replace 语句的使用。

- **use dml_test_1**;

insert into test_seq(col1,col2)values(seq_test.nextval,seq_test.currval);

```
mysql> insert into test_seq(col1,col2)values(seq_test.nextval,seq_test.currval);
Query OK, 1 row affected (0.31 sec)

mysql> select * from test_seq;
+------+------+
| col1 | col2 |
+------+------+
|    1 |    1 |
+------+------+
1 row in set (0.05 sec)
```

- **use dml_test_2**;

insert into test_seq(col1,col2)values(dml_test_1.seq_test.nextval,dml_test_1.seq_test.currval);

```
mysql> use dml_test_2;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> create table test_seq(col1 bigint,col2 bigint) dbpartition by hash(col1);
Query OK, 0 rows affected (0.52 sec)

mysql> insert into test_seq(col1,col2)values(dml_test_1.seq_test.nextval,dml_test_1.seq_test.currval);
Query OK, 1 row affected (0.04 sec)

mysql> select * from test_seq;
+------+------+
| col1 | col2 |
+------+------+
|    2 |    2 |
+------+------+
1 row in set (0.05 sec)
```

由于全局序列是创建在逻辑库 dml_test_1 下的，在逻辑库 dml_test_2 下使用全局序列需要显式指定逻辑库 dml_test_1.seq_test.nextval、dml_test_1.seq_test.currval。

说明

- 全局序列结合 insert 和 replace 的使用只支持拆分表，不支持广播表和单表。
- nextval 和 currval 在 insert 和 replace 语句中是从左到右执行的，如果一条语句使用同一个全局序列 nextval 多次，每出现一次就递增一次。
- 全局序列是属于逻辑库的，删除逻辑库，所在删除逻辑库的全局序列也会被删除。

----结束

14.8 数据库管理语法

支持的数据库管理语法

- SHOW Syntax
- SHOW COLUMNS Syntax
- SHOW CREATE TABLE Syntax
- SHOW TABLE STATUS Syntax
- SHOW TABLES Syntax
- SHOW DATABASES

  若未搜索到对应的库，请先检查账号的细粒度权限。
- SHOW INDEX FROM
- SHOW VARIABLES Syntax

支持的数据库工具命令

- DESC Syntax
- USE Syntax
- EXPLAIN Syntax

  与 MySQL 内部的 explain 有所区别，DRDS 的 explain 显示的结果是当前语句路由到的节点描述。

不支持的数据库管理语法

表 14-30 数据库管理语句的限制

数据库管理语句	限制条件
数据库管理语句	• 不支持 SET Syntax 修改全局变量。 • 不支持 SHOW TRIGGERS 语法。 下列的 SHOW 指令会随机发到某个物理分片，每个物理分片如果在不同的 RDS for MySQL 实例上，查得的变量或者表信息可能不同： • SHOW TABLE STATUS • SHOW VARIABLES Syntax • check table 不支持 hash 和 key 分区表 • SHOW WARNINGS Syntax 不支持 LIMIT/COUNT 的组合 • SHOW ERRORS Syntax 不支持 LIMIT/COUNT 的组合

14.9 SQL 高级功能

表 14-31 SQL 高级功能的限制

SQL 高级功能	限制条件
SQL 高级功能	• 暂不支持 Prepare\EXECUTE 语法。 • 暂不支持用户自定义数据类型、自定义函数。 • 暂不支持视图、存储过程、触发器、游标。 • 暂不支持 BEGIN…END、LOOP…END LOOP、REPEAT…UNTIL…END REPEAT、WHILE…DO…END WHILE 等复合语句。 • 暂不支类似 IF ， WHILE 等流程控制类语句。 • 暂不支持的预处理类型： **PREPARE** Syntax **EXECUTE** Syntax • 不支持在建表语句中，对索引增加 COMMENT 形式的注释。

15 常见问题

15.1 DRDS 通用类

15.1.1 DRDS 提供哪些高可靠保障

数据完整性

DRDS 实例故障不会影响数据的完整性。

- 业务数据存储在数据节点分片中，DRDS 不存储业务数据。
- 逻辑库与逻辑表等配置信息存储在 DRDS 数据库中，DRDS 数据库主备高可用。

高可用机制

DRDS 采用多个无状态节点集群式部署模式，通过弹性负载均衡地址提供服务。

- DRDS 节点自身宕机类故障，对于已建立在故障节点上的连接会断连报错，DRDS 集群整体服务不受影响，通常情况下可在 5 秒内将故障节点从集群中剔除。
- 下挂数据节点故障，通常情况下可以在下挂数据节点恢复后 30 秒内完全恢复正常服务能力。

15.1.2 DRDS 自身是否会存储业务数据

DRDS 实例自身不存储客户业务相关数据，客户业务相关数据都存储在数据节点分片中。

DRDS 节点目前硬盘主要用来存储日志和一些临时文件，日志和临时文件会做定期清理，空间足够使用。

15.1.3 如何选择和配置安全组

DRDS 实例采用了 VPC 和安全组等网络安全保护措施，以下内容帮助您正确配置安全组。

通过 VPC 内网访问 DRDS 实例

DRDS 实例的访问和使用，包括客户端所在 ECS 访问 DRDS 实例，以及 DRDS 实例访问其关联的数据节点。

除了 ECS、DRDS 实例、数据节点必须处于相同 VPC 之外，还需要他们的安全组分别配置了正确的规则，允许网络访问。

1. 建议 ECS、DRDS、数据节点配置相同的安全组。安全组创建后，默认包含同一安全组内网络访问不受限制的规则。
2. 如果配置了不同安全组，可参考如下配置方式：

说明

- 假设 ECS、DRDS、RDS 分别配置了安全组：sg-ECS、sg-DDM、sg-RDS。
- 假设 DRDS 实例服务端口为 5066，RDS for MySQL 实例服务端口为 3306。
- 以下规则，远端可使用安全组，也可以使用具体的 IP 地址。

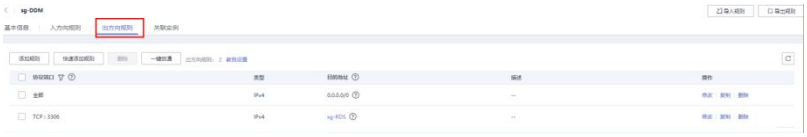
ECS 所在安全组需要增加图 17-1 中的规则，以保证客户端能正常访问 DRDS 实例：

图 15-1 ECS 安全组策略

DRDS 所在安全组需要增加图 17-2 和图 17-3 中的规则，以保证能访问数据节点，且被客户端访问。

图 15-2 DRDS 安全组入方向配置

图 15-3 DRDS 安全组出方向配置



数据节点所在安全组需要增加图 17-4 中的规则，以保证能被 DRDS 访问。

图 15-4 RDS 安全组配置

15.1.4 一个 DRDS 实例关联的不同数据节点之间是否可以共享数据

一个 DRDS 实例关联的不同数据节点之间数据是互相独立的，无法共享。

15.2 DRDS 使用类

15.2.1 如何解决 JDBC 驱动方式连接 DRDS 异常问题

MySQL 驱动（ JDBC）通过 Loadbalance 方式连接 DRDS，在某些场景下连接切换时会陷入死循环，最终导致栈溢出。

问题定位

1. 查看 APP 日志，定位异常原因。

   例如，从以下日志中分析出异常最终原因为栈溢出。

```
Caused by: java.lang.StackOverflowError
    at java.nio.HeapByteBuffer.<init>(HeapByteBuffer.java:57)
    at java.nio.ByteBuffer.allocate(ByteBuffer.java:335)
    at java.nio.charset.CharsetEncoder.encode(CharsetEncoder.java:795)
    at java.nio.charset.Charset.encode(Charset.java:843)
    at com.mysql.jdbc.StringUtils.getBytes(StringUtils.java:2362)
    at com.mysql.jdbc.StringUtils.getBytes(StringUtils.java:2344)
    at com.mysql.jdbc.StringUtils.getBytes(StringUtils.java:568)
    at com.mysql.jdbc.StringUtils.getBytes(StringUtils.java:626)
    at com.mysql.jdbc.Buffer.writeStringNoNull(Buffer.java:670)
    at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2636)
```

2. 分析溢出源。

   例如，从以下日志可以分析出，溢出原因为驱动内部陷入死循环。

```
at
com.mysql.jdbc.LoadBalancedConnectionProxy.pickNewConnection(LoadBalancedConnec
tionProxy.java:344)
at
com.mysql.jdbc.LoadBalancedAutoCommitInterceptor.postProcess(LoadBalancedAutoCo
mmitInterceptor.java:104)
at com.mysql.jdbc.MysqlIO.invokeStatementInterceptorsPost(MysqlIO.java:2885)
at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2808)
at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2483)
at com.mysql.jdbc.ConnectionImpl.setReadOnlyInternal(ConnectionImpl.java:4961)
```

```
at com.mysql.jdbc.ConnectionImpl.setReadOnly(ConnectionImpl.java:4954)
at
com.mysql.jdbc.MultiHostConnectionProxy.syncSessionState(MultiHostConnectionPro
xy.java:381)
at
com.mysql.jdbc.MultiHostConnectionProxy.syncSessionState(MultiHostConnectionPro
xy.java:366)
at
com.mysql.jdbc.LoadBalancedConnectionProxy.pickNewConnection(LoadBalancedConnec
tionProxy.java:344)
```

3. 查看使用的 MySQL 版本，为 5.1.44。

   查看该版本源代码，发现获取连接时，**LoadBalance** 会根据负载均衡策略更新连接，并将老连接的配置复制给新连接，在新连接 **AutoCommit** 为 **true**，新连接部分参数和老连接不一致，**loadBalanceAutoCommitStatementThreshold** 参数没有配置的场景下，会陷入死循环，更新连接函数调用同步参数函数，同步参数又调用更新连接，最终导致栈溢出。

解决方法

在连接 DRDS 的 URL 添加 **loadBalanceAutoCommitStatementThreshold=5&retriesAllDown=10** 参数。

```
//使用负载均衡的连接示例
//jdbc:mysql:loadbalance://ip1:port1,ip2:port2..ipN:portN/{db_name}
String url =
"jdbc:mysql:loadbalance://192.168.0.200:5066,192.168.0.201:5066/db_5133?loadBalance
AutoCommitStatementThreshold=5&retriesAllDown=10";
```

- loadBalanceAutoCommitStatementThreshold：表示连接上执行多少个语句后会重新选择连接。

  假设 loadBalanceAutoCommitStatementThreshold 设为 5，则当执行 5 个 sql 后（Queries 或者 updates 等），将会重新选择连接。若为 0 表示“粘性连接，不重新选择连接”。关闭自动提交时（autocommit=false）会等待事务完成再考虑是否重新选择连接。

15.2.2 如何选择 JDBC 驱动方式的版本和参数

DRDS 暂不支持使用 5.1.46 版本的 JDBC 驱动连接 DRDS，建议您使用以下版本的 JDBC 驱动：5.1.35-5.1.45。

JDBC 驱动下载地址：https://dev.mysql.com/doc/index-connectors.html。

JDBC URL 中推荐参数如表 17-1 所示。

表 15-1 参数

参数名称	参数说明	推荐取值
ip:port	连接地址和端口，用于连接 DRDS。	在 DRDS 管理控制台 DRDS 实例管理中查看连接地址。

参数名称	参数说明	推荐取值
db_name	连接逻辑库名称。	在 DRDS 管理控制台，DRDS 实例管理 > 逻辑库管理下查看逻辑库名称。
loadBalanceAutoCommitStatementThreshold	表示连接上执行多少个语句后会重新选择连接。 • 若取值为 5，则当执行 5 个 sql 后（Queries 或者 updates 等），将会重新选择连接。 • 若取值为 0，则表示“粘性连接，不重新选择连接”。 关闭自动提交时（autocommit=false）会等待事务完成再考虑是否重新选择连接。	5
loadBalanceHostRemovalGracePeriod	设置主机从负载均衡连接中移除的宽限时间。	15000
loadBalanceBlacklistTimeout	设置服务器在全局黑名单中存留的时间。	60000
loadBalancePingTimeout	使用负载均衡连接时，等待每个负载均衡连接 ping 响应的毫秒数。	5000
retriesAllDown	当所有的连接地址都无法连接时，轮询重试的最大次数。 重试次数达到阈值仍然无法获取有效连接，将会抛出 SQLException。	10
connectTimeout	和数据库服务器建立 socket 连接时的超时。 单位：毫秒，0 表示永不超时，适用于 JDK 1.4 及更高版本。	10000
socketTimeout	socket 操作（读写）超时。 单位：毫秒， 0 表示永不超时	根据业务实际情况合理配置。

15.2.3 使用 mysqldump 从 MySQL 导出数据非常缓慢的原因

mysqldump 客户端的版本和 DRDS 所支持的 MySQL 版本不一致，可能会导致从 MySQL 导出数据非常缓慢。

建议版本保持一致。

15.2.4 导入数据到 DRDS 后出现主键重复

在 DRDS 中创表时设置自增起始值，并确保起始值大于导入数据自增键的最大值。

15.2.5 如何处理数据迁移过程中自增列后报错：主键重复

重新设置自增主键的初始值为大于当前已有数据的最大值，执行如下语句：

```
ALTER SEQUENCE 库名.SEQ 名 START WITH 新初始值
```

15.2.6 如何处理配置参数未超时却报错

建议可将参数 SocketTimeOut 值调整或者去掉，默认为 0 则不断开连接。

15.2.7 如何处理 DRDS 逻辑库与 RDS 实例的先后关系

DRDS 逻辑库与关联的 RDS 强相关，不允许直接删除关联的 RDS，这会导致业务不可用且逻辑库也会删除失败。如果需要删除，先删除逻辑库再删除 RDS。

15.2.8 DRDS 逻辑库删除后，数据节点里面残留着部分预留的 DRDS 数据库和一些 DRDS 的账户，这些是否需要手动删除

如果不需要了，可以直接手动删除，释放空间。

15.3 SQL 语法类

15.3.1 DRDS 是否支持 SQL 跨库访问

不支持 SQL 通过带数据库名称的方式跨逻辑库访问。

DRDS 会自动去除 SQL 中的库名，比如 select * from dn1.item,会自动改成 select * from item。

15.3.2 DRDS 是否支持分布式 JOIN

DRDS 支持分布式 JOIN。

- 表设计时，增加字段冗余
- 支持跨分片的 JOIN，主要实现的方式有三种：广播表，ER 分片和 ShareJoin。
- DRDS 目前禁止多个表的跨库 update 和 delete。

15.3.3 如何进行 SQL 优化

- 尽量避免使用 LEFT JOIN 或 RIGHT JOIN，建议使用 INNER。
- 在使用 LEFT 或 RIGHT JOIN 时，ON 会优先执行，WHERE 条件在最后执行，所以在使用过程中，条件尽可能在 ON 语句中判断，减少 WHERE 的执行。
- 尽量少用子查询，改用 JOIN，避免大表全表扫描。

15.3.4 DRDS 是否支持数据类型强制转换

数据类型转换属于高级用法，DRDS 对 SQL 的兼容性会逐步完善，如有需要请提工单处理。

15.3.5 如何处理 INSERT 语句批量插入多条数据时报错

解决方案

建议拆分为多条 INSERT 语句插入，或在控制台配置参数管理页面，修改 max_allowed_packet 参数值，并重启实例使之生效。

15.4 RDS 相关类

15.4.1 数据库表名是否区分大小写

DRDS 默认对 databaseName、tableName、columnName 不区分大小写。

15.4.2 RDS for MySQL 哪些高危操作会影响 DRDS

RDS for MySQL 相关高危操作如表 17-2 所示。

表 15-2 RDS for MySQL 高危操作

操作类别	操作	操作影响
RDS for MySQL 控制台操作类	删除 RDS for MySQL 实例	RDS for MySQL 实例删除后，DRDS 关联该 RDS for MySQL 实例的逻辑库、逻辑表都无法使用。
	切换 RDS for MySQL 主备实例	切换主备实例可能造成短时间内的 RDS for MySQL 服务闪断，并有可能在主备同步时延过大的情况下，导致少量数据丢失。 • RDS for MySQL 实例主备切换过程中，DRDS 将无法进行创建逻辑库、创建表等操作。 • RDS for MySQL 实例主备切换后，DRDS 中 RDS for MySQL 实例 ID 不变。
	重启实例	重启过程中，RDS for MySQL 实例将不可用，DRDS 业务将会受影响。
	重置密码	RDS for MySQL 重置密码后，DRDS 这边创建逻辑库时输入重置后的密码即可。

操作类别	操作	操作影响
	修改参数模板	其中如下参数为固定值，如果修改，将会影响 DRDS 正常运行。 • 数据表名和序列名称不区分大小写，“lower_case_table_names”固定为“1”。 • 扩容场景，必须将“local_infile”配置为“ON”。
	修改安全组	将导致 DRDS 服务无法连接 RDS for MySQL 实例。
	修改 VPC	DRDS 实例与 RDS for MySQL 实例不在同一 VPC 中将导致无法互通。
	恢复	恢复数据可能会破坏数据完整性。
RDS for MySQL 客户端类	删除 DRDS 创建的物理库	删除物理库后，原数据将会丢失，新数据将无法写入。
	删除 DRDS 创建的物理帐号	删除物理帐号后将无法在 DRDS 上创建逻辑表。
	删除 DRDS 创建的物理表	删除物理表后，将导致 DRDS 数据丢失，DRDS 后续无法正常使用该逻辑表。
	修改 DRDS 创建的物理表名	将导致 DRDS 无法获取该逻辑表的数据，且后续无法正常使用。
	修改记录	如修改全局表记录，将会影响各分片数据一致性。
	修改白名单	需要确保 DRDS 服务在 RDS for MySQL 实例的白名单内，否则 DRDS 服务将无法访问 RDS for MySQL 实例。

15.4.3 如何处理表中存在主键重复的数据

场景一

DRDS 实例的逻辑表中已存在主键数据类型边界值的记录，如果插入的数据超过主键数据类型的范围，表中会出现主键重复的数据。

场景一处理方法

步骤 1 登录云服务管理控制台。

步骤 2 在 RDS for MySQL 的“实例管理”页面，查找 DRDS 实例对应的 RDS for MySQL 实例，单击目标 RDS for MySQL 实例名称，进入实例的“基本信息”页面。

步骤 3 在基本信息页面的左侧导航栏中选择“参数修改”。

步骤 4 在“参数”页签搜索“sql_mode”，单击“值”列中的下拉框，勾选“STRICT_ALL_TABLES”或“STRICT_TRANS_TABLES”方式，单击“保存”。

说明

“STRICT_ALL_TABLES”和“STRICT_TRANS_TABLES”方式属于严格模式。严格模式控制 MySQL 如何处理非法或丢失的输入值。

- 非法：数据类型错误或超出范围。
- 丢失：如果某列定义为非空列且没有 DEFAULT 值，当新插入的行不包含该列时，该行记录丢失。
- 在进行扩容时，若 DRDS 的实例版本低于 2.4.1.3。在选择 MySQL 实例的参数 sql_mode 时，请不要选择 ANSI_QUOTES。不能使用双引号来引用文字字符串，因为它们被解释为标识符。

  例如：select * from test where tb = "logic"。

关于“sql_mode”更多信息，请参考 Server SQL Modes。

步骤 5 在“DRDS 实例管理”页面，重启 DRDS 实例。

----结束

场景二

DRDS 实例的拆分表（hash\range\mod）联合主键，如果拆分键数据中有插入 0 和 1 的话，一定会出现重复主键。

场景二处理方法

步骤 1 登录云服务管理控制台。

步骤 2 在 RDS for MySQL 的“实例管理”页面，查找 DRDS 实例对应的 RDS for MySQL 实例，单击目标 RDS for MySQL 实例名称，进入实例的“基本信息”页面。

步骤 3 在基本信息页面的左侧导航栏中选择“参数修改”。

步骤 4 在“参数”页签搜索“sql_mode”，单击“值”列中的下拉框，勾选“NO_AUTO_VALUE_ON_ZERO”方式，单击“保存”。

步骤 5 在 DRDS 实例管理”页面，重启 DRDS 实例。

----结束

15.4.4 如何通过 show full innodb status 指令查询 RDS for MySQL 相关信息

通过 MySQL 客户端连接 DRDS 实例后，可直接输入 show full innodb status 指令查询该 DRDS 实例所关联的 RDS for MySQL 实例信息。可查询信息如：

- 当前的时间及自上次输出以来经过的时长。
- 可以使用命令 show full innodb status 来查看 master thread 的状态信息。
- 如果有高并发的工作负载，您需关注 SEMAPHORES 信号量，它包含了两种数据：事件计数器以及可选的当前等待线程的列表，如果有性能上的瓶颈，可使用这些信息来找出瓶颈。

15.5 连接管理类

15.5.1 本地环境是否可以连接 DRDS 实例

可以连接，首先绑定弹性公网 IP 到 DRDS 实例，就可以在本地环境使用连接工具（例如：Navicat）通过 EIP 访问 DRDS 实例。

Navicat 客户端连接 DRDS 实例

步骤 1 登录分布式关系型数据库服务，单击需要连接的 DRDS 实例名称，进入实例基本信息页面。

步骤 2 在“实例信息”模块的弹性公网 IP 单击“绑定”。绑定已申请分配的公网 IP。

步骤 3 在 DRDS 管理控制台左侧选择虚拟私有云图标。单击“访问控制>安全组”

步骤 4 在安全组界面，单击操作列的“配置规则”，进入安全组详情界面。在安全组详情界面，单击“添加规则”，弹出添加规则窗口。根据界面提示配置安全组规则，设置完成后单击“确定”即可。

说明

绑定弹性公网 IP 后，建议您在内网安全组中设置严格的出入规则，以加强数据库安全性。

步骤 5 打开 Navicat 客户端，单击“连接”。在新建连接窗口中填写主机 IP 地址（弹性公网 IP 地址）、用户名和密码（DRDS 帐号、密码）。

步骤 6 单击“连接测试”，若显示连接成功，单击“确定”，等待 1-2 分钟即可连接成功。连接失败会直接弹出失败原因，请修改后重试。

----结束

说明

通过其他可视化的 MySQL 工具（例如 Workbench）连接 DRDS 实例的操作与此章基本一致，不做详细描述。

15.5.2 MySQL 连接 DRDS 时出现乱码如何解决

MySQL 连接的编码和实际的编码不一致，可能导致 DRDS 解析时出现乱码。

通过“default-character-set=utf8”指定客户端连接的编码即可。

如下所示：

```
mysql -h 127.0.0.1 -P 5066 -D database --default-character-set=utf8 -u ddmuser -p
password
```