

Alibaba Cloud Apsara Stack Agility SE

User Guide

Version: 1912, Internal: V3.1.0

Issue: 20200311

Legal disclaimer

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

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequent

ial, exemplary, incidental, special, or punitive damages, including lost profits arising from the use or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

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

.

Document conventions

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

Style	Description	Example
{} or {a b}	This format is used for a required value, where only one item can be selected.	switch { <i>active</i> <i>stand</i> }

Contents

Legal disclaimer	I
Document conventions	I
1 ASCM console	1
1.1 What is the ASCM console?.....	1
1.2 User roles and permissions.....	2
1.3 Log on to the ASCM console.....	3
1.4 Webpage introduction.....	4
1.5 Initial configuration.....	5
1.5.1 Configuration description.....	5
1.5.2 Configuration process.....	6
1.6 Enterprise.....	7
1.6.1 Organizations.....	8
1.6.1.1 Create an organization.....	8
1.6.1.2 Query an organization.....	8
1.6.1.3 Modify organization information.....	8
1.6.1.4 Delete an organization.....	9
1.6.1.5 Obtain the AccessKey pair of an organization.....	9
1.6.2 Resource sets.....	10
1.6.2.1 Create a resource set.....	10
1.6.2.2 View the details of a resource set.....	10
1.6.2.3 Modify the name of a resource set.....	10
1.6.2.4 Add a member to a resource set.....	11
1.6.2.5 Delete a resource set.....	12
1.6.3 Roles.....	12
1.6.3.1 Create a custom role.....	12
1.6.3.2 View the details of a role.....	14
1.6.3.3 Modify custom role information.....	14
1.6.3.4 Delete a custom role.....	15
1.6.4 Users.....	15
1.6.4.1 System users.....	15
1.6.4.1.1 Create a user.....	15
1.6.4.1.2 Query a user.....	17
1.6.4.1.3 Modify user information.....	17
1.6.4.1.4 Change user roles.....	17
1.6.4.1.5 Modify a user logon policy.....	18
1.6.4.1.6 View the initial password of a user.....	18
1.6.4.1.7 Reset the password of a user.....	19
1.6.4.1.8 Disable and enable a user.....	20
1.6.4.1.9 Delete a user.....	20
1.6.4.2 Historical users.....	20

1.6.4.2.1 Query historical users.....	20
1.6.4.2.2 Restore historical users.....	21
1.6.5 Logon policies.....	21
1.6.5.1 Create a logon policy.....	21
1.6.5.2 Query a logon policy.....	23
1.6.5.3 Modify a logon policy.....	23
1.6.5.4 Delete a logon policy.....	24
1.6.6 User groups.....	24
1.6.6.1 Create a user group.....	24
1.6.6.2 Add users to a user group.....	25
1.6.6.3 Delete users from a user group.....	26
1.6.6.4 Add a role.....	27
1.6.6.5 Delete a role.....	27
1.6.6.6 Modify the name of a user group.....	27
1.6.6.7 Delete a user group.....	28
1.7 Configurations.....	28
1.7.1 Password policies.....	28
1.7.2 Menus.....	28
1.7.2.1 Create a menu.....	29
1.7.2.2 Modify a menu.....	30
1.7.2.3 Delete a menu.....	31
1.7.2.4 Display or hide menus.....	31
1.7.3 Specifications.....	32
1.7.3.1 Create specifications.....	32
1.7.3.2 View specifications.....	32
1.7.3.3 Disable specifications.....	32
1.7.3.4 Export specifications.....	33
1.8 Security.....	33
1.8.1 View operation logs.....	33
1.9 RAM.....	34
1.9.1 RAM introduction.....	34
1.9.2 Permission policy structure and syntax.....	35
1.9.3 RAM roles.....	38
1.9.3.1 View basic information about a RAM role.....	38
1.9.3.2 Create a RAM role.....	38
1.9.3.3 Add a permission policy.....	39
1.9.3.4 Modify the content of a RAM permission policy.....	39
1.9.3.5 Modify the name of a RAM permission policy.....	40
1.9.3.6 Add a RAM role to a user group.....	40
1.9.3.7 Grant permissions to a RAM role.....	41
1.9.3.8 Remove permissions from a RAM role.....	41
1.9.3.9 Modify a RAM role name.....	42
1.9.3.10 Delete a RAM role.....	42
1.9.4 RAM authorization policies.....	42
1.9.4.1 Create a RAM role.....	42

1.9.4.2 View the details of a RAM role.....	43
1.9.4.3 View RAM authorization policies.....	43
1.10 Personal information management.....	44
1.10.1 Modify personal information.....	44
1.10.2 Change your logon password.....	44
1.10.3 Switch the current role.....	45
1.10.4 View the AccessKey pair of your Apsara Stack tenant account....	46
2 Object Storage Service (OSS).....	47
2.1 What is OSS?.....	47
2.2 Instructions.....	47
2.3 Quick start.....	48
2.3.1 Log on to the OSS console.....	48
2.3.2 Create buckets.....	49
2.3.3 Upload objects.....	52
2.3.4 Obtain object URLs.....	53
2.4 Buckets.....	53
2.4.1 View bucket information.....	53
2.4.2 Delete buckets.....	54
2.4.3 Modify bucket ACLs.....	54
2.4.4 Configure static website hosting.....	55
2.4.5 Configure logging.....	56
2.4.6 Configure hotlink protection.....	57
2.4.7 Configure CORS.....	58
2.4.8 Manage lifecycle rules.....	60
2.5 Objects.....	62
2.5.1 Search for objects.....	62
2.5.2 Delete objects.....	63
2.5.3 Configure ACL for objects.....	64
2.5.4 Create folders.....	65
3 ApsaraDB for RDS.....	66
3.1 What is ApsaraDB for RDS?.....	66
3.2 Limits on ApsaraDB RDS for MySQL.....	67
3.3 Log on to the ApsaraDB for RDS console.....	68
3.4 Quick Start.....	69
3.4.1 Procedure.....	69
3.4.2 Create an instance.....	71
3.4.3 Initialization.....	73
3.4.3.1 Configure a whitelist.....	73
3.4.3.2 Create a privileged account.....	75
3.4.3.3 Create a standard account.....	77
3.4.3.4 Create a database.....	79
3.4.4 Connect to an ApsaraDB RDS for MySQL instance.....	80
3.5 Instances.....	82
3.5.1 Create an instance.....	83

3.5.2	View basic information about an instance.....	85
3.5.3	Restart an instance.....	85
3.5.4	Change specifications.....	86
3.5.5	Set a maintenance window.....	86
3.5.6	Change the data replication mode.....	87
3.5.7	Release an instance.....	89
3.6	Accounts.....	89
3.6.1	Create a database account.....	89
3.6.2	Reset your password.....	92
3.6.3	Modify account permissions.....	92
3.6.4	Delete an account.....	93
3.7	Databases.....	93
3.7.1	Create a database.....	93
3.7.2	Delete a database.....	94
3.8	Database connection.....	94
3.8.1	Change the endpoint of an instance.....	95
3.8.2	Switch the access mode.....	95
3.9	Monitoring and alerts.....	96
3.9.1	View resource and engine monitoring data.....	96
3.9.2	Set a monitoring frequency.....	98
3.10	Data security.....	100
3.10.1	Configure a whitelist.....	100
3.10.2	SQL audit.....	102
3.11	Database backup and restoration.....	104
3.11.1	Automatic backup.....	104
3.11.2	Manual backup.....	106
3.11.3	Restore data to a new instance (formerly known as cloning an instance).....	107
3.12	Read-only instances.....	110
3.12.1	Overview.....	110
3.12.2	Create a read-only instance.....	111
3.12.3	View the details of read-only instances.....	113
3.12.3.1	View instance details through a read-only instance.....	113
3.12.3.2	View instance details through the primary instance.....	114
3.13	Logs.....	115
3.14	Migrate data from an on-premises database to an ApsaraDB for RDS instance.....	115
3.14.1	Compress data.....	115
3.14.2	Migrate MySQL data.....	117
3.14.2.1	Use mysqldump to migrate MySQL data.....	117
4	AnalyticDB for PostgreSQL.....	121
4.1	What is AnalyticDB for PostgreSQL?.....	121
4.2	Quick start.....	121
4.2.1	Overview.....	121
4.2.2	Log on to the AnalyticDB for PostgreSQL console.....	122

4.2.3 Create an instance.....	123
4.2.4 Configure a whitelist.....	124
4.2.5 Create an initial account.....	126
4.2.6 Obtain the client tool.....	126
4.2.7 Connect to a database.....	127
4.3 Instances.....	132
4.3.1 Reset the password.....	132
4.3.2 View monitoring information.....	133
4.3.3 Switch the network type of an instance.....	133
4.3.4 Restart an instance.....	134
4.3.5 Import data.....	135
4.3.5.1 Import or export data from or to OSS in parallel.....	135
4.3.5.2 Import data from MySQL.....	146
4.3.5.3 Import data from PostgreSQL.....	148
4.3.5.4 Import data by using the \COPY statement.....	150
4.4 Databases.....	151
4.4.1 Overview.....	151
4.4.2 Create a database.....	151
4.4.3 Create a partition key.....	152
4.4.4 Construct data.....	152
4.4.5 Query data.....	153
4.4.6 Manage extensions.....	154
4.4.7 Manage users and permissions.....	155
4.4.8 Manage JSON data.....	156
4.4.9 Use HyperLogLog.....	163
4.4.10 Use the CREATE LIBRARY statement.....	165
4.4.11 Create and use the PL/Java UDF.....	166
4.5 Table.....	167
4.5.1 Create a table.....	168
4.5.2 Principles and scenarios of row store, column store, heap tables, and AO tables.....	174
4.5.3 Enable the column store and compression features.....	176
4.5.4 Add a field to a column store table and set the default value.....	177
4.5.5 Configure the table partition.....	179
4.5.6 Configure the sort key.....	180
4.6 Best practices.....	182
4.6.1 Configure memory and load parameters.....	182
5 Data Transmission Service (DTS).....	195
5.1 What is DTS?.....	195
5.2 Log on to the DTS console.....	195
5.3 Data migration.....	196
5.3.1 Migrate data between two user-created MySQL databases.....	196
5.3.2 Precheck items.....	203
5.3.2.1 Source database connectivity.....	203
5.3.2.2 Check the destination database connectivity.....	205

5.3.2.3 Binlog configurations in the source database.....	206
5.3.2.4 Referential integrity constraint.....	207
5.3.2.5 Existence of Federated tables.....	207
5.3.2.6 Permissions.....	208
5.3.2.7 Object name conflict.....	208
5.3.2.8 Schema existence.....	209
5.3.2.9 Source database server_id.....	210
5.3.2.10 Source database version.....	210
5.3.3 Database, table, and column name mapping.....	210
5.3.4 Configure an SQL filter for filtering the data to be migrated.....	213
5.3.5 Troubleshoot migration errors.....	213
5.4 Change tracking.....	215
5.4.1 Track data changes from a user-created MySQL database.....	215
5.4.2 Create consumer groups.....	219
5.4.3 Manage consumer groups.....	220
5.4.4 Modify objects for change tracking.....	221
6 KVStore for Redis.....	223
6.1 What is KVStore for Redis?.....	223
6.2 Quick start.....	223
6.2.1 Get started with KVStore for Redis.....	224
6.2.2 Log on to the KVStore for Redis console.....	225
6.2.3 Create an instance.....	226
6.2.4 Configure a whitelist.....	228
6.2.5 Connect to an instance.....	231
6.2.5.1 Use a Redis client.....	231
6.2.5.2 Use redis-cli.....	243
6.3 Instance management.....	244
6.3.1 Change the password.....	245
6.3.2 Configure a whitelist.....	245
6.3.3 Change the configuration of an instance.....	249
6.3.4 Set a maintenance window.....	250
6.3.5 Upgrade the minor version.....	251
6.3.6 Configure SSL encryption.....	251
6.3.7 Clear instance data.....	252
6.3.8 Release an instance.....	253
6.3.9 Manage database accounts.....	253
6.3.10 Use Lua scripts.....	255
6.3.11 Restart an instance.....	256
6.3.12 Export the instance list.....	257
6.4 Connection management.....	257
6.4.1 View endpoints.....	257
6.4.2 Apply for a public endpoint.....	258
6.4.3 Modify an endpoint.....	258
6.5 System parameters.....	259
6.6 Backup and recovery.....	268

6.7 Performance monitoring.....	270
6.7.1 View monitoring data.....	270
6.7.2 Customize metrics.....	270
6.7.3 Modify the monitoring frequency.....	272
7 Distributed Relational Database Service (DRDS).....	273
7.1 What is DRDS?.....	273
7.2 Quick start.....	275
7.3 Log on to the DRDS console.....	275
7.4 Instance management.....	276
7.4.1 Create a DRDS instance.....	276
7.4.2 Change configurations.....	278
7.4.3 Restart an instance.....	278
7.4.4 Release an instance.....	279
7.4.5 Monitor instances.....	279
7.4.5.1 View the monitoring information.....	280
7.4.5.2 Metrics.....	280
7.4.5.3 How metrics work.....	282
7.4.5.4 Prevent performance problems.....	283
7.4.5.4.1 Example 1: DRDS CPU utilization.....	283
7.4.5.4.2 Example 2: Logical RT and physical RT.....	285
7.4.5.4.3 Example 3: Logical QPS and physical QPS.....	288
7.4.5.4.4 Example 4: High memory usage.....	291
7.4.6 View the instance version.....	291
7.5 Account management.....	292
7.5.1 Overview.....	292
7.5.2 Create an account.....	293
7.5.3 Reset the password.....	295
7.5.4 Modify account permissions.....	296
7.5.5 Delete an account.....	299
7.6 Database management.....	300
7.6.1 Create a database.....	300
7.6.2 View a database.....	302
7.6.3 Perform smooth scale-out.....	302
7.6.4 View the monitoring information.....	305
7.6.5 Configure a whitelist.....	306
7.6.6 Delete a database.....	307
7.6.7 Fix database shard connections.....	307
7.7 Custom control commands.....	308
7.7.1 Help statement.....	308
7.7.2 Statements for viewing rules and topology.....	309
7.7.3 SQL tuning statements.....	316
7.7.4 Statistics query statements.....	325
7.7.5 SHOW PROCESSLIST and KILL commands.....	331
7.7.6 SHOW PROCESSLIST and KILL commands in earlier versions....	335
7.8 Custom hints.....	337

7.8.1 Introduction to hints.....	337
7.8.2 Read/write splitting.....	340
7.8.3 Specify a timeout period for an SQL statement.....	342
7.8.4 Specify a database shard to run an SQL statement.....	343
7.8.5 Scan all or some of database shards and table shards.....	346
7.8.6 INDEX hint.....	349
7.9 Custom hints for DRDS 5.2.....	351
7.9.1 Introduction to hints.....	351
7.9.2 Read/write splitting.....	352
7.9.3 Prevent the delay from a read-only ApsaraDB for RDS instance...	354
7.9.4 Specify a timeout period for an SQL statement.....	355
7.9.5 Specify a database shard to run an SQL statement.....	356
7.9.6 Scan all database shards and table shards.....	362
7.10 Distributed transactions.....	364
7.10.1 Distributed transactions based on MySQL 5.7.....	364
7.10.2 Distributed transactions based on MySQL 5.6.....	365
7.11 DDL operations.....	366
7.11.1 DDL statements.....	366
7.11.2 CREATE TABLE statement.....	367
7.11.2.1 Overview.....	367
7.11.2.2 Create a single-database non-partition table.....	368
7.11.2.3 Create a non-partition table in database shards.....	369
7.11.2.4 Create table shards in database shards.....	370
7.11.2.5 Use the primary key as the shard key.....	384
7.11.2.6 Create a broadcast table.....	385
7.11.2.7 Other attributes of the MySQL CREATE TABLE statement.....	385
7.11.3 ALTER TABLE statement.....	385
7.11.4 DROP TABLE statement.....	386
7.11.5 FAQ about DDL statements.....	386
7.11.6 DDL sharding functions.....	388
7.11.6.1 Overview.....	388
7.11.6.2 HASH.....	391
7.11.6.3 UNI_HASH.....	392
7.11.6.4 RIGHT_SHIFT.....	395
7.11.6.5 RANGE_HASH.....	396
7.11.6.6 MM.....	397
7.11.6.7 DD.....	398
7.11.6.8 WEEK.....	399
7.11.6.9 MMDD.....	400
7.11.6.10 YYYYMM.....	401
7.11.6.11 YYYYWEEK.....	402
7.11.6.12 YYYYDD.....	403
7.11.6.13 YYYYMM_OPT.....	404
7.11.6.14 YYYYWEEK_OPT.....	407
7.11.6.15 YYYYDD_OPT.....	409

7.12 Automatic protection of important SQL statements.....	410
7.13 DRDS sequence.....	411
7.13.1 Overview.....	411
7.13.2 Explicit sequence usage.....	416
7.13.3 Implicit sequence usage.....	421
7.13.4 Limits and precautions for DRDS sequence.....	423
7.14 Best practices.....	425
7.14.1 Select a shard key.....	425
7.14.2 Select the number of shards.....	427
7.14.3 Basic concepts of SQL optimization.....	428
7.14.4 SQL optimization methods.....	434
7.14.4.1 Overview.....	434
7.14.4.2 Single-table SQL optimization.....	435
7.14.4.3 JOIN query optimization.....	440
7.14.4.4 Subquery optimization.....	444
7.14.5 Select connection pools for an application.....	445
7.14.6 Connections in a DRDS instance.....	446
7.14.7 Perform instance upgrade.....	449
7.14.8 Perform scale-out.....	451
7.14.9 Troubleshoot slow SQL statements in DRDS.....	454
7.14.9.1 Details about a low SQL statement.....	454
7.14.9.2 Locate slow SQL statements.....	458
7.14.9.3 Locate nodes with performance loss.....	460
7.14.9.4 Troubleshoot performance loss.....	463
7.14.10 Handle DDL exceptions.....	464
7.14.11 Efficiently scan DRDS data.....	469
7.15 Appendix: DRDS terms.....	471
8 AnalyticDB for MySQL.....	482
8.1 What is AnalyticDB for MySQL?.....	482
8.2 Limits.....	483
8.3 Quick start.....	484
8.3.1 Log on to the AnalyticDB for MySQL console.....	485
8.3.2 Create a database cluster.....	485
8.3.3 Configure a whitelist.....	487
8.3.4 Create a database account.....	489
8.3.5 Connect to a database cluster.....	491
8.3.6 Create a database.....	492
8.3.7 Apply for a public endpoint.....	492
8.3.8 Synchronize data.....	493
8.4 Connect to a database cluster.....	494
8.4.1 Use the MySQL command-line tool to connect to AnalyticDB for MySQL.....	494
8.4.2 Use the code in a business system to connect to AnalyticDB for MySQL.....	494
8.4.2.1 C#.....	494

8.4.2.2 PHP.....	495
8.4.2.3 Python.....	496
8.4.2.4 Configure the Druid connection pool.....	497
8.4.2.5 Java.....	498
8.4.3 Enable PreparedStatement for a client in different programming languages.....	502
8.4.4 Use a client to connect to AnalyticDB for MySQL.....	503
8.4.4.1 SQL Workbench/J.....	503
8.4.4.2 DbVisualizer.....	505
8.4.4.3 DBeaver.....	506
8.4.4.4 Navicat.....	507
8.5 Manage database clusters.....	509
8.5.1 View monitoring information.....	509
8.5.2 Change specifications.....	509
8.5.3 Delete a cluster.....	510
8.6 Backup and restoration.....	510
8.6.1 Back up data.....	510
8.6.2 Restore data.....	511
8.7 Diagnostics and optimization.....	511
8.7.1 Use functions related to slow SQL statements.....	511
8.8 Account and permission management.....	511
8.8.1 Permission model.....	511
8.8.2 Manage database accounts and permissions.....	513
8.9 Data visualization.....	514
8.9.1 Tableau.....	514
8.9.2 QlikView.....	515
8.9.3 FineReport.....	516
8.10 Data migration and synchronization.....	518
8.10.1 Use Kettle to synchronize the local data to AnalyticDB for MySQL.....	518
8.11 SQL manual.....	521
8.11.1 Data types.....	521
8.11.2 Data definition statements.....	525
8.11.2.1 CREATE DATABASE.....	525
8.11.2.2 CREATE TABLE.....	526
8.11.2.3 ALTER TABLE.....	531
8.11.2.4 CREATE VIEW.....	534
8.11.2.5 DROP DATABASE.....	534
8.11.2.6 DROP TABLE.....	535
8.11.2.7 DROP VIEW.....	535
8.11.3 Data manipulation statements.....	536
8.11.3.1 INSERT INTO.....	536
8.11.3.2 REPLACE INTO.....	537
8.11.3.3 INSERT SELECT FROM.....	538
8.11.3.4 REPLACE SELECT FROM.....	539

8.11.3.5 INSERT OVERWRITE INTO SELECT.....	540
8.11.3.6 UPDATE.....	540
8.11.3.7 DELETE.....	541
8.11.3.8 TRUNCATE TABLE.....	542
8.11.3.9 KILL PROCESS.....	543
8.11.3.10 SHOW PROCESSLIST.....	543
8.11.4 SELECT.....	545
8.11.4.1 Syntax.....	545
8.11.4.2 WITH.....	546
8.11.4.3 GROUP BY.....	547
8.11.4.4 HAVING.....	549
8.11.4.5 JOIN.....	550
8.11.4.6 LIMIT.....	550
8.11.4.7 ORDER BY.....	551
8.11.4.8 Subqueries.....	551
8.11.4.9 UNION, INTERSECT, and EXCEPT.....	552
8.11.5 CREATE USER.....	553
8.11.6 GRANT.....	554
8.11.7 REVOKE.....	555
8.11.8 Query users.....	556
8.11.9 RENAME USER.....	556
8.11.10 DROP USER.....	557
8.11.11 SHOW.....	557
8.12 System functions.....	559
8.12.1 Aggregate functions.....	559
8.12.1.1 AVG.....	559
8.12.1.2 BIT_AND.....	559
8.12.1.3 BIT_OR.....	560
8.12.1.4 BIT_XOR.....	560
8.12.1.5 COUNT.....	560
8.12.1.6 MAX.....	561
8.12.1.7 MIN.....	561
8.12.1.8 STD/STDDEV.....	561
8.12.1.9 STDDEV_POP.....	562
8.12.1.10 STDDEV_SAMP.....	562
8.12.1.11 SUM.....	562
8.12.1.12 VAR_POP.....	563
8.12.1.13 VAR_SAMP.....	563
8.12.1.14 VARIANCE.....	563
8.12.2 Date and time functions.....	564
8.12.2.1 ADDDATE/DATE_ADD.....	564
8.12.2.2 ADDTIME.....	565
8.12.2.3 CONVERT_TZ.....	566
8.12.2.4 CURDATE.....	566
8.12.2.5 CURTIME.....	566

8.12.2.6 DATE.....	567
8.12.2.7 DATE_FORMAT.....	567
8.12.2.8 SUBDATE/DATE_SUB.....	569
8.12.2.9 DATEDIFF.....	570
8.12.2.10 DAY/DAYOFMONTH.....	570
8.12.2.11 DAYNAME.....	571
8.12.2.12 DAYOFWEEK.....	571
8.12.2.13 DAYOFYEAR.....	572
8.12.2.14 EXTRACT.....	572
8.12.2.15 FROM_DAYS.....	573
8.12.2.16 FROM_UNIXTIME.....	573
8.12.2.17 HOUR.....	574
8.12.2.18 LAST_DAY.....	574
8.12.2.19 LOCALTIME/LOCALTIMESTAMP/NOW.....	574
8.12.2.20 MAKEDATE.....	575
8.12.2.21 MAKETIME.....	575
8.12.2.22 MINUTE.....	576
8.12.2.23 MONTH.....	576
8.12.2.24 MONTHNAME.....	577
8.12.2.25 PERIOD_ADD.....	577
8.12.2.26 PERIOD_DIFF.....	577
8.12.2.27 QUARTER.....	578
8.12.2.28 SEC_TO_TIME.....	578
8.12.2.29 SECOND.....	579
8.12.2.30 STR_TO_DATE.....	579
8.12.2.31 SUBTIME.....	579
8.12.2.32 SYSDATE.....	580
8.12.2.33 TIME.....	580
8.12.2.34 TIME_FORMAT.....	581
8.12.2.35 TIME_TO_SEC.....	581
8.12.2.36 TIMEDIFF.....	582
8.12.2.37 TIMESTAMP.....	582
8.12.2.38 TIMESTAMPADD.....	582
8.12.2.39 TIMESTAMPDIFF.....	583
8.12.2.40 TO_DAYS.....	584
8.12.2.41 TO_SECONDS.....	584
8.12.2.42 UNIX_TIMESTAMP.....	585
8.12.2.43 UTC_DATE.....	585
8.12.2.44 UTC_TIME.....	585
8.12.2.45 UTC_TIMESTAMP.....	586
8.12.2.46 WEEK.....	586
8.12.2.47 WEEKDAY.....	587
8.12.2.48 WEEKOFYEAR.....	588
8.12.2.49 YEAR.....	588
8.12.2.50 YEARWEEK.....	588

8.12.3 String functions	589
8.12.3.1 ASCII.....	589
8.12.3.2 BIN.....	589
8.12.3.3 BIT_LENGTH.....	590
8.12.3.4 CHAR.....	590
8.12.3.5 CHAR_LENGTH/CHARACTER_LENGTH.....	590
8.12.3.6 CONCAT.....	591
8.12.3.7 CONCAT_WS.....	591
8.12.3.8 ELT.....	591
8.12.3.9 EXPORT_SET.....	592
8.12.3.10 FIELD.....	592
8.12.3.11 FIND_IN_SET.....	593
8.12.3.12 FORMAT.....	593
8.12.3.13 HEX.....	593
8.12.3.14 INSTR.....	594
8.12.3.15 LEFT.....	594
8.12.3.16 LENGTH/OCTET_LENGTH.....	594
8.12.3.17 LIKE.....	595
8.12.3.18 LOCATE.....	595
8.12.3.19 LOWER/LCASE.....	596
8.12.3.20 LPAD.....	596
8.12.3.21 LTRIM.....	596
8.12.3.22 MAKE_SET.....	597
8.12.3.23 MID.....	597
8.12.3.24 OCT.....	597
8.12.3.25 POSITION.....	598
8.12.3.26 REPEAT.....	598
8.12.3.27 REPLACE.....	598
8.12.3.28 REVERSE.....	599
8.12.3.29 RIGHT.....	599
8.12.3.30 RLIKE/REGEXP.....	599
8.12.3.31 RPAD.....	600
8.12.3.32 RTRIM.....	600
8.12.3.33 SPACE.....	600
8.12.3.34 STRCMP.....	601
8.12.3.35 SUBSTR/SUBSTRING.....	601
8.12.3.36 SUBSTRING_INDEX.....	602
8.12.3.37 TRIM.....	602
8.12.3.38 UPPER/UCASE.....	603
8.12.4 Numeric functions	603
8.12.4.1 ABS.....	603
8.12.4.2 ROUND.....	603
8.12.4.3 SQRT.....	604
8.12.4.4 LN.....	604
8.12.4.5 LOG.....	605

8.12.4.6 LOG2.....	605
8.12.4.7 PI.....	605
8.12.4.8 LOG10.....	605
8.12.4.9 POWER/POW.....	606
8.12.4.10 RADIANS.....	606
8.12.4.11 DEGREES.....	606
8.12.4.12 SIGN.....	607
8.12.4.13 CEILING/CEIL.....	607
8.12.4.14 FLOOR.....	607
8.12.4.15 EXP.....	608
8.12.4.16 COS.....	608
8.12.4.17 ACOS.....	608
8.12.4.18 TAN.....	609
8.12.4.19 ATAN.....	609
8.12.4.20 ATAN2.....	609
8.12.4.21 COT.....	610
8.12.4.22 ASIN.....	610
8.12.4.23 SIN.....	610
8.12.5 Arithmetic operators.....	610
8.12.6 Bit functions and operators.....	612
8.12.7 Control flow functions.....	614

1 ASCM console

1.1 What is the ASCM console?

The Apsara Stack Cloud Management (ASCM) console is a service capability platform based on the Alibaba Cloud Apsara Stack platform and designed for government and enterprise customers. This platform improves IT management and troubleshooting and is dedicated to providing a leading service capability platform of the cloud computing industry. It provides large-scale and cost-efficient end-to-end cloud computing and big data services for customers in industries such as government, education, healthcare, finance, and enterprise.

Overview

The ASCM console simplifies the management and deployment of physical and virtual resources by building an Apsara Stack platform that supports various business types of government and enterprise customers. The console helps you build your business systems in a simple and quick manner, fully improve resource utilization, and reduce O&M costs, allowing you to shift your focus from O&M to business. The console brings the Internet economy model to government and enterprise customers, and builds a new ecosystem chain based on cloud computing

Workflow

ASCM console operations are divided into the following parts:

- 1. System initialization: This part is designed to complete basic system configurations, such as creating organizations, resource sets, and users, creating basic resources such as VPCs, and creating contacts and contact groups in CloudMonitor.**
- 2. Cloud resource creation: This part is designed to create resources as needed.**
- 3. Cloud resource management: This part is designed to complete resource management operations, such as starting, using, and releasing resources and changing resource configurations.**

1.2 User roles and permissions

This topic describes roles and their permissions.

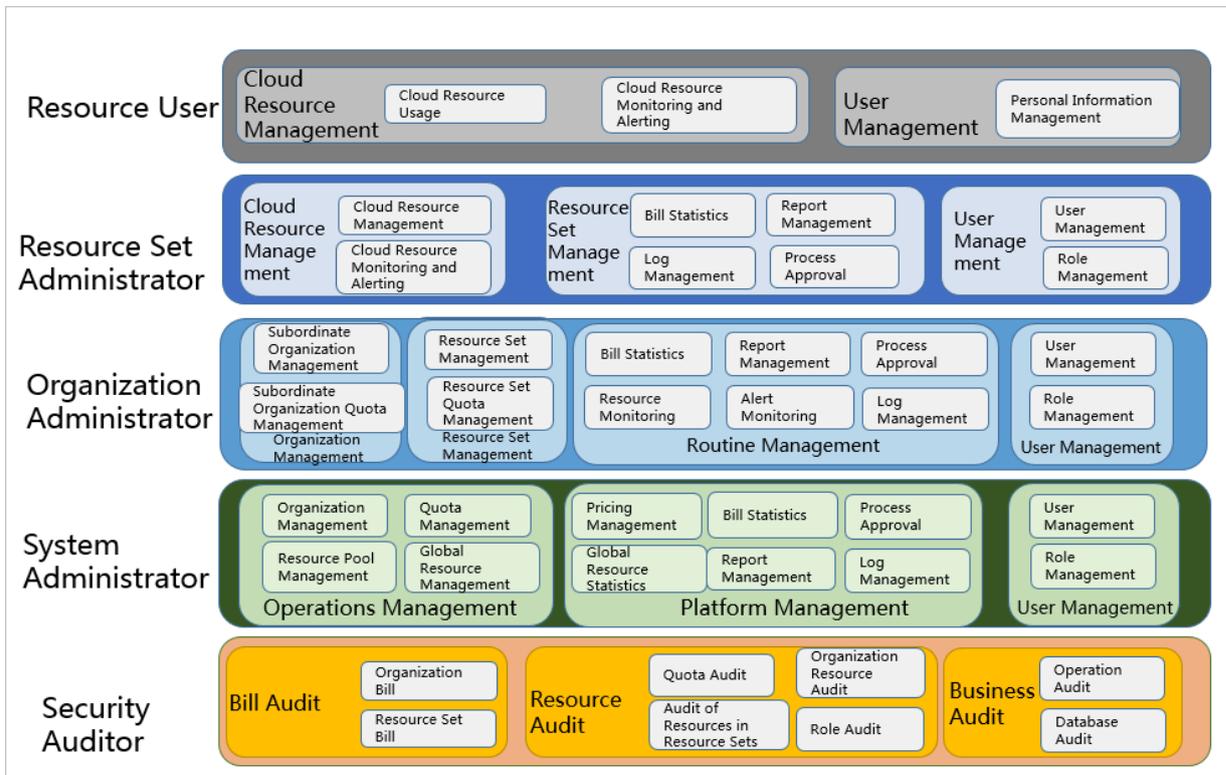


Table 1-1: Roles and permissions

Role name	Role permission
Resource user	This role has the permissions to view and modify resources in a resource set and create alarm rules.
Resource set administrator	This role has the permissions to create, modify, and delete resources in a resource set and manage the users of the resource set.
Organization administrator	This role has the permissions to manage an organization and its subordinate organizations, create, modify, and delete the resources of the organizations, create and view alarm rules for the resources, and export reports.
System administrator	This role has read and write permissions on all resources.

Role name	Role permission
Security auditor	This role has read-only permissions on all resources.
Super administrator	This role has the permissions to initialize the system and create system administrators.

1.3 Log on to the ASCM console

This topic describes how to log on to the Apsara Stack Cloud Management (ASCM) console.

Prerequisites

- Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel . The URL used to access the ASCM console is in the following format: `http://IP address or domain name of the ASCM console/manage`.
- We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to access the ASCM console. Press Enter.
2. Enter your username and password.

The system has a default super administrator, whose username is super. The super administrator can create system administrators. A system administrator can create system users and notify the users of the default passwords by SMS or email.



Note:

When you log on to the ASCM console for the first time, you must modify the password of your username as instructed. For security concerns, your password must meet the minimum complexity requirements: The password must be 8 to 20 characters in length and must contain at least two types of the following characters: letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs (\$), and percent signs (%).

3. Click Login to go to the ASCM console homepage.

1.4 Webpage introduction

The webpage of the ASCM console consists of the main menu bar, information area of the currently logged-on user, and operation area.

ASCM console page

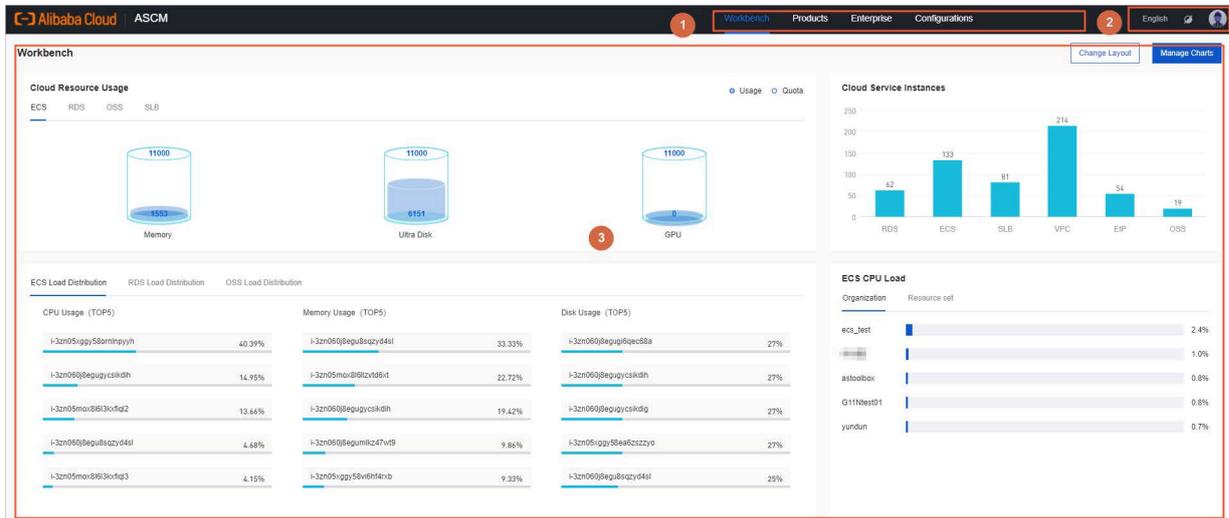
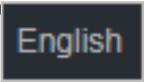


Table 1-2: Functional areas of the webpage

Area		Description
1	Main menu bar	<p>It includes the following modules:</p> <ul style="list-style-type: none"> • Home: uses charts to display the usage and monitoring data of existing system resources in each region. • Product: manages all types of basic cloud products and resources. • Enterprise: manages organizations, resource sets, roles, users, and logon policies for enterprises. • Configurations: manages system configurations, password policies, specifications, and message center. • Security: provides operation logs and system logs.

Area		Description
2	Information area of the currently logged-on user	<p> : allows you to switch between English, simplified Chinese, and traditional Chinese.</p> <p> : allows you to switch between styles.</p> <p> : Move the pointer over the icon of the currently logged-on user. The User Information and Exit menu items are displayed.</p> <p>On the User Information page, you can perform the following operations:</p> <ul style="list-style-type: none"> - View basic information. - Modify personal information. - Change the logon password. - View the AccessKey pair of your Apsara Stack tenant account
3	Operation area	Operation area: the information display and operation area.

1.5 Initial configuration

1.5.1 Configuration description

Before using the Apsara Stack Cloud Management (ASCM) console, you must complete a series of basic configuration operations as an administrator, such as creating organizations, resource sets, users, and roles and initializing resources. This is the initial system configuration.

Based on the service-oriented principle, the ASCM console manages the organizations, resource sets, users, and roles of cloud data centers in a centralized manner to grant different resource access permissions to different users.

- **Organization**

After the ASCM console is deployed, a root organization is automatically generated. You can create other organizations under the root organization.

Organizations are displayed in a hierarchical structure. You can create subordinate organizations under each organization level.

- **Resource set**

A resource set is a container used to store resources. Each resource must belong to a resource set.

- **User**

A user is a resource manager and user.

- **Role**

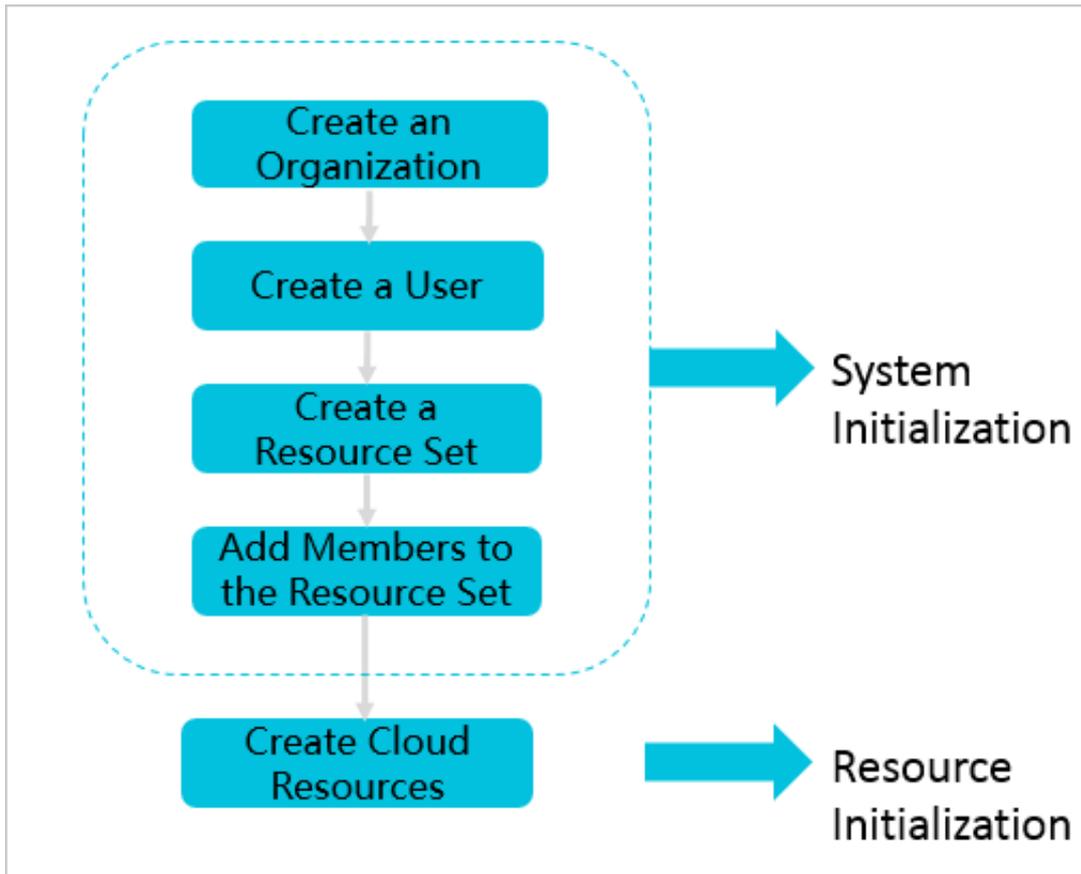
A role is a set of access permissions. You can assign different roles to different users to meet different requirements for system access control.

The following table describes the relationships among organizations, resource sets, users, roles, and cloud resources.

1.5.2 Configuration process

This topic describes the initial configuration process.

Before using the Apsara Stack Cloud Management (ASCM) console, you must complete the initial system configurations as an administrator according to the process shown in the following figure.



1. *Create an organization*

You can create organizations to store resource sets and their resources.

2. *Create a user*

You can create users and assign different roles to different users to meet different requirements for system access control.

3. *Create a resource set*

Before applying for resources, you must create a resource set.

4. *Add a member to a resource set*

Add members to the resource set.

5. Create cloud resources

You can create instances in each service console based on project requirements. For more information about how to create cloud service instances, see the detailed introduction of each cloud service.

1.6 Enterprise

1.6.1 Organizations

1.6.1.1 Create an organization

You can create organizations to store resource sets and their resources.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Organizations.
4. In the organization navigation tree, move the pointer over the name of a parent organization, and click  on the right.
5. Choose Add Organization from the shortcut menu.
6. In the dialog box that appears, enter an organization name and click OK.

1.6.1.2 Query an organization

You can query an organization by name to view its resource sets, users, and user groups.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Organizations.
4. In the search box below Organizations, enter an organization name to query information about the corresponding organization.

1.6.1.3 Modify organization information

An administrator can modify the information of an organization.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Organizations.
4. In the organization navigation tree, move the pointer over the name of an organization, and click  on the right.
5. Choose Edit Organization from the shortcut menu.

6. In the dialog box that appears, modify the name of the organization and click OK.

1.6.1.4 Delete an organization

Administrators can delete organizations that are no longer needed.

Prerequisites



Note:

Before deleting an organization, make sure that the organization does not contain any users, resource sets, or subordinate organizations. Otherwise, the organization cannot be deleted.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Organizations.
4. In the organization navigation tree, move the pointer over the name of an organization, and click  on the right.
5. Choose Delete Organization from the shortcut menu.
6. In the message that appears, click OK.

1.6.1.5 Obtain the AccessKey pair of an organization

An administrator can obtain the AccessKey pair of an organization.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Organizations.
4. In the organization navigation tree, move the pointer over the name of the target organization and click  on the right.
5. Choose AccessKey from the shortcut menu.
6. In the message that appears, view the AccessKey information of the organization.

1.6.2 Resource sets

1.6.2.1 Create a resource set

Before applying for resources, you must create a resource set.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Resource Sets.
4. In the upper-right corner of the page, click Create.
5. In the Create Resource Set dialog box that appears, set Name and Organization.
6. Click OK.

1.6.2.2 View the details of a resource set

When you need to use a cloud resource in your organization, you can view the details of the resource set that contains the resource, including all resource instances and members of the resource set.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Resource Sets.
4. Select an organization from the Organization drop-down list, or enter a resource set name in the search bar.
5. Click Search.
6. Click the name of the resource set that you want to view to go to the Resource Set Details page.

Click the Resources and Members tabs to view information about all resource instances and members of the resource set.

1.6.2.3 Modify the name of a resource set

An administrator can modify the name of a resource set to keep it up-to-date.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.

3. In the left-side navigation pane of the Enterprise page, click Resource Sets.
4. Click More in the Actions column corresponding to a resource set, and choose Edit Name from the shortcut menu.
5. In the dialog box that appears, enter the new name.
6. Click OK.

1.6.2.4 Add a member to a resource set

You can add a member to a resource set so that the member can use the resources in the resource set.

Prerequisites

Before adding a member, make sure that the following prerequisites are met:

- A resource set is created. For more information, see [Create a resource set](#).
- A user is created. For more information, see [Create a user](#).

Context

Members of a resource set have the permissions to use resources in the resource set

.

Deleting resources from a resource set does not affect the members of the resource set. Similarly, deleting members from a resource set does not affect the resources in the resource set.

You can delete a member that is no longer in use in a resource set. After the member is deleted, it will no longer be able to access the resource set.

Procedure

1. [Log on to the ASCM console](#) as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Resource Sets.
4. Click More in the Actions column corresponding to a resource set, and choose Add Member from the shortcut menu.
5. In the dialog box that appears, select a username.
6. Click OK.

1.6.2.5 Delete a resource set

Administrators can delete resource sets that are not needed.

Context

Ensure that the resource set to be deleted does not contain any members or resources.



Notice:

A resource set cannot be deleted if it contains resources or members.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Resource Sets.
4. Click More in the Actions column corresponding to a resource set, and choose Delete from the shortcut menu.
5. In the message that appears, click OK.

1.6.3 Roles

1.6.3.1 Create a custom role

You can add custom roles in the Apsara Stack Cloud Management (ASCM) console to more efficiently grant permissions to users so that different personnel can work with different functions.

Context

A role is a set of access permissions. Each role has a range of permissions. A user can have multiple roles, which means that the user is granted all the permissions defined for these roles. A role can be used to grant the same permissions to a group of users.

Before adding a custom role, note that the total number of custom and default roles cannot exceed 20.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.

3. In the left-side navigation pane of the Enterprise page, click Roles.
4. In the upper-right corner of the page, click Create Custom Role.
5. On the Roles page that appears, set the role name and management permissions.

The screenshot shows the 'Roles' page with four numbered steps: 1. Role Name and Management Permissions, 2. Application Permissions, 3. Menu Permissions, and 4. Associated Users. Step 1 is active, displaying a form with the following fields: 'Role Name' (0/15), 'Description' (0/100), and 'Scope' (radio buttons for 'All Organizations', 'Specified Organization and Subordinate Organizations', and 'Resource Set').

The following table describes the role creation parameters.

Table 1-3: Role creation parameters

Parameter	Description
Role Name	The name of the role. The name can be up to 15 characters in length and can contain only letters and digits.
Description	Optional. The description of the role. The description can be up to 100 characters in length and can contain letters, digits, commas (,), semicolons (;), and underscores (_).
Management Permissions	<p>All Organizations</p> <p>The permissions apply to all organizations involved.</p> <ul style="list-style-type: none"> • Specified Organization and Subordinate Organizations <p>The permissions apply to the organization to which the user belongs and its subordinate organizations.</p> <ul style="list-style-type: none"> • Resource Set <p>The permissions apply to the resource sets assigned to the user.</p>

6. Set the parameters in the Application Permissions, Menu Permissions, and Associated Users steps in sequence.



Note:

The system automatically selects the required permissions that the specified operation depends on. Removing the dependency may cause the operation to fail.

1.6.3.2 View the details of a role

If you are not certain about the specific permissions of a role, go to the Roles page to view the role permissions.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Roles.
4. Click the name of the role that you want to view. On the Roles page, view information about the role.

1.6.3.3 Modify custom role information

An administrator can modify the name and permissions of a custom role.

Context

**Note:**

Information about preset roles cannot be modified.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Roles.
4. In the role name list, click More in the Actions column corresponding to a custom role, and choose Modify from the shortcut menu to go to the Roles page.
5. You can modify the name, permissions, and associated users of a custom role.
 - **Procedure of modifying a role name:** Move the pointer over the role name and click  to enter a new role name.
 - **Procedure of modifying permissions:** Click the Management Permissions, Application Permissions, or Menu Permissions tab, select or remove related permissions from the corresponding tab, and then click Update.
 - **Procedure of binding a user to a role:** Click the Associated Users tab, and select a user from the Select one or more users drop-down list to add the user. To unbind the user from the role, click Remove.

1.6.3.4 Delete a custom role

You can delete custom roles that are no longer needed.

Context



Note:

Default or preset roles cannot be deleted.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Roles.
4. Click More in the Actions column corresponding to a role, and choose Delete from the shortcut menu.
5. In the message that appears, click OK.

1.6.4 Users

1.6.4.1 System users

1.6.4.1.1 Create a user

An administrator can create a user and assign the user different roles to meet different requirements for system access control.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Users.
4. Click Create.
5. In the dialog box that appears, configure the parameters.

Parameter	Description
Username	The Apsara Stack account name of the user. The name must be 3 to 30 characters in length and can contain letters, digits, hyphens (-), underscores (_), periods (.), and at signs (@). It must start with a letter or digit.

Parameter	Description
Display Name	The display name of the user. The name must be 2 to 30 characters in length and can contain letters, digits, hyphens (-), underscores (_), periods (.), and at signs (@).
Roles	The roles for the user.
Organization	The organization to which the user belongs.
Logon Policy	<p>The logon policy that restricts the logon time and IP addresses of the user. By default, the default policy is bound to new users.</p> <div style="background-color: #f0f0f0; padding: 5px;">  Note: The default policy does not restrict the time period and IP addresses for users to log on. To restrict the logon time and IP addresses of a user, you can modify the user's logon policy or create a logon policy for the user. For more information, see Create a logon policy. </div>
Mobile Number	<p>The mobile number of the user. The mobile number is used by the system to notify users of resource application and usage. Make sure that the entered mobile number is correct.</p> <div style="background-color: #f0f0f0; padding: 5px;">  Note: If the mobile number is changed, be sure to update it on the system in a timely manner. </div>
Landline Number	Optional. The landline number of the user. It must be 4 to 20 characters in length and can contain only digits (0 to 9) and hyphens (-).
Email	<p>The email address of the user. The email address is used by the system to notify users of resource application and usage. Make sure that the entered email address is correct.</p> <div style="background-color: #f0f0f0; padding: 5px;">  Note: If the email address is changed, be sure to update it on the system in a timely manner. </div>

6. Click OK.

1.6.4.1.2 Query a user

You can view user information such as name, organization, mobile number, email address, role, logon time, and initial password.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Users.
4. Click the System Users tab.
5. Set Username, Organization, or Role, and then click Search.
6. Click More in the Actions column corresponding to a user, and choose User Information from the shortcut menu to view basic information about the user.

1.6.4.1.3 Modify user information

You can modify user information such as display name, mobile number, and email address to keep it up-to-date.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Users.
4. Click the System Users tab.
5. Click More in the Actions column corresponding to a user, and choose Edit from the shortcut menu.
6. In the Modify User Information dialog box, enter the relevant information and click OK.

1.6.4.1.4 Change user roles

You can add, change, and delete roles for a user.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Users.
4. Click the System Users tab.

5. Click **More** in the **Actions** column corresponding to a user, and choose **Authorize** from the shortcut menu.
6. In the **Role** field, add, delete, or change user roles as needed.
7. Click **OK**.

1.6.4.1.5 Modify a user logon policy

An administrator can modify a user's logon policy to restrict the permitted logon time and IP addresses of the user.

Prerequisites

A new logon policy is created. For more information about how to create a logon policy, see [Create a logon policy](#).

Modify a user logon policy

1. [Log on to the ASCM console](#) as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Users**.
4. Click the **System Users** tab.
5. Click **More** in the **Actions** column corresponding to a user, and choose **Logon Policy** from the shortcut menu.
6. In the **Assign Logon Policy** dialog box, select a logon policy and click **OK**.

Modify multiple user logon policies at a time

1. [Log on to the ASCM console](#) as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Users**.
4. Click the **System Users** tab.
5. Select multiple users.
6. In the upper-right corner of the page, click **Logon Policy**.
7. In the **Assign Logon Policies** dialog box, select a logon policy.

1.6.4.1.6 View the initial password of a user

After a user is created, the system automatically generates an initial password for the user.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Users.
4. Click the System Users tab.
5. Select the user for which you want to view the initial password.
6. You can use one of the following methods to view the initial password of a user:
 - Click View Initial Password in the upper-right corner of the Users page to view the initial password.
 - Click More in the Actions column corresponding to the user, and choose User Information from the shortcut menu. On the user information page, click View Password to view the initial password.

1.6.4.1.7 Reset the password of a user

If users forget their logon passwords, the system administrator can reset the logon passwords for them.

Prerequisites

The logon password of a user can be reset by only the user and those who have the permissions to create resource sets for the user.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Users.
4. Click the System Users tab.
5. Select the user for which you want to reset the password.
6. Click More in the Actions column corresponding to the user, and choose User Information from the shortcut menu.
7. Click Reset Password.

After the password is reset, a message is displayed, indicating that the password has been reset. If you want to view the initial password after password reset, click View Password.

1.6.4.1.8 Disable and enable a user

You can disable a user to prevent the user from logging on to the Apsara Stack Cloud Management (ASCM) console. Disabled users must be re-enabled before they can log on to the ASCM console again.

Context

By default, users are enabled when they are created.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Users.
4. Click the System Users tab.
5. You can perform the following operations on the current tab:
 - Select a user whose Status is Enabled, click More in the Actions column, and choose Disable from the shortcut menu to disable the user.
 - Select a user whose Status is Disabled, click More in the Actions column, and choose Enable from the shortcut menu to enable the user.

1.6.4.1.9 Delete a user

An administrator can delete a specified user as needed.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Users.
4. Click the System Users tab.
5. Click More in the Actions column corresponding to a user, and choose Delete from the shortcut menu.

1.6.4.2 Historical users

1.6.4.2.1 Query historical users

You can check whether a user has been deleted, or quickly find and restore the user.

Procedure

1. *Log on to the ASCM console* as an administrator.

2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Users.
4. Click the Historical Users tab.
5. Enter the username that you want to query in the Username search box.
6. Click Search.

1.6.4.2 Restore historical users

An administrator can restore a deleted user account from the Historical Users tab.

Context

The basic information such as logon password of a restored user will be the same as it was before the user was deleted, except for the organization and role.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Users.
4. Click the Historical Users tab.
5. Find the user to be restored and click Restore in the Actions column.
6. In the Restore User dialog box that appears, select an organization and a role.
7. Click OK.

1.6.5 Logon policies

1.6.5.1 Create a logon policy

To improve the security of the Apsara Stack Cloud Management (ASCM) console, an administrator can create a logon policy to control the logon time and IP addresses of a user.

Context

Logon policies are used to control the time period and IP addresses for users to log on. After a user is bound to a logon policy, the user's logons will be restricted based on the logon time and IP addresses specified in the policy.

When providing services, the ASCM console automatically generates a default policy without restrictions on the logon time and IP addresses. The default policy cannot be deleted.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the Enterprise page, click **Logon Policies**.
4. In the upper-right corner of the page, click **Create**.
5. In the Create Logon Policy dialog box that appears, set the **Name**, **Policy Properties**, **Time Period**, and **IP Address** fields.

Table 1-4: Logon policy parameters

Parameter	Description
Name	The name of the logon policy. The name must be 2 to 50 characters in length and can contain only letters and digits. The name must be unique in the system.
Description	The description of the logon policy.
Policy Properties	<p>The authentication method for filtering user logons.</p> <ul style="list-style-type: none"> • Whitelist: Logon is allowed if the following parameters are met. • Blacklist: Logon is denied if the following parameters are met.

Parameter	Description
Time Period	The permitted logon time period. When this policy is configured, users can log on to the ASCM console only during the configured period.
IP Address	The permitted CIDR block. When this policy is configured, users can log on to the ASCM console only from IP addresses within the specified CIDR block.

1.6.5.2 Query a logon policy

When providing services, the Apsara Stack Cloud Management (ASCM) console automatically generates a default policy without restrictions on the logon time and IP addresses.

Context

When providing services, the Apsara Stack Cloud Management (ASCM) console automatically generates a default policy without restrictions on the logon time and IP addresses.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Logon Policies.
4. Enter the name of the policy that you want to view and click Search.
5. View the logon policy, including the permitted logon time and IP addresses.

1.6.5.3 Modify a logon policy

You can modify the policy name, policy properties, permitted logon time period, and IP addresses of a logon policy.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Logon Policies.
4. Click More in the Actions column corresponding to a policy, and choose Modify from the shortcut menu.
5. In the Modify Logon Policy dialog box that appears, modify the logon policy information.

6. Click OK.

1.6.5.4 Delete a logon policy

You can delete logon policies that are no longer needed.

Context



Note:

The default policy cannot be deleted.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Logon Policies.
4. Click More in the Actions column corresponding to a policy, and choose Delete from the shortcut menu.
5. In the message that appears, click OK.

1.6.6 User groups

1.6.6.1 Create a user group

You can create a user group in a selected organization and grant batch authorizations to users in the group.

Prerequisites

Before creating a user group, you must create an organization. For more information, see *Create an organization*.

Context

Relationship between user groups and users:

- A user group can contain zero or more users.
- You can add users to user groups as needed.
- You can add a user to multiple user groups.

Relationship between user groups and organizations:

- A user group can only belong to a single organization.
- You can create multiple user groups in an organization.

Relationship between user groups and roles:

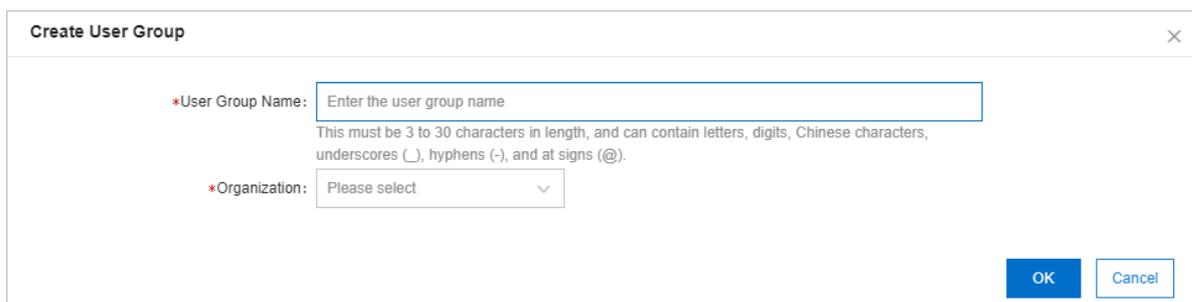
- A user group can only be bound to a single role.
- A role can be associated with multiple user groups.
- When a role is associated with a user group, the role permissions are automatically granted to users in the user group.

Relationship between user groups and resource sets:

- You can add zero or more user groups to a resource set.
- A user group can be added to multiple resource sets.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **User Groups**.
4. In the upper-right corner of the page, click **Create User Group**.
5. In the dialog box that appears, set **User Group Name** and **Organization**.



6. Click **OK**.

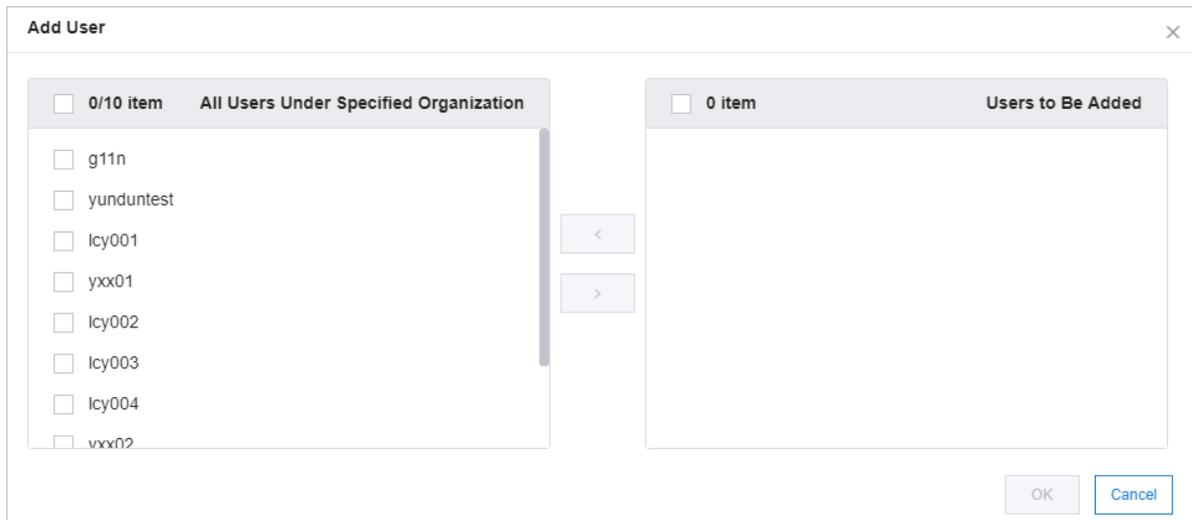
1.6.6.2 Add users to a user group

You can add users to a user group.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **User Groups**.
4. Click **Add User** in the **Actions** column corresponding to a user group.

5. Select the names of users to be added from the left list, and click the right arrow to move them to the right list.



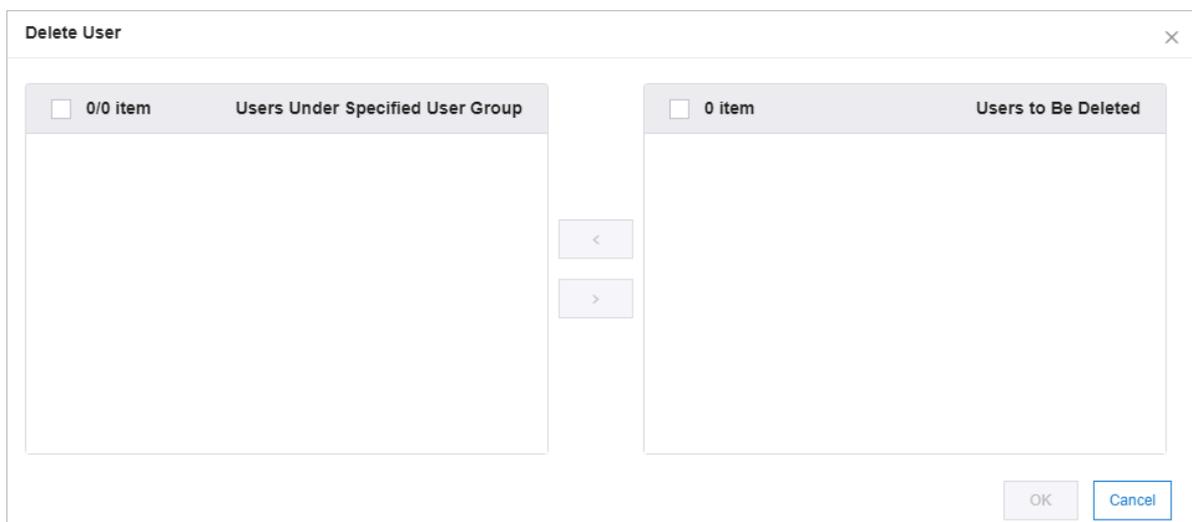
6. Click OK.

1.6.6.3 Delete users from a user group

You can delete users from a user group.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click User Groups.
4. Click Delete User in the Actions column corresponding to a user group.
5. Select the names of users to be deleted from the Users Under Specified User Group list, and click the right arrow to move them to the Users to Be Deleted list.



6. Click OK.

1.6.6.4 Add a role

You can add a role to a user group and assign the role to all users in the group.

Context

**Note:**

You can add only one role to a user group.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click User Groups.
4. Click Add Role in the Actions column corresponding to a user group.
5. In the dialog box that appears, select a role.
6. Click OK.

1.6.6.5 Delete a role

You can delete existing roles.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click User Groups.
4. Click Delete Role in the Actions column corresponding to a user group.
5. In the Confirm message that appears, click OK.

1.6.6.6 Modify the name of a user group

You can modify the names of user groups.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click User Groups.
4. Click Edit User Group in the Actions column corresponding to a user group.
5. In the dialog box that appears, enter the new name.
6. Click OK.

1.6.6.7 Delete a user group

You can delete user groups that are no longer needed.

Procedure

1. *Log on to the ASCM console as an administrator.*
2. **In the top navigation bar, click Enterprise.**
3. **In the left-side navigation pane of the Enterprise page, click User Groups.**
4. **Click Delete User Group in the Actions column corresponding to a user group.**
5. **In the Confirm message that appears, click OK.**

1.7 Configurations

1.7.1 Password policies

You can configure password policies for user logons.

Procedure

1. *Log on to the ASCM console as an administrator.*
2. **In the top navigation bar, click Configurations.**
3. **In the left-side navigation pane of the Configurations page, click Password Policies.**
4. **On the Password Policy page, set the password policy parameters.**

The screenshot displays the 'Password Policy' configuration interface. It includes the following settings:

- Password Length:** A text input field containing '10', with a note 'To 32 Digits (Minimum: 8)'.
- The Password Must Contain:** Four checked checkboxes: 'Lowercase Letters', 'Uppercase Letters', 'Digits', and 'Special Characters'.
- Logon Disabled After Password Expires:** Radio buttons for 'Yes' (selected) and 'No'.
- Password Validity Period (Days):** A text input field containing '90', with a note '(The value must be 0 to 1095. The value 0 specifies that the password will not expire.)'.
- Password Attempts:** A text input field containing '5', with a note '(The value must be 0 to 32. The value 0 specifies that the password history check is disabled.)'.
- Password History Check:** A text input field containing '5', with a note '(The value must be 0 to 24. The value 0 specifies that the password history check is disabled.)'.

At the bottom of the form, there are two buttons: 'Save' (highlighted in blue) and 'Reset'.

To restore to the default password policy, click Reset.

1.7.2 Menus

1.7.2.1 Create a menu

You can create a menu and add its URL to the ASCM console for quick access.

Procedure

1. *Log on to the ASCM console as an administrator.*
2. **In the top navigation bar, click Configurations.**
3. **In the left-side navigation pane of the Configurations page, click Menu Settings.**
4. **On the Main Menu page, click Create in the upper-right corner.**
5. **In the Create dialog box that appears, set the menu parameters.**

The screenshot shows a 'Create' dialog box with the following fields and options:

- *Title:** Please input
- URL:** Please input
- *Console Type:** asconsole oneconsole other
Different console types correspond to different service endpoints. If you select Other, the endpoint configured in the URL field is used.
- Icon:** Please input
- *Identifier:** Please input
- *Order:** 0 (with + and - buttons)
- *Parent Level:** Please select (dropdown menu)
- *Open With:** Default New Window
- Description:** Please input

Buttons: OK, Cancel

Table 1-5: Menu parameters

Parameter	Description
Title	The display name of the menu.
URL	The URL of the menu.

Parameter	Description
Console Type	<p>Different console types correspond to different domain names.</p> <ul style="list-style-type: none"> · oneconsole: You only need to enter the path in the URL field. The domain name is automatically matched. · asconsole: You only need to enter the path in the URL field. The domain name is automatically matched. · other: You must enter the domain name in the URL field.
Icon	The icon displayed in the left-side navigation pane. The icon cannot be changed.
Identifier	The unique identifier of the menu in the system. This identifier can be used to indicate whether the menu is selected in the navigation bar. The identifier cannot be changed.
Order	The display order among the same-level menus. The larger the value, the lower the display order. Leave the Order field empty.
Parent Level	The displayed tree structure.
Open With	Specifies whether to open the menu in the current window or in a new window.
Description	The description of the menu.

1.7.2.2 Modify a menu

You can modify an existing menu, including the menu name, URL, icon, and menu order.

Prerequisites

Default menus cannot be modified.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Menu Settings**.
4. Click **Edit** in the **Actions** column corresponding to a menu.

5. In the Edit dialog box that appears, modify relevant information about the menu.

Edit [Close]

*Title: [Redacted]

URL: /module/config?identifier=blink&jumpUrl=true#jump/blink

*Console Type: asconsole oneconsole other
 Different console types correspond to different service endpoints. If you select Other, the endpoint configured in the URL field is used.

Icon: wind-rc-product-icon glyph-sc rotate-0

*Identifier: blink

*Order: 21 [+] [-]

*Parent Level: Products [v]

*Group: Please select [v]

*Open With: Default New Window

Description: Please input

[OK] [Cancel]

1.7.2.3 Delete a menu

You can delete menus that are no longer needed.

Prerequisites

Default menus cannot be deleted.

Procedure

1. *Log on to the ASCM console as an administrator.*
2. **In the top navigation bar, click Configurations.**
3. **In the left-side navigation pane of the Configurations page, click Menu Settings.**
4. **Click Delete in the Actions column corresponding to a menu.**
5. **In the message that appears, click OK.**

1.7.2.4 Display or hide menus

You can display or hide menus as follows:

Procedure

1. *Log on to the ASCM console as an administrator.*

2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Menu Settings**.
4. Select or clear the check box in the **Displayed** column corresponding to a menu.

1.7.3 Specifications

1.7.3.1 Create specifications

You can customize specifications for each resource type.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Specifications**.
4. Click the resource type for which you want to create specifications.
5. In the upper-right corner of the page, click **Create Specifications**.
6. In the dialog box that appears, set the parameters.
7. Click **OK**.

1.7.3.2 View specifications

You can view the specifications of each resource type.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Specifications**.
4. Click the resource type for which you want to view specifications.
5. In the list of specifications, view information about the specifications.

1.7.3.3 Disable specifications

By default, the status of newly created specifications is **Enabled**.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Specifications**.
4. Select the resource type for which you want to disable specifications.

5. Click **Disable** in the **Actions** column corresponding to the target specifications.
6. In the message that appears, click **OK**.

1.7.3.4 Export specifications

You can export specifications that you want to view and share.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click **Configurations**.
3. In the left-side navigation pane of the **Configurations** page, click **Specifications**.
4. Click the resource type for which you want to create specifications.
5. In the upper-right corner of the page, click **Export**.
6. Save the specifications file to the target path.

1.8 Security

1.8.1 View operation logs

You can view operation logs to obtain up-to-date information about various resources and functional modules in the Apsara Stack Cloud Management (ASCM) console. You can also export operation logs to your personal computer.

Procedure

1. *Log on to the ASCM console* as a security administrator.
2. In the top navigation bar, click **Security**.
3. You can filter logs by username, object, level, source IP address, details, start time, and end time.

The following table describes the fields in the query result.

Table 1-6: Fields in the query result

Log field	Description
Username	The name of the operator.

Log field	Description
Object	The Apsara Stack service on which operations are performed. The operations include creating, modifying, deleting, querying, updating, binding, unbinding, enabling, and disabling service instances, applying for and releasing service instances, and changing the ownership of service instances.
Level	The operation level. Valid values: INFO, DEBUG, and ERROR.
Source IP	The IP address of the operator.
Details	The brief introduction of the operation.
Start Time	The time when the operation started.
End Time	The time when the operation ended.

- Optional: Click Export to export the logs displayed on the current page to your personal computer in.xls format.

The exported log file is named *log.xls* and stored in the *C:\Users\Username\Downloads* directory.

1.9 RAM

1.9.1 RAM introduction

Resource Access Management (RAM) is a resource access control service provided by Apsara Stack.

You can use RAM to manage users and control which resources are accessible to employees, systems, and applications.

RAM provides the following features:

- RAM role

To authorize a cloud service in a level-1 organization to use other resources in the organization, you must create a RAM role. This role contains the operations that the cloud service can perform on resources.

Only system administrators and level-1 organization administrators can create RAM roles.

- **User group**

You can create multiple users within an organization and grant them different operation permissions on cloud resources.

You can create RAM user groups to classify and authorize RAM users within your Apsara Stack tenant account. This simplifies the management of RAM users and their permissions.

You can create RAM permission policies to grant different operation permissions to different user groups.

1.9.2 Permission policy structure and syntax

This topic describes the structure and syntax used to create or update permission policies in Resource Access Management (RAM).

Policy characters and usage rules

- **Characters in a policy**

- The following characters are JSON tokens and are included in policies: `{ }`
`[] " , : .`
- The following characters are special characters in the syntax and are not included in policies: `= < > () |`.

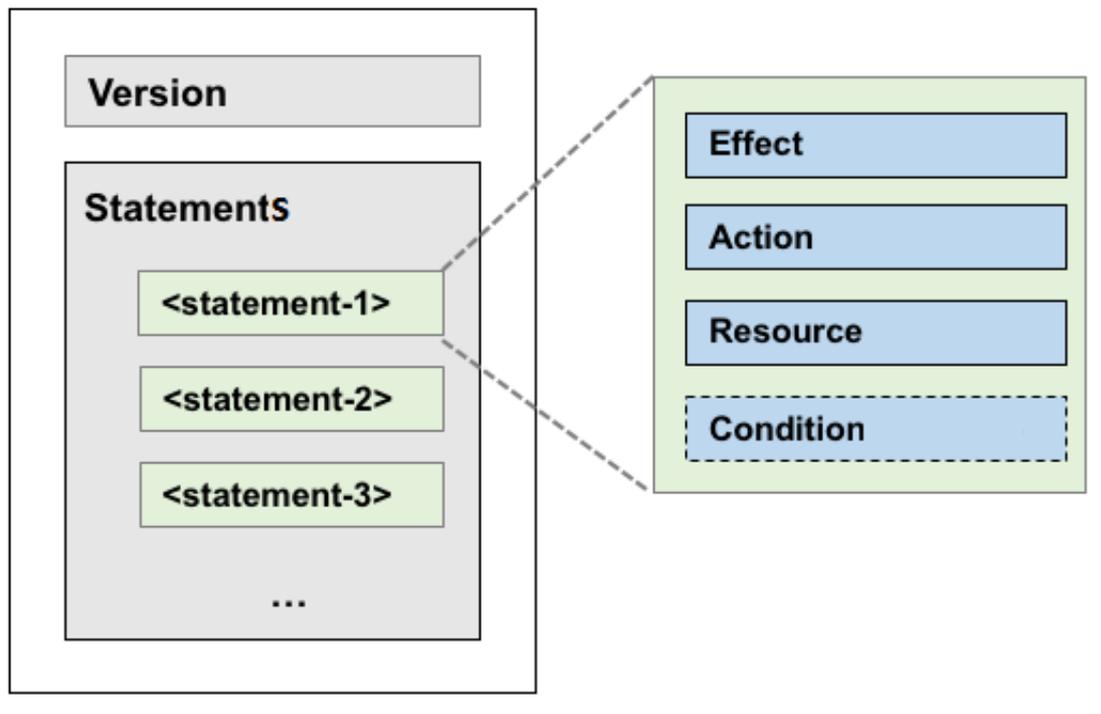
- **Use of characters**

- If an element can have more than one value, you can perform the following operations:
 - Separate multiple values by using commas (,) as delimiters between each value and use an ellipsis (...) to describe the remaining values. Example: `[<action_string>, <action_string>, ...]`.
 - Include only one value. Examples: `"Action": [<action_string>]` and `"Action": <action_string>`.
- A question mark (?) following an element indicates that the element is optional. Example: `<condition_block? >`.
- A vertical bar (|) between elements indicates multiple options. Example: `("Allow" | "Deny")`.
- Elements that must be text strings are enclosed in double quotation marks ("). Example: `<version_block> = "Version" : ("1")`.

Policy structure

The policy structure includes the following components:

- **The version number.**
- **A list of statements. Each statement contains the following elements: Effect, Action, Resource, and Condition. The Condition element is optional.**



Policy syntax

```

policy = {
    <version_block>,
    <statement_block>
}
<version_block> = "Version" : ("1")
<statement_block> = "Statement" : [ <statement>, <statement>, ... ]
<statement> = {
    <effect_block>,
    <action_block>,
    <resource_block>,
    <condition_block? >
}
<effect_block> = "Effect" : ("Allow" | "Deny")
<action_block> = ("Action" | "NotAction") :
    ("*" | [<action_string>, <action_string>, ...])
<resource_block> = ("Resource" | "NotResource") :
    ("*" | [<resource_string>, <resource_string>, ...])
<condition_block> = "Condition" : <condition_map>
<condition_map> = {
    <condition_type_string> : {
        <condition_key_string> : <condition_value_list>,
        <condition_key_string> : <condition_value_list>,
        ...
    }
}

```

```

    },
    <condition_type_string> : {
      <condition_key_string> : <condition_value_list>,
      <condition_key_string> : <condition_value_list>,
      ...
    }, ...
  }
  <condition_value_list> = [<condition_value>, <condition_value>, ...]
  <condition_value> = ("String" | "Number" | "Boolean")

```

Description:

- The current policy version is 1.
- The policy can have multiple statements.
 - The effect of each statement can be either `Allow` or `Deny`.

**Note:**

In a statement, both the `Action` and `Resource` elements can have multiple values.

- Each statement can have its own conditions.

**Note:**

A condition block can contain multiple conditions with different operators and logical combinations of these conditions.

- You can attach multiple policies to a RAM user. If policies that apply to a request include an `Allow` statement and a `Deny` statement, the `Deny` statement overrides the `Allow` statement.
- Element value:
 - If an element value is a number or Boolean value, it must be enclosed in double quotation marks ("") in the same way as strings.
 - If an element value is a string, characters such as the asterisk (*) and question mark (?) can be used for fuzzy matching.
 - The asterisk (*) indicates any number (including zero) of allowed characters. For example, `ecs:Describe*` indicates all ECS API operations that start with `Describe`.
 - The question mark (?) indicates an allowed character.

Policy format check

Policies are stored in RAM as JSON documents. When you create or update a policy, RAM first checks whether the JSON format is valid.

- **For more information about JSON syntax standards, see [RFC 7159](#).**
- **We recommend that you use tools such as JSON validators and editors to check whether the policies meet JSON syntax standards.**

1.9.3 RAM roles

1.9.3.1 View basic information about a RAM role

You can view basic information about a RAM role, including its user groups and existing permission policies.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. **In the top navigation bar, click Enterprise.**
3. **In the left-side navigation pane of the Enterprise page, click Roles.**
4. **On the Roles page, click the name of the target RAM role.**
5. **In the basic information section, click the User Groups and Permissions tabs to view relevant information.**

1.9.3.2 Create a RAM role

To authorize a cloud service in a level-1 organization to use other resources in the organization, you must create a RAM role. This role contains the operations that the cloud service can perform on resources.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. **In the top navigation bar, click Enterprise.**
3. **In the left-side navigation pane of the Enterprise page, click Roles.**
4. **In the upper-right corner of the page, click Create RAM Role.**
5. **On the Roles - Create RAM Role page that appears, set Role Name and Description.**
6. **Click Create.**

1.9.3.3 Add a permission policy

To use a cloud service to access other cloud resources, you must create a permission policy and attach it to a user group.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the Enterprise page, click **Roles**.
4. In the role name list, click **More** in the Actions column corresponding to a RAM role, and choose **Modify** from the shortcut menu to go to the Roles page.
5. Click the **Permissions** tab.
6. Click **Add Permission Policy**.
7. In the dialog box that appears, enter information about the permission policy.

Add Permission Policy [X]

*Policy Name:
 0/15

Description:
 0/100

*Policy Details:
 1 The details of the specified policy must be 2,048 characters in length, and follow the JSON format

[OK] [Cancel]

For more information about how to enter the policy content, see [Permission policy structure and syntax](#).

1.9.3.4 Modify the content of a RAM permission policy

You can modify the content of a RAM permission policy as needed.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Roles.
4. In the role name list, click More in the Actions column corresponding to a RAM role, and choose Modify from the shortcut menu to go to the Roles page.
5. Click the Permissions tab.
6. Click the name of a permission policy in the Permission Policy Name column.
7. In the Modify Permission Policy dialog box that appears, modify the relevant information and click OK.

For more information about how to modify the policy content, see [Permission policy structure and syntax](#).

1.9.3.5 Modify the name of a RAM permission policy

You can modify the name of a RAM permission policy as needed.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Roles.
4. In the role name list, click More in the Actions column corresponding to a RAM role, and choose Modify from the shortcut menu to go to the Roles page.
5. Click the Permissions tab. Click the name of a permission policy in the Permission Policy Name column.
6. In the Modify Permission Policy dialog box that appears, modify the permission policy name.

1.9.3.6 Add a RAM role to a user group

You can bind RAM roles to user groups as needed.

Prerequisites

You must create a user group before RAM roles can be added. If no user groups have been created, see [Create a user group](#).

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.

3. In the left-side navigation pane of the Enterprise page, click Roles.
4. In the role name list, click More in the Actions column corresponding to a RAM role, and choose Modify from the shortcut menu to go to the Roles page.
5. Click the User Groups tab.
6. Click Add User Group. In the dialog box that appears, select a user group.
7. Click OK.

1.9.3.7 Grant permissions to a RAM role

When you grant permissions to a RAM role, all users in the user groups that are assigned this role will share the granted permissions.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Roles.
4. In the role name list, click More in the Actions column corresponding to a RAM role, and choose Modify from the shortcut menu to go to the Roles page.
5. Click the Permissions tab.
6. Click Select Existing Permission Policy.
7. In the dialog box that appears, select a RAM permission policy and click OK.

If no RAM permission policies are available, see [Add a permission policy](#).

1.9.3.8 Remove permissions from a RAM role

You can remove permissions that are no longer needed from RAM roles.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Roles.
4. In the role name list, click More in the Actions column corresponding to a RAM role, and choose Modify from the shortcut menu to go to the Roles page.
5. Click the Permissions tab.
6. Click Remove in the Actions column corresponding to the permission policy that you want to remove.

1.9.3.9 Modify a RAM role name

Administrators can modify the names of RAM roles.

Context



Note:

The name of a preset role cannot be modified.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Roles.
4. In the role name list, click More in the Actions column corresponding to a RAM role, and choose Modify from the shortcut menu to go to the Roles page.
5. Move the pointer over the role name and click  to enter a new role name.

1.9.3.10 Delete a RAM role

This topic describes how to delete a RAM user.

Prerequisites

Before you delete a RAM role, make sure that no policies are attached to the RAM role.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Enterprise.
3. In the left-side navigation pane of the Enterprise page, click Roles.
4. In the role name list, click More in the Actions column corresponding to a RAM role, and choose Delete from the shortcut menu.
5. In the message that appears, click OK.

1.9.4 RAM authorization policies

1.9.4.1 Create a RAM role

You can create authorization policies and grant them to organizations as needed.

Procedure

1. *Log on to the ASCM console* as an administrator.

2. In the top navigation bar, click Configurations.
3. In the left-side navigation pane of the Configurations page, click RAM Roles.
4. In the upper-right corner of the page, click Create RAM User.
5. On the Create RAM User page, set Organization and Service.
6. Click OK.

1.9.4.2 View the details of a RAM role

You can view the details of a RAM role, including its role name, creation time, description, and Alibaba Cloud Resource Name (ARN).

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Configurations.
3. In the left-side navigation pane of the Configurations page, click RAM Roles.
4. On the RAM Users page, set Role Name or Service Name, and click Search in the upper-right corner.
To perform another search, click Clear.
5. Find the RAM role that you want to view and click Details in the Actions column.
6. Click the Role Details tab to view the details of the RAM role.

1.9.4.3 View RAM authorization policies

You can view the details of a RAM authorization policy, including its policy name, policy type, default version, description, association time, and policy content.

Prerequisites

A RAM authorization policy is created. For more information, see [Create a RAM role](#).

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the top navigation bar, click Configurations.
3. In the left-side navigation pane of the Configurations page, click RAM Roles.
4. On the RAM Users page, set Role Name or Service Name, and click Search in the upper-right corner.
To perform another search, click Clear.
5. Find the RAM role that you want to view and click Details in the Actions column.

6. Click the Role Policy tab to view information about the role authorization policy.
Click Details in the Actions column to view the policy details.

1.10 Personal information management

1.10.1 Modify personal information

You can modify your personal information to keep it up-to-date.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the upper-right corner of the homepage, move the pointer over the user profile picture and choose User Information from the shortcut menu.



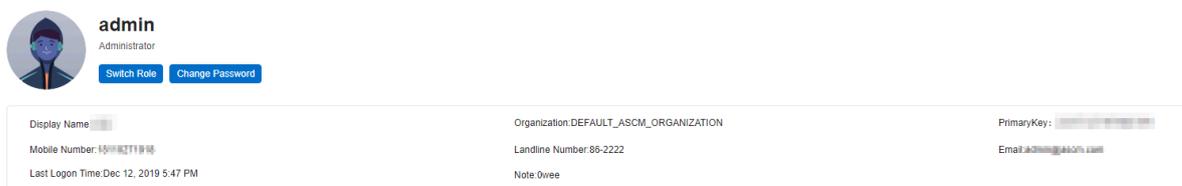
3. Click  next to the item you want to modify.
4. In the Modify User Information dialog box that appears, modify the relevant information.
5. Click OK.

1.10.2 Change your logon password

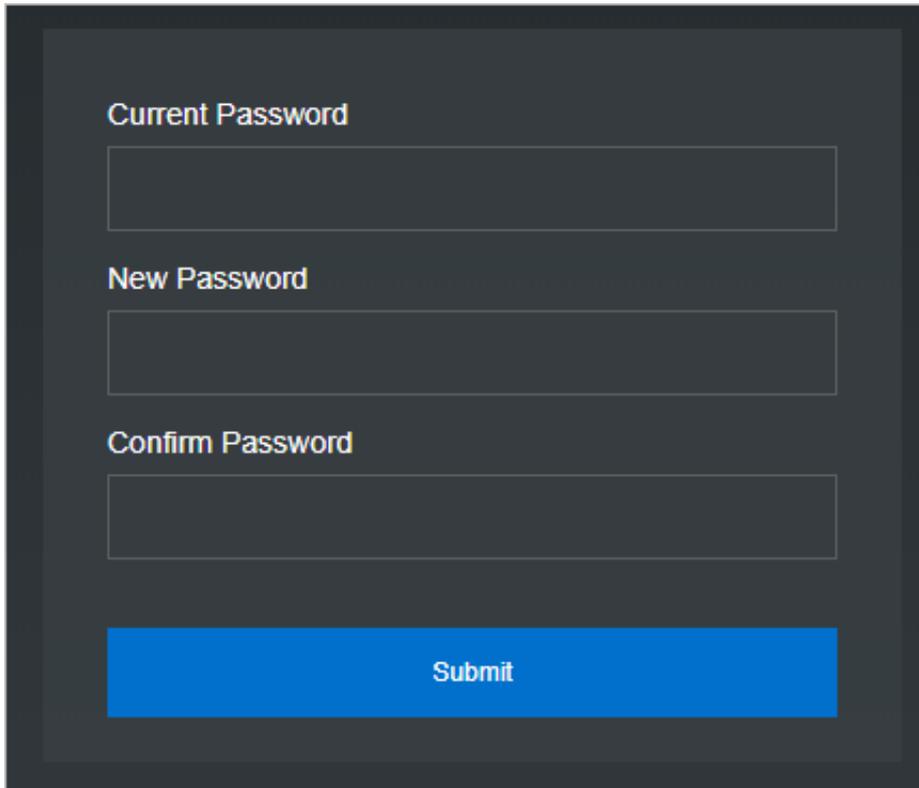
To improve security, you must change your logon password in a timely manner.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the upper-right corner of the homepage, move the pointer over the user profile picture and choose User Information from the shortcut menu.



3. Click **Change Password**. On the page that appears, set **Current Password**, **New Password**, and **Confirm Password**.



The screenshot shows a dark-themed form for changing a password. It consists of three vertically stacked text input fields. The first field is labeled 'Current Password', the second 'New Password', and the third 'Confirm Password'. Below these fields is a prominent blue button with the text 'Submit' in white.

4. Click **Submit**.

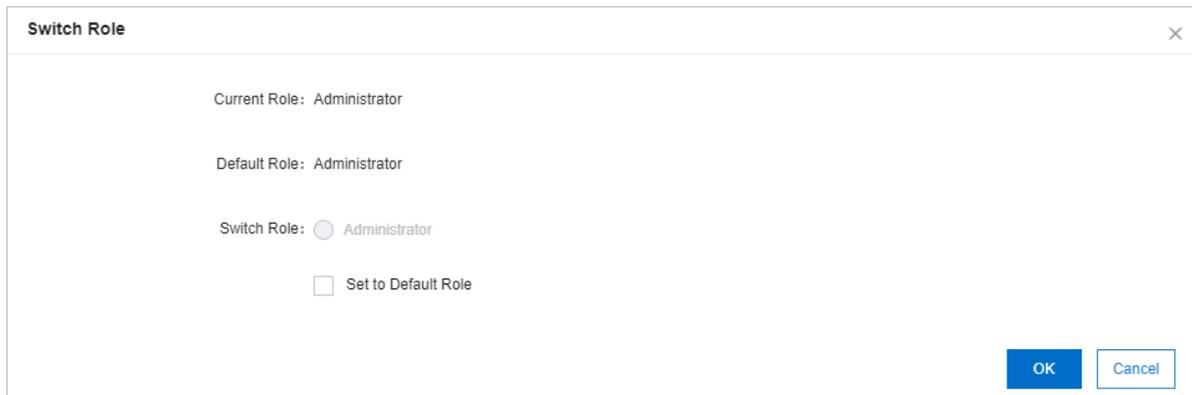
1.10.3 Switch the current role

You can switch the scope of your current role.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the upper-right corner of the homepage, move the pointer over the user profile picture and choose **User Information** from the shortcut menu.
3. Click **Switch Role**.

4. In the Switch Role dialog box that appears, select the role that you want to switch to.



The image shows a 'Switch Role' dialog box with a close button (X) in the top right corner. Inside the dialog, it displays the following information:

- Current Role: Administrator
- Default Role: Administrator
- Switch Role: Administrator
- Set to Default Role

At the bottom right of the dialog, there are two buttons: 'OK' (in blue) and 'Cancel' (in white with a blue border).

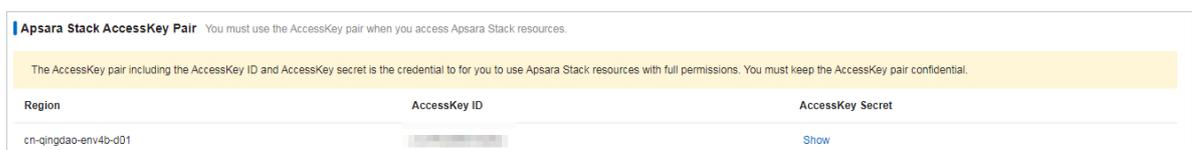
You can also switch back to the default role.

1.10.4 View the AccessKey pair of your Apsara Stack tenant account

To secure cloud resources, the system must verify the identity of visitors and ensure that they have the relevant permissions. You must obtain the AccessKey ID and AccessKey secret of your personal account to access cloud resources.

Procedure

1. *Log on to the ASCM console* as an administrator.
2. In the upper-right corner of the homepage, move the pointer over the user profile picture and choose **User Information** from the shortcut menu.
3. In the **Apsara Stack AccessKey Pair** section, view your AccessKey pair.



The image shows a section titled 'Apsara Stack AccessKey Pair' with a warning message: 'You must use the AccessKey pair when you access Apsara Stack resources.' Below this is a yellow warning box: 'The AccessKey pair including the AccessKey ID and AccessKey secret is the credential for you to use Apsara Stack resources with full permissions. You must keep the AccessKey pair confidential.'

Region	AccessKey ID	AccessKey Secret
cn-qingdao-env4b-d01	[Redacted]	Show



Note:

The AccessKey pair is made up of the AccessKey ID and AccessKey secret. These credentials provide you full permissions on Apsara Stack resources. You must keep the AccessKey pair confidential.

2 Object Storage Service (OSS)

2.1 What is OSS?

Alibaba Cloud Object Storage Service (OSS) is a massive, secure, low-cost, and highly reliable cloud storage service provided by Alibaba Cloud.

It can be considered as an out-of-the-box storage solution with unlimited storage capacity. Compared with the user-created server storage, OSS has many outstanding advantages in reliability, security, cost, and data processing capabilities. Using OSS, you can store and retrieve a variety of unstructured data files, such as text files, images, audios, and videos, over the network at any time.

OSS uploads data files as objects to buckets. OSS is an object storage service that uses a key-value pair format. You can retrieve object content based on unique object names (keys).

On OSS, you can:

- **Create a bucket and upload objects to the bucket.**
- **Obtain an object URL from OSS to share or download an object.**
- **Complete the ACL settings of a bucket or object by modifying its properties or metadata.**
- **Perform basic and advanced OSS tasks through the OSS console.**
- **Perform basic and advanced OSS tasks using the Alibaba Cloud SDKs or directly calling the RESTful APIs in your application.**

2.2 Instructions

Before you use OSS, you must understand the following content:

- **To allow other users to use all or part of OSS functions, you must create RAM users and grant permissions to the users by configuring their RAM policies.**

- Before you use OSS, you must also understand the following limits.

Item	Description
Bucket	<ul style="list-style-type: none"> - You can create up to 100 buckets. - After a bucket is created, its name and region cannot be modified.
Object upload	<ul style="list-style-type: none"> - Objects larger than 5 GB cannot be uploaded by using the following modes: console upload, simple upload, form upload, or append upload . To upload an object that is larger than 5 GB, you must use multipart upload. The size of an object uploaded by using multipart upload cannot exceed 48.8 TB. - You can upload an object with the same name as that of an existing object, but the existing object is overwritten.
Object deletion	<ul style="list-style-type: none"> - Deleted objects cannot be recovered. - You can delete up to 50 objects at a time in the console. To delete more objects at a time, you must use APIs or SDKs.
Lifecycle	You can configure up to 1,000 lifecycle rules for each bucket.

2.3 Quick start

2.3.1 Log on to the OSS console

This topic describes how to log on to the OSS console.

Prerequisites

- Before you log on to the Apsara Stack Cloud Management (ASCM) console, you must obtain the IP address or domain name of the ASCM console from the deployment personnel. The URL to access the ASCM console is in format of `http://the IP address or domain name of the ASCM console/manage`.
- We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL to access the ASCM console. Press Enter.

2. Enter your username and password.

The system has a default super administrator with username **super**. The super administrator can create system administrators. The system administrator can create other users and notify them of their default passwords by SMS or email.



Note:

You must modify the password of your username as instructed when you log on to the ASCM console for the first time. To improve security, the password must meet the minimum complexity requirements. The password must be 8 to 20 characters in length and contain at least two types of the following characters: letters, numbers, or special characters such as exclamation marks (!), at signs (@), number signs (#), dollar signs (\$), and percent signs (%).

3. Click **Login** to go to the ASCM console homepage.

4. In the top navigation bar, choose **Products > Object Storage Service**.

2.3.2 Create buckets

Objects uploaded to buckets are stored in OSS. Before you upload an object to a bucket, you must have a bucket to store objects.

Context

The attributes of a bucket include the region, ACL, and storage class.

Procedure

1. *Log on to the OSS console.*

2. In the left-side navigation pane, click **Create Bucket** if no buckets are available. In the **Create OSS Bucket** dialog box that appears, configure parameters.



Object Storage Service (OSS)
 OSS provides secure, cost-effective, and highly reliable services for storing large amounts of data in the cloud. OSS provides RESTful APIs to facilitate storing and accessing data over the Internet. The capacity and processing capability of OSS can be scaled elastically.



Note:

In the left-side navigation pane, click the **+** icon next to **Create Bucket** if there are buckets available. The **Create OSS Bucket** dialog box appears.

The following table describes the parameters for creating a bucket.

Table 2-1: Parameter descriptions

Parameter	Configuration method
Organization	Select an organization from the drop-down list for the bucket.
Resource Set	Select a resource set from the drop-down list for the bucket.
Region	Select a region from the drop-down list for the bucket  Note: <ul style="list-style-type: none"> After a bucket is created, the region cannot be changed. If you want to access OSS from your ECS instance over the internal network, select the region where your ECS instance is deployed.
Cluster	Select a cluster for the bucket. Two OSS clusters can be deployed in Apsara Stack.

Parameter	Configuration method
Bucket Name	<p data-bbox="858 271 1289 304">Enter the name of the bucket.</p> <div data-bbox="868 331 1433 712" style="background-color: #f0f0f0; padding: 5px;">  Note: <ul style="list-style-type: none"> • The bucket name must comply with the naming conventions. • The bucket name must be globally unique in Apsara Stack OSS. • The bucket name cannot be changed after the bucket is created </div>
Storage Type	<p data-bbox="858 750 1305 824">Set the value to Standard. Only Standard is supported.</p>
Bucket Access	<p data-bbox="858 853 1321 927">Set the ACL for the bucket. The following options are available:</p> <ul style="list-style-type: none"> • Private: Only the owner or authorized users of this bucket can read and write objects in the bucket. Other users, including anonymous users cannot access the objects in the bucket without authorization. • Public Read: Only the owner or authorized users of this bucket can write objects in the bucket. Other users, including anonymous users can only read objects in the bucket. • Public Read/Write: Any users, including anonymous users can read and write objects in the bucket. Fees incurred by such operations are paid by the owner of the bucket. Exercise caution when you configure this option. <div data-bbox="868 1727 1433 1928" style="background-color: #f0f0f0; padding: 5px;">  Note: <p data-bbox="868 1805 1358 1917">After a bucket is created, you can modify its ACL. For more information, see Modify bucket ACLs.</p> </div>

3. Click Submit.

2.3.3 Upload objects

After you create a bucket, you can upload objects to it.

Context

You can upload an object of any format to a bucket. You can use the OSS console to upload an object no larger than 5 GB to a bucket. To upload an object larger than 5 GB, use an SDK or call an API operation.

Procedure

1. *Log on to the OSS console.*
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. Click the Files tab.
4. Click Upload. The Upload dialog box appears.
5. In the Upload To section, set the directory to which the object will be uploaded.
 - **Current:** Objects are uploaded to the current folder.
 - **Specified:** Objects are uploaded to the specified folder. OSS creates the specified folder automatically and uploads the object to it.



Note:

For more information about folders, see [Create folders](#).

6. In the File ACL section, select the ACL of the object to upload. By default, an object inherits the ACL of the bucket to which it belongs.
7. Drag and drop one or more objects to upload to the Upload field, or click Upload to select one or more objects to upload.



Note:

- If the uploaded object has the same name as an existing object in the bucket, the existing object will be overwritten.
- Do not refresh or close the upload page when objects are being uploaded. Otherwise, the upload tasks are interrupted and the upload object list is cleared.

- The name of the uploaded object must comply with the following conventions:
 - The name can contain only UTF-8 characters.
 - The name is case-sensitive.
 - The name must be 1 to 1,023 bytes in length.
 - The name cannot start with a forward slash (/) or backslash (\).

8. After the object is uploaded, refresh the Files tab to view the uploaded object.

2.3.4 Obtain object URLs

You can obtain the URL of an object uploaded to a bucket. This URL can be used to share or download the object.

Prerequisites

Before you obtain an object URL, you must have a bucket and upload an object to it.

Procedure

1. [Log on to the OSS console](#).
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. Click the Files tab. The list of objects appears.
4. Click View Details in the Actions column corresponding to the target object. In the Preview dialog box that appears, click Copy File URL below the URL field. You can also choose More > Copy File URL in the Actions column corresponding to the bucket. In the dialog box that appears, click Copy.

What's next

You can send the URL to other users so that they can view or download the object.

2.4 Buckets

2.4.1 View bucket information

You can view the details of created buckets in the OSS console.

Prerequisites

Before you view bucket information, you must complete the procedure instructed in [Create buckets](#), or at least one bucket exists in the current region.

Procedure

1. [Log on to the OSS console](#).
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. On the bucket details page that appears, click the Overview tab. View the bucket domain names and basic settings.

2.4.2 Delete buckets

You can delete buckets in the OSS console.

Prerequisites

Before you delete a bucket, you must complete the procedure instructed in [Create buckets](#), or at least one bucket exists in the current region.



Note:

To delete a bucket, ensure that all objects in the bucket are deleted, including parts generated from incomplete multipart upload operations. Otherwise, the bucket cannot be deleted.

Procedure

1. [Log on to the OSS console](#).
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. On the bucket details page that appears, click Delete Bucket in the upper right corner. In the message that appears, click OK.



Notice:

Deleted buckets cannot be recovered. Exercise caution when you perform this operation.

2.4.3 Modify bucket ACLs

You can modify the Access Control List (ACL) of a bucket in the OSS console to control access to the bucket.

Prerequisites

Before you modify the ACL of a bucket, you must complete the procedure instructed in [Create buckets](#), or at least one bucket exists in the current region.

Context

OSS provides ACL to control access to buckets. By default, the ACL of a bucket is private when you create the bucket. You can modify the ACL of the bucket after the bucket is created.

OSS provides ACL for buckets. The following ACLs are available for a bucket.

- **Private:** Only the owner or authorized users of this bucket can read and write objects in the bucket. Other users, including anonymous users cannot access the objects in the bucket without authorization.
- **Public read:** Only the owner or authorized users of this bucket can write objects in the bucket. Other users, including anonymous users can only read objects in the bucket.
- **Public read/write:** Any users, including anonymous users can read and write objects in the bucket. Fees incurred by such operations are paid by the owner of the bucket. Exercise caution when you configure this option.



Notice:

If you set ACL to public read or public read/write, other users can directly read the data in the bucket without authentication, resulting in security risks. For data security reasons, we recommend that you set the bucket ACL to private.

Procedure

1. [Log on to the OSS console](#).
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. On the bucket details page, click the Basic Settings tab. Find the Access Control List (ACL) section.
4. Click Configure. Modify the bucket ACL.
5. Click Save.

2.4.4 Configure static website hosting

You can configure static website hosting in the OSS console so that users can access the static website through the bucket domain name.

Prerequisites

Before you configure static website hosting for a bucket, you must complete the procedure instructed in *Create buckets*, or at least one bucket exists in the current region.

Context

Static website hosting is not enabled if the default pages are not specified.

After the default homepage is configured, the default homepage is displayed if you access the root domain name of the static website or any URL ending with a forward slash (/) under this domain name.

Procedure

1. *Log on to the OSS console.*
2. In the left-side navigation pane, click the name of the target bucket.
3. On the bucket details page, click the Basic Settings tab. Find the Static Pages section.
4. Click **Configure**. Configure the following parameters:
 - **Default Homepage:** Specify the name of the index document that links to the index page. The index page functions similar to index.html. Only the HTML object in the root folder of the bucket can be used.
 - **Default 404 Page:** Specify the name of the error document that links to the error page displayed when the requested resource does not exist. Only the HTML, JPG, PNG, BMP, or WebP object in the root folder can be used. The default 404 page is not enabled if you do not specify this parameter.
5. Click **Save**.

2.4.5 Configure logging

You can enable or disable bucket logging in the OSS console.

Prerequisites

Before you enable or disable logging for a bucket, you must complete the procedure instructed in *Create buckets*, or at least one bucket exists in the current region.

Context

You can store access logs in the current bucket or in a new bucket.

Procedure

1. *Log on to the OSS console.*

2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. On the bucket details page that appears, click the Basic Settings tab. Find the Logging section.
4. Click Configure. Turn on Logging. Set Destination Bucket and Log Prefix.
 - **Destination Bucket:** Select the name of the bucket in which access logs are to store from the drop-down list. You must be the owner of the selected bucket and the bucket must be in the same region as the bucket for which logging is enabled.
 - **Log Prefix:** Enter the directory where the access logs are generated and stored and the prefix of the access logs. If you specify `log/<TargetPrefix>`, the access logs are stored in the `log/` directory.
5. Click Save.

2.4.6 Configure hotlink protection

You can configure hotlink protection for a bucket in the OSS console to prevent unauthorized domain names from accessing the data in your bucket.

Prerequisites

Before you configure hotlink protection for a bucket, you must complete the procedure instructed in [Create buckets](#), or at least one bucket exists in the current region.

Context

OSS provides hotlink protection to prevent other domain names from accessing your data in OSS. You can configure the Referer field in the HTTP header to implement hotlink protection. You can configure a Referer whitelist in the OSS console for a bucket or configure whether to accept requests whose Referer field is empty. For example, you can add `http://www.aliyun.com` for a bucket named `oss-example`. Then, requests whose Referer field is set to `http://www.aliyun.com` can access the objects in the `oss-example` bucket.

Procedure

1. [Log on to the OSS console](#).
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.

3. On the bucket details page, click the Basic Settings tab. Find the Hotlinking Protection section.
4. Click Configure. Configure the following parameters:
 - **Referer:** Add URLs to the whitelist. Referers are typically in URL format. Separate multiple Referers with new lines. You can use question marks (?) and asterisks (*) as wildcard characters.
 - **Allow Empty Referer:** Specify whether to allow requests whose Referer field is empty. If you do not allow empty Referers fields, only HTTP or HTTPS requests which include an allowed Referer field value can access the objects in the bucket.
5. Click Save.

2.4.7 Configure CORS

You can configure cross-origin resource sharing (CORS) in the OSS console to enable cross-origin access.

Prerequisites

Before you configure CORS, you must complete the procedure instructed in [Create buckets](#), or at least one bucket exists in the current region.

Context

OSS provides CORS in HTML5 to enable cross-origin access. When OSS receives a cross-origin request (or an OPTIONS request), it reads the CORS rules of the bucket and then checks the relevant permissions. OSS matches the rules one by one. When OSS finds the first match, it returns a corresponding header. If none of the rules match, OSS does not attach any CORS header.

Procedure

1. [Log on to the OSS console](#).
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. On the bucket details page, click the Basic Settings tab. Find the Cross-Origin Resource Sharing (CORS) section. Click Configure.

4. Click **Create Rule**. In the **Create Rule** dialog box that appears, configure the following parameters.

Parameter	Required	Description
Sources	Yes	Specifies the sources from which you want to allow cross-origin requests . You can configure multiple origins and separate them with new lines . Each origin can contain only one asterisk (*) wildcard. If Sources is set to asterisk (*), all cross-origin requests are allowed.
Allowed Methods	Yes	Specifies the cross-origin request methods that are allowed.
Allowed Headers	No	Specifies the allowed headers in a cross-origin request. Allowed headers are case-insensitive. You can configure multiple headers and separate them with new lines. Each allowed header can contain only one asterisk (*) wildcard. If there are no special header requirements, we recommend that you set Allowed Headers to asterisk (*) to allow all requests.
Exposed Headers	No	Specifies the list of headers that can be exposed to the browser. The headers are the response headers that allow access from an application such as XMLHttpRequest in JavaScript. No asterisk (*) wildcards are allowed.
Cache Timeout (Seconds)	No	Specifies the time the browser can cache the response to a preflight (OPTIONS) request to a specific resource.



Note:

You can configure up to 10 rules for each bucket.

5. Click **OK**.

2.4.8 Manage lifecycle rules

You can define and manage lifecycle rules for a bucket in the OSS console.

Prerequisites

Before you manage lifecycle rules of a bucket, you must complete the procedure instructed in [Create buckets](#), or at least one bucket exists in the current region.

Context

You can define a rule for a full set or a subset (by specifying the prefix keyword) of objects in a bucket. A rule is automatically applied to all objects that match the rule. You can manage lifecycle rules to perform operations, such as object management and automatic part deletion.

- If an object matches a rule, data of the object is cleared within two days from the effective date.
- Data that is deleted based on a lifecycle rule cannot be recovered. Configure a rule only when necessary.

Procedure

1. [Log on to the OSS console](#).
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. Click the Basic Settings tab. Find the Lifecycle section. Click Configure.
4. Click Create Rule. In the Create Rule dialog box that appears, configure the following parameters:
 - **Status:** Configure the status of the rule: Enabled or Disabled.
 - **Applied To:** You can select Files with Specified Prefix or Whole Bucket. Files with Specified Prefix indicates that this rule applies to objects that have a specified prefix. Whole Bucket indicates that this rule applies to all objects in the bucket.



Note:

If you select `Files with Specified Prefix`, you can configure multiple lifecycle rules that have different prefixing configurations for objects. If you select `Whole Bucket`, only one lifecycle rule can be configured.

- **Prefix:** If you set `Applied To` to `Files with Specified Prefix`, you must enter the prefix of the objects to which to apply the rule. If you want to match objects whose names start with `img`, enter `img`.
- **File Lifecycle:** Configure operations to perform on expired objects. You can select `Validity Period (Days)`, `Expiration Date`, or `Disabled`.
 - **Validity Period (Days):** Specify the number of days within which objects can be retained after they are last modified, and the operation to perform on these objects after they expire. Objects that meet the specified conditions are retained for the specified validity period after the objects are last modified. The specified operation is performed on these objects after they expire. If you set `Validity Period (Days)` to 30, objects that are last modified on January 1, 2016 are scanned for by the backend application and deleted on January 31, 2016.
 - **Expiration Date:** Specify the date before which objects that are last modified expire and the operation to perform on these objects after they expire. All objects that are last modified before this date expire and the specified operation is performed on these objects. If you select `Transit to Archive Storage Class` and set `Expiration Date` to 2012-12-21, the backend application scans for objects that are last modified before December 21, 2012 and converts their storage class to `Archive`.
 - **Disabled:** The automatic object deletion or storage class conversion function is not enabled.
- **Fragment Lifecycle:** Configure the delete operation to perform on expired parts. You can select `Validity Period (Days)`, `Expiration Date`, or `Disabled`.
 - **Validity Period (Days):** Specify the number of days within which parts can be retained after they are last modified. After the validity period, expired parts are deleted. If you set `Validity Period (Days)` to 30, the

backend application scans for parts that are last modified before January 1, 2016 and deletes them on January 31, 2016.

- **Expiration Date:** Specify the date before which parts that are last modified expire and the operation to perform on these parts after they expire. If you set Expiration Date to 2012-12-21, parts that are last modified before this date are scanned for and deleted by the backend application.
- **Disabled:** The automatic part deletion function is not enabled.



Notice:

In each lifecycle rule, you must configure at least object lifecycle or part lifecycle. In other words, you must select Delete or configure conversion actions for object lifecycle or select Delete for part lifecycle.

5. Click OK.



Notice:

- Lifecycle rules are run after they are configured and saved. Check the configurations before you save them.
- Object deletion is irreversible. Configure lifecycle rules as needed.

2.5 Objects

2.5.1 Search for objects

You can search buckets or folders for objects whose names contain a specified prefix in the OSS console.

Prerequisites

Before you search for objects, you must complete the procedure instructed in [Create buckets](#) and [Upload objects](#), or at least one bucket exists in the current region and at least one object exists in the bucket.

Context

When you search for objects based on a prefix, the search string is case-sensitive and cannot contain a forward slash (/). The search range is limited to the root directory of the current bucket or the objects in the current folder (excluding subfolders and the objects in them).

Procedure

1. [Log on to the OSS console](#).
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. On the bucket details page that appears, click the Files tab. The Files tab appears.
4. On the right side of the Files tab, enter a prefix in the search box and press Enter or click the search icon to search for the related objects.

The system lists the names of objects and folders that match the prefix in the root directory of the bucket.



Note:

To search a specific folder for objects, open the folder and enter a prefix in the search box. The system lists the names of objects and subfolders in the folder that match the prefix.

2.5.2 Delete objects

You can delete uploaded objects in the OSS console.

Prerequisites

Before you delete objects, you must complete the procedure instructed in [Create buckets](#) and [Upload objects](#), or at least one bucket exists in the current region and at least one object exists in the bucket.

Context

You can delete one or more objects at a time. A maximum of 1,000 objects can be deleted at a time. You can use an SDK or call an API operation to delete a specific object or more than 1,000 objects.



Notice:

Deleted objects cannot be recovered. Exercise caution when you delete objects.

Procedure

1. [Log on to the OSS console](#).
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. On the bucket details page that appears, click the Files tab.

4. Select one or more objects. Choose **Batch Operation > Delete**. You can also choose **More > Delete** in the **Actions** column corresponding to the target object.
5. In the **Delete File** message that appears, click **OK**.

2.5.3 Configure ACL for objects

You can configure ACL for an object in the OSS console to control access to the object.

Prerequisites

Before you configure ACL for an object, you must complete the procedure instructed in *Create buckets* and *Upload objects*, or at least one bucket exists in the current region and at least one object exists in the bucket.

Procedure

1. *Log on to the OSS console.*
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. On the bucket details page that appears, click the **Files** tab.
4. On the **Files** tab, click the name of the target object. The **View Details** dialog box appears.
5. Click **Set ACL** on the right side of **File ACL**. The **Set ACL** dialog box appears.

ACLs are described as follows:

- **Inherited from Bucket:** The ACL of each object is the same as that of the bucket.
 - **Private:** Only the owner or authorized users of this bucket can read and write objects in the bucket. Other users, including anonymous users cannot access the objects in the bucket without authorization.
 - **Public Read:** Only the owner or authorized users of this bucket can write objects in the bucket. Other users, including anonymous users can only read objects in the bucket.
 - **Public Read/Write:** Any users, including anonymous users can read and write objects in the bucket. Fees incurred by such operations are paid by the owner of the bucket. Configure this option only when necessary.
6. Click **OK**.

2.5.4 Create folders

You can create a folder in a bucket in the OSS console.

Prerequisites

Before you create a folder, you must complete the procedure instructed in [Create buckets](#), or at least one bucket exists in the current region.

Context

OSS does not use traditional folders. Folders are virtual in OSS. All elements are stored as objects. In the OSS console, a folder is an object whose size is 0 and has a name that ends with a forward slash (/). A folder is used to sort objects of the same type and process them at a time. The OSS console displays objects that end with a forward slash (/) as folders. These objects can be uploaded and downloaded. You can use OSS folders in the OSS console the way you use folders in Windows.



Note:

The OSS console displays any objects that end with a forward slash (/) as folders, regardless of whether these objects contain data. You can download these objects only by calling an API operation or using an SDK.

Procedure

1. [Log on to the OSS console](#).
2. In the left-side navigation pane, click the name of a bucket to go to the bucket details page.
3. Click the Files tab. On the Files tab that appears, click Create Folder.
4. In the Create Folder dialog box that appears, set Folder Name.

The folder name must comply with the following conventions:

- The name can contain only UTF-8 characters. The name cannot contain emojis.
 - The name cannot start with a forward slash (/) or backslash (\). The name cannot contain consecutive forward slashes (/). Folders are separated with forward slashes (/). Subfolders in this path are automatically created.
 - The subfolder name cannot contain two consecutive periods (. .).
 - The name must be 1 to 254 characters in length.
5. Click OK.

3 ApsaraDB for RDS

3.1 What is ApsaraDB for RDS?

ApsaraDB for RDS is a stable, reliable, and scalable online database service. Based on the distributed file system and high-performance storage, ApsaraDB for RDS allows you to easily perform database operations and maintenance with its set of solutions for disaster recovery, backup, restoration, monitoring, and migration.

ApsaraDB RDS for MySQL

Originally based on a branch of MySQL, ApsaraDB RDS for MySQL is a tried and tested solution for handling high-volume concurrent traffic during Double 11, providing excellent performance. ApsaraDB RDS for MySQL provides whitelist configuration, backup and restoration, transparent data encryption, data migration, and management for instances, accounts, and databases. ApsaraDB RDS for MySQL also provides the following advanced features:

- **Read-only instance:** In scenarios where RDS has a small number of write requests but a large number of read requests, you can enable read/write splitting to distribute read requests away from the primary instance. Read-only instances allow ApsaraDB RDS for MySQL 5.6 to automatically scale the reading capability and increase the application throughput when a large amount of data is being read.
- **Data compression:** ApsaraDB RDS for MySQL 5.6 allows you to use the TokuDB storage engine to compress data. Data transferred from the InnoDB storage engine to the TokuDB storage engine can be reduced by 80% to 90% in volume. 2 TB of data in InnoDB can be compressed to 400 GB or less in TokuDB. In addition to data compression, TokuDB supports transaction and online DDL operations. TokuDB is compatible with MyISAM and InnoDB applications.

3.2 Limits on ApsaraDB RDS for MySQL

Before using ApsaraDB RDS for MySQL, you must understand its limits and take precautions.

To ensure instance stability and security, ApsaraDB RDS for MySQL has some service limits, as listed in [Table 3-1: Limits on ApsaraDB RDS for MySQL](#).

Table 3-1: Limits on ApsaraDB RDS for MySQL

Operation	Limit
Database parameter modification	Database parameters can be modified only by using the ApsaraDB RDS console or API operations. Due to security and stability considerations, only specific parameters can be modified.
Root permissions of databases	The root and SA permissions are not provided.
Database backup	<ul style="list-style-type: none"> You can use the command line interface (CLI) or graphical user interface (GUI) to perform logical backup. You can use the ApsaraDB for RDS console or API operations to perform physical backup.
Database restoration	<ul style="list-style-type: none"> You can use the CLI or GUI to perform logical restoration. You can use the ApsaraDB for RDS console or API operations to perform physical restoration.
Data import	<ul style="list-style-type: none"> You can use the CLI or GUI to perform logical import. You can use the MySQL CLI or DTS to import data.
ApsaraDB RDS for MySQL storage engine	<ul style="list-style-type: none"> Only InnoDB and TokuDB are supported. Due to the inherent shortcomings of the MyISAM engine, some data may be lost. Only some existing instances use the MyISAM engine. MyISAM engine tables in newly created instances are automatically converted to InnoDB engine tables. For performance and security considerations, we recommend that you use the InnoDB storage engine. The Memory engine is not supported. Newly created Memory tables are automatically converted into InnoDB tables.

Operation	Limit
Database replication	ApsaraDB RDS for MySQL provides dual-node clusters based on a primary/secondary replication architecture. The secondary instances in this replication architecture are hidden and cannot be accessed directly.
RDS instance restart	Instances must be restarted by using the ApsaraDB for RDS console or API operations.
Account and database management	You can use the ApsaraDB for RDS console to manage accounts and databases for ApsaraDB RDS for MySQL instances. ApsaraDB RDS for MySQL also allows you to create a privileged account to manage users, passwords, and databases.
Standard account	<ul style="list-style-type: none"> • Custom authorization is not supported. • Both the account management and database management UIs are provided in the ApsaraDB for RDS console. • Instances that support standard accounts also support privileged accounts.
Privileged account	<ul style="list-style-type: none"> • Custom authorization is supported. • On the Accounts and Databases pages in the ApsaraDB for RDS console, the management feature is unavailable. Related operations can only be performed by using code. • A privileged account cannot be reverted back to a standard account.

3.3 Log on to the ApsaraDB for RDS console

This topic describes how to log on to the ApsaraDB for RDS console.

Prerequisites

- Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel. The URL used to access the ASCM console is in the following format: `http://IP address or domain name of the ASCM console/manage`.
- We recommend that you use the Google Chrome browser.

1. In the address bar, enter the URL used to access the ASCM console. Press Enter.

2. Enter your username and password.

The system has a default super administrator, whose username is super. The super administrator can create system administrators. A system administrator can create system users and notify the users of the default passwords by SMS or email.



Note:

When you log on to the ASCM console for the first time, you must modify the password of your username as instructed. For security concerns, your password must meet the minimum complexity requirements: The password must be 8 to 20 characters in length and must contain at least two types of the following characters: letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs (\$), and percent signs (%).

3. Click Login to go to the ASCM console homepage.
4. In the top navigation bar, choose Products > Database Services > ApsaraDB for RDS.

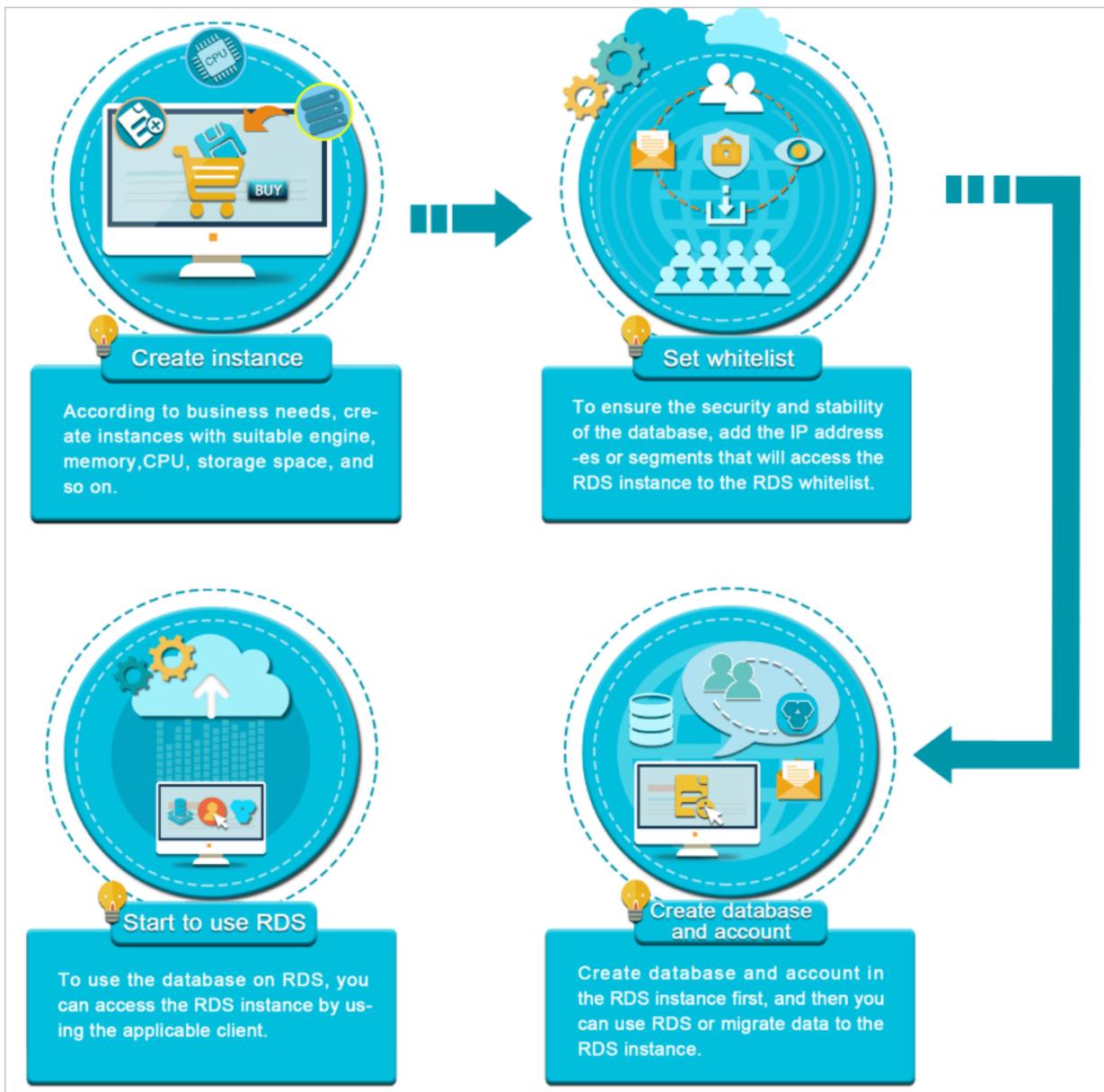
3.4 Quick Start

3.4.1 Procedure

ApsaraDB for RDS quick start covers the following topics: creating an RDS instance, configuring a whitelist, creating a database, creating an account, and connecting

to the instance. This topic uses ApsaraDB RDS for MySQL as an example to describe how to use RDS. It provides all the necessary information to create an RDS instance. Typically, you must complete several operations after instance creation to make it ready for use, as shown in *Figure 3-1: Quick start flowchart*.

Figure 3-1: Quick start flowchart



- *Create an instance*

An instance is a virtualized database server on which you can create and manage multiple databases.

- [Configure a whitelist](#)

After creating an RDS instance, you must configure its whitelist to allow access from external devices.

The whitelist improves the access security of your RDS instance. We recommend that you maintain the whitelist on a regular basis. The whitelist configuration process does not affect the normal operations of the RDS instance.

- [Create a database and an account](#)

Before using a database, you must first create the database and an account in the RDS instance. Different engines support different account modes. For more information, see the console UI and documentation.

- [Connect to an ApsaraDB RDS for MySQL instance](#)

After creating an RDS instance, configuring a whitelist, and creating a database and an account, you can connect to the instance from a database client.

3.4.2 Create an instance

This topic describes how to create an instance in the ApsaraDB for RDS console.

Prerequisites

Before you create an RDS instance, you must apply for an Apsara Stack account.

1. [Log on to the ApsaraDB for RDS console](#).
2. On the Instances page, click Create Instance in the upper-right corner.
3. Configure the following parameters.

Category	Parameter	Description
Region	Region	The region where the RDS instance resides. Services in different regions cannot communicate over the internal network. After a region is selected, it cannot be changed.
Basic Settings	Organization	The organization to which the instance belongs.
	Resource Set	The resource set to which the instance belongs.

Category	Parameter	Description
Specifications	Instance Name	<p>The name of the instance.</p> <p> Note: The name must be 2 to 64 characters in length and can contain letters, digits, underscores (_), hyphens (-), periods (.), colons (:), and commas (,). It must start with a letter and cannot start with http:// or https://.</p>
	Database Engine	The engine of the database, which is automatically set to MySQL.
	Engine Version	The version of the database engine. Set the value to 5.6 or 5.7.
	Instance Type	The type of the instance. Select one from the drop-down list.
	Storage	<p>The storage space of the instance, including the space for data, system files, binlog files, and transaction files. The minimum storage space is 50 GB. You can adjust the storage space in 5 GB increments.</p> <p> Note: For dedicated instances with local SSDs, the storage space is associated with the instance type.</p>
Network Type	Network Access	The network access of the instance. Set the value to Internal Network or Public Network.
	Network Type	<p>The network type of the instance, which is automatically set to Classic Network.</p> <p>Classic Network: Cloud services on a classic network are not isolated. Unauthorized access to a cloud service is blocked only by the security group or whitelist policy of the service.</p>
	IP Whitelist	You can add IP addresses to allow them to connect to the RDS instance.

Category	Parameter	Description
Access Mode	Access Mode	<p>RDS instances support two access modes: Standard and Database Proxy.</p> <ul style="list-style-type: none"> • Standard: RDS uses SLB to eliminate the impact of instance high-availability switching on the application layer. This mode reduces the response time, but slightly increases the probability of transient disconnections and disables SQL interception. • Database Proxy: This mode prevents 90% of transient disconnections and intercepts SQL injection attacks based on semantic analysis. However, it increases the response time by over 20%.

4. After you configure the preceding parameters, click **Submit**.

3.4.3 Initialization

3.4.3.1 Configure a whitelist

To ensure database security and reliability, you must modify the whitelist of an ApsaraDB for RDS instance before you enable the instance. You must add the IP addresses or CIDR blocks that are used for database access to the whitelist.

Context

The whitelist improves the access security of your RDS instance. We recommend that you maintain the whitelist on a regular basis. The whitelist configuration process does not affect the normal operations of the RDS instance.

Precautions

- The default whitelist can only be edited or cleared, but cannot be deleted.
- You can add up to 1,000 IP addresses or CIDR blocks to a whitelist. If you want to add a large number of IP addresses, we recommend that you merge them into CIDR blocks, such as 192.168.1.0/24.

Edit a whitelist

1. [Log on to the ApsaraDB for RDS console](#).
2. Click the ID of an instance.
3. In the left-side navigation pane, click **Data Security**.

4. On the Whitelist Settings tab, click Edit corresponding to the default whitelist.



Note:

- To connect an ECS instance to an RDS instance by using an internal endpoint, you must make sure that the two instances are in the same region and have the same network type. Otherwise, the connection fails.
- You can also click Create Whitelist to create a new whitelist.

5. In the dialog box that appears, specify the IP addresses or CIDR blocks used to access the instance and click OK.

- If you specify the CIDR block 10.10.10.0/24, any IP addresses in the 10.10.10.X format are allowed to access the RDS instance.
- If you want to add multiple IP addresses or CIDR blocks, separate each entry with a comma (without spaces), such as 192.168.0.1,172.16.213.9.
- After you click Add Internal IP Addresses of ECS Instances, the IP addresses of all the ECS instances under your Apsara Stack account are displayed. You can select the required IP addresses to add to the whitelist.



Note:

If you add a new IP address or CIDR block to the default whitelist, the default address 127.0.0.1 is automatically deleted.

Create a whitelist



Note:

You can create up to 50 whitelists for an instance.

The procedure is as follows:

1. [Log on to the ApsaraDB for RDS console](#).
2. Click the ID of an instance to go to the Basic Information page of the instance.
3. In the left-side navigation pane, click Data Security.
4. On the Whitelist Settings tab, click Create Whitelist.

5. In the Create Whitelist dialog box, configure the following parameters.

Parameter	Description
Whitelist Name	<p>The name of the whitelist.</p> <ul style="list-style-type: none"> The whitelist name must be 2 to 32 characters in length. It can contain lowercase letters, digits, and underscores (_). It must start with a lowercase letter and end with a letter or digit.
Whitelist	<p>The IP addresses or CIDR blocks used to access the instance.</p> <ul style="list-style-type: none"> If you specify the CIDR block 10.10.10.0/24, any IP addresses in the 10.10.10.X format are allowed to access the RDS instance. If you want to add multiple IP addresses or CIDR blocks, separate each entry with a comma (without spaces), such as 192.168.0.1,172.16.213.9. After you click Add Internal IP Addresses of ECS Instances, the IP addresses of all the ECS instances under your Apsara Stack account are displayed. You can select the required IP addresses to add to the whitelist.

6. Click OK.

3.4.3.2 Create a privileged account

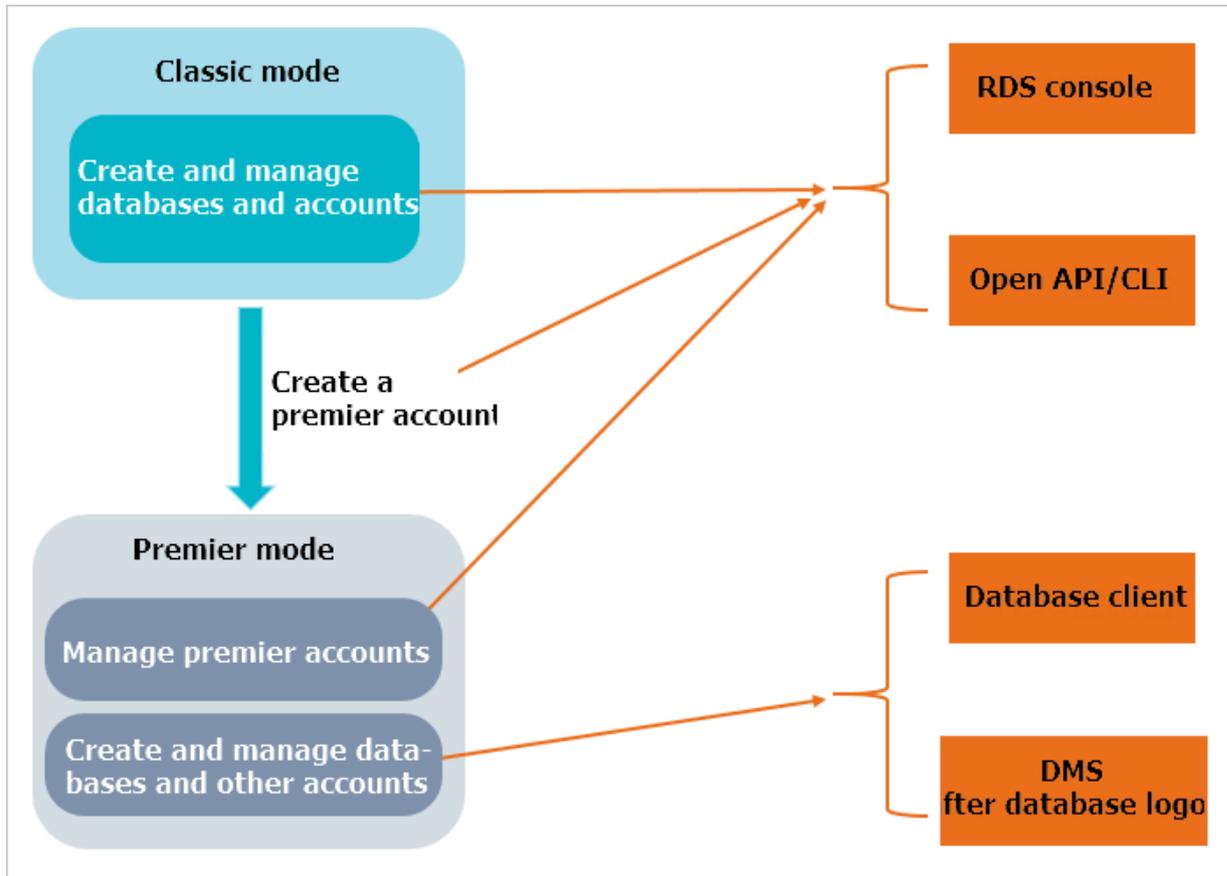
ApsaraDB for RDS supports two account management modes: classic and privileged. For ApsaraDB RDS for MySQL 5.6 or 5.7 instances, you can upgrade the account management mode from classic to privileged by creating a privileged account.

Context

Compared with the classic mode, the privileged mode enables more permissions and allows you to use SQL to directly manage databases and accounts. Therefore, we recommend that you use the privileged mode. After a privileged account is created for a primary instance, the privileged account is synchronized to read-only instances.

Figure 3-2: Comparison of account management modes shows the differences between the two modes in creating and managing databases and accounts for ApsaraDB RDS for MySQL 5.6 instances.

Figure 3-2: Comparison of account management modes



Note:

- In an ApsaraDB RDS for MySQL 5.6 or 5.7 instance, the account management mode can be upgraded from classic to privileged, but cannot be downgraded from privileged to classic.
- The instance restarts when a privileged account is created, causing a transient disconnection of less than 30 seconds. To avoid service impacts from transient disconnections, choose an appropriate time and ensure that your applications can be automatically reconnected.

Procedure

1. *Log on to the ApsaraDB for RDS console.*
2. **Click the ID of an instance.**

3. In the left-side navigation pane, click **Accounts**.
4. On the **Accounts** page, click **Create Account**.
5. Configure the following parameters.

Parameter	Description
Database Account	Enter an account name. The requirements are as follows: <ul style="list-style-type: none"> • The name must be 2 to 16 characters in length. • It must start with a letter and end with a letter or digit. • It can contain lowercase letters, digits, and underscores (_).
Account Type	Select Privileged Account.
Password	Enter an account password. The requirements are as follows: <ul style="list-style-type: none"> • The password must be 8 to 32 characters in length. • The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. • Special characters include ! @ # \$ % ^ & * () _ + - =
Re-enter Password	Enter the password again.
Description	Enter information about the account to facilitate subsequent management. The description can be up to 256 characters in length.

6. Click **Create**.

3.4.3.3 Create a standard account

After you create an ApsaraDB for RDS instance and configure its whitelist, you must create a database and an account in the instance. This topic describes how to create a standard account.

Context

To migrate data from an on-premises database to an ApsaraDB for RDS instance, you must create a database and an account in the RDS instance identical to those in the on-premises database. Databases in an instance share all resources that belong to the instance. You can create up to 500 databases and 500 accounts for an ApsaraDB RDS for MySQL 5.6 instance.

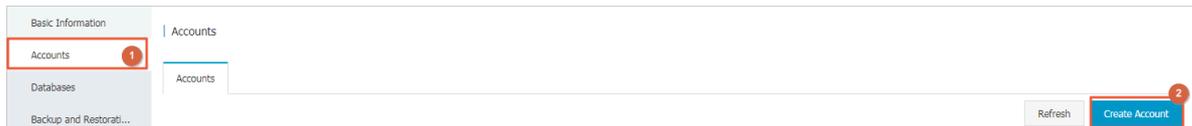
Follow the principle of least privilege to create accounts and assign role permissions. Assign appropriate read-only and read/write permissions to accounts. When necessary, you can split database accounts and databases into smaller units so that each database account can only access data for its own services.



Notice:

To ensure database security, set strong account passwords and change the passwords on a regular basis.

1. *Log on to the ApsaraDB for RDS console.*
2. **Click the ID of an instance.**
3. **In the left-side navigation pane, click Accounts.**
4. **Click Create Account.**



5. **Configure the following parameters.**

Parameter	Description
Database Account	<p>Enter an account name. The requirements are as follows:</p> <ul style="list-style-type: none"> • The name must be 2 to 16 characters in length. • It must start with a letter and end with a letter or digit. • It can contain lowercase letters, digits, and underscores (_).

Parameter	Description
Authorized Databases	<p>Grant permissions on one or more databases to the account. You do not have to configure this parameter at this time. You can authorize databases after the account is created.</p> <ol style="list-style-type: none"> Select one or more databases from the left-side box, and click Add to add them to the right-side box. In the right-side box, select Read/Write or Read-only for a database. <p>If you want to grant the same permissions on multiple databases to the account, click the button in the upper-right corner of the right-side box. The button may appear as Full Control Read/Write.</p> <div style="background-color: #f0f0f0; padding: 5px;">  Note: The button in the upper-right corner changes after you click. For example, after you click Full Control Read/Write, the button changes to Full Control Read-only. </div>
Password	<p>Enter an account password. The requirements are as follows:</p> <ul style="list-style-type: none"> The password must be 8 to 32 characters in length. The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. Special characters include ! @ # \$ % ^ & * () _ + - =
Re-enter Password	Enter the password again.
Description	Optional. Enter information about the account to facilitate subsequent management. The description can be up to 256 characters in length.

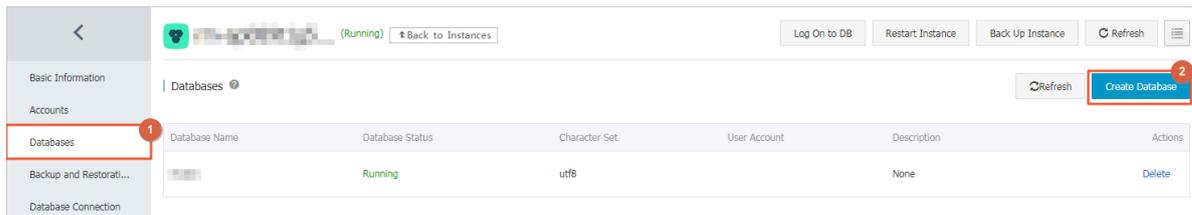
6. Click Create.

3.4.3.4 Create a database

After you create an ApsaraDB for RDS instance and configure its whitelist, you must create a database and an account in the instance.

- Log on to the ApsaraDB for RDS console.
- Click the ID of an instance.
- In the left-side navigation pane, click Databases.

4. Click Create Database.



5. On the Create Database page that appears, configure the following parameters.

Parameter	Description
Database Name	<ul style="list-style-type: none"> The database name must be 2 to 64 characters in length. It must start with a letter and end with a letter or digit. It can contain lowercase letters, digits, underscores (_), and hyphens (-). Each database name must be unique in an instance.
Supported Character Set	<p>Select utf8, gbk, latin1, or utf8mb4.</p> <p>If you want to use other character sets, select all, and then select the required character set from the list.</p>
Description	<p>Optional. Enter information about the database to facilitate subsequent management. The description can be up to 256 characters in length.</p>

6. Click Create.

3.4.4 Connect to an ApsaraDB RDS for MySQL instance

After completing the initial configurations, you can use an ECS instance or a database client to connect to an ApsaraDB RDS for MySQL instance.

Prerequisites

- An ApsaraDB RDS for MySQL instance is created. For more information about how to create an instance, see [Create an instance](#).
- An account is created to connect to the MySQL instance. For more information about how to create an account, see [Create a privileged account](#) and [Create a standard account](#).
- The IP address of the server that needs to connect to the MySQL instance is added to the whitelist. For more information about how to configure a whitelist, see [Configure a whitelist](#).

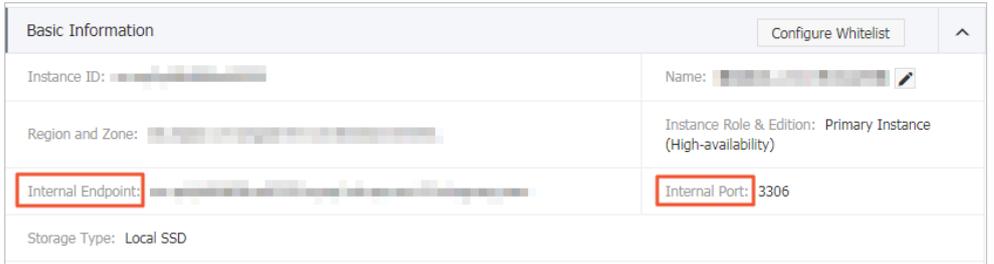
If you need to connect an ECS instance to an ApsaraDB for RDS instance, you must make sure that both instances are in classic networks or in the same VPC, and the IP address of the ECS instance is correctly configured in the RDS whitelist.

Connect to an instance from a client

ApsaraDB RDS for MySQL is fully compatible with MySQL. You can connect to an ApsaraDB for RDS instance from a general database client in the similar way you connect to a MySQL database. The following example uses the *HeidiSQL* client:

1. Start the HeidiSQL client.
2. In the lower-left corner, click Create.
3. Enter information about the RDS instance you want to connect. The parameters are described as follows.

Parameter	Description
Network Type	The network type of the database you want to connect. Select MariaDB or MySQL (TCP/IP).

Parameter	Description
Hostname IP	<p>Enter the internal or public IP address of the RDS instance.</p> <ul style="list-style-type: none"> If your client is deployed in an ECS instance, and the instance is in the same region and has the same network type as the destination RDS instance, you can use the internal IP address. For example, if your ECS and RDS instances are both in a VPC located in the China (Hangzhou) region, you can use the internal IP address provided to create a secure connection. Use the public IP address for other situations. <p>To view the internal and public endpoints together with the corresponding port numbers of the RDS instance, perform the following steps:</p> <ol style="list-style-type: none"> Log on to the ApsaraDB for RDS console. Find the target instance and click its ID. On the Basic Information page that appears, you can view the internal endpoint and internal port number of the instance. 
User	Enter the name of the account used to access the RDS instance.
Password	The password of the account.
Port	The port number of the RDS instance. If you connect to the instance over the internal network, enter the internal port number of the instance. If you connect to the instance over the Internet, enter the public port number of the instance.

- Click Open. If the connection information is correct, you can connect to the instance.

3.5 Instances

3.5.1 Create an instance

This topic describes how to create an instance in the ApsaraDB for RDS console.

Prerequisites

Before you create an ApsaraDB for RDS instance, you must apply for an Apsara Stack tenant account.

1. [Log on to the ApsaraDB for RDS console.](#)
2. On the Instances page, click Create Instance in the upper-right corner.
3. Configure the following parameters.

Category	Parameter	Description
Region	Region	The region where the ApsaraDB for RDS instance resides. Services in different regions cannot communicate over the internal network. After a region is selected, it cannot be changed.
	Zone	The zone of the instance. Common ApsaraDB for RDS instances use a hot standby architecture. Single zone indicates that the primary and secondary nodes are in the same zone.
Basic Settings	Organization	The organization to which the instance belongs.
	Resource Set	The resource set to which the instance belongs.
Specifications	Instance Name	The name of the instance. The name must be 2 to 64 characters in length and can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter.
	Database Type	The database type. Set the value to MySQL.
	Database Version	The database version. Set the value to 5.6.

Category	Parameter	Description
	Instance Specifications	<p>The specifications of the instance. Available specifications are as follows:</p> <ul style="list-style-type: none"> • Dedicated: This type of specifications is followed by the "Dedicated" suffix. • Dedicated Instance: This type of specifications is followed by the "Dedicated Instance" suffix. • Dedicated Host: This type of specifications is followed by the "Dedicated Host" suffix. • Basic Edition: This type of specifications is followed by the "Basic Edition" suffix. <p>Memory size determines the maximum number of connections and IOPS. The actual values are displayed on the console UI.</p>
	Storage	<p>The storage space of the instance, including the space for data, system files, binlog files, and transaction files. The smallest unit interval available to adjust the storage space is 5 GB.</p> <div style="background-color: #f0f0f0; padding: 5px;">  Note: For dedicated instances with local SSDs, the storage space is associated with the instance specifications. </div>
Network Type	Instance Type	The type of the instance.
	Network Type	<p>The network type supported by the RDS instance. Set the value to Classic Network.</p> <p>Classic Network: Cloud services on a classic network are not isolated. Unauthorized access to a cloud service is blocked only by the security group or whitelist policy of the service.</p>
	IP Whitelist	You can add IP addresses to allow them to connect to the RDS instance.

Category	Parameter	Description
Access Mode	Access Mode	<p>ApsaraDB for RDS instances support two access modes: Standard Mode and Database Proxy Mode.</p> <ul style="list-style-type: none"> • Standard Mode: ApsaraDB for RDS uses SLB to eliminate the impact of high-availability switching between database engines on the application layer. This mode reduces the response time, but slightly increases the probability of transient disconnections and disables SQL interception. • Database Proxy Mode: This mode prevents 90% of transient disconnections and intercepts SQL injection attacks based on semantic analysis. However, it increases the response time by over 20%.

4. After you set the preceding parameters, click Submit.

3.5.2 View basic information about an instance

You can view the details of an instance, such as its basic information, internal network connection information, running status, and configurations.

1. [Log on to the ApsaraDB for RDS console.](#)
2. You can use one of the following methods to access the Basic Information page of an instance:
 - On the RDS Management page, click the ID of the instance to access its Basic Information page.
 - On the RDS Management page, click Management in the Actions column corresponding to the instance to access its Basic Information page.

3.5.3 Restart an instance

If the number of connections exceeds the threshold or any performance issue occurs on an instance, you can manually restart the instance.



Notice:

A restart will disconnect the instance. We recommend that you make appropriate service arrangements before you restart an instance. Proceed with caution.

1. [Log on to the ApsaraDB for RDS console.](#)

2. Click the ID of an instance.
3. In the upper-right corner of the page, click Restart Instance.
4. In the Restart Instance message that appears, click Confirm to restart the instance.

3.5.4 Change specifications

You can change specifications of your instance, such as the instance type and storage space, if the specifications do not meet the requirements of your application.

Procedure

1. *Log on to the ApsaraDB for RDS console.*
2. Click the name of an instance to go to the Basic Information page.
3. Click Change Specifications.
4. In the dialog box that appears, click Next.
5. On the Change Specifications page that appears, select Instance Type and Storage.
6. After you configure the preceding parameters, click Submit.

3.5.5 Set a maintenance window

This topic describes how to set a maintenance window for an ApsaraDB for RDS instance.

Context

To ensure the stability of ApsaraDB for RDS instances, the backend system performs maintenance of the instances at irregular intervals. The default maintenance window is from 02:00 to 06:00. You can set the maintenance window to the off-peak period of your business to avoid impact on business.

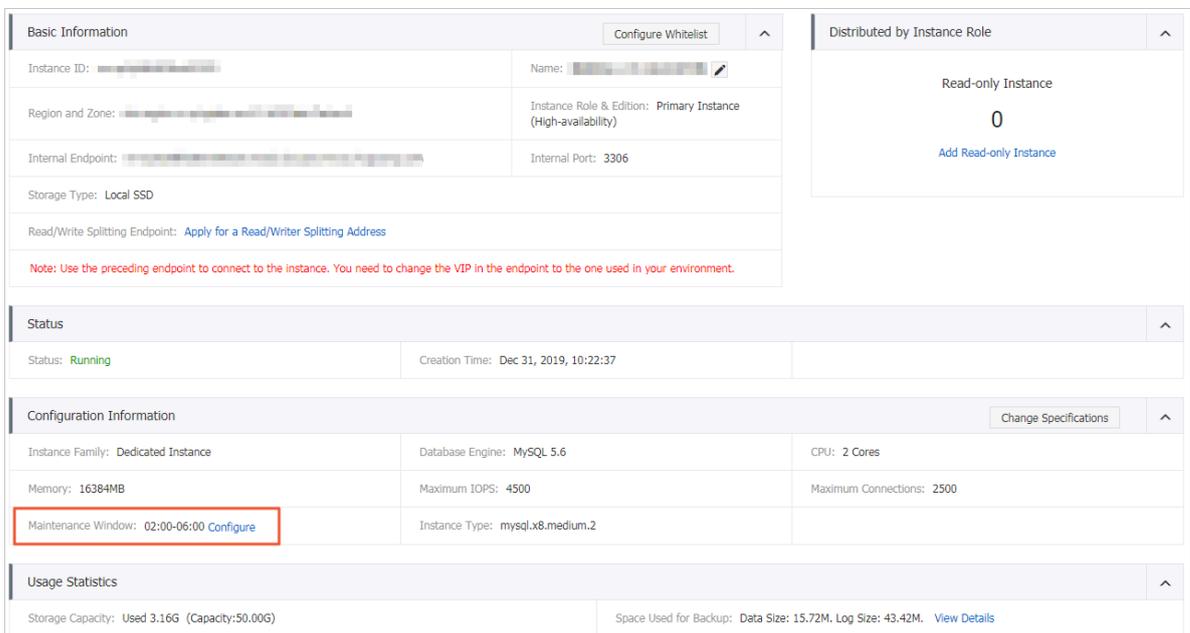
Precautions

- To ensure the stability of the maintenance process, the instance will enter the Instance Maintaining state before the maintenance time. When the instance is in this state, access to data in the database and query operations such as performance monitoring are not affected. However, apart from account and database management and IP address whitelist configuration, modification operations such as upgrade, downgrade, and restart are temporarily unavailable.

- **During the maintenance window, the instance is disconnected once or twice. Make sure that you configure automatic reconnection policies for your applications to avoid service disruptions.**

Procedure

1. [Log on to the ApsaraDB for RDS console.](#)
2. **Click the ID of an instance or click Management in the Actions column corresponding to the instance.**
3. **In the Configuration Information section, click Configure to the right of Maintenance Window.**



The screenshot displays the configuration page for an RDS instance. It is divided into several sections:

- Basic Information:** Includes Instance ID, Name, Region and Zone, Internal Endpoint, Storage Type (Local SSD), and Read/Write Splitting Endpoint. A note at the bottom states: "Note: Use the preceding endpoint to connect to the instance. You need to change the VIP in the endpoint to the one used in your environment."
- Status:** Shows the instance is in a "Running" state with a creation time of Dec 31, 2019, 10:22:37.
- Configuration Information:** Lists Instance Family (Dedicated Instance), Database Engine (MySQL 5.6), CPU (2 Cores), Memory (16384MB), Maximum IOPS (4500), Maximum Connections (2500), and Instance Type (mysql.x8.medium.2). The **Maintenance Window** is set to 02:00-06:00, and the **Configure** link next to it is highlighted with a red box.
- Usage Statistics:** Shows Storage Capacity (Used 3.16G, Capacity: 50.00G) and Space Used for Backup (Data Size: 15.72M, Log Size: 43.42M).

4. **Select a maintenance window and click Save.**



Note:

The maintenance window is in UTC+8.

3.5.6 Change the data replication mode

This topic describes how to change the mode of data replication between a primary ApsaraDB for RDS instance and its secondary instances to increase database availability.

You can use the following methods to replicate data:

- **Semi-sync**

After an application-initiated update is completed on the primary instance of a cluster, logs are synchronized to all secondary databases. This transaction is considered committed after at least one node in the cluster has received the logs. This way, there is no need to wait for the logs to be applied.

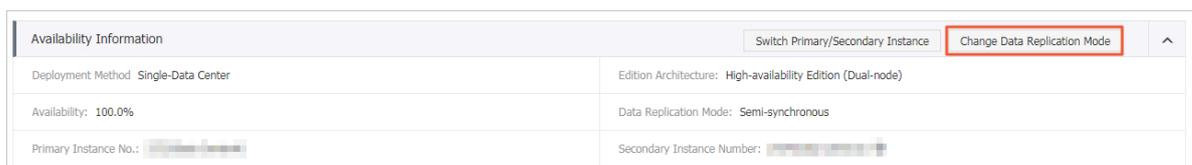
If the secondary instances are unavailable or a network exception occurs between the primary and secondary instances, semi-synchronous replication will degrade to Async mode.

- **Async**

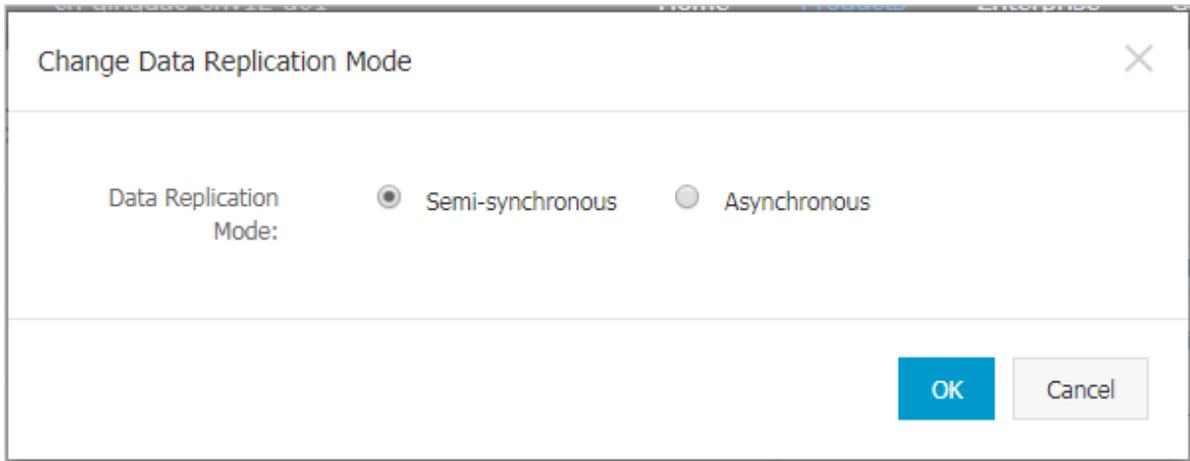
When an application initiates an update request to add, delete, or modify data, the primary instance responds to the application immediately after completing the operation. The primary instance then replicates data to the secondary instances asynchronously. During asynchronous data replication, the unavailability of secondary instances does not affect the operations on the primary instance. Data will remain consistent even if the primary instance is unavailable.

Procedure

1. [Log on to the ApsaraDB for RDS console.](#)
2. Click the ID of an instance.
3. In the left-side navigation pane, click **Service Availability**.
4. Click **Change Data Replication Mode**.



5. In the dialog box that appears, select a data replication mode and click OK.



3.5.7 Release an instance

You can manually release instances as needed.

Precautions

- You can manually release only instances that are in the running state.
- After an instance is released, the instance data is immediately cleared. We recommend that you back up your data before you release an instance.

1. [Log on to the ApsaraDB for RDS console.](#)
2. In the Actions column corresponding to the instance you want to release, choose More > Release Instance.
3. In the Release Instance message that appears, click Confirm.

3.6 Accounts

3.6.1 Create a database account

This topic describes the functions and features of accounts in classic and privileged modes, and how to create accounts for both modes.

You must create an account in an ApsaraDB for RDS instance before you can use the database. ApsaraDB for RDS supports two account management modes: classic and privileged. The classic mode is a management mode retained from earlier versions of ApsaraDB for RDS. In the classic mode, databases and accounts cannot be managed by using SQL statements. The privileged mode is a management mode introduced in later versions that enables more permissions. In the privileged mode

, databases and accounts can be managed by using SQL statements. We recommend that you use the privileged mode for personalized and fine-grained control over database permissions.

Account modes

In the classic mode, all accounts are created by using the ApsaraDB for RDS console or API operations, instead of by using SQL statements. All accounts are equal. There is not one account with more management permissions over others. You can use the ApsaraDB for RDS console to create and manage all accounts and databases.

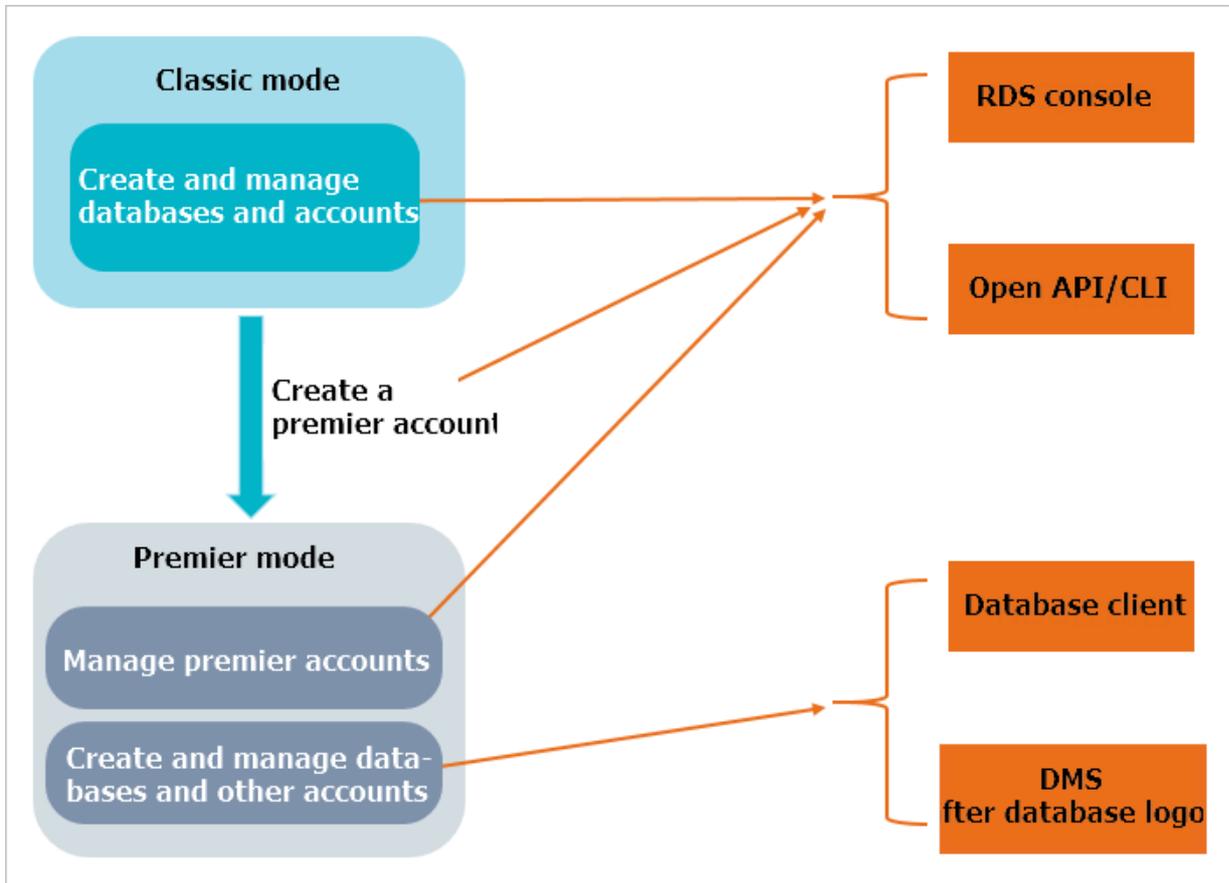
In the privileged mode, the first account that you create is the initial account. You must use the ApsaraDB for RDS console or API operations to create and manage the initial account. Log on to a database with your initial account. You can then use SQL statements or DMS to create and manage standard accounts. However, you cannot use the initial account to change the passwords of standard accounts. To change the password of a standard account, you must delete the account and create a new one . Run the following commands to log on to the database by using the initial account named root and create a standard account named jeffrey:

```
mysql -hxxxxxxxxx.mysql.rds.aliyuncs.com -uroot -pxxxxxx -e "  
    CREATE USER 'jeffrey'@'%' IDENTIFIED BY 'password';  
    CREATE DATABASE DB001;  
    "
```

In the privileged mode, you cannot manage databases by using the ApsaraDB for RDS console or API operations. You must use SQL statements or DMS to create and manage databases.

Figure 3-3: Difference between standard and privileged accounts shows how to create and manage databases and accounts in classic and privileged modes.

Figure 3-3: Difference between standard and privileged accounts



How to create an account



Note:

- Use service roles to create accounts and follow the principle of least privilege to assign appropriate read-only and read/write permissions to accounts. When necessary, you can split database accounts and databases into smaller units so that each database account can only access data for its own services. If an account does not need to write data to a database, assign read-only permissions to the account.
- Use strong passwords for database accounts and change the passwords on a regular basis.
- For more information about how to create a standard account for ApsaraDB RDS for MySQL, see [Create a standard account](#).

- For more information about how to create a privileged account for ApsaraDB RDS for MySQL, see [Create a privileged account](#).

3.6.2 Reset your password

You can use the ApsaraDB for RDS console to reset the password of your database account.



Note:

To ensure data security, we recommend that you change your password on a regular basis.

1. [Log on to the ApsaraDB for RDS console](#).
2. Click the ID of an instance.
3. In the left-side navigation pane, click **Accounts**.
4. Find the target account and click **Reset Password** in the **Actions** column.
5. In the dialog box that appears, enter and confirm the new password, and then click **OK**.



Note:

The password must meet the following requirements:

- The password must be 8 to 32 characters in length.
- It must contain characters from at least three of the following categories: uppercase letters, lowercase letters, digits, and special characters.
- Special characters include ! @ # \$ % ^ & * () _ + - =

3.6.3 Modify account permissions

You can modify the account permissions of your instances at any time when using ApsaraDB for RDS.



Note:

You can modify the permissions of a standard account as needed. The permissions of privileged accounts can only be reset to the default settings and cannot be changed to a specific set of permissions.

1. [Log on to the ApsaraDB for RDS console](#).
2. Click the ID of an instance.

3. In the left-side navigation pane, click **Accounts**.
4. Find the target account and click **Modify Permissions** in the **Actions** column.
5. Configure the following parameters.

Parameter	Description
Authorize Databases	Select a database and click Add or Remove .
Authorized Databases	<p>In the Authorized Databases list, you can set the account permissions to Read/Write or Read-only. You can also click Full Control Read/Write or Full Control Read-only to set the permissions of the account on all authorized databases.</p> <ul style="list-style-type: none"> • Read-only: grants the database read-only permissions to the account. • Read/Write: grants the database read/write permissions to the account.

6. Click **OK**.

3.6.4 Delete an account

You can delete a database account from the ApsaraDB for RDS console.

You can use the console to delete privileged and standard accounts that are no longer used.

1. [Log on to the ApsaraDB for RDS console.](#)
2. Click the ID of an instance.
3. In the left-side navigation pane, click **Accounts**.
4. Find the account you want to delete and click **Delete** in the **Actions** column.
5. In the message that appears, click **Confirm**.

3.7 Databases

3.7.1 Create a database

After you create an ApsaraDB for RDS instance and configure its whitelist, you must create a database and an account in the instance.

1. [Log on to the ApsaraDB for RDS console.](#)
2. Click the ID of an instance.

3. In the left-side navigation pane, click Databases.

4. Click Create Database.



5. On the Create Database page that appears, configure the following parameters.

Parameter	Description
Database Name	<ul style="list-style-type: none"> • The database name must be 2 to 64 characters in length. • It must start with a letter and end with a letter or digit. • It can contain lowercase letters, digits, underscores (_), and hyphens (-). • Each database name must be unique in an instance.
Supported Character Set	<p>Select utf8, gbk, latin1, or utf8mb4.</p> <p>If you want to use other character sets, select all, and then select the required character set from the list.</p>
Description	<p>Optional. Enter information about the database to facilitate subsequent management. The description can be up to 256 characters in length.</p>

6. Click Create.

3.7.2 Delete a database

You can delete out-of-date databases in the ApsaraDB for RDS console.

1. [Log on to the ApsaraDB for RDS console.](#)
2. Click the ID of an instance.
3. In the left-side navigation pane, click Databases.
4. Find the database you want to delete and click Delete in the Actions column.
5. In the message that appears, click Confirm.

3.8 Database connection

3.8.1 Change the endpoint of an instance

This topic describes how to change the endpoint of an instance.

1. [Log on to the ApsaraDB for RDS console](#)
2. Click the ID of an instance.
3. In the left-side navigation pane, click Database Connection.
4. Click Change Endpoint.
5. In the dialog box that appears, set Connection Type, Endpoint, and Port, and click OK.



Note:

- The prefix of the endpoint must be 8 to 64 characters in length and can contain letters, digits, and hyphens (-). It must start with a lowercase letter.
- The port number must be in the range of 1000 to 65535.

3.8.2 Switch the access mode

ApsaraDB for RDS supports two access modes: Standard Mode and Safe Mode. This topic describes the differences between the two access modes and their configuration methods.

Standard Mode and Safe Mode have the following differences:

- **Standard Mode:** ApsaraDB for RDS uses SLB to eliminate the impact of high-availability switching between database engines on the application layer. This mode reduces the response time, but slightly increases the probability of transient disconnections and disables SQL interception.
- **Safe Mode:** This mode prevents 90% of transient disconnections and intercepts SQL injection attacks based on semantic analysis. However, it increases the response time by more than 20%.

1. [Log on to the ApsaraDB for RDS console](#).
2. Click the ID of an instance.
3. In the left-side navigation pane, click Database Connection.
4. Click Switch Access Mode.



Note:

When the access mode change is in progress, Status of the instance changes to Switching the access mode. When Status changes to Running, the access mode is changed.

3.9 Monitoring and alerts

The ApsaraDB for RDS console provides a variety of performance metrics for you to monitor the status of your instances.

3.9.1 View resource and engine monitoring data

The ApsaraDB for RDS console provides a variety of performance metrics for you to monitor the status of your instances.

Procedure

1. [Log on to the ApsaraDB for RDS console.](#)
2. Click the ID of an instance.
3. In the left-side navigation pane, click Monitoring and Alerts.
4. On the Monitoring and Alerts page, select Resource Monitoring or Engine Monitoring, and select a time range to view the corresponding monitoring data. The specific metrics are described as follows.

Category	Metric	Description
Resource Monitoring	Disk Space (MB)	<p>The disk space usage of the instance, including:</p> <ul style="list-style-type: none"> • Instance size • Data usage • Log size • Temporary file size • Other system file size <p>Unit: MB.</p>
	IOPS (Input/Output Operations per Second)	The number of I/O requests per second for the instance.
	Total Connections	The total number of connections to the instance, including the number of active connections and the total number of connections.

Category	Metric	Description
	CPU and Memory Usage (%)	The CPU and memory usage of the instance (excluding the CPU and memory usage for the operating system).
	Network Traffic (KB)	The inbound and outbound traffic of the instance per second. Unit: KB.
Engine Monitoring	TPS (Transactions per Second)/ QPS (Queries per Second)	The average number of transactions per second and the average number of SQL statements executed per second.
	InnoDB Buffer Pool Read Hit Ratio, Usage Ratio, and Dirty Block Ratio (%)	The read hit ratio, usage ratio, and dirty block ratio of the InnoDB buffer pool.
	InnoDB Read/Write Volume (KB)	The amount of data that InnoDB reads and writes per second. Unit: KB.
	InnoDB Buffer Pool Read/Write Frequency	The number of read and write operations that InnoDB performs per second.
	InnoDB Log Read/Write/fsync	The average frequency of physical writes to log files per second by InnoDB, the log write request frequency, and the average frequency of fsync writes to log files.
	Number of Temporary Tables Created Automatically on the Hard Disk when MySQL Statements Are Being Executed	The number of temporary tables that are automatically created on the hard disk when the database executes SQL statements.

Category	Metric	Description
	MySQL_COMDML	<p>The number of SQL statements that the database executes per second. The SQL statements include:</p> <ul style="list-style-type: none"> • Insert • Delete • Insert_Select • Replace • Replace_Select • Select • Update
	MySQL_RowDML	<p>The number of operations that InnoDB performs per second, including:</p> <ul style="list-style-type: none"> • The average number of physical writes to log files per second • The number of rows that are read, updated, deleted, and inserted from InnoDB tables per second

3.9.2 Set a monitoring frequency

The ApsaraDB for RDS console provides a variety of performance metrics for which you can set a monitoring frequency.

ApsaraDB for RDS provides the following monitoring frequencies:

- Every 5 seconds for the first 7 days. After seven days, performance metrics are monitored every minute.
- Every 60 seconds for 30 days.
- Every 300 seconds for 30 days.

The following table lists the monitoring configuration policies in detail.

Instance type	Every 5 seconds	Every 60 seconds	Every 300 seconds
Basic Edition	Not supported	Supported for free	Default configuration

Instance type	Every 5 seconds	Every 60 seconds	Every 300 seconds
High-availability Edition and Enterprise Edition (formerly known as Finance Edition): less than 8 GB memory	Not supported	Supported for free	Default configuration
High-availability Edition and Enterprise Edition (formerly known as Finance Edition): at least 8 GB memory	Supported on a pay-as-you-go basis	Default configuration	Supported for free

1. [Log on to the ApsaraDB for RDS console.](#)
2. Click the ID of an instance.
3. In the left-side navigation pane, click **Monitoring and Alerts**.
4. On the **Resource Monitoring** tab, click **Set Monitoring Frequency**.
5. In the **Set Monitoring Frequency** dialog box, select the monitoring frequency you want.

6. Click **OK**.



Note:

If your instance does not support the selected monitoring frequency, a prompt is displayed in the Set Monitoring Frequency dialog box. Select a monitoring frequency supported by the instance as prompted.

3.10 Data security

3.10.1 Configure a whitelist

To ensure database security and reliability, you must modify the whitelist of an ApsaraDB for RDS instance before you enable the instance. You must add the IP addresses or CIDR blocks that are used for database access to the whitelist.

Context

The whitelist improves the access security of your RDS instance. We recommend that you maintain the whitelist on a regular basis. The whitelist configuration process does not affect the normal operations of the RDS instance.

Precautions

- The default whitelist can only be edited or cleared, but cannot be deleted.
- You can add up to 1,000 IP addresses or CIDR blocks to a whitelist. If you want to add a large number of IP addresses, we recommend that you merge them into CIDR blocks, such as 192.168.1.0/24.

Edit a whitelist

1. [Log on to the ApsaraDB for RDS console](#).
2. Click the ID of an instance.
3. In the left-side navigation pane, click Data Security.
4. On the Whitelist Settings tab, click Edit corresponding to the default whitelist.



Note:

- To connect an ECS instance to an RDS instance by using an internal endpoint, you must make sure that the two instances are in the same region and have the same network type. Otherwise, the connection fails.
- You can also click Create Whitelist to create a new whitelist.

5. In the dialog box that appears, specify the IP addresses or CIDR blocks used to access the instance and click OK.

- If you specify the CIDR block 10.10.10.0/24, any IP addresses in the 10.10.10.X format are allowed to access the RDS instance.
- If you want to add multiple IP addresses or CIDR blocks, separate each entry with a comma (without spaces), such as 192.168.0.1,172.16.213.9.
- After you click Add Internal IP Addresses of ECS Instances, the IP addresses of all the ECS instances under your Apsara Stack account are displayed. You can select the required IP addresses to add to the whitelist.



Note:

If you add a new IP address or CIDR block to the default whitelist, the default address 127.0.0.1 is automatically deleted.

Create a whitelist



Note:

You can create up to 50 whitelists for an instance.

The procedure is as follows:

1. [Log on to the ApsaraDB for RDS console.](#)
2. Click the ID of an instance to go to the Basic Information page of the instance.
3. In the left-side navigation pane, click Data Security.
4. On the Whitelist Settings tab, click Create Whitelist.
5. In the Create Whitelist dialog box, configure the following parameters.

Parameter	Description
Whitelist Name	<p>The name of the whitelist.</p> <ul style="list-style-type: none"> • The whitelist name must be 2 to 32 characters in length. • It can contain lowercase letters, digits, and underscores (_). • It must start with a lowercase letter and end with a letter or digit.

Parameter	Description
Whitelist	<p>The IP addresses or CIDR blocks used to access the instance.</p> <ul style="list-style-type: none"> • If you specify the CIDR block 10.10.10.0/24, any IP addresses in the 10.10.10.X format are allowed to access the RDS instance. • If you want to add multiple IP addresses or CIDR blocks, separate each entry with a comma (without spaces), such as 192.168.0.1,172.16.213.9. • After you click Add Internal IP Addresses of ECS Instances, the IP addresses of all the ECS instances under your Apsara Stack account are displayed. You can select the required IP addresses to add to the whitelist.

6. Click OK.

3.10.2 SQL audit

You can use the SQL audit feature to audit SQL executions and check the details. SQL audit does not affect instance performance.



Note:

You cannot view the logs that are generated before you enable SQL audit.

You can view the incremental data of your ApsaraDB RDS for MySQL instance by using SQL audit logs or binlogs. However, these two methods differ in the following aspects:

- SQL audit logs are similar to audit logs in MySQL and record all DML and DDL operations by using network protocol analysis. SQL audit does not parse the actual parameter values. Therefore, a small amount of information may be lost if a large number of SQL statements are executed to query data. The incremental data obtained by using this method may be inaccurate.
- Binlogs record all add, delete, and modify operations and the incremental data used for data restoration. Binlogs are temporarily stored in your ApsaraDB for RDS instance after they are generated. The system transfers full binlog files to OSS on a regular basis. OSS then stores the files for seven days. However, a binlog file cannot be transferred if data is being written to it. Therefore, you may find that some binlog files fail to be uploaded to OSS after you click Upload Binlogs.

Binlogs are not generated in real time, but you can obtain accurate incremental data from them.



Note:

You can click Upload Binlogs on the Backup and Restoration page.

Precautions

- **SQL audit is disabled by default. SQL audit does not affect instance performance.**
- **SQL audit records are retained for 30 days.**
- **Files exported from SQL audit are retained for two days. The system clears files that are retained for more than two days.**

Enable SQL audit

1. *Log on to the ApsaraDB for RDS console.*
2. **Click the ID of an instance.**
3. **In the left-side navigation pane, click Data Security.**
4. **Click the SQL Audit tab.**

The screenshot shows the SQL Audit configuration page in the ApsaraDB for RDS console. It includes tabs for 'Whitelist Settings', 'SQL Audit', and 'SSL Encryption'. The 'SQL Audit' tab is active. There is a 'Select Time Range' field with a date range of 'Jan 6, 2020, 09:46 - Jan 6, 2020, 13:46'. Below this are input fields for 'DB:', 'User:', and 'Keyword:', each followed by a 'Search' button. A red box highlights the 'Enable SQL Audit Log' button. At the bottom, a table header is visible with columns: 'Connection IP Address', 'Database Name', 'Executing Account', 'SQL Details', 'Thread ID', 'Time Consumed', 'Number of Returned Records', and 'Execution Time'. A message at the bottom of the page reads: 'You have not yet turned on SQL audit. Enable now.'

5. **Click Enable SQL Audit Log.**
6. **In the message that appears, click Confirm.**

After enabling SQL audit, you can query SQL information based on conditions such as time, database, user, and keyword.

Disable SQL audit



Note:

If SQL audit is disabled, all the SQL audit records are cleared. We recommend that you export and store the audit records locally before you disable SQL audit.

You can disable SQL audit when you do not need it to avoid charges. To disable SQL audit, follow these steps:

1. *Log on to the ApsaraDB for RDS console.*

2. Click the ID of an instance.
3. In the left-side navigation pane, click Data Security.
4. Click the SQL Audit tab.

5. Click Export File to export and store the SQL audit content locally.
6. After the file is exported, click Disable SQL Audit Log.
7. In the message that appears, click Confirm.

3.11 Database backup and restoration

3.11.1 Automatic backup

RDS automatic backup supports full physical backups. ApsaraDB for RDS automatically backs up data based on pre-configured policies. This topic describes how to configure a policy for automatic backup.

1. [Log on to the ApsaraDB for RDS console.](#)
2. Click the ID of an instance.
3. In the left-side navigation pane, click Backup and Restoration.
4. Click the Backup Settings tab.

5. Click Edit.

Backup Settings
✕

Data Retention Period: Days

Backup Cycle: Monday Tuesday Wednesday Thursday
 Friday Saturday Sunday

Backup Time: ▼

Log Backup: Enable Disable

Log Retention Period: Days

6. Configure the following parameters.

Parameter	Description
Data Retention Period (Days)	The number of days for which data backup files are retained. Valid values: 7 to 730. Default value: 7.
Backup Cycle	One or multiple days in a week.
Backup Time	Any period of time within a day. Unit: hours. We recommend that you back up data during off-peak hours.
Log Backup	<p>Specifies whether to enable log backup.</p> <div style="background-color: #f0f0f0; padding: 5px; border: 1px solid #ccc;">  Notice: If you disable log backup, all the log backup files are deleted, and you cannot restore data to a saved point in time. </div>

Parameter	Description
Log Retention Period (Days)	<p>The number of days for which log backup files are retained. Valid values: 7 to 730. Default value: 7.</p> <p> Note: The log backup retention period must be shorter than or equal to the data backup retention period.</p>

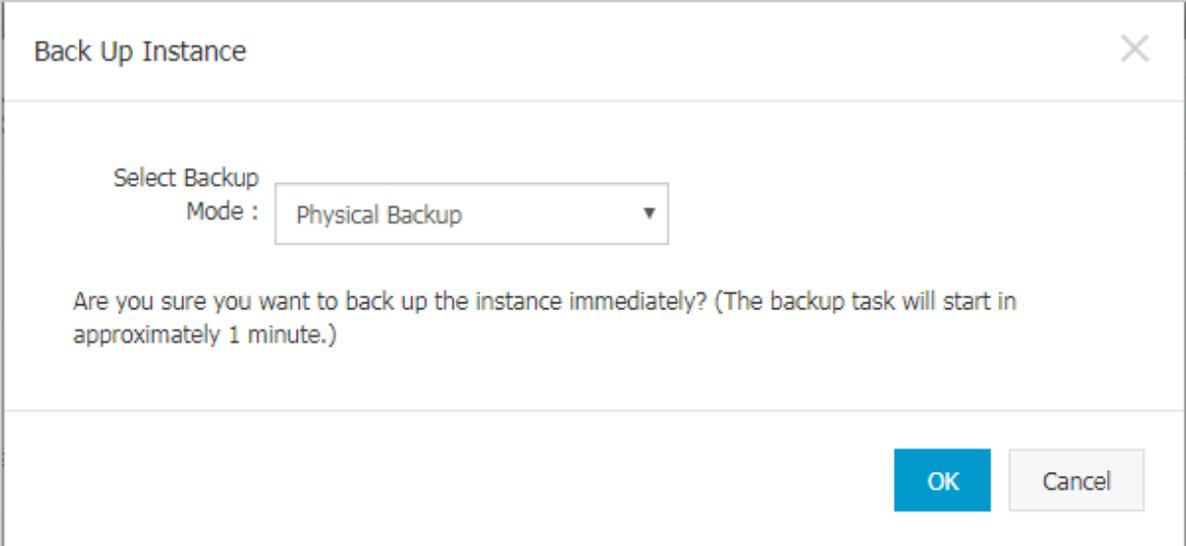
7. After you set the preceding parameters, click OK.

3.11.2 Manual backup

Manual backup supports both full physical backups and full logical backups. This topic describes how to manually back up ApsaraDB for RDS data.

Procedure

1. [Log on to the ApsaraDB for RDS console.](#)
2. Click the ID of an instance.
3. In the upper-right corner, click Back Up Instance.



4. Set the backup mode and backup policy, and click OK.

Parameter	Value	Description
Backup Mode	Physical Backup	This mode dumps the physical files of the RDS database, such as data files, control files, and log files. In case the database fails, these files can be used to restore data.

Parameter	Value	Description
	Logical Backup	This mode stores all schema definition statements and data insertion statements of the RDS database . You can execute these SQL statements to restore data. A database that is exactly the same as the original database is created.

**Note:**

If you choose Logical Backup > Single-Database Backup, select the database to back up on the left, click > to add the database to the list on the right, and then click OK.

Back Up Instance

Select Backup Mode : Logical Backup

Backup Policy : Instance Backup Single-Database Backup

Are you sure you want to back up the instance immediately? (The backup task will start in approximately 1 minute.)

OK Cancel

3.11.3 Restore data to a new instance (formerly known as cloning an instance)

A cloned instance is a new instance with the same content as the primary instance, including data and settings. This feature allows you to restore data of the primary instance or create multiple instances that are the same as the primary instance.

Prerequisites

- The primary instance is in the running state.
- The primary instance does not have an ongoing migration task.
- Data backup and log backup are enabled.
- The primary instance has at least one completed backup set before you clone the instance by backup set.

Features

You can specify a backup set or any point in time within the backup retention period to clone an instance.



Note:

- A cloned instance copies only the data of the primary instance. The copied data includes database information, account information, and instance settings such as whitelist settings, backup settings, parameter settings, and alert threshold settings.
- The database engine of a cloned instance must be the same as that of the primary instance. Other settings, such as the instance edition, zone, network type, instance type, and storage space, can be different. If you want to clone an instance to restore the data of a primary instance, we recommend that you select the instance type and storage space that are higher than those of the primary instance to speed up the data restoration process.
- The account type of a cloned instance must be the same as that of the primary instance. The account password of the cloned instance can be changed.

Procedure

1. [Log on to the ApsaraDB for RDS console.](#)
2. Click the ID of an instance.
3. In the left-side navigation pane, click Backup and Restoration.
4. On the Data Backup tab, find the backup set you want to restore and click Restore in the Actions column.
5. In the dialog box that appears, select Restore Database and click OK.

6. On the Restore RDS Instance page, configure the following parameters.

Category	Parameter	Description
Region	Region	The region where the ApsaraDB for RDS instance resides.
	Zone	The zone where the ApsaraDB for RDS instance resides.
Database Restoration	Restore Mode	Select By Time or By Backup Set to restore the database.
	Time	Select the point in time to which you want to restore the database.  Note: When Restore Mode is set to By Time, you must specify this parameter.
	Backup Set	Select the backup set for restoration.  Note: When Restore Mode is set to By Backup Set, you must specify this parameter.
Specifications	Database Engine	The engine of the database, which is consistent with that of the primary instance and automatically set to MySQL.
	Engine Version	The version of the database engine, which is the same as that of the primary instance and automatically set.
	Instance Type	The type of the cloned instance.  Note: We recommend that you select the instance type and storage space that are higher than those of the primary instance to speed up the data restoration process.

Category	Parameter	Description
	Storage	<p>The storage space of the instance, including the space for data, system files, binlog files, and transaction files. You can adjust the storage space in 5 GB increments.</p> <div style="background-color: #f0f0f0; padding: 5px;">  Note: For dedicated instances with local SSDs, the storage space is associated with the instance type. </div>
Network Type	Network Access	Set the value to Internal Network or Public Network.
	Network Type	The network type of the instance, which is automatically set to Classic Network.

7. After you configure the preceding parameters, click Submit.

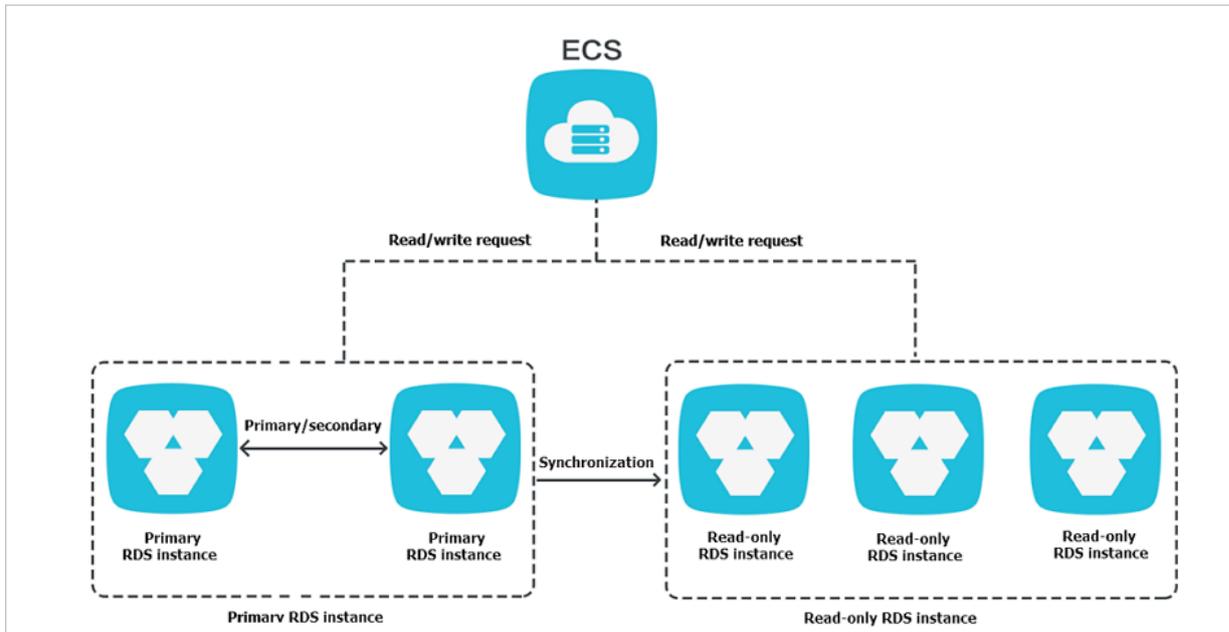
3.12 Read-only instances

3.12.1 Overview

RDS for MySQL 5.6 allows you to create read-only instances. In scenarios where there are a few write requests but a large number of read requests, you can create read-only instances to relieve the database access load on the primary instance. This topic describes the features and limits of read-only instances.

To scale the reading capability and distribute database access loads, you can create one or more read-only instances in a region. Read-only instances allow RDS to increase the application throughput when a large amount of data is being read.

A read-only instance with a single physical node and no backup node uses the native replication capability of MySQL to synchronize changes from the primary instance to all its read-only instances. Read-only instances must be in the same region as the primary instance but do not have to be in the same zone as the primary instance. The following figure shows the topology of read-only instances.



Read-only instances have the following features:

- Specifications of a read-only instance can be different from those of the primary instance and can be changed at any time, which facilitates elastic scaling.
- Read-only instances do not require account or database maintenance. Account and database information is synchronized from the primary instance.
- The whitelists of read-only instances can be configured independently.
- System performance monitoring is provided.

RDS provides up to 20 system performance monitoring views, including those for disk capacity, IOPS, connections, CPU utilization, and network traffic. You can view the load of instances easily.

- RDS provides a variety of optimization recommendations, such as storage engine check, primary key check, large table check, and check for excessive indexes and missing indexes. You can optimize your databases based on the optimization recommendations and specific applications.

3.12.2 Create a read-only instance

You can create read-only instances of different specifications based on your business requirements.

Precautions

- A maximum of five read-only instances can be created for a primary instance.
- Backup settings and temporary backup are not supported.

- Instance restoration is not supported.
- Data migration to read-only instances is not supported.
- Database creation and deletion are not supported.
- Account creation, deletion, authorization, and password changes are not supported.
- After a read-only instance is created, you cannot restore data by directly overwriting the primary instance with a backup set.

Procedure

1. [Log on to the ApsaraDB for RDS console.](#)
2. Click the ID of an instance.
3. On the Basic Information page, click Add Read-only Instance on the right.
4. On the Create Read-only RDS Instance page, configure the read-only instance parameters.

Category	Parameter	Description
Region	Region	The region where the ApsaraDB for RDS instance resides.
	Zone	The zone where the ApsaraDB for RDS instance resides.
Specifications	Database Engine	The database engine of the read-only instance, which is automatically set to MySQL.
	Engine Version	The engine version of the read-only instance, which is the same as that of the primary instance.
	Instance Type	The type of the read-only instance. The type of the read-only instance can be different from that of the primary instance, and can be modified at any time to facilitate flexible upgrade and downgrade.

Category	Parameter	Description
	Storage	The storage space of the read-only instance. To ensure sufficient I/O throughput for data synchronization, we recommend that you select at least the same instance type and storage space as the primary instance for the read-only instance.
Network Type	Network Type	The network type of the read-only instance, which is automatically set to Classic Network.

5. After you configure the preceding parameters, click **Submit**.

3.12.3 View the details of read-only instances

3.12.3.1 View instance details through a read-only instance

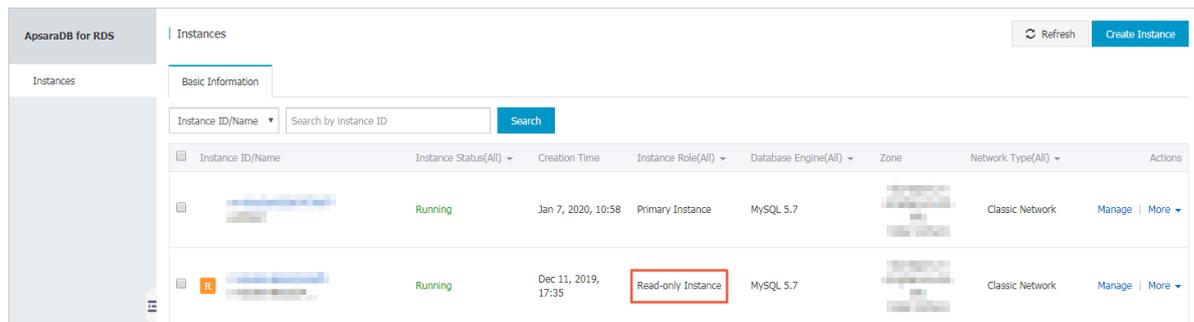
You can go to the read-only instance management page from the **Instances** page or the **Read-Only Instance** page of the primary instance. Read-only instances are managed in the same way as ordinary instances. The read-only instance management page shows the management operations that can be performed. This topic describes how to go to the read-only instance management page from the **Instances** page.

Procedure

1. [Log on to the ApsaraDB for RDS console.](#)

2. On the Instances page, click the ID of a read-only instance. The Basic Information page that appears allows you to manage the read-only instance. In the instance list, Instance Type of read-only instances is displayed as Read-only Instance, as shown in *Figure 3-4: View read-only instances*.

Figure 3-4: View read-only instances

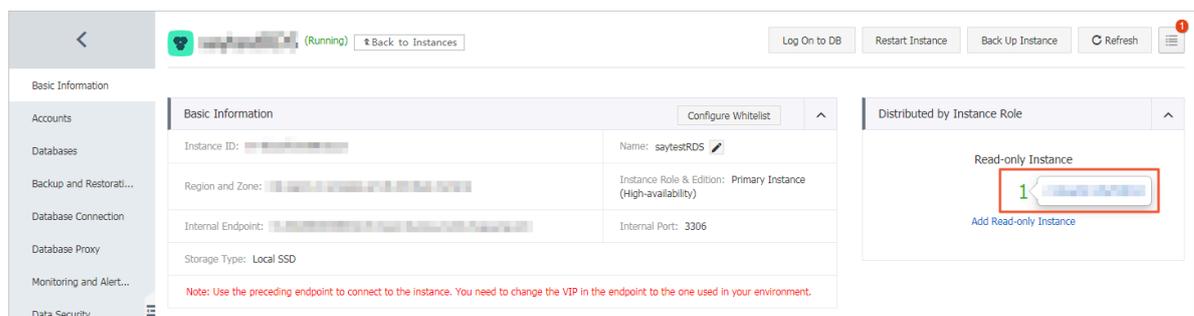


3.12.3.2 View instance details through the primary instance

You can go to the read-only instance management page from the Instances page or the Read-Only Instance page of the primary instance. Read-only instances are managed in the same way as ordinary instances. The read-only instance management page shows the management operations that can be performed. This topic describes how to go to the read-only instance management page from the Read-Only Instance page of the primary instance.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. Click the ID of an instance.
3. On the Basic Information page, move the pointer over the number below Read-only Instance in the Distributed by Instance Role section. The ID of the read-only instance is displayed.



4. Click the ID of the read-only instance to go to the read-only instance management page.

3.13 Logs

1. [Log on to the ApsaraDB for RDS console.](#)
2. Click the ID of an instance.
3. In the left-side navigation pane, click Logs. Click the Error Logs, Slow Log Details, or Primary/Secondary Switching Logs tab, select a time range, and click Search.

Tab	Description
Error Logs	Records database running errors that occurred within the last month.
Slow Log Details	Records SQL statements that took longer than one second to execute from the last month, and deletes redundant SQL statements.  Note: Slow query logs in the ApsaraDB for RDS console are updated once every minute. However, you can query real-time slow query logs from the <code>mysql.slow_log</code> table.
Primary/ Secondary Switching Logs	This feature is suitable for ApsaraDB RDS for MySQL High-availability Edition instances.

3.14 Migrate data from an on-premises database to an ApsaraDB for RDS instance

3.14.1 Compress data

ApsaraDB RDS for MySQL 5.6 allows you to use the TokuDB storage engine to compress data. This topic describes how to compress data.

Context

Data transferred from the InnoDB storage engine to the TokuDB storage engine can be reduced by 80% to 90% in volume. 2 TB of data in InnoDB can be compressed to 400 GB or less in TokuDB. In addition to data compression, TokuDB supports

transactions and online DDL operations. TokuDB is compatible with MyISAM and InnoDB applications.



Note:

- The TokuDB storage engine does not support foreign keys.
- The TokuDB storage engine is not applicable to scenarios that require frequent reading of large amounts of data.

Procedure

1. Run the following command to check the MySQL version:

```
SELECT version();
```



Note:

For more information about how to log on to a database, see [Connect to an ApsaraDB RDS for MySQL instance](#).

2. Run the following command and set `loose_tokudb_buffer_pool_ratio` to indicate the proportion of cache that TokuDB occupies in the shared cache of TokuDB and InnoDB:

```
select sum(data_length) into @all_size from information_schema.
tables where engine='innodb';
select sum(data_length) into @change_size from information_schema
.tables where engine='innodb' and concat(table_schema, '.',
table_name) in ('XX.XXXX', 'XX.XXXX', 'XX.XXXX');
select round(@change_size/@all_size*100);
```



Note:

In the preceding command, `XX.XXXX` indicates the name of the database or table to be transferred to the TokuDB storage engine.

3. Restart the instance. For more information, see [Restart an instance](#)
4. Run the following command to change the storage engine:

```
ALTER TABLE XX.XXXX ENGINE=TokuDB
```



Note:

In the preceding command, `XX.XXXX` indicates the name of the database or table to be transferred to the TokuDB storage engine.

3.14.2 Migrate MySQL data

3.14.2.1 Use mysqldump to migrate MySQL data

This topic describes how to use `mysqldump` to migrate local data to an ApsaraDB RDS for MySQL instance.

Prerequisites

An ECS instance is created.

Context

`mysqldump` is easy to use but requires extensive downtime. This tool is suitable for scenarios where the amount of data is small or extensive downtime is allowed.

ApsaraDB RDS for MySQL is fully compatible with the native database service. The procedure of migrating the original database to an ApsaraDB RDS for MySQL instance is similar to that of migrating data from one MySQL server to another.

Before you migrate data, create a migration account in the on-premises database, and grant read and write permissions on the database to the migration account.

Procedure

1. Run the following command to create a migration account in the on-premises database:

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

Parameter description:

- **username:** specifies the name of the account to be created.
- **host:** specifies the host from which you log on to the database. As a local user, you can use `localhost` to log on to the database. To log on from any host, you can use the wildcard `%`.
- **password:** specifies the logon password for the account.

For example, if you want to create account `William` with password `Changme123` for logging on to the on-premises database from any host, run the following command:

```
CREATE USER 'William'@'%' IDENTIFIED BY 'Changme123';
```

2. Run the following command to grant permissions to the migration account in the on-premises database:

```
GRANT SELECT ON databasename.tablename TO 'username'@'host' WITH
GRANT OPTION; GRANT REPLICATION SLAVE ON databasename.tablename
TO 'username'@'host' WITH GRANT OPTION; GRANT REPLICATION SLAVE ON
databasename.tablename TO 'username'@'host' WITH GRANT OPTION;
```

Parameter description:

- **privileges:** specifies the operation permissions granted to the account, such as SELECT, INSERT, and UPDATE. To grant all permissions to the account, use ALL
- **databasename:** specifies the database name. To grant all database permissions to the account, use wildcard *.
- **tablename:** specifies the table name. To grant all table permissions to the account, use the wildcard *.
- **username:** specifies the name of the account to which you want to grant permissions.
- **host:** specifies the host from which the account is authorized to log on to the database. As a local user, you can use localhost to log on to the database. To log on from any host, you can use the wildcard %.
- **WITH GRANT OPTION:** an optional parameter that enables the account to use the GRANT command.

For example, if you want to grant all of the database and table permissions to the William account and use the account to log on to the on-premises database from any host, run the following command:

```
GRANT ALL ON *. * TO 'William'@'%';
```

3. Use the data export tool of mysqldump to export data from the database as a data file.



Notice:

Do not update data during data export. This step only exports data. It does not export stored procedures, triggers, or functions.

```
mysqldump -h localIp -u userName -p --opt --default-character-set=utf8
--hex-blob dbName --skip-triggers > /tmp/dbName.sql
```

Parameter description:

- **localIp:** specifies the IP address of the on-premises database server.
- **userName:** specifies the migration account of the on-premises database.
- **dbName:** specifies the name of the database you want to migrate.
- **/tmp/dbName.sql:** specifies the name of the backup file.

4. Use mysqldump to export stored procedures, triggers, and functions.



Notice:

Skip this step if no stored procedures, triggers, or functions are used in the database. When exporting stored procedures, triggers, and functions, you must remove the DEFINER to ensure compatibility with ApsaraDB RDS for MySQL.

```
mysqldump -h localIp -u userName -p --opt --default-character-set=utf8
--hex-blob dbName -R | sed -e 's/DEFINER[ ]*=[ ]*[^\]*\*/\*/' > /tmp/
triggerProcedure.sql
```

Parameter description:

- **localIp:** specifies the IP address of the on-premises database server.
- **userName:** specifies the migration account of the on-premises database.
- **dbName:** specifies the name of the database you want to migrate.
- **/tmp/triggerProcedure.sql:** specifies the name of the backup file.

5. Upload the data file and stored procedure file to the ECS instance.

The example in this topic shows how to upload files to the following paths:

```
/tmp/dbName.sql
```

```
/tmp/triggerProcedure.sql
```

6. Log on to the ECS console and import both the data file and stored procedure file to the target ApsaraDB RDS for MySQL instance.

```
mysql -h intranet4example.mysql.rds.aliyuncs.com -u userName -p dbName  
< /tmp/dbName.sql
```

```
mysql -h intranet4example.mysql.rds.aliyuncs.com -u userName -p dbName  
< /tmp/triggerProcedure.sql
```

Parameter description:

- **intranet4example.mysql.rds.aliyuncs.com:** the endpoint of the ApsaraDB RDS for MySQL instance. An internal endpoint is used as an example.
- **userName:** specifies the migration account of the ApsaraDB RDS for MySQL database.
- **dbName:** specifies the name of the database you want to import.
- **/tmp/dbName.sql:** specifies the name of the data file you want to import.
- **/tmp/triggerProcedure.sql:** specifies the name of the stored procedure file you want to import.

4 AnalyticDB for PostgreSQL

4.1 What is AnalyticDB for PostgreSQL?

AnalyticDB for PostgreSQL (formerly known as HybridDB for PostgreSQL) is a distributed cloud database that uses multiple compute groups to provide Massively Parallel Processing (MPP) data warehousing service.

AnalyticDB for PostgreSQL is developed based on the Greenplum Open Source Database project and has been enhanced by Alibaba Cloud. This service has the following features:

- **Compatible with Greenplum, allowing you to use all tools that support Greenplum.**
- **Supports OSS, JSON, and HyperLogLog, a probability cardinality estimation algorithm.**
- **Supports flexible hybrid analysis through the SQL:2003 standard and OLAP aggregate functions.**
- **Provides a hybrid mode that supports both column store and row store, enhancing analytics performance.**
- **Supports data compression technology to reduce storage costs.**
- **Provides online expansion and performance monitoring services to free you from managing and maintaining large numbers of MPP clusters. This enables DBAs, developers, and data analysts to focus on improving enterprise productivity and creating core business by using SQL.**

4.2 Quick start

4.2.1 Overview

This topic provides a quick start guide about how to perform management tasks for AnalyticDB for PostgreSQL instances such as creating an instance and logging on to a database.

- [Log on to the AnalyticDB for PostgreSQL console](#)

This topic describes how to log on to the AnalyticDB for PostgreSQL console.

- [Create an instance](#)

You can create an instance in the console and then manage the instance.

- [Configure a whitelist](#)

To ensure a secure and stable database, you must add IP addresses or CIDR blocks that are allowed to access the database to a whitelist of the instance before you use the AnalyticDB for PostgreSQL instance.

- [Create an initial account](#)

After you create an instance, you must create an initial account to log on to the database.

- [Connect to a database](#)

You can use a client that supports PostgreSQL or Greenplum to connect to the database.

4.2.2 Log on to the AnalyticDB for PostgreSQL console

This topic describes how to log on to the AnalyticDB for PostgreSQL console.

Prerequisites

- **Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel . The URL used to access the ASCM console is in the following format: `http://IP address or domain name of the ASCM console/manage`.**
- **We recommend that you use the Google Chrome browser.**

- 1. In the address bar, enter the URL used to access the ASCM console. Press Enter.**
- 2. Enter your username and password.**

The system has a default super administrator, whose username is super. The super administrator can create system administrators. A system administrator can create system users and notify the users of the default passwords by SMS or email.



Note:

When you log on to the ASCM console for the first time, you must modify the password of your username as instructed. For security concerns, your password must meet the minimum complexity requirements: The password must be 8 to 20 characters in length and must contain at least two types of the following characters: letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs (\$), and percent signs (%).

3. Click Login to go to the ASCM console homepage.
4. In the top navigation bar, choose Products > AnalyticDB for PostgreSQL.

4.2.3 Create an instance

You can create an instance in the console and then manage the instance.

1. [Log on to the AnalyticDB for PostgreSQL console.](#)
2. In the upper-right corner of the page, click Create Instance.
3. On the AnalyticDB for PostgreSQL buy page, configure the following parameters.

Section	Parameter	Description
Region	Region	The region of the instance.  Note: If you need to access the AnalyticDB for PostgreSQL instance from an ECS instance over VPC, you must deploy the instance in the same region and zone as those of the ECS instance.
	Zone	The zone of the instance.
Basic Settings	Organization	The organization to which the instance belongs.
	Resource Set	The resource set to which the instance belongs.
	Engine	Currently, only the integrated computing and storage version is supported.
	Node Type	The unit of computing resources. Different group types have different storage capacities and computing capabilities.
	Nodes	The number of compute nodes. An instance must contain at least two compute nodes. The performance of an instance scales linearly with the number of compute nodes.

Section	Parameter	Description
Network	Network Type	<p>Valid values:</p> <ul style="list-style-type: none"> • Classic Network: Cloud services on a classic network are not isolated from each other. Unauthorized access to a cloud service is blocked only by the security group or whitelist policy of the service. • VPC: A VPC helps you to build an isolated network environment in Alibaba Cloud. You can customize the route table, IP address range, and gateway in a VPC. We recommend that you select VPC for enhanced security. <p>You can create a VPC in advance, or change the network type to VPC after instance creation.</p>
	VPC	<p>The VPC where the AnalyticDB for PostgreSQL instance is located.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p> Note: Virtual Private Cloud (VPC): You can use a VPC to build an isolated network environment in Alibaba Cloud. You can customize the route table, IP address range, and gateway in a VPC.</p> </div>
	VSwitch	The VSwitch where the AnalyticDB for PostgreSQL instance is located.
	IP Whitelist	The IP addresses that are allowed to access the instance.

4. After you have configured the preceding parameters, click **Submit**.

4.2.4 Configure a whitelist

To ensure a secure and stable database, you must add IP addresses or CIDR blocks that are allowed to access the database to a whitelist.

1. [Log on to the AnalyticDB for PostgreSQL console](#).
2. Find the target instance and click its ID. The **Basic Information** page appears.
3. In the left-side navigation pane, click **Security Controls**. The **Security Controls** page appears.

4. On the Whitelist Settings tab, click **Modify** corresponding to the *default* whitelist. The **Modify Group** page appears.

**Note:**

You can also click **Clear** corresponding to the *default* whitelist to delete the IP addresses of the default whitelist, and then click **Add Group** to create a new whitelist.

5. Delete 127.0.0.1 in the *default* whitelist and enter your IP addresses in the whitelist. The following table lists the parameters.

Parameter	Description
Group Name	Specify the name of the whitelist. The whitelist name must be 2 to 32 characters in length and can contain lowercase letters, digits, and underscores (_). It must start with a lowercase letter and end with a letter or digit. The default whitelist cannot be modified or deleted.
Whitelist	<p>Enter the CIDR blocks or IP addresses that are allowed to access the database. Use commas (,) to separate multiple CIDR blocks or IP addresses.</p> <ul style="list-style-type: none"> • A whitelist can contain IP addresses such as 10.10.10.1 and CIDR blocks such as 10.10.10.0/24. This CIDR block indicates that any IP addresses in the 10.10.10.X format have access to the database. • The percent sign (%) or 0.0.0.0/0 indicates that any IP addresses are allowed to access the database. <div style="background-color: #f0f0f0; padding: 5px; margin: 10px 0;">  Notice: This configuration is not recommended because it reduces the security of the database. </div> <ul style="list-style-type: none"> • Default whitelists of new instances contain the loopback address 127.0.0.1. This configuration allows no access from external IP addresses.

6. Click **OK** to create a whitelist.

What's next

- We recommend that you regularly maintain the whitelist to ensure secure access for AnalyticDB for PostgreSQL.
- You can click **Modify** or **Delete** to modify or delete custom whitelists.

4.2.5 Create an initial account

After you create an instance, you must create an initial account to log on to the database.

1. [Log on to the AnalyticDB for PostgreSQL console.](#)
2. Find the target instance and click its ID. The Basic Information page appears.
3. In the left-side navigation pane, click Account Management. The Account Management page appears.
4. In the upper-right corner of the page, click Create Account. The Create Account page appears.
5. Enter the database account and password, and click OK.

Parameter	Description
Account	The name of the account must be 2 to 16 characters in length and can contain lowercase letters, digits, and underscores (_). It must start with a letter and end with a letter or digit.
New Password	The password must be 8 to 32 characters in length . It must contain at least three of the following character types: uppercase letters, lowercase letters , digits, and special characters.
Password	Enter the password again.

4.2.6 Obtain the client tool

The interface protocol of AnalyticDB for PostgreSQL is compatible with Greenplum Community Edition and PostgreSQL 8.2. Because of this, you can use the Greenplum or PostgreSQL client to connect to AnalyticDB for PostgreSQL.



Note:

Apsara Stack is an isolated environment. You must deploy software installation packages to the internal environment.

Graphical client tools

AnalyticDB for PostgreSQL users can directly use client tools that support Greenplum, such as [SQL Workbench](#), [Navicat Premium](#), [Navicat for PostgreSQL](#), and [pgAdmin III \(1.6.3\)](#).

Command-line client psql (for RHEL 6, RHEL 7, CentOS 6, and CentOS 7)

For Red Hat Enterprise Linux (RHEL) and CentOS 6 or 7, you can download the tools from the following addresses and decompress the packages to use them:

- For RHEL 6 or CentOS 6, click [hybriddb_client_package_el6](#).
- For RHEL version 7 or CentOS version 7, click [hybriddb_client_package_el7](#).

Command-line client psql (for other Linux systems)

The compilation methods for client tools applicable to other Linux systems are as follows:

1. Obtain the source code by using one of the following methods:

- **Obtain the git directory. You must first install the git tool.**

```
git clone https://github.com/greenplum-db/gpdb.git
cd gpdb
git checkout 5d870156
```

- **Download the code.**

```
wget https://github.com/greenplum-db/gpdb/archive/5d87015609abd330c68a5402c1267fc86cbc9e1f.zip
unzip 5d87015609abd330c68a5402c1267fc86cbc9e1f.zip
cd gpdb-5d87015609abd330c68a5402c1267fc86cbc9e1f
```

2. Use gcc and other compilers.

```
./configure
make -j32
make install
```

3. Use psql and pg_dump. The paths of the two tools are as follows:

psql: `/usr/local/pgsql/bin/psql`

pg_dump: `/usr/local/pgsql/bin/pg_dump`

Command-line client psql (for Windows and other systems)

For client tools for Windows and other systems, go to the Pivotal website to download [HybridDB Client](#)

4.2.7 Connect to a database

The Greenplum Database and AnalyticDB for PostgreSQL are both developed based on PostgreSQL 8.2 and fully compatible with its messaging protocol. AnalyticDB for

PostgreSQL users can use tools that support the PostgreSQL 8.2 message protocol, such as libpq, JDBC, ODBC, psycopg2, and pgAdmin III.

Context

AnalyticDB for PostgreSQL provides `psql`, a binary program of Red Hat. For more information about the download link, see [Obtain the client tool](#). The Greenplum official website provides an easy-to-install installation package that includes JDBC, ODBC, and libpq. For more information, see [Greenplum official documentation](#).



Note:

- Apsara Stack is an isolated environment. To access Apsara Stack, you must prepare the necessary software installation packages in advance.
- AnalyticDB for PostgreSQL instances can only be accessed by clients deployed on ECS instances within the same region and zone.

psql

`psql` is a common tool used together with Greenplum, and provides a variety of command functions. Its binary files are located in the `bin` directory of Greenplum. The procedure is as follows:

1. Connect to AnalyticDB for PostgreSQL by using one of the following methods:

- **Connection string**

```
psql "host=yourgpdbaddress.gpdb.rds.aliyuncs.com port=3432 dbname=postgres user=gpdbaccount password=gpdbpassword"
```

- **Specified parameters**

```
psql -h yourgpdbaddress.gp.aliyun-inc.com -p 3432 -d postgres -U gpdbaccount
```

Parameters:

- **-h:** specifies the host address.
- **-p:** specifies the port number.
- **-d:** specifies the database. The default database is postgres.
- **-U:** specifies the user to connect to the database.

In `psql`, you can run the `psql --help` command to view more options. You can run the `\?` command to view the commands supported in `psql`.

2. Enter the password to go to the psql shell interface.

```
postgres=>
```

References

- For more information about the Greenplum psql usage, see [psql](#).
- AnalyticDB for PostgreSQL also supports psql statements of PostgreSQL. Pay attention to the differences between Greenplum psql and PostgreSQL psql. For more information, see [PostgreSQL 8.3.23 Documentation - psql](#).

pgAdmin III

pgAdmin III is a PostgreSQL graphical client and can be directly used to connect to AnalyticDB for PostgreSQL. For more information, click [here](#). For more information about other graphical clients, see [Obtain the client tool](#).

1. Download pgAdmin III 1.6.3 or earlier versions.

You can download pgAdmin III 1.6.3 from the [PostgreSQL website](#). PgAdmin III 1.6.3 supports various operating systems, such as Windows, macOS, and Linux.



Note:

AnalyticDB for PostgreSQL is compatible with PostgreSQL 8.2. Therefore, you must use pgAdmin III 1.6.3 or earlier to connect to AnalyticDB for PostgreSQL. pgAdmin 4 and later versions are not supported.

2. Choose File > Add Server.
3. In the New Server Registration dialog box that appears, enter the configuration information.
4. Click OK to connect to AnalyticDB for PostgreSQL.

JDBC

JDBC uses the interface provided by PostgreSQL. The download methods are as follows:

- Click [PostgreSQL JDBC Driver](#) to download the official JDBC of PostgreSQL, and then add it to the environment variables.
- Use the tools provided by the Greenplum website. For more information, see [Greenplum Database 4.3 Connectivity Tools for UNIX](#).

The sample code is as follows:

```
import java.sql.Connection; import java.sql.DriverManager; import java.
sql.ResultSet; import java.sql.SQLException; import java.sql.Statement
; public class gp_conn { public static void main(String[] args) { try
{ Class.forName("org.postgresql.Driver"); Connection db = DriverMana
ger.getConnection("jdbc:postgresql://mygpdbpub.gpdb.rds.aliyuncs.com:
3432/postgres","mygpdb","mygpdb"); Statement st = db.createStatement();
ResultSet rs = st.executeQuery("select * from gp_segment_configuration
"); while (rs.next()) { System.out.print(rs.getString(1)); System.out
.print(" | "); System.out.print(rs.getString(2)); System.out.print(" |
"); System.out.print(rs.getString(3)); System.out.print(" | "); System
.out.print(rs.getString(4)); System.out.print(" | "); System.out.print(
rs.getString(5)); System.out.print(" | "); System.out.print(rs.getString
(6)); System.out.print(" | "); System.out.print(rs.getString(7)); System
.out.print(" | "); System.out.print(rs.getString(8)); System.out.print
(" | "); System.out.print(rs.getString(9)); System.out.print(" | ");
System.out.print(rs.getString(10)); System.out.print(" | "); System.out.
println(rs.getString(11)); } rs.close(); st.close(); } catch (ClassNotFo
undException e) { e.printStackTrace(); } catch (SQLException e) { e.
printStackTrace(); } } }
```

Python

Python uses psycopg2 to connect to Greenplum and PostgreSQL. Procedure:

1. Install psycopg2. There are three installation methods in CentOS:

- **Method 1: Run the yum -y install python-psycopg2 command.**
- **Method 2: Run the pip install psycopg2 command.**
- **Method 3: Run the source code:**

```
yum install -y postgresql-devel*
wget http://initd.org/psycopg/tarballs/PSYCOPG-2-6/psycopg2-2.6.
tar.gz
tar xf psycopg2-2.6.tar.gz
cd psycopg2-2.6
python setup.py build
sudo python setup.py install
```

2. Run the following commands to set PYTHONPATH and reference it:

```
import psycopg2
sql = 'select * from gp_segment_configuration;'
conn = psycopg2.connect(database='gpdb', user='mygpdb', password='
mygpdb', host='mygpdbpub.gpdb.rds.aliyuncs.com', port=3432)
conn.autocommit = True
cursor = conn.cursor()
cursor.execute(sql)
rows = cursor.fetchall()
for row in rows:
    print row
conn.commit()
conn.close()
```

A similar output is displayed:

```
(1, -1, 'p', 'p', 's', 'u', 3022, '192.168.2.158', '192.168.2.158',
None, None)(6, -1, 'm', 'm', 's', 'u', 3019, '192.168.2.47', '192.168
.2.47', None, None)(2, 0, 'p', 'p', 's', 'u', 3025, '192.168.2.148',
```

```
'192.168.2.148', 3525, None)(4, 0, 'm', 'm', 's', 'u', 3024, '192.168.2.158', '192.168.2.158', 3524, None)(3, 1, 'p', 'p', 's', 'u', 3023, '192.168.2.158', '192.168.2.158', 3523, None)(5, 1, 'm', 'm', 's', 'u', 3026, '192.168.2.148', '192.168.2.148', 3526, None)
```

libpq

libpq is the C language interface to AnalyticDB for PostgreSQL. You can use the libpq library to access and manage PostgreSQL databases in a C program. You can locate its static and dynamic libraries under the lib directory.

For the example programs, visit [Example Programs](#).

For more information about libpq, see [PostgreSQL 9.4.17 Documentation - Chapter 31. libpq - C Library](#).

ODBC

PostgreSQL ODBC is an open-source version based on the GNU Lesser General Public License (LGPL) protocol. You can download it from the [PostgreSQL website](#).

1. Install the driver.

```
yum install -y unixODBC.x86_64
yum install -y postgresql-odbc.x86_64
```

2. View the driver configuration.

```
cat /etc/odbcinst.ini
# Example driver definitions
# Driver from the postgresql-odbc package
# Setup from the unixODBC package
[PostgreSQL]
Description = ODBC for PostgreSQL
Driver = /usr/lib/psqlodbcw.so
Setup = /usr/lib/libodbcpsqlS.so
Driver64 = /usr/lib64/psqlodbcw.so
Setup64 = /usr/lib64/libodbcpsqlS.so
FileUsage = 1
# Driver from the mysql-connector-odbc package
# Setup from the unixODBC package
[MySQL]
Description = ODBC for MySQL
Driver = /usr/lib/libmyodbc5.so
Setup = /usr/lib/libodbcmyS.so
Driver64 = /usr/lib64/libmyodbc5.so
Setup64 = /usr/lib64/libodbcmyS.so
FileUsage = 1
```

3. Configure the DSN. Replace the **** in the following code with the corresponding connection information.

```
[mygpdb]
Description = Test to gp
Driver = PostgreSQL
Database = ****
```

```
Servername = ****.gpdb.rds.aliyuncs.com
Username = ****
Password = ****
Port = ****
ReadOnly = 0
```

4. Test connectivity.

```
echo "select count(*) from pg_class" | isql mygpdb
+-----+
| Connected! |
|           |
| sql-statement |
| help [tablename] |
| quit         |
+-----+
SQL> select count(*) from pg_class
+-----+
| count |
+-----+
| 388   |
+-----+
SQLRowCount returns 1
1 rows fetched
```

5. After ODBC is connected to the instance, connect the application to ODBC. For more information, see [PostgreSQL ODBC Driver](#) and [psqlODBC HOWTO - C#](#).

References

- [Pivotal Greenplum documentation](#)
- [PostgreSQL psqlODBC](#)
- [Compiling psqlODBC on Unix](#)
- [Download ODBC connectors](#)
- [Download JDBC connectors](#)
- [The PostgreSQL JDBC Interface](#)

4.3 Instances

4.3.1 Reset the password

If you forget the password of your database account, you can reset the password in the AnalyticDB for PostgreSQL console.



Note:

We recommend that you change your password periodically to ensure data security.

1. [Log on to the AnalyticDB for PostgreSQL console.](#)
2. Find the target instance and click its ID. The Basic Information page appears.
3. In the left-side navigation pane, click Account Management. The Account Management page appears.
4. Click Reset Password in the corresponding Actions column of the account. The Reset Account Password page appears.
5. After you enter and confirm the new password, click OK.

**Note:**

The password must be 8 to 32 characters in length. It must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. We recommend that you do not use a previously used password.

4.3.2 View monitoring information

You can go to the monitoring information page in the console to view the operation status of an instance.

1. [Log on to the AnalyticDB for PostgreSQL console.](#)
2. Find the target instance and click its ID. The Basic Information page appears.
3. In the left-side navigation pane, click Monitoring and Alarms. The Monitoring and Alarms page appears.

Specify a duration of time *n* up to seven days in length to view the metrics for that last *n* period.

4.3.3 Switch the network type of an instance

The default network type of an instance is Virtual Private Cloud (VPC). After an instance has been created, you can switch its network type between classic network and VPC as needed.

Context

AnalyticDB for PostgreSQL supports two network types: classic network and VPC. Both network types use BGP connections, and are independent of the public network of your service provider. These network types only differ in function, and you can choose a network type based on your requirements. The two network types are applicable to different scenarios:

- **Classic network:** IP addresses are allocated by Alibaba Cloud. Classic networks are easy to configure and use. This network type is suitable for users who do not need to perform complex operations, or who only require short deployment cycles.
- **VPC:** a logically isolated private network. You can customize the network topology and IP addresses and connect through a leased line. This network type is suitable for advanced users.

**Warning:**

Switching the network type will cause the database service to stop. Proceed with caution.

1. [Log on to the AnalyticDB for PostgreSQL console.](#)
2. Find the target instance and click its ID. The Basic Information page appears.
3. In the left-side navigation pane, click Database Connection. The Database Connection page appears.
4. In the upper-right corner of the page, click Switch to Classic Network or Switch to VPC.
5. If you click Switch to VPC, you must select the destination VPC and VSwitch. Click OK.

**Note:**

To switch the network type to VPC, a VPC and VSwitch must exist or be created in the zone where the instance is located.

6. If you click Switch to Classic Network, click OK in the displayed message.

**Note:**

After you switch the network type, it takes 3 to 30 minutes for the instance to enter the running state.

4.3.4 Restart an instance

To better meet your needs, AnalyticDB for PostgreSQL automatically updates the database kernel version. When you create an instance, the latest database kernel is used by default. After a new version is released, you can restart your instance to

update the database kernel and use its extended features. This topic describes how to restart an instance.

**Warning:**

Restarting an instance will cause the database service to stop. Proceed with caution.

1. [Log on to the AnalyticDB for PostgreSQL console](#).
2. Find the target instance and click its ID. The Basic Information page appears.
3. In the upper-right corner of the page, click Restart Instance.

**Note:**

The restart process typically takes from 3 to 30 minutes. During the restart period, the instance cannot provide external services. We recommend that you take precautionary measures before restarting instances. After the instance has been restarted and enters the running state, you can access the database.

4.3.5 Import data

4.3.5.1 Import or export data from or to OSS in parallel

AnalyticDB for PostgreSQL can import or export data from or to OSS tables in parallel by using the OSS external table feature, `gpossext`. AnalyticDB for PostgreSQL also supports GZIP compression for OSS external tables to reduce file size and storage costs. `gpossext` can read from and write to TEXT and CSV files, even when they are compressed in GZIP packages.

- Create an OSS external table extension (`oss_ext`)

To use an OSS external table, you must first create an OSS external table extension in AnalyticDB for PostgreSQL. You must create an extension for each database that you need to access.

- **Creation statement:** `CREATE EXTENSION IF NOT EXISTS oss_ext;`
- **Deletion statement:** `DROP EXTENSION IF EXISTS oss_ext;`

- **Import data in parallel**

1. **Distribute data evenly among multiple OSS files for storage. We recommend that you set the number of OSS files to an integer that is the multiple of the number of compute nodes in AnalyticDB for PostgreSQL.**
2. **Create a READABLE external table in AnalyticDB for PostgreSQL.**
3. **Execute the following statement to import data in parallel:**

```
INSERT INTO <destination table> SELECT * FROM <external table>
```



Note:

- **The data import performance depends on the OSS performance and resources of the AnalyticDB for PostgreSQL instance, such as CPU, I/O, memory, and network resources. To ensure the best import performance, we recommend that you use column store and compression when you create a table. For example, you can specify the following clause: WITH (APPENDONLY=true, ORIENTATION=column, COMPRESSTYPE=zlib, COMPRESSLEVEL=5, BLOCKSIZE=1048576). For more information, see [Greenplum Database official documentation on database table creation syntax](#).**
- **We recommend that you configure OSS and AnalyticDB for PostgreSQL instances within the same region to implement the best import performance. For more information, see [Endpoints](#).**

- **Export data in parallel**

1. **Create a WRITABLE external table in AnalyticDB for PostgreSQL.**
2. **Execute the following statement to export data to OSS in parallel:**

```
INSERT INTO <external table> SELECT * FROM <source table>
```

- **Create OSS external tables**



Note:

The syntax to create and use external tables is the same as that of Greenplum Database, except for the syntax of location-related parameters.

```
CREATE [READABLE] EXTERNAL TABLE tablename
( columnname datatype [, ...] | LIKE othertable )
LOCATION ('ossprotocol')
FORMAT 'TEXT'
    [( [HEADER]
      [DELIMITER [AS] 'delimiter' | 'OFF']
```

```

        [NULL [AS] 'null string']
        [ESCAPE [AS] 'escape' | 'OFF']
        [NEWLINE [ AS ] 'LF' | 'CR' | 'CRLF']
        [FILL MISSING FIELDS] )]]
    | 'CSV'
    [( [HEADER]
        [QUOTE [AS] 'quote']
        [DELIMITER [AS] 'delimiter']
        [NULL [AS] 'null string']
        [FORCE NOT NULL column [, ...]]
        [ESCAPE [AS] 'escape']
        [NEWLINE [ AS ] 'LF' | 'CR' | 'CRLF']
        [FILL MISSING FIELDS] )]]
[ ENCODING 'encoding' ]
[ [LOG ERRORS [INTO error_table]] SEGMENT REJECT LIMIT count
  [ROWS | PERCENT] ]
CREATE WRITABLE EXTERNAL TABLE table_name
( column_name data_type [, ...] | LIKE other_table )
LOCATION ('ossprotocol')
FORMAT 'TEXT'
        [( [DELIMITER [AS] 'delimiter']
          [NULL [AS] 'null string']
          [ESCAPE [AS] 'escape' | 'OFF'] )]]
    | 'CSV'
    [( [QUOTE [AS] 'quote']
      [DELIMITER [AS] 'delimiter']
      [NULL [AS] 'null string']
      [FORCE QUOTE column [, ...]] ]
      [ESCAPE [AS] 'escape'] )]]
[ ENCODING 'encoding' ]
[ DISTRIBUTED BY (column, [ ... ] ) | DISTRIBUTED RANDOMLY ]
ossprotocol:
    oss://oss_endpoint prefix=prefix_name
        id=userossid key=userosskey bucket=ossbucket compressiontype=[
none|gzip] async=[true|false]
ossprotocol:
    oss://oss_endpoint dir=[folder/[folder/]...]/file_name
        id=userossid key=userosskey bucket=ossbucket compressiontype=[
none|gzip] async=[true|false]
ossprotocol:
    oss://oss_endpoint filepath=[folder/[folder/]...]/file_name

```

```
id=userossid key=userosskey bucket=ossbucket compressiontype=[
none|gzip] async=[true|false]
```

Parameters

Table 4-1: Common parameters

Parameter	Description
Protocol and endpoint	<p>It is in the protocol name://oss_endpoint format. The protocol name is oss. oss_endpoint is the domain name used by users to access OSS in a region.</p> <div style="background-color: #f0f0f0; padding: 5px;">  Note: You can access the database from a VPC host by using an internal endpoint containing "internal" in the name in order not to generate public traffic. </div>
id	The AccessKey ID of the OSS account.
key	The AccessKey secret of the OSS account.
bucket	The bucket where the data file is located. You must use OSS to create the bucket before data import.

Parameter	Description
prefix	<p>The prefix of the path name corresponding to the data file. Prefixes are directly matched and cannot be controlled by regular expressions. The prefix, filepath, and dir parameters are mutually exclusive and only one parameter can be specified at a time.</p> <ul style="list-style-type: none"> · If you create a READABLE external table for data import, all OSS files that contain the specified prefix will be imported. <ul style="list-style-type: none"> - If you set prefix to test/filename, the following files will be imported: <ul style="list-style-type: none"> ■ test/filename ■ test/filenameexxx ■ test/filename/aa ■ test/filenameeyyy/aa ■ test/filenameeyyy/bb/aa - If you set prefix to test/filename/, only the following file out of the preceding files will be imported: <p style="margin-left: 20px;">test/filename/aa</p> · If you create a WRITABLE external table for data export, each exported file will have a unique name based on this parameter. <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p> Note: One or more files can be exported for each compute node. The names of exported files are in the <code>prefix_tablename_uuid.x</code> format. <code>uuid</code> indicates a timestamp in microseconds as an int64 value. <code>x</code> indicates the node ID. You can use an external table for multiple export operations. Each export operation is assigned a <code>uuid</code> value. The files exported during each operation share a <code>uuid</code> value.</p> </div>

Parameter	Description
dir	<p>The virtual folder path in OSS. The prefix, filepath, and dir parameters are mutually exclusive and only one parameter can be specified at a time.</p> <ul style="list-style-type: none">• A folder path must end with a forward slash (/) such as <code>test/mydir/</code>.• If you use this parameter when creating an external table for data import, all files under the specified virtual directory (except for its subdirectories and contained files) will be imported. Unlike filepath, dir does not require you to specify the names of files in the directory.• If this parameter is used in creating an external table for data export, all data will be exported to multiple files within the specified directory. The names of exported files are in the <code>filename.x</code> format, where x is a digit. The values of x may not be consecutive.

Parameter	Description
filepath	<p>The file name that contains a path in OSS. The prefix , filepath, and dir parameters are mutually exclusive and only one parameter can be specified at a time. You can only specify the filepath parameter when you create a READABLE external table for data import.</p> <ul style="list-style-type: none"> • The file name includes the file path, but not the bucket name. • The file name specified for data import must be in the filename or filename.x format. The values of x must be consecutive digits starting from 1. <p>For example, if filepath is set to filename and OSS contains the following files, the imported files include filename, filename.1, and filename.2, but filename.4 is not imported because filename.3 does not exist.</p> <pre>filename filename.1 filename.2 filename.4</pre>

Table 4-2: Import mode parameters

Parameter	Description
async	<p>Specifies whether to load data asynchronously.</p> <ul style="list-style-type: none"> • Asynchronous data import is enabled by default. You can set async to false or f to disable asynchronous data import. • Enables the worker thread to load data from OSS to accelerate the import performance. The default import mode is asynchronous mode. • Asynchronous data import consumes more hardware resources than normal data import.

Parameter	Description
compressiontype	The compression format of the imported file. Valid values: <ul style="list-style-type: none"> · none: specifies to import files without compressing them. This is the default value. · gzip: specifies compress imported files in the GZIP format. Only the GZIP format is supported.
compressionlevel	The compression level of the files written to OSS. Valid values: 1 to 9. Default value: 6.

Table 4-3: Export mode parameters

Parameter	Description
oss_flush_block_size	The size of each data block written to OSS. Valid values: 1 MB to 128 MB. Default value: 32 MB.
oss_file_max_size	The maximum size of each file written to OSS. If the limit is exceeded, subsequent data is written to another file. Valid values: 8 MB to 4000 MB. Default value: 1024 MB.
num_parallel_worker	The number of parallel compression threads for data written to OSS. Valid values: 1 to 8. Default value: 3.

Additionally, you must pay attention to the following items for the export mode:

- **WRITABLE** is the keyword of the external table for data export. You must specify this keyword when creating an external table.
- Only the **prefix** and **dir** parameters are supported for data export. The **filepath** parameter is not supported.
- You can use the **DISTRIBUTED BY** clause to write data from compute nodes to OSS based on the specified distribution keys.

Other common parameters

The following error-tolerance parameters can be used for data import and export:

Table 4-4: Error-tolerance parameters

Parameter	Description
<code>oss_connect_timeout</code>	The connection timeout period. Unit: seconds. Default value: 10.
<code>oss_dns_cache_timeout</code>	The DNS timeout period. Unit: seconds. Default value : 60.
<code>oss_speed_limit</code>	The minimum rate tolerated. Default value: 1024 bit/s (1 Kbit/s).
<code>oss_speed_time</code>	The maximum amount of time tolerated. Unit: seconds. Default value: 15.

If the default values are used for the preceding parameters, a timeout will occur when the transmission rate is lower than 1 Kbit/s for 15 consecutive seconds. For more information, *see* [Troubleshooting in OSS SDK reference](#).

The other parameters are compatible with the original external table syntax of Greenplum Database. For more information about the syntax, *see* [Greenplum Database official documentation on external table syntax](#). These parameters include:

- **FORMAT:** indicates the supported file format, such as TEXT and CSV.
- **ENCODING:** indicates the data encoding format of a file, such as UTF-8.
- **LOG ERRORS:** indicates that the clause can ignore imported erroneous data and write the data to `error_table`. You can also use the count parameter to specify the error reporting threshold.

Examples

```
#Create a READABLE external table of OSS. create readable external table
ossexample (date text, time text, open float, high float, low float,
volume int) location('oss://oss-cn-hangzhou.aliyuncs.com prefix=osstest
/example id=XXX key=XXX bucket=testbucket compressiontype=gzip') FORMAT
'csv' (QUOTE ''' DELIMITER E'\t') ENCODING 'utf8' LOG ERRORS INTO
my_error_rows SEGMENT REJECT LIMIT 5;create readable external table
ossexample (date text, time text, open float, high float, low float,
volume int) location('oss://oss-cn-hangzhou.aliyuncs.com dir=osstest
/ id=XXX key=XXX bucket=testbucket') FORMAT 'csv' LOG ERRORS SEGMENT
REJECT LIMIT 5;create readable external table ossexample (date text,
time text, open float, high float, low float, volume int) location('oss
://oss-cn-hangzhou.aliyuncs.com filepath=osstest/example.csv id=XXX key
=XXX bucket=testbucket') FORMAT 'csv' LOG ERRORS SEGMENT REJECT LIMIT
5;#Create a WRITABLE external table of OSS. create WRITABLE external
table ossexample_exp (date text, time text, open float, high float,
low float, volume int) location('oss://oss-cn-hangzhou.aliyuncs.com
prefix=osstest/exp/outfromhdb id=XXX key=XXX bucket=testbucket') FORMAT
'csv' DISTRIBUTED BY (date);create WRITABLE external table ossexample
_exp (date text, time text, open float, high float, low float, volume
```

```

int) location('oss://oss-cn-hangzhou.aliyuncs.com dir=osstest/exp/
id=XXX key=XXX bucket=testbucket') FORMAT 'csv' DISTRIBUTED BY (date
);#Create a heap table to load data. create table example (date text,
time text, open float, high float, low float, volume int) DISTRIBUTED
BY (date);#Load data from ossexample to example in parallel. insert
into example select * from ossexample;#Export data from example to OSS
. insert into ossexample_exp select * from example;#Each compute node
is involved. #Each compute node pulls data from OSS in parallel. The
redistribution motion node calculates the hash value of the data, and
then distributes the hash value to the corresponding compute node. That
compute node then imports the data to the database through the insert
node. explain insert into example select * from ossexample; QUERY PLAN
-----
Insert (slice0; segments: 4) (rows=250000 width=92) -> Redistribute
Motion 4:4 (slice1; segments: 4) (cost=0.00..11000.00 rows=250000
width=92) Hash Key: ossexample.date -> External Scan on ossexample (
cost=0.00..11000.00 rows=250000 width=92)(4 rows)#The compute node
exports the local data to OSS. Data redistribution is not performed.
explain insert into ossexample_exp select * from example; QUERY PLAN
-----
Insert
(slice0; segments: 3) (rows=1 width=92) -> Seq Scan on example (cost=0.
00..0.00 rows=1 width=92)(2 rows)

```

TEXT and CSV format description

The following parameters specify the formats of files read from and written to OSS. You can specify the parameters in the external DDL parameters.

- **\n: a line delimiter or line break for TEXT and CSV files.**
- **DELIMITER: specifies the delimiter of columns.**
 - **If the DELIMITER parameter is specified, the QUOTE parameter must also be specified.**
 - **Recommended column delimiters include commas (,), vertical bars (|), \t, and other special characters.**
- **QUOTE: encloses user data that contains special characters by column.**
 - **Strings that contain special characters will be enclosed by QUOTE to differentiate user data and the control characters.**
 - **To optimize the efficiency, it is unnecessary to enclose data such as integers in QUOTE characters.**
 - **QUOTE cannot be the same string as specified in DELIMITER. The default value of QUOTE is double quotation marks (").**
 - **User data that contains QUOTE characters must also contain ESCAPE characters to differentiate the user data from code for the machine.**

- **ESCAPE:** specifies the escape character.
 - Place an escape character before a special character that needs to be escaped to indicate that it is not a special character.
 - If ESCAPE is not specified, the default value is the same as QUOTE.
 - You can also use other characters as ESCAPE characters such as backslashes (\), which is used by MySQL.

Default control characters for TEXT and CSV files

Table 4-5: Default control characters for TEXT and CSV files

Control character	TEXT	CSV
DELIMITER	\t (tab)	, (comma)
QUOTE	" (double quotation mark)	" (double quotation mark)
ESCAPE	N/A	Same as QUOTE
NULL	\N (backslash-N)	Empty string without quotation marks



Note:

All control characters must be single-byte characters.

SDK troubleshooting

The following [Table 4-6: Error log information](#) table lists the error logs generated when an error occurs during the import or export process.

Table 4-6: Error log information

Keyword	Description
code	The HTTP status code of the error request.
error_code	The error code returned by OSS.
error_msg	The error message returned by OSS.
req_id	The UUID that identifies the request. If you require assistance in solving a problem, you can submit a ticket containing the req_id of the failed request to OSS developers.

References

- [Greenplum Database official documentation on database external table syntax](#)
- [Greenplum Database official documentation on database table creation syntax](#)

4.3.5.2 Import data from MySQL

You can use the `mysql2pgsql` tool to migrate tables from MySQL to AnalyticDB for PostgreSQL, Greenplum Database, PostgreSQL, or PPAS.

Background information

`mysql2pgsql` connects a source MySQL database to a destination AnalyticDB for PostgreSQL database, queries data to be exported from the MySQL database, and then imports the data to the destination database by using the `\COPY` statement. The tool supports multi-thread import. Each worker thread imports a part of database tables.

To download the binary installation package of `mysql2pgsql`, click [here](#).

To view instructions on source code compilation of `mysql2pgsql`, click [here](#).

Procedure

1. **Modify the `my.cfg` configuration file to configure the connection information of source and destination databases.**
 - a. **Modify the connection information of the source MySQL database.**

**Note:**

You must have the read permissions on all user tables.

```
[src.mysql]
host = "192.168.1.1"
port = "3306"
user = "test"
password = "test"
db = "test"
encodingdir = "share"
encoding = "utf8"
```

- b. **Modify the connection information of the destination PostgreSQL, PPAS, or AnalyticDB for PostgreSQL database.**

**Note:**

You must have write permissions on the destination table.

```
[desc.pgsql]
```

```
connect_string = "host=192.168.1.2 dbname=test port=3432 user=
test password=pgsql"
```

2. Import data by using mysql2pgsql.

```
./mysql2pgsql -l <tables_list_file> -d -n -j <number of threads> -s
<schema of target able>
```

Table 4-7: Parameters

Parameter	Description
-l	<p>Optional. Used to specify a text file that contains tables to be synchronized. If you do not specify this parameter, all the tables in the database that is specified in the configuration file will be synchronized. <tables_list_file> is the name of a file that contains a collection of tables to be synchronized and conditions for table queries. The content format is as follows:</p> <pre>table1 : select * from table_big where column1 < '2016 -08-05' table2 : table3 table4: select column1, column2 from tableX where column1 != 10 table5: select * from table_big where column1 >= '2016 -08-05'</pre>
-d	Optional. Indicates the table creation DDL statement that creates the destination table but does not synchronize data.
-n	Optional. Must be used along with -d to specify that the table partition definition is not included in the DDL statement.
-j	Optional. Used to specify the number of threads used for data synchronization. If you do not specify this parameter, five concurrent threads will be used by default.
-s	Optional. Used to specify the schema of the destination table. Only one schema at a time can be specified by the command. If you do not specify the parameter, the data is imported into the table under the public schema.

Typical usage

Full database migration

1. Obtain the DDL statements of the corresponding destination table by running the following command:

```
./mysql2pgsql -d
```

2. Create a table in the destination database based on these DDL statements with the distribution key information added.
3. Run the following command to synchronize all tables:

```
./mysql2pgsql
```

This command will migrate the data from all MySQL tables in the database that is specified in the configuration file to the destination database. By default, five concurrent threads are used to read and import data from involved tables.

Partial table migration

1. Create a new file `tab_list.txt` and enter the following content:

```
t1  
t2 : select * from t2 where c1 > 138888
```

2. Run the following command to synchronize the specified `t1` and `t2` tables (note that for the `t2` table, only data that meets the `c1 > 138888` condition is migrated):

```
./mysql2pgsql -l tab_list.txt
```

4.3.5.3 Import data from PostgreSQL

You can use the `pgsql2pgsql` tool to migrate tables across AnalyticDB for PostgreSQL, Greenplum Database, PostgreSQL, and PPAS.

Context

`pgsql2pgsql` supports the following features:

- Full migration across PostgreSQL, PPAS, Greenplum Database, and AnalyticDB for PostgreSQL.
- Full migration and incremental migration from PostgreSQL or PPAS (version 9.4 or later) to AnalyticDB for PostgreSQL or ApsaraDB RDS for PPAS.

You can download the software packages from the [dbsync project](#) library.

- To download the binary installation package of `pgsql2pgsql`, click [here](#).
- To view instructions on source code compilation of `pgsql2pgsql`, click [here](#).

Procedure

1. Modify the my.cfg configuration file to configure the connection information of source and destination databases.

a) Modify the connection information of the source PostgreSQL database.



Note:

In the connection information of the source PostgreSQL database, we recommend that you set the user to the owner of the source database.

```
[src.pgsql]
connect_string = "host=192.168.1.1 dbname=test port=3432 user=
test password=pgsql"
```

b) Modify the connection information of the local temporary PostgreSQL database.

```
[local.pgsql]
connect_string = "host=192.168.1.2 dbname=test port=3432 user=
test2 password=pgsql"
```

c) Modify the connection information of the destination PostgreSQL database.



Note:

You must have the write permissions on the destination table.

```
[desc.pgsql]
connect_string = "host=192.168.1.2 dbname=test port=3432 user=
test3 password=pgsql"
```



Note:

- If you need to synchronize incremental data, you must have the permissions to create replication slots in the source database.
- PostgreSQL versions 9.4 and later support logic flow replication, meaning that source databases of the versions support incremental migration. The kernel supports logic flow replication only if you configure the following kernel parameters:

```
wal_level = logical
```

```
max_wal_senders = 6
```

```
max_replication_slots = 6
```

2. Use `pgsql2pgsql` to perform full database migration.

```
./pgsql2pgsql
```

By default, the migration program migrates the table data of all users from the source PostgreSQL database to the destination PostgreSQL database.

3. View the status information.

You can view the status information in a single migration process by connecting to the local temporary database. The information is stored in the `db_sync_status` table, including the start and end time of the full migration, the start time of the incremental migration, and the status of incremental synchronization.

4.3.5.4 Import data by using the `\COPY` statement

You can use the `\COPY` statement to import the data of local text files into AnalyticDB for PostgreSQL databases. The local text files must be formatted, such as files that use commas (,), semicolons (;), or special characters as delimiters.

Context

- Parallel writing of large amounts of data is not available because the `\COPY` statement writes data in serial using the coordinator node. If you need to import a large amount of data in parallel, you can use the OSS-based data import method.
- The `\COPY` statement is a `psql` instruction. If you use the database statement `COPY` instead of the `\COPY` statement, you must note that only `stdin` is supported. This `COPY` statement does not support file because the root user does not have the superuser permissions to perform operations on files.
- AnalyticDB for PostgreSQL also allows you to use JDBC to execute the `COPY` statement. The `CopyIn` method is encapsulated within JDBC. For more information, see [Interface CopyIn](#).
- For more information about how to use the `COPY` statement, see [COPY](#).

Procedure

Import data by using the following sample code:

```
\COPY table [(column [, ...])] FROM {'file' | STDIN}
  [ [WITH]
    [OIDS]
    [HEADER]
    [DELIMITER [ AS ] 'delimiter']
    [NULL [ AS ] 'null string']
    [ESCAPE [ AS ] 'escape' | 'OFF']
```

```

[NEWLINE [ AS ] 'LF' | 'CR' | 'CRLF']
[CSV [QUOTE [ AS ] 'quote']
  [FORCE NOT NULL column [, ...]]
[FILL MISSING FIELDS]
[[LOG ERRORS [INTO error_table] [KEEP]
SEGMENT REJECT LIMIT count [ROWS | PERCENT] ]
\COPY {table [(column [, ...])] | (query)} TO {'file' | STDOUT}
[ [WITH]
  [OIDS]
  [HEADER]
  [DELIMITER [ AS ] 'delimiter']
  [NULL [ AS ] 'null string']
  [ESCAPE [ AS ] 'escape' | 'OFF']
  [CSV [QUOTE [ AS ] 'quote']
    [FORCE QUOTE column [, ...]] ]
[IGNORE EXTERNAL PARTITIONS ]

```

4.4 Databases

4.4.1 Overview

The operations based on the Greenplum Database in AnalyticDB for PostgreSQL are the same as those in the Greenplum Database, including schema, supported data types, and user permissions. Except for certain operations exclusive to the Greenplum Database (such as the partition keys and AO tables), you can refer to PostgreSQL for other operations.

References

- [Pivotal Greenplum Official Documentation](#)
- [Greenplum 4.3 Best Practices](#)
- [Golden Rules of Greenplum Data Distribution](#)

4.4.2 Create a database

After you log on to the AnalyticDB for PostgreSQL instance, you can execute SQL statements to create databases.

Similar to PostgreSQL, in AnalyticDB for PostgreSQL you can execute SQL statements to create databases. For example, after psql is connected to Greenplum, execute the following statements:

```

=> create database mygpdb;
CREATE DATABASE
=> \c mygpdb
psql (9.4.4, server 8.3devel)

```

You are now connected to database "mygpdb" as user "mygpdb".

4.4.3 Create a partition key

AnalyticDB for PostgreSQL is a distributed database and data is distributed across all the data nodes. You must create partition keys to distribute the data. The partition keys are vital to query performance. Partition keys are used to ensure even data distribution. Proper selection of keys can significantly improve query performance.

Specify a partition key

In AnalyticDB for PostgreSQL, tables can be distributed across all compute nodes in either hash or random mode. You must specify the partition key when creating a table. Imported data will be distributed to the specific compute node based on the hash value calculated by the partition key.

```
=> create table vtbl(id serial, key integer, value text, shape cuboid,
location geometry, comment text) distributed by (key);
CREATE TABLE
```

If you do not specify the partition key (that means a statement without the `distributed by (key)` field), AnalyticDB for PostgreSQL will randomly allocate the ID field by using the round-robin algorithm.

Rules for selecting the partition key

- Select evenly distributed columns or multiple columns to prevent data skew.
- Select fields commonly used for connection operations, especially for highly concurrent statements.
- Select the condition columns that feature high concurrency queries and high filterability.
- Do not use random distribution.

4.4.4 Construct data

In some test scenarios, you must construct data to fill the database.

1. Create a function that generates random strings.

```
CREATE OR REPLACE FUNCTION random_string(integer) RETURNS text AS $
body$
SELECT array_to_string(array
                        (SELECT substring('0123456789ABCDEFGHIJ
KLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
FROM (ceil(random()*62))::
int
```

```

                                FOR 1)
                                FROM generate_series(1, $1)), '');
$body$
LANGUAGE SQL VOLATILE;

```

2. Create a partition key.

```

CREATE TABLE tbl(id serial, KEY integer, locate geometry, COMMENT
text) distributed by (key);

```

3. Construct data.

```

INSERT INTO tbl(KEY, COMMENT, locate)
SELECT
    KEY,
    COMMENT,
    ST_GeomFromText(locate) AS locate
FROM
    (SELECT
        (a + 1) AS KEY,
        random_string(ceil(random() * 24)::integer) AS COMMENT,
        'POINT(' || ceil(random() * 36 + 99) || ' ' || ceil(random
() * 24 + 50) || ') ' AS locate
    FROM
        generate_series(0, 99999) AS a
    AS t;

```

4.4.5 Query data

This topic describes the query statements and how to view the query plans.

Query statement sample

```

=> select * from tbl where key = 751;
 | id | key | value | shape          | locate
 |----+----+-----+-----+-----+
 | 751 | 751 | red   | 010100000000000000000000C05B400000000000004A40 |
 B9hPhjeNWPqV |
 (1 row)
Time: 513.101 ms

```

View a query plan

```

=> explain select * from tbl where key = 751;
Gather Motion 1:1 (slice1; segments: 1) (cost=0.00..1519.28 rows=1
width=53)
-> Seq Scan on tbl (cost=0.00..1519.28 rows=1 width=53)
    Filter: key = 751
Settings: effective_cache_size=8GB; gp_statistics_use_fkeys=on

```

```
Optimizer status: legacy query optimizer
```

4.4.6 Manage extensions

You can use extensions to expand database features. AnalyticDB for PostgreSQL enables you to manage extensions.

Extension types

AnalyticDB for PostgreSQL supports the following extensions:

- **PostGIS:** supports geographic information data.
- **MADlib:** supports the machine learning function library.
- **fuzzystrmatch:** supports the fuzzy matching of strings.
- **orafunc:** compatible with some Oracle functions.
- **oss_ext:** supports reading data from OSS.
- **hll:** collects statistics by using the HyperLogLog algorithm.
- **pljava:** supports compiling user-defined functions (UDF) in the PL/Java language.
- **pgcrypto:** supports cryptographic hash functions.
- **intarray:** supports integer array-related functions, operators, and indexes.

Create an extension

Execute the following statements to create an extension:

```
CREATE EXTENSION <extension name>;  
CREATE SCHEMA <schema name>;  
CREATE EXTENSION IF NOT EXISTS <extension name> WITH SCHEMA <schema  
name>;
```



Note:

Before you create the MADlib extension, you must create the plpythonu extension first.

```
CREATE EXTENSION plpythonu;  
CREATE EXTENSION madlib;
```

Delete an extension

Execute the following statements to delete an extension:

```
DROP EXTENSION <extension name>;
```

```
DROP EXTENSION IF EXISTS <extension name> CASCADE;
```

**Note:**

If there are objects dependent on the extension, you must add the **CASCADE** keyword to delete all dependent objects.

4.4.7 Manage users and permissions

This topic describes how to manage users and permissions in AnalyticDB for PostgreSQL.

Manage users

The system prompts you to specify an initial username and password when you create an instance. This initial user is the root user. After the instance is created, you can use the root user account to connect to the database. The system also creates superusers such as `aurora` and `replicator` for internal management.

You can run the `\du+` command to view the information of all the users after you connect to the database by using the client tool of PostgreSQL or Greenplum.

Example:

```
postgres=> \du+
          List of roles
Role name | Attributes | Member of |
Description
-----+-----+-----
root_user |           | rds_superuser
...
```

AnalyticDB for PostgreSQL does not provide superuser permissions, but offers a similar role, `RDS_SUPERUSER`, which is consistent with the permission system of ApsaraDB RDS for PostgreSQL. The root user (such as `root_user` in the preceding example) has the permissions of the `RDS_SUPERUSER` role. You can only identify this permission attribute by viewing the user description.

The root user has the following permissions:

- Can create databases and users and perform actions such as `LOGIN`, excluding the `SUPERUSER` permissions.
- Can view and modify the data tables of other users and perform actions such as `SELECT`, `UPDATE`, `DELETE`, and changing owners.

- Can view the connection information of other users, cancel their SQL statements , and kill their connections.
- Can create and delete extensions.
- Can create other users with RDS_SUPERUSER permissions. Example:

```
CREATE ROLE root_user2 RDS_SUPERUSER LOGIN PASSWORD 'xyz' ;
```

Manage permissions

You can manage permissions at the database, schema, and table levels. For example , if you want to grant read permissions on a table to a user and revoke their write permissions, you can execute the following statements:

```
GRANT SELECT ON TABLE t1 TO normal_user1;
REVOKE UPDATE ON TABLE t1 FROM normal_user1;
REVOKE DELETE ON TABLE t1 FROM normal_user1;
```

4.4.8 Manage JSON data

JavaScript Object Notation (JSON) has become a basic data type in the Internet and IoT fields. For more information about JSON, visit [JSON official website](#). PostgreSQL support for JSON has been well developed. Optimized by Alibaba Cloud, AnalyticDB for PostgreSQL supports the JSON type based on the PostgreSQL syntax.

Check whether the current version supports JSON

Execute the following statement to check whether the current version supports JSON:

```
=> SELECT ' '::json;
```

If the following output is displayed, it indicates the JSON type is supported and the instance is ready for use. If the operation fails, restart the instance.

```
json
-----
 ""
(1 row)
```

If the following output is displayed, it indicates the JSON type is not supported.

```
ERROR: type "json" does not exist
LINE 1: SELECT ' '::json;
              ^
```

The preceding command converts data from the string type to the JSON type. PostgreSQL supports operations on JSON data based on this conversion.

JSON conversion in the database

Database operations include reading and writing. The written data is typically converted from the string type to the JSON type. The contents of a string must meet the JSON standard, such as strings, digits, arrays, and objects. Example:

String

```
=> SELECT '"hijson"'::json;
   json
-----
'hijson'
(1 row)
```

`::` is used for explicit type conversion in PostgreSQL, Greenplum, and AnalyticDB for PostgreSQL. The database calls the input function in JSON type during the conversion. Therefore, the JSON format check is performed as follows:

```
=> SELECT '{hijson:1024}'::json;
ERROR:  invalid input syntax for type json
LINE 1: SELECT '{hijson:1024}'::json;
                ^
DETAIL:  Token "hijson" is invalid.
CONTEXT:  JSON data, line 1: {hijson...
```

In the preceding example, `hijson` must be enclosed in double quotation marks (") because JSON requires the KEY value to be a string. A syntax error is returned when `{hijson:1024}` is entered.

Apart from explicit type conversion, database records can also be converted to JSON.

Typically, JSON is not used for a string or a digit, but an object that contains one or more key-value pairs. AnalyticDB for PostgreSQL can support most JSON scenarios after data is converted from the string type to objects. Example:

```
=> select row_to_json(row('{a":"a"}', 'b'));
   row_to_json
-----
{"f1": "{\\a\\":\\a\\}", "f2": "b"}
(1 row)
=> select row_to_json(row('{a":"a"}::json, 'b'));
   row_to_json
-----
{"f1": {"a": "a"}, "f2": "b"}
(1 row)
```

You can see the differences between the string and JSON here. The whole record is conveniently converted into the JSON type.

JSON data types

- **Object**

The object is the most frequently used data type in JSON. Example:

```
=> select '{"key":"value"}'::json;
      json
-----
{"key":"value"}
(1 row)
```

- **Integer and floating point number**

JSON only supports three data types for numeric values: integer, floating point number, and constant expression. AnalyticDB for PostgreSQL supports all three types.

```
=> SELECT '1024'::json;
      json
-----
1024
(1 row)
=> SELECT '0.1'::json;
      json
-----
0.1
(1 row)
```

The following information is required in some special situations:

```
=> SELECT '1e100'::json;
      json
-----
1e100
(1 row)
=> SELECT '{"f":1e100}'::json;
      json
-----
{"f":1e100}
(1 row)
```

Extra-long numbers are also supported. Example:

```
=> SELECT '9223372036854775808'::json;
      json
-----
9223372036854775808
(1 row)
```

- **Array**

```
=> SELECT '[[1,2], [3,4,5]]'::json;
      json
-----
[[1,2], [3,4,5]]
```

```
(1 row)
```

Operators

Operators supported by JSON

```
=> select oprname,oprname,opcode from pg_operator where oprleft = 3114;
oprname |          opcode          |
-----+-----
->      | json_object_field       |
->>    | json_object_field_text  |
->      | json_array_element      |
->>    | json_array_element_text |
#>     | json_extract_path_op    |
#>>   | json_extract_path_text_op
(6 rows)
```

Basic usage

```
=> SELECT '{"f":"1e100"}'::json -> 'f';
? column?
-----
"1e100"
(1 row)
=> SELECT '{"f":"1e100"}'::json ->> 'f';
? column?
-----
1e100
(1 row)
=> select '{"f2":{"f3":1},"f4":{"f5":99,"f6":"stringy"}}'::json#>array
['f4','f6'];
? column?
-----
"stringy"
(1 row)
=> select '{"f2":{"f3":1},"f4":{"f5":99,"f6":"stringy"}}'::json#>'f4,
f6';
? column?
-----
"stringy"
(1 row)
=> select '{"f2":["f3",1],"f4":{"f5":99,"f6":"stringy"}}'::json#>>'f2,
,0';
? column?
-----
f3
(1 row)
```

JSON functions

Supported JSON functions

```
postgres=# \df *json*
List of
functions
 Schema | Name | Result data type |
 Argument data types | Type
-----+-----+-----+-----
+-----+-----+-----+-----
```

pg_catalog	array_to_json	json		anyarray
pg_catalog	array_to_json	json	normal	anyarray
, boolean			normal	
pg_catalog	json_array_element	json		from_json
json, element_index integer			normal	
pg_catalog	json_array_element_text	text		from_json
json, element_index integer			normal	
pg_catalog	json_array_elements	SETOF json		from_json
json, OUT value json			normal	
pg_catalog	json_array_length	integer		json
			normal	
pg_catalog	json_each	SETOF record		from_json
json, OUT key text, OUT value json			normal	
pg_catalog	json_each_text	SETOF record		from_json
json, OUT key text, OUT value text			normal	
pg_catalog	json_extract_path	json		from_json
json, VARIADIC path_elems text[]			normal	
pg_catalog	json_extract_path_op	json		from_json
json, path_elems text[]			normal	
pg_catalog	json_extract_path_text	text		from_json
json, VARIADIC path_elems text[]			normal	
pg_catalog	json_extract_path_text_op	text		from_json
json, path_elems text[]			normal	
pg_catalog	json_in	json		cstring
			normal	
pg_catalog	json_object_field	json		from_json
json, field_name text			normal	
pg_catalog	json_object_field_text	text		from_json
json, field_name text			normal	
pg_catalog	json_object_keys	SETOF text		json
			normal	
pg_catalog	json_out	cstring		json
			normal	
pg_catalog	json_populate_record	anyelement		base
anyelement, from_json json, use_json_as_text boolean			normal	
pg_catalog	json_populate_recordset	SETOF anyelement		base
anyelement, from_json json, use_json_as_text boolean			normal	
pg_catalog	json_recv	json		internal
			normal	
pg_catalog	json_send	bytea		json
			normal	
pg_catalog	row_to_json	json		record
			normal	
pg_catalog	row_to_json	json		record,
boolean			normal	
pg_catalog	to_json	json		
anyelement				normal
(24 rows)				

Basic usage

```

=> SELECT array_to_json('{{1,5},{99,100}}'::int[]);
   array_to_json
-----
 [[1,5],[99,100]]
 (1 row)
=> SELECT row_to_json(row(1,'foo'));
   row_to_json
-----
 {"f1":1,"f2":"foo"}
 (1 row)
=> SELECT json_array_length('[1,2,3,{"f1":1,"f2":[5,6]},4]');

```

```

json_array_length
-----
                    5
(1 row)
=> select * from json_each('{"f1":[1,2,3],"f2":{"f3":1},"f4":null,"f5":99,"f6:"stringy"}') q;
 key | value
-----+-----
  f1 | [1,2,3]
  f2 | {"f3":1}
  f4 | null
  f5 | 99
  f6 | "stringy"
(5 rows)
=> select json_each_text('{"f1":[1,2,3],"f2":{"f3":1},"f4":null,"f5":null}');
 json_each_text
-----
(f1,"[1,2,3]")
(f2,"{"f3":1}")
(f4,)
(f5,null)
(4 rows)
=> select json_array_elements('[1,true,[1,[2,3]],null,{"f1":1,"f2":[7,8,9]},false]');
 json_array_elements
-----
1
true
[1,[2,3]]
null
{"f1":1,"f2":[7,8,9]}
false
(6 rows)
create type jpop as (a text, b int, c timestamp);
=> select * from json_populate_record(null::jpop,'{"a":"blurfl","x":43.2}.2}', false) q;
  a      | b | c
-----+---+---
 blurfl |   | 
(1 row)
=> select * from json_populate_recordset(null::jpop,['{"a":"blurfl","x":43.2},{ "b":3,"c":"2012-01-20 10:42:53"}'],false) q;
  a      | b | c
-----+---+---
 blurfl | 3 | Fri Jan 20 10:42:53 2012
(2 rows)

```

Code examples

Create a table

```

create table tj(id serial, ary int[], obj json, num integer);
=> insert into tj(ary, obj, num) values('{1,5}'::int[], '{"obj":1}', 5);
INSERT 0 1
=> select row_to_json(q) from (select id, ary, obj, num from tj) as q;
 row_to_json
-----
{"f1":1,"f2":[1,5],"f3":{"obj":1},"f4":5}
(1 row)

```

```
=> insert into tj(ary, obj, num) values('{2,5}'::int[], '{"obj":2}', 5);
INSERT 0 1
=> select row_to_json(q) from (select id, ary, obj, num from tj) as q;
      row_to_json
-----
{"f1":1,"f2":[1,5],"f3":{"obj":1},"f4":5}
{"f1":2,"f2":[2,5],"f3":{"obj":2},"f4":5}
(2 rows)
```

Join multiple tables

```
create table tj2(id serial, ary int[], obj json, num integer);
=> insert into tj2(ary, obj, num) values('{2,5}'::int[], '{"obj":2}', 5);
INSERT 0 1
=> select * from tj, tj2 where tj.obj->>'obj' = tj2.obj->>'obj';
 id | ary | obj | num | id | ary | obj | num
-----+-----+-----+-----+-----+-----+-----+-----
  2 | {2,5} | {"obj":2} | 5 | 1 | {2,5} | {"obj":2} | 5
(1 row)
=> select * from tj, tj2 where json_object_field_text(tj.obj, 'obj')
= json_object_field_text(tj2.obj, 'obj');
 id | ary | obj | num | id | ary | obj | num
-----+-----+-----+-----+-----+-----+-----+-----
  2 | {2,5} | {"obj":2} | 5 | 1 | {2,5} | {"obj":2} | 5
(1 row)
```

Use the JSON function index

```
CREATE TEMP TABLE test_json (
    json_type text,
    obj json
);
=> insert into test_json values('aa', '{"f2":{"f3":1},"f4":{"f5":99,"f6":"foo"}}');
INSERT 0 1
=> insert into test_json values('cc', '{"f7":{"f3":1},"f8":{"f5":99,"f6":"foo"}}');
INSERT 0 1
=> select obj->'f2' from test_json where json_type = 'aa';
? column?
-----
{"f3":1}
(1 row)
=> create index i on test_json (json_extract_path_text(obj, '{f4}'));
CREATE INDEX
=> select * from test_json where json_extract_path_text(obj, '{f4}') =
'{"f5":99,"f6":"foo"}';
 json_type | obj
-----+-----
 aa | {"f2":{"f3":1},"f4":{"f5":99,"f6":"foo"}}
(1 row)
```



Note:

JSON data cannot be used as the partition key and does not support JSON aggregate functions.

Example of using Python to access the database:

```

#!/bin/env python
import time
import json
import psycopg2
def gpquery(sql):
    conn = None
    try:
        conn = psycopg2.connect("dbname=sanity1x2")
        conn.autocommit = True
        cur = conn.cursor()
        cur.execute(sql)
        return cur.fetchall()
    except Exception as e:
        if conn:
            try:
                conn.close()
            except:
                pass
            time.sleep(10)
        print e
    return None
def main(_):
    sql = "select obj from tj;"
    #rows = Connection(host, port, user, pwd, dbname).query(sql)
    rows = gpquery(sql)
    for row in rows:
        print json.loads(row[0])
if __name__ == '__main__':
    main()

```

4.4.9 Use HyperLogLog

AnalyticDB for PostgreSQL is highly optimized by Alibaba Cloud, and not only has the features of Greenplum Database, but also supports HyperLogLog. It is suitable for industries such as Internet advertising and estimation analysis that require quick estimation of business metrics such as PV and UV.

Create a HyperLogLog extension

You can execute the following statement to create a HyperLogLog extension:

```
CREATE EXTENSION hll;
```

Basic types

- **Execute the following statement to create a table containing the hll field:**

```
create table agg (id int primary key,userid hll);
```

- **Execute the following statement to convert int to hll_hashval:**

```
select 1::hll_hashval;
```

Basic operators

- **The hll type supports =, !=, <>, ||, and #.**

```
select hll_add_agg(1::hll_hashval) = hll_add_agg(2::hll_hashval);
select hll_add_agg(1::hll_hashval) || hll_add_agg(2::hll_hashval);
select #hll_add_agg(1::hll_hashval);
```

- **The hll_hashval type supports =, !=, and <>.**

```
select 1::hll_hashval = 2::hll_hashval;
select 1::hll_hashval <> 2::hll_hashval;
```

Basic functions

- **Hash functions such as Hll_hash_boolean, hll_hash_smallint, and hll_hash_bigint.**

```
select hll_hash_boolean(true);
select hll_hash_integer(1);
```

- **hll_add_agg: converts the int format to the hll format.**

```
select hll_add_agg(1::hll_hashval);
```

- **hll_union: aggregates the hll fields.**

```
select hll_union(hll_add_agg(1::hll_hashval),hll_add_agg(2::hll_hashval));
```

- **hll_set_defaults: sets the precision.**

```
select hll_set_defaults(15,5,-1,1);
```

- **hll_print: displays debug information.**

```
select hll_print(hll_add_agg(1::hll_hashval));
```

Examples

```
create table access_date (acc_date date unique, userids hll);
insert into access_date select current_date, hll_add_agg(hll_hash_integer(user_id)) from generate_series(1,10000) t(user_id);
insert into access_date select current_date-1, hll_add_agg(hll_hash_integer(user_id)) from generate_series(5000,20000) t(user_id);
insert into access_date select current_date-2, hll_add_agg(hll_hash_integer(user_id)) from generate_series(9000,40000) t(user_id);
postgres=# select #userids from access_date where acc_date=current_date;
? column?
-----
9725.85273370708
(1 row)
postgres=# select #userids from access_date where acc_date=current_date-1;
? column?
```

```

-----
 14968.6596883279
(1 row)
postgres=# select #userids from access_date where acc_date=current_da
te-2;
          ? column?
-----
 29361.5209149911
(1 row)

```

4.4.10 Use the CREATE LIBRARY statement

AnalyticDB for PostgreSQL introduces the **CREATE LIBRARY** and **DROP LIBRARY** statements to allow you to import custom software packages.

Syntax

```

CREATE LIBRARY library_name LANGUAGE [JAVA] FROM oss_location OWNER
ownername
CREATE LIBRARY library_name LANGUAGE [JAVA] VALUES file_content_hex
OWNER ownername
DROP LIBRARY library_name

```

Table 4-8: Parameters

Parameter	Description
library_name	The name of the library to be installed. If the library to be installed has the same name as an existing library, you must delete the existing library before installing the new one.
LANGUAGE [JAVA]	The programming language to be used. Only PL/Java is supported.
oss_location	The location of the package. You can specify the OSS bucket and object names. Only one object can be specified and the specified object cannot be a compressed file. The format is as follows: <pre>oss://oss_endpoint filepath=[folder/[folder /]...]/file_name id=userossid key=userosskey bucket=ossbucket</pre>
file_content_hex	The content of the file. The byte stream is in hexadecimal notation. For example, 73656c6563742031 indicates the hexadecimal byte stream of "select 1". You can use this syntax to import packages without using OSS.
ownername	Specifies the user.
DROP LIBRARY	Deletes a library.

Examples

- **Example 1: Install a JAR package named analytics.jar.**

```
create library example language java from 'oss://oss-cn-hangzhou.
aliyuncs.com filepath=analytics.jar id=xxx key=yyy bucket=zzz';
```

- **Example 2: Import the file content with the byte stream in hexadecimal notation.**

```
create library pglib LANGUAGE java VALUES '73656c6563742031' OWNER "
myuser";
```

- **Example 3: Delete a library.**

```
drop library example;
```

- **Example 4: View installed libraries.**

```
select name, lanname from pg_library;
```

4.4.11 Create and use the PL/Java UDF

AnalyticDB for PostgreSQL allows you to compile and upload JAR software packages written in PL/Java language, and use these JAR packages to create user-defined functions (UDFs). The PL/Java language supported by AnalyticDB for PostgreSQL is Community Edition PL/Java 1.5.0 and the JVM version is 1.8. This topic describes how to create a PL/Java UDF. For more PL/Java examples, see [PL/Java code](#). For the compiling method, see [PL/Java documentation](#).

Procedure

1. In AnalyticDB for PostgreSQL, execute the following statement to create a PL/Java extension. You only need to execute the statement once for each database.

```
create extension pljava;
```

2. Compile the UDF based on your business needs. For example, you can use the following code to compile the Test.java file:

```
public class Test
{
    public static String substring(String text, int beginIndex,
        int endIndex)
    {
        try {
            Process process = null;
            process = Runtime.getRuntime().exec("echo
Test running");
        } catch (Exception e) {
            return "" + e;
        }
        return text.substring(beginIndex, endIndex);
    }
}
```

```
}
```

3. Compile the manifest.txt file.

```
Manifest-Version: 1.0
Main-Class: Test
Specification-Title: "Test"
Specification-Version: "1.0"
Created-By: 1.7.0_99
Build-Date: 01/20/2016 21:00 AM
```

4. Run the following commands to compile and package the program.

```
javac Test.java
jar cfm analytics.jar manifest.txt Test.class
```

5. Upload the analytics.jar file generated in step 4 to OSS by using the following OSS console command.

```
osscmd put analytics.jar oss://zzz
```

6. In AnalyticDB for PostgreSQL, execute the CREATE LIBRARY statement to import the file to AnalyticDB for PostgreSQL.

```
create library example language java from 'oss://oss-cn-hangzhou.
aliyuncs.com filepath=analytics.jar id=xxx key=yyy bucket=zzz';
```



Note:

You can only use the filepath variable in the CREATE LIBRARY statement to import files one at a time. Additionally, the CREATE LIBRARY statement also supports byte streams to import files without using OSS. For more information, see [Use the CREATE LIBRARY statement](#).

7. In AnalyticDB for PostgreSQL, execute the following statements to create and use the UDF.

```
create table temp (a varchar) distributed randomly;
insert into temp values ('my string');
create or replace function java_substring(varchar, int, int)
returns varchar as 'Test.substring' language java;
select java_substring(a, 1, 5) from temp;
```

4.5 Table

4.5.1 Create a table

You can create tables within your databases.

Syntax

The complete syntax for creating a table is as follows. Depending on your business needs, not all clauses will be required. Use the clauses that can fulfill your business needs.

```
CREATE [[GLOBAL | LOCAL] {TEMPORARY | TEMP}] TABLE table_name (
  [ { column_namedata_type [ DEFAULT default_expr ]
    [column_constraint [ ... ] ]
  [ ENCODING ( storage_directive [,...] ) ]
  ]
  | table_constraint
  | LIKE other_table [{INCLUDING | EXCLUDING}
    {DEFAULTS | CONSTRAINTS}] ...}
  [, ... ] ]
)
[ INHERITS ( parent_table [, ... ] ) ]
[ WITH ( storage_parameter=value [, ... ] ) ]
[ ON COMMIT {PRESERVE ROWS | DELETE ROWS | DROP} ]
[ TABLESPACE tablespace ]
[ DISTRIBUTED BY (column, [ ... ] ) | DISTRIBUTED RANDOMLY ]
[ PARTITION BY partition_type (column)
  [ SUBPARTITION BY partition_type (column) ]
  [ SUBPARTITION TEMPLATE ( template_spec ) ]
  [...]
  ( partition_spec )
  | [ SUBPARTITION BY partition_type (column) ]
  [...]
  ( partition_spec
  [ ( subpartition_spec
    [ (...)]
  ) ]
  ) ]
)
```

The *column_constraint* clause can be defined as follows:

```
[CONSTRAINT constraint_name]
  NOT NULL | NULL
  | UNIQUE [USING INDEX TABLESPACE tablespace]
    [WITH ( FILLFACTOR = value )]
  | PRIMARY KEY [USING INDEX TABLESPACE tablespace]
    [WITH ( FILLFACTOR = value )]
  | CHECK ( expression )
  | REFERENCES table_name [ ( column_name [, ... ] ) ]
    [ key_match_type ]
    [ key_action ]
```

The *storage_directive* clause of columns can be defined as follows:

```
COMPRESSTYPE={ZLIB | QUICKLZ | RLE_TYPE | NONE}
[COMPRESSLEVEL={0-9} ]
```

```
[BLOCKSIZE={8192-2097152} ]
```

The `storage_parameter` clause of tables can be defined as follows:

```
APPENDONLY={TRUE|FALSE}
BLOCKSIZE={8192-2097152}
ORIENTATION={COLUMN|ROW}
CHECKSUM={TRUE|FALSE}
COMPRESSTYPE={ZLIB|QUICKLZ|RLE_TYPE|NONE}
COMPRESSLEVEL={0-9}
FILLFACTOR={10-100}
OIDS[=TRUE|FALSE]
```

The `table_constraint` clause can be defined as follows:

```
[CONSTRAINT constraint_name]
UNIQUE ( column_name [, ... ] )
    [USING INDEX TABLESPACE tablespace]
    [WITH ( FILLFACTOR=value )]
| PRIMARY KEY ( column_name [, ... ] )
    [USING INDEX TABLESPACE tablespace]
    [WITH ( FILLFACTOR=value )]
| CHECK ( expression )
| FOREIGN KEY ( column_name [, ... ] )
    REFERENCES table_name [ ( column_name [, ... ] ) ]
    [ key_match_type ]
    [ key_action ]
    [ key_checking_mode ]
```

Valid values of `key_match_type`:

```
MATCH FULL
| SIMPLE
```

Valid values of `key_action`:

```
ON DELETE
| ON UPDATE
| NO ACTION
| RESTRICT
| CASCADE
| SET NULL
| SET DEFAULT
```

Valid values of `key_checking_mode`:

```
DEFERRABLE
| NOT DEFERRABLE
| INITIALLY DEFERRED
| INITIALLY IMMEDIATE
```

Valid values of `partition_type`:

```
LIST
```

```
| RANGE
```

The partition_specification clause can be defined as follows:

```
partition_element [, ...]
```

The partition_element clause can be defined as follows:

```
DEFAULT PARTITION name
| [PARTITION name] VALUES (list_value [,...])
| [PARTITION name]
  START ([datatype] 'start_value') [INCLUSIVE | EXCLUSIVE]
  [ END ([datatype] 'end_value') [INCLUSIVE | EXCLUSIVE] ]
  [ EVERY ([datatype] [number | INTERVAL] 'interval_value') ]
| [PARTITION name]
  END ([datatype] 'end_value') [INCLUSIVE | EXCLUSIVE]
  [ EVERY ([datatype] [number | INTERVAL] 'interval_value') ]
[ WITH ( partition_storage_parameter=value [, ... ] ) ]
[ TABLESPACE tablespace ]
```

The subpartition_spec or template_spec clause can be defined as follows:

```
subpartition_element [, ...]
```

The subpartition_element clause can be defined as follows:

```
DEFAULT SUBPARTITION name
| [SUBPARTITION name] VALUES (list_value [,...])
| [SUBPARTITION name]
  START ([datatype] 'start_value') [INCLUSIVE | EXCLUSIVE]
  [ END ([datatype] 'end_value') [INCLUSIVE | EXCLUSIVE] ]
  [ EVERY ([datatype] [number | INTERVAL] 'interval_value') ]
| [SUBPARTITION name]
  END ([datatype] 'end_value') [INCLUSIVE | EXCLUSIVE]
  [ EVERY ([datatype] [number | INTERVAL] 'interval_value') ]
[ WITH ( partition_storage_parameter=value [, ... ] ) ]
[ TABLESPACE tablespace ]
```

The storage_parameter clause of partitions can be defined as follows:

```
APPENDONLY={TRUE|FALSE}
BLOCKSIZE={8192-2097152}
ORIENTATION={COLUMN|ROW}
CHECKSUM={TRUE|FALSE}
COMPRESSTYPE={ZLIB|QUICKLZ|RLE_TYPE|NONE}
COMPRESSLEVEL={1-9}
FILLFACTOR={10-100}
OIDS[=TRUE|FALSE]
```

Parameters

The [Table 4-9: Table creation parameters](#) table describes the key parameters for creating a table.

Table 4-9: Table creation parameters

Parameter	Description
TABLE_NAME	The name of the table to be created.
column_name	The name of a column to be created in the new table.
data_type	The data type of the column. For columns that contain textual data, set the data type to VARCHAR or TEXT. We do not recommend the CHAR type.
DEFAULT default_expr	Specifies a default value for the column. The system will assign this default value to all columns that do not have a value. The default values can be any variable-free expression. Subqueries or cross-references to other columns in the table are not allowed. The data type of the default expression must match the data type of the column. If a column does not have a default value, the default value is null.
ENCODING storage_directive	Specifies the type of compression and block size for the column data. This clause is valid only for append-optimized, column-oriented tables. Column compression settings are inherited from the table level to the partition level to the sub-partition level. The lowest-level settings have priority over inherited settings.
INHERITS	Specifies that all columns in the new table automatically inherit a parent table. You can use INHERITS to create a persistent relationship between the new child table and its parent table. Schema modifications to the parent table are applied to the child table as well. When the parent table is also scanned, the data of the child table is scanned as well.
LIKE other_table	Specifies a table from which the new table automatically copies all column names, data types, NOT NULL constraints, and distribution policies. Storage properties such as append-optimized or partition structure are not copied. Unlike INHERITS, the new table is completely decoupled from the original table after the new table is created.

Parameter	Description
CONSTRAINT constraint_name	Configures a column or table constraint. When a constraint is violated, the constraint name will be displayed in the error message. Constraint names can be used to communicate helpful information to client applications. Constraint names that contain spaces must be enclosed by double quotation marks ("").
WITH (storage_option=value)	Configures storage options for the table or its indexes.
ON COMMIT	The operation that the system performs on the temporary tables at the end of a transaction. Valid values: <ul style="list-style-type: none"> • PRESERVE ROWS: No special action is taken. The data will be retained after the transaction is completed. The data will only be released when the session is disconnected. • DELETE ROWS: All rows in the temporary table are deleted. • DROP: The temporary table is deleted.
TABLESPACE tablespace	Specifies the name of the tablespace in which the new table is to be created. If not specified, the default tablespace of the database is used.
DISTRIBUTED BY	Specifies the distribution policy for the database. <ul style="list-style-type: none"> • DISTRIBUTED BY (column, [...]): specifies the partition key. The system uses hash distribution based on the distribution key. To evenly distribute data, you must set the partition key to the primary key of the table or a unique column or a set of columns. • DISTRIBUTED RANDOMLY: distributes data randomly. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;">  Note: We recommend that you do not use random distribution. </div>

Parameter	Description
PARTITION BY	<p>Configures the partition key to partition the table. Partitioning large tables improves data access efficiency.</p> <p>To partition a table is to create a top-level (parent) table and multiple lower-level (child) tables. A parent table is always empty when the partition table is created. Data is stored in the lowest-level child tables. In a multi-level partition table, data is only stored in the lowest-level sub-partitions.</p> <p>Valid values: RANGE, LIST, and a combination of the two.</p>
SUBPARTITION BY	Configures a multi-level partitioned table.
SUBPARTITION TEMPLATE	You can specify a sub-partition template to create sub-partitions (lower-level child tables). This sub-partition template is applied to all parent partitions to ensure the same sub-partition structure.

Examples

Create a table and configure the partition key. The primary key is the default partition key in AnalyticDB for PostgreSQL.

```
CREATE TABLE films (
code          char(5) CONSTRAINT firstkey PRIMARY KEY,
title         varchar(40) NOT NULL,
did           integer NOT NULL,
date_prod    date,
kind         varchar(10),
len          interval hour to minute
);

CREATE TABLE distributors (
did          integer PRIMARY KEY DEFAULT nextval('serial'),
name        varchar(40) NOT NULL CHECK (name <> '')
);
```

Create a compressed table and configure the partition key.

```
CREATE TABLE sales (txn_id int, qty int, date date)
WITH (appendonly=true, compresslevel=5)
DISTRIBUTED BY (txn_id);
```

Use sub-partition templates of each level and the default partition to create a three-level partition table.

```
CREATE TABLE sales (id int, year int, month int, day int,
region text)
DISTRIBUTED BY (id)
PARTITION BY RANGE (year)
```

```

SUBPARTITION BY RANGE (month)
  SUBPARTITION TEMPLATE (
    START (1) END (13) EVERY (1),
    DEFAULT SUBPARTITION other_months )

SUBPARTITION BY LIST (region)
  SUBPARTITION TEMPLATE (
    SUBPARTITION usa VALUES ('usa'),
    SUBPARTITION europe VALUES ('europe'),
    SUBPARTITION asia VALUES ('asia'),
    DEFAULT SUBPARTITION other_regions)

( START (2008) END (2016) EVERY (1),
  DEFAULT PARTITION outlying_years);

```

4.5.2 Principles and scenarios of row store, column store, heap tables, and AO tables

AnalyticDB for PostgreSQL supports row store, column store, heap tables, and AO tables. This topic describes their principles and scenarios.

Row store and column store

Table 4-10: Comparison

Dimension	Row store	Column store
Definition	Row store stores data in the form of rows. Each row is a tuple. To read a column, you must deform all of the columns that precede the target column. Because of this, the costs for accessing the first and the last columns are different.	Column store stores data as columns corresponding to a file or a batch of files. The cost of reading any column is the same. However, if you need to read multiple columns, you must access multiple files. The more columns you access, the higher the overheads are.
Compression ratio	Low.	High.
Cost of reading any column	Columns with larger column numbers cost more.	Same.
Vector computing and JIT architecture	Not suitable. Not suitable for batch computation.	Suitable. More efficient when accessing and obtaining statistics of a batch of data.

Dimension	Row store	Column store
Scenarios	<p>If you need to perform a large number of update and delete operations due to OLTP requirements such as when querying table details where multiple columns are returned, you can use row store.</p> <p>You can use partition tables if you have diversified requirements. For example, if you need to partition the data based on time, you can use row store to query the details of recent data and use column store to obtain more statistics from historical data.</p>	<p>You can use column store if you need data statistics because of the OLAP requirements.</p> <p>If you need a higher compression ratio, you can use column store.</p>

Heap tables

A heap table is heap storage. All changes to the heap table generate redo logs that can be used to restore data by time point. However, heap tables cannot implement logical incremental backup because any data block in the table may be changed and it is not convenient to record the position by using the heap storage.

Commit and redo logs are used to ensure reliability when transactions are finished. You can also implement redundancy by building secondary nodes through redo logs.

Append-optimized (AO) tables

AO tables are used to append data for storage. When you delete the updated data, you can use another bitmap file to mark the row to be deleted and use the bit and offset to determine whether a row is deleted.

When the transaction is finished, you must call the fsync function to record the offset of the data block that performs the last write operation. Even if the data block only contains one record, a new data block will be appended for the next transaction. The data block is synchronized to the secondary node for data redundancy.

AO tables are not suitable for small transactions because the fsync function is called at the end of each transaction, and this data block will not be reused even if there is space left.

AO tables are suitable for OLAP scenarios, batch data writing, high compression ratio, and logical backup that supports incremental backup. During backup, you only need to record the offset from the backup and the bitmap deletion mark for each full backup.

Usage scenarios of heap tables

- **When multiple small transactions are handled, use a heap table.**
- **When you need to restore data by time point, use a heap table.**

Usage scenarios of AO tables

- **When you need to use column store, use an AO table.**
- **When data is written in batches, use an AO table.**

4.5.3 Enable the column store and compression features

If you want to improve performance, speed up data import, or reduce costs for tables with infrequent updates and multiple fields, we recommend that you use column store and compression. This will increase the compression ratio threefold to ensure faster performance and import speed.

To enable the column store and compression features, you must specify the column store and compression options when creating a table. For example, you can add

the following clause to the CREATE statement to enable the two features. For more information about the table creation syntax, see [Create a table](#).

```
with (APPENDONLY=true, ORIENTATION=column, COMPRESSTYPE=zlib,
COMPRESSLEVEL=5, BLOCKSIZE=1048576, OIDS=false)
```



Note:

AnalyticDB for PostgreSQL only supports zlib and RLE_TYPE compression algorithms. If you specify the quicklz algorithm, it is automatically converted to zlib.

4.5.4 Add a field to a column store table and set the default value

This topic describes how to add a field to a column store table and set the default value for the field, and how to use the ANALYZE statement to view the impact of updated data on the size of the column store table.

Context

In a column store table, each column is stored as a file, and two columns in the same row correspond to each other by using the offset. For example, if you add two fields of the INT8 type, you can quickly locate column B from column A by using the offset.

When you add the field, AO tables are not rewritten. If an AO table contains the records of deleted data, the added field must be filled with the deleted records before using the relative offset.

Procedure

1. Create three AO column store tables.

```
postgres=# create table tbl1 (id int, info text) with (appendonly=
true, blocksize=8192, compresstype=none, orientation=column);
NOTICE: Table doesn't have 'DISTRIBUTED BY' clause -- Using column
named 'id' as the Greenplum Database data distribution key for this
table.
HINT: The 'DISTRIBUTED BY' clause determines the distribution of
data. Make sure column(s) chosen are the optimal data distribution
key to minimize skew.
CREATE TABLE

postgres=# create table tbl2 (id int, info text) with (appendonly=
true, blocksize=8192, compresstype=none, orientation=column);
NOTICE: Table doesn't have 'DISTRIBUTED BY' clause -- Using column
named 'id' as the Greenplum Database data distribution key for this
table.
HINT: The 'DISTRIBUTED BY' clause determines the distribution of
data. Make sure column(s) chosen are the optimal data distribution
key to minimize skew.
```

```
CREATE TABLE

postgres=# create table tbl3 (id int, info text) with (appendonly=
true, blocksize=8192, compressstype=none, orientation=column);
NOTICE: Table doesn't have 'DISTRIBUTED BY' clause -- Using column
named 'id' as the Greenplum Database data distribution key for this
table.
HINT: The 'DISTRIBUTED BY' clause determines the distribution of
data. Make sure column(s) chosen are the optimal data distribution
key to minimize skew.
CREATE TABLE
```

2. Insert 10 million entries to the first two tables and 20 million entries to the third one.

```
postgres=# insert into tbl1 select generate_series(1,10000000),'test
';
INSERT 0 10000000
postgres=# insert into tbl2 select generate_series(1,10000000),'test
';
INSERT 0 10000000
postgres=# insert into tbl3 select generate_series(1,20000000),'test
';
INSERT 0 20000000
```

3. Analyze the tables and display their sizes.

```
postgres=# analyze tbl1;
ANALYZE
postgres=# analyze tbl2;
ANALYZE
postgres=# analyze tbl3;
ANALYZE

postgres=# select pg_size_pretty(pg_relation_size('tbl1'));
pg_size_pretty
-----
88 MB
(1 row)
postgres=# select pg_size_pretty(pg_relation_size('tbl2'));
pg_size_pretty
-----
88 MB
(1 row)
postgres=# select pg_size_pretty(pg_relation_size('tbl3'));
pg_size_pretty
-----
173 MB
(1 row)
```

4. Update all the data in the first table. Display the table size after the update. The size is twice as large as the size before the update.

```
postgres=# update tbl1 set info='test';
UPDATE 10000000
postgres=# analyze tbl1;
ANALYZE
postgres=# select pg_size_pretty(pg_relation_size('tbl1'));
pg_size_pretty
-----
```

```
173 MB
(1 row)
```

5. Add fields to the three tables and set the default values.

```
postgres=# alter table tbl1 add column c1 int8 default 1;
ALTER TABLE
postgres=# alter table tbl2 add column c1 int8 default 1;
ALTER TABLE
postgres=# alter table tbl3 add column c1 int8 default 1;
ALTER TABLE
```

6. Analyze the tables and view the table sizes.

```
postgres=# analyze tbl1;
ANALYZE
postgres=# analyze tbl2;
ANALYZE
postgres=# analyze tbl3;
ANALYZE

postgres=# select pg_size_pretty(pg_relation_size('tbl1'));
pg_size_pretty
-----
325 MB
(1 row)

postgres=# select pg_size_pretty(pg_relation_size('tbl2'));
pg_size_pretty
-----
163 MB
(1 row)

postgres=# select pg_size_pretty(pg_relation_size('tbl3'));
pg_size_pretty
-----
325 MB
(1 row)
```

When you add fields to the AO tables, the number of entries in the existing files will prevail. Even if all the entries are deleted, you must initialize the original data in the newly added fields.

4.5.5 Configure the table partition

For fact tables or large-sized tables in the database, we recommend that you configure table partitions.

Configure the table partition

You can use the table partitioning feature to delete data by using the `ALTER TABLE DROP PARTITION` statement to delete all the data in a partition, and import data by using the `ALTER TABLE EXCHANGE PARTITION` statement to add a new data partition on a regular basis.

AnalyticDB for PostgreSQL supports range partitioning, list partitioning, and composite partitioning. Range partitioning only supports partitioning by fields of the numeric or datetime data types.

The following example shows a table that uses range partitioning.

```
CREATE TABLE LINEITEM (
  L_ORDERKEY          BIGINT NOT NULL,
  L_PARTKEY           BIGINT NOT NULL,
  L_SUPPKEY           BIGINT NOT NULL,
  L_LINENUMBER        INTEGER,
  L_QUANTITY           FLOAT8,
  L_EXTENDEDPRICE     FLOAT8,
  L_DISCOUNT         FLOAT8,
  L_TAX               FLOAT8,
  L_RETURNFLAG        CHAR(1),
  L_LINESTATUS        CHAR(1),
  L_SHIPDATE          DATE,
  L_COMMITDATE        DATE,
  L_RECEIPTDATE       DATE,
  L_SHIPINSTRUCT      CHAR(25),
  L_SHIPMODE          CHAR(10),
  L_COMMENT           VARCHAR(44)
) WITH (APPENDONLY=true, ORIENTATION=column, COMPRESSTYPE=zlib
, COMPRESSLEVEL=5, BLOCKSIZE=1048576, OIDS=false) DISTRIBUTED BY (
  l_orderkey)
PARTITION BY RANGE (L_SHIPDATE) (START (date '1992-01-01') INCLUSIVE
END (date '2000-01-01') EXCLUSIVE EVERY (INTERVAL '1 month' ));
```

Principles of table partitioning

The purpose of partitioning is to minimize the amount of data that needs to be scanned during a query, so partitions must be associated with the query conditions

-
- **Principle 1: Select the fields related to the query conditions to configure partitions and reduce the amount of data to be scanned.**
- **Principle 2: When multiple query conditions exist, configure sub-partitions to further reduce the amount of data to be scanned.**

4.5.6 Configure the sort key

A sort key is an attribute of a table. Data on disks is stored in the order of the sort key.

Context

Sort keys have two major advantages:

- **Speed up and optimize column-store operations. The min and max meta information the system collects seldom overlaps with each other, which features good filterability.**
- **Eliminate the need to perform ORDER BY and GROUP BY operations. The data directly read from the disk is ordered as required by the sorting conditions.**

Create a table

```

Command:      CREATE TABLE
Description:  define a new table
Syntax:
CREATE [[GLOBAL | LOCAL] {TEMPORARY | TEMP}] TABLE table_name (
[ { column_name data_type [ DEFAULT default_expr ]      [column_con
straint [ ... ]
[ ENCODING ( storage_directive [,...] ) ]
]
| table_constraint
| LIKE other_table [{INCLUDING | EXCLUDING}
{DEFAULTS | CONSTRAINTS}] ...}
[, ... ] ]
[column_reference_storage_directive [, ] ]
)
[ INHERITS ( parent_table [, ... ] ) ]
[ WITH ( storage_parameter=value [, ... ] ) ]
[ ON COMMIT {PRESERVE ROWS | DELETE ROWS | DROP} ]
[ TABLESPACE tablespace ]
[ DISTRIBUTED BY (column, [ ... ] ) | DISTRIBUTED RANDOMLY ]
[ SORTKEY (column, [ ... ] ) ]
[ PARTITION BY partition_type (column)
  [ SUBPARTITION BY partition_type (column) ]
  [ SUBPARTITION TEMPLATE ( template_spec ) ]
  [...]
( partition_spec )
  | [ SUBPARTITION BY partition_type (column) ]
  [...]
( partition_spec
  [ ( subpartition_spec
    [(...)]
  ) ]
) ]

```

```
)
```

Examples:

```
create table test(date text, time text, open float, high float, low
  float, volume int) with(APPENDONLY=true,ORIENTATION=column) sortkey (
  volume);
```

Sort the table

```
VACUUM SORT ONLY [tablename]
```

Modify the sort key

This statement only modifies the catalog and does not sort data. You must execute the `VACUUM SORT ONLY` statement to sort the table.

```
ALTER [[GLOBAL | LOCAL] {TEMPORARY | TEMP}] TABLE table_name SET
  SORTKEY (column, [ ... ] )
```

Examples:

```
alter table test set sortkey (high,low);
```

4.6 Best practices

4.6.1 Configure memory and load parameters

You must configure memory and load parameters to improve database stability.

Background information

AnalyticDB for PostgreSQL is an MPP database with high computational and resource requirements. It consumes all of the resources provided to it, allowing AnalyticDB for PostgreSQL to have higher processing speeds but making it very easy to reach its limits.

The worst-case scenario in the event of the CPU, network, or hard disk exceeding its limits is a hardware bottleneck. However, in the event that memory is completely consumed, the database may crash.

How to avoid OOM errors

Out of memory (OOM) indicates that the system is unable to provide sufficient memory requested by a process. The following prompt appears when OOM errors occur:

```
Out of memory (seg27 host.example.com pid=47093) VM Protect failed to allocate 4096 bytes, 0 MB available
```

Causes

Possible causes of the OOM error include:

- **The memory of the database node is insufficient.**
- **Kernel parameters related to the memory of the operating system are incorrectly configured.**
- **Data skew has occurred, causing a compute node to request a large amount of memory.**
- **Query skew has occurred. For example, if the grouping fields of some aggregate or window functions are not distribution keys, the data must be redistributed. After redistribution, data will be skewed in a certain computer node and result in the node requesting a large amount of memory.**

Solutions

1. **Modify the queries to request less memory.**
2. **Use the resource queue provided by AnalyticDB for PostgreSQL to limit the number of concurrent queries. Reduce the number of queries executed within the cluster at the same time to reduce the overall memory requested by the system.**
3. **Reduce the number of compute nodes deployed on a host. For example, deploy 8 compute nodes instead of 16 compute nodes on a host with 128 GB of memory. This allows each compute node to use twice the amount of memory compared with the latter.**
4. **Increase the memory of a host.**
5. **Set the `gp_vmem_protect_limit` parameter to limit the maximum VMEM that can be used by a single compute node. The memory size of a single host and the number of compute nodes deployed on the host determine the maximum memory size that a single compute node can use on average.**
6. **For SQL statements that have unpredictable memory usage, you can set the `statement_mem` parameter in the session to limit the memory usage of a single SQL statement, so as to prevent a single SQL statement from consuming all available memory.**
7. **Set the `statement_mem` parameter at the database level to apply to all the sessions in the database.**

8. Use the resource queue to limit the maximum memory usage of the resource group. Add database users to the resource group to limit the overall memory used by these users.

Configure memory-related parameters

Properly configuring the operating system, database parameters, and resource queue can effectively reduce the probability of OOM.

When calculating the average memory usage of a single compute node on a single host, you must consider both the primary and secondary compute nodes. When the cluster encounters a host failure, the system will switch the service from primary compute nodes to the corresponding secondary compute nodes. During this time, the number of compute nodes on the host will be greater than usual. Therefore, you must consider the amount of resources that will be occupied by the secondary compute nodes during failover.

The following tables describe how to configure parameters of the operating system kernel and database to avoid OOM.

Table 4-11: Operating system kernel parameters describes the parameter configuration of the operating system kernel.

Table 4-11: Operating system kernel parameters

Parameter	Description
huge page	Do not configure the huge page parameter of the system. AnalyticDB for PostgreSQL does not support the latest version of PostgreSQL and therefore does not support the huge page feature. The huge page parameter locks a part of the allocated memory. Database nodes will not be able to use this part of the memory.

Parameter	Description
<code>vm.overcommit_memory</code>	<p>If you use the swap space, set this parameter to 2. If you do not use the swap space, set this parameter to 0.</p> <p>Valid values:</p> <ul style="list-style-type: none"> • 0: The requested memory space cannot exceed the difference between the total memory and the resident set size (RSS). An error is returned only when the memory has been exceeded. • 1: Most processes use the malloc function to apply for the memory, but do not use all the memory applied. When this parameter is set to 1, the memory requested by the malloc function will be allocated under any circumstances unless there is not sufficient memory. • 2: The swap space is also considered when the system calculates the memory space that can be applied for. You can apply for a large amount of memory even if the swap space is triggered.
<code>overcommit_ratio</code>	<p>The larger the value, the more memory that process can apply for and the less that will be reserved for the operating system. For the formula used to calculate the memory parameters, see Examples to calculate the memory parameters.</p> <p>When this parameter is set to 2, the memory address that can be applied for cannot exceed $\text{swap} + \text{memory} \times \text{overcommit_ratio}$.</p>

[Database parameters](#) describes the parameter configuration of the database.

Table 4-12: Database parameters

Parameter	Description
<code>gp_vmem_protect_limit</code>	<p>Specifies the maximum amount of memory that all processes can apply for on each node. If the value is too large, it may result in a system OOM error or even more serious problems. If the value is too small, SQL statements may not be executed even when the system has enough memory</p>

Parameter	Description
runaway_detector_activation_percent	<p>Default value: 90. This value is specified as a percentage . When the memory used by any compute node exceeds $\text{runaway_detector_activation_percent} \times \text{gp_vmem_protect_limit}/100$, the query is terminated to prevent OOM.</p> <p>The termination starts from the query that occupies the maximum memory until the memory reaches a value lower than $\text{runaway_detector_activation_percent} \times \text{gp_vmem_protect_limit}/100$.</p> <p>You can use the <code>gp_toolkit.session_level_memory_consumption</code> view to observe the memory usage of each session and runaway information.</p>

Parameter	Description
statement_mem	<p data-bbox="595 286 1431 479">Specifies the maximum amount of memory that a single SQL statement can apply. When the maximum memory is exceeded, spill files are created. Default value: 125. Unit : MB.</p> <p data-bbox="595 517 1414 600">We recommend that you set this parameter according to the following formula:</p> <div data-bbox="595 629 1431 748" style="background-color: #f0f0f0; padding: 5px;"> $(gp_vmem_protect_limit \times 0.9) / \max_expected_concurrent_queries$ </div> <div data-bbox="595 768 1431 1574" style="background-color: #f0f0f0; padding: 5px;"> <p data-bbox="608 775 767 835"> Note:</p> <ul data-bbox="616 864 1414 1563" style="list-style-type: none"> <li data-bbox="616 864 1414 1057">• You can specify the statement_mem parameter in a session. If the current concurrency is low and a session needs to run a query that requires a large amount of memory, you must specify this parameter in the session. <li data-bbox="616 1077 1414 1563">• Statement_mem is suitable for limiting memory usage in low concurrency scenarios. Statement_mem is suitable for limiting memory usage in low concurrency scenarios. If you use statement_mem to limit the memory for high concurrency scenarios , each query is allocated with a very small amount of memory. As a result, the performance of a small number of queries with high memory requirements in high concurrency scenarios is affected. We recommend that you use the resource queue to limit the maximum memory usage in high concurrency scenarios. </div>

Parameter	Description
<code>gp_workfile_limit_files_per_query</code>	<p>Specifies the maximum number of spill files that can be created by each query. When the memory requested by the query exceeds the <code>statement_mem</code> limit, spill files (also known as work files) are created, which is similar to the swap space of the operating system. When the number of spill files used exceeds the limit, the query will be terminated.</p> <p>Default value: 0, which indicates that an unlimited number of spill files can be created.</p>
<code>gp_workfile_compress_algorithm</code>	<p>Specifies the compression algorithm for spill files. Valid values: none and zlib.</p> <p>Specifies the compression algorithm. The values optimize storage space or I/O by sacrificing CPU. You can set this parameter when the disk is insufficient or the spill files meet a write bottleneck.</p>

Examples to calculate the memory parameters

The environment is as follows:

- **Host configuration:**

```
Total RAM = 256 GB
SWAP = 64 GB
```

- **Four hosts, each deployed with eight primary compute nodes and eight secondary compute nodes.**

When a host fails, the eight primary compute nodes are distributed to the remaining three hosts. A single host can be deployed with at most three extra primary compute nodes from the failed host. A single host can be deployed with at most 11 primary compute nodes.

1. Calculate the total memory allocated to AnalyticDB for PostgreSQL by the operating system.

Reserve 7.5 GB and 5% of memory for the operating system and calculate the available memory for all applications, and divide the available memory by the empirical coefficient of 1.7.

```
gp_vmem = ((SWAP + RAM) - (7.5 GB + 0.05 × RAM))/1.7
         = ((64 + 256) - (7.5 + 0.05 × 256))/1.7
         = 176
```

2. Use the empirical coefficient of 0.026 to calculate `overcommit_ratio`.

```
vm.overcommit_ratio = (RAM - (0.026 × gp_vmem))/RAM
                    = (256 - (0.026 × 176))/256
                    = .982
```

Set `vm.overcommit_ratio` to 98.

3. Calculate `gp_vmem_protect_limit` (the protection parameter of the maximum memory usage for each compute node), and divide `gp_vmem` by `maximum_acting_primary_segments` (the number of primary compute nodes to be run on each other host after one host fails).

```
gp_vmem_protect_limit calculation
gp_vmem_protect_limit = gp_vmem/maximum_acting_primary_segments
                     = 176/11
                     = 16 GB
                     = 16384 MB
```

Configure the resource queue

You can use resource queues to limit the number of concurrent queries and the total memory usage. When a query is running, it is added to the corresponding queue and the resources used are recorded in the queue. The resource limit of the queue is applied to all sessions in the queue.

The resource queue in AnalyticDB for PostgreSQL is similar to `cgroup` in Linux.

The syntax to create a resource queue is as follows:

```
Command:      CREATE RESOURCE QUEUE
Description:  create a new resource queue for workload management
Syntax:
CREATE RESOURCE QUEUE name WITH (queue_attribute=value [, ... ])
where queue_attribute is:
    ACTIVE_STATEMENTS=integer
    [ MAX_COST=float [COST_OVERCOMMIT={TRUE|FALSE}] ]
    [ MIN_COST=float ]
    [ PRIORITY={MIN|LOW|MEDIUM|HIGH|MAX} ]
```

```

[ MEMORY_LIMIT='memory_units' ]
| MAX_COST=float [ COST_OVERCOMMIT={TRUE|FALSE} ]
[ ACTIVE_STATEMENTS=integer ]
[ MIN_COST=float ]
[ PRIORITY={MIN|LOW|MEDIUM|HIGH|MAX} ]
[ MEMORY_LIMIT='memory_units' ]

```

Table 4-13: Resource queue creation parameters describes the parameters for creating the resource queue.

Table 4-13: Resource queue creation parameters

Parameter	Description
ACTIVE_STATEMENTS	<p>The number of SQL statements that are allowed to run (in the active status) concurrently.</p> <p>The value -1 indicates an unlimited number of SQL statements can run concurrently.</p>

Parameter	Description
MEMORY_LIMIT ' memory_units kB, MB or GB '	<p>Specifies the maximum memory usage allowed by all SQL statements in the resource queue. The value -1 indicates unlimited memory usage, but it is easy to trigger OOM errors because it is limited by the database or system parameters mentioned in the preceding sections.</p> <p>The memory usage of SQL statements is limited by resource queues and parameters.</p> <ul style="list-style-type: none"> • When the <code>gp_resqueue_memory_policy</code> parameter is set to <code>none</code>, the limit is the same as that in the Greenplum databases earlier than version 4.1. • When the <code>gp_resqueue_memory_policy</code> parameter is set to <code>auto</code> and you have specified the <code>statement_mem</code> parameter for a session or at the database level, the allowed memory of a single query will exceed the <code>MEMORY_LIMIT</code> of the resource queue. <p>Example:</p> <pre data-bbox="596 1111 1433 1256">=> SET statement_mem='2GB'; => SELECT * FROM my_big_table WHERE column='value' ORDER BY id; => RESET statement_mem;</pre> <ul style="list-style-type: none"> • The system parameter <code>max_statement_mem</code> can limit the maximum memory usage at the compute node level. The memory requested by a single query cannot exceed <code>max_statement_mem</code>. <p>You can modify the <code>statement_mem</code> parameter at the session level, but do not modify the <code>max_statement_mem</code> parameter. We recommend that you specify <code>max_statement_mem</code> as follows:</p> <pre data-bbox="596 1675 1433 1794">(seghost_physical_memory) / (average_number_concurrent_queries)</pre> <ul style="list-style-type: none"> • When the <code>gp_resqueue_memory_policy</code> parameter is set to <code>eager_free</code>, it indicates that the query is divided into several stages and that the database allocates the memory requested in the current stage. For example, if a query requests 1 GB of memory in total but only needs 100 MB during each stage, the database will allocate 100 MB of memory to the query. You can use <code>eager_free</code> to reduce the possibility of insufficient memory for the query.

Parameter	Description
MAX_COST float	<p>The maximum cost of the queries that are allowed to execute concurrently by the resource group. The cost is the estimated total cost in the SQL execution plan.</p> <p>The value of the parameter can be specified as a floating-point number (such as 100.0) or an exponent (such as 1e+2). A value of -1 indicates the cost is unlimited.</p>
COST_OVERCOMMIT boolean	Specifies whether the limit of max_cost can be exceeded when the system is idle. The value TRUE indicates the limit can be exceeded.
MIN_COST float	When the resources requested exceed the limit, the queries are queued. However, when the cost of a query is lower than the min_cost, the query can run without queuing.
PRIORITY ={MIN LOW MEDIUM HIGH MAX}	<p>The priority of the current resource queue. When resources are insufficient, CPU resources are allocated to the resource queue with a higher priority. The SQL statements in the resource queue with a higher priority can obtain CPU resources first. We recommend that you allocate users that initiate queries with high real-time requirements to resource queues with higher priority.</p> <p>This parameter is similar to the CPU resource group in the Linux cgroup and the time slice policy of real-time and common tasks.</p>

Example of modifying resource queue limits:

```
ALTER RESOURCE QUEUE myqueue WITH (MAX_COST=-1.0, MIN_COST= -1.0);
```

Example of putting the user in the resource queue:

```
ALTER ROLE sammy RESOURCE QUEUE poweruser;
```

The following table describes the parameters of resource queues.

Table 4-14: Resource queue parameters

Parameter	Description
<code>gp_resqueue_memory_policy</code>	Specifies the memory management policy of the resource queue.
<code>gp_resqueue_priority</code>	Specifies whether to enable query prioritization. Valid values: <ul style="list-style-type: none"> · On · Off If this parameter is disabled, existing priority settings are not evaluated.
<code>gp_resqueue_priority_cpucores_per_segment</code>	Specifies the number of CPU cores allocated to each compute node. For example, if an 8-core host is configured with two primary compute nodes, you can set the parameter to 4. If there are no other nodes on the primary node, set the parameter to 8. When the CPU is preempted, the SQL statements running in the resource group with higher priority are allocated with CPU resources first.
<code>gp_resqueue_priority_sweeper_interval</code>	Specifies the interval at which CPU utilization is recalculated for all active statements. The share value is calculated when the SQL statement is executed. You can calculate the share value based on the priority and <code>gp_resqueue_priority_cpucores_per_segment</code> . The smaller the value and the more frequent the calculation, the better the result brought by the priority settings and the larger the overhead.

Tips for configuring resource queues:

- We recommend that you create a resource queue for each user.

The default resource queue of AnalyticDB for PostgreSQL is `pg_default`. If no queue is created, all users are assigned to `pg_default`. This operation is not recommended. We recommend that you create a resource queue for each user. Typically, a database user corresponds to a business. Different database users may correspond to different businesses or users, such as business users, analysts, developers, and DBAs.

- We do not recommend that you use superusers to execute queries.

Queries initiated by superusers are only limited by the preceding parameters and not by the resource queue. We do not recommend that you use superusers to execute queries if you want to use resource queues to limit the use of resources.

- `ACTIVE_STATEMENTS` indicates the SQL statements that can be executed concurrently within the resource queue. When the cost of a query is lower than the `min_cost`, the query can run without queuing.
- You can specify the `MEMORY_LIMIT` parameter to set the allowed maximum memory usage of all the SQL statements in a resource queue. The `statement_mem` parameter has higher priority that can break through the limit of resource queues.



Note:

The memory of all resource queues cannot exceed `gp_vmem_protect_limit`.

- You can distinguish businesses by configuring the priorities of resource queues. For example, assume that report forms have top priority, while common businesses and analysts have lower priorities. In this case, you can create three resource queues with the max, high, and medium priorities, respectively.
- If the amount of resources requested at different times vary, you can use the `crontab` command to adjust the limits of resource queues periodically based on usage patterns.

For example, the queue of analysts has top priority during the day, while the queue of forms has lower priority at night. AnalyticDB for PostgreSQL does not support resource limits by time period. Therefore, you can only deploy tasks externally by using the `ALTER RESOURCE QUEUE` statement.

- You can use the view provided by `gp_toolkit` to observe the resource usage of the resource queues.

```
gp_toolkit.gp_resq_activity
gp_toolkit.gp_resq_activity_by_queue
gp_toolkit.gp_resq_priority_backend
gp_toolkit.gp_resq_priority_statement
gp_toolkit.gp_resq_role
gp_toolkit.gp_resqueue_status
```

5 Data Transmission Service (DTS)

5.1 What is DTS?

Data Transmission Service (DTS) is a data service provided by Alibaba Cloud. DTS supports data transmission between various types of data sources, such as relational databases.

DTS provides data transmission capabilities such as data migration and change tracking. DTS can be used in many scenarios, such as interruption-free data migration, geo-disaster recovery, cross-border data synchronization, and cache updates. DTS helps you build a data architecture that features high availability, scalability, and security.

- DTS allows you to simplify data transmission and focus on business development
-
- DTS supports MySQL as the data source type.

5.2 Log on to the DTS console

This topic uses the Google Chrome browser as an example to describe how to log on to the DTS console.

Prerequisites

- Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel
- The URL used to access the ASCM console is in the following format: `http://IP address or domain name of the ASCM console/manage`.
- We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to access the ASCM console. Press Enter.
2. Enter your username and password.

The system has a default super administrator, whose username is super. The super administrator can create system administrators. A system administrator

can create system users and notify the users of the default passwords by SMS or email.



Note:

When you log on to the ASCM console for the first time, you must modify the password of your username as instructed. For security concerns, your password must meet the minimum complexity requirements: The password must be 8 to 20 characters in length and must contain at least two types of the following characters: letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs (\$), and percent signs (%).

3. Click Login to go to the ASCM console homepage.
4. In the top navigation bar, choose Products > Data Transmission Service.
5. On the DTS page, select an organization and region, and then click DTS.

5.3 Data migration

5.3.1 Migrate data between two user-created MySQL databases

This topic describes how to migrate data between two user-created MySQL databases by using Data Transmission Service (DTS). DTS supports schema migration, full data migration, and incremental data migration. To migrate data from a user-created MySQL database, you can select all of the supported migration types to ensure service continuity.

Prerequisites

- The version of the user-created MySQL database is 5.1, 5.5, 5.6, 5.7, or 8.0.
- The available storage space of the destination MySQL database is larger than the total size of the data in the source MySQL database.
- The service port of the user-created MySQL database is accessible through the Internet.

Migration types

DTS supports schema migration, full data migration, and incremental data migration between MySQL databases.

- **Schema migration**

DTS migrates the schemas of the required objects to the destination database. DTS supports schema migration for the following types of objects: table, view, trigger, stored procedure, and function.

- **Full data migration**

DTS migrates historical data of the required objects from the source MySQL database to the destination MySQL database. If you also select incremental data migration, non-transaction tables without primary keys are locked during the full data migration process. Data cannot be written to these locked tables, and the locking duration depends on the data volume of the tables. The locks are released only after these tables are migrated. This ensures data consistency.

- **Incremental data migration**

In incremental data migration, data changes made during the migration are updated to the destination database. If DDL operations are performed during data migration, the schema changes are not migrated to the destination database

.

Limits

Data migration between user-created MySQL databases is subject to the following limits:

- DDL operations that are performed during incremental data migration cannot be synchronized to the destination database.
- DTS does not support schema migration for events.
- If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.
- If you select incremental data migration, the binary logging feature of the source MySQL database must be enabled and the value of the `binlog_format` parameter must be set to `row`. If the version of the source MySQL database is 5.6 or later, the value of the `binlog_row_image` parameter must be set to `full`.

Preparations

Before data migration, you must create accounts for the source and destination MySQL databases. You must also grant the read/write permissions to the accounts

. The following table lists the permissions required for database accounts when different migration types are used.

Table 5-1: Migration types and required permissions

Migration type	Schema migration	Full data migration	Incremental data migration
Source MySQL database	SELECT	SELECT	<ul style="list-style-type: none"> • SELECT • REPLICATION SLAVE • REPLICATION CLIENT
Destination MySQL database	<ul style="list-style-type: none"> • SELECT • REPLICATION SLAVE • REPLICATION CLIENT 	<ul style="list-style-type: none"> • SELECT • REPLICATION SLAVE • REPLICATION CLIENT 	<ul style="list-style-type: none"> • SELECT • REPLICATION SLAVE • REPLICATION CLIENT

1. Run the following command to create an account for an on-premises database:

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

Parameters:

- **username:** the account that you want to create.
- **host:** the host from which the account is allowed to log on to the database. To allow a local user to log on to the database, set the value of this parameter to

`localhost`. To allow the account to log on to the database from all hosts, set the value of this parameter to a percent sign (%).

- **password:** the logon password for the account.

For example, you can run the following command to create an account named `William`. The password is `Changme123`. The account is allowed to log on to the database from all hosts.

```
CREATE USER 'William'@'%' IDENTIFIED BY 'Changme123';
```

2. Run the following command to grant permissions to the account of the on-premises database:

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

Parameters:

- **privileges:** the operations that the account is allowed to perform, such as the `SELECT`, `INSERT`, and `UPDATE` operations. To allow the account to perform all operations, set the value of this parameter to `ALL`.
- **databasename:** the database name. To allow the account to perform operations on all databases, set the value of this parameter to an asterisk (*).
- **tablename:** the table name. To allow the account to perform operations on all tables, set the value of this parameter to an asterisk (*).
- **username:** the account to which you want to grant permissions.
- **host:** the host from which the account is allowed to log on to the database. To allow a local user to log on to the database, set the value of this parameter to `localhost`. To allow the account to log on to the database from all hosts, set the value of this parameter to a percent sign (%).
- **WITH GRANT OPTION:** the permissions that authorize the account to use the `GRANT` command. This parameter is optional.

For example, you can run the following command to grant all permissions on tables and databases to the `William` account. The account is allowed to log on to the database from all hosts.

```
GRANT ALL ON *.* TO 'William'@'%';
```



Note:

If you want to perform incremental data migration, you must enable binary logging for the on-premises database and set the relevant parameters.

3. Run the following command to check whether binary logging is enabled for the on-premises database:

```
show global variables like "log_bin";
```

If the query result is `log_bin=OFF`, binary logging is not enabled for the on-premises database. To ensure that incremental data generated during data migration is synchronized to the destination database, modify the following parameters in the `my.cnf` configuration file:

```
log_bin=mysql_bin binlog_format=row server_id=An integer greater than 1 binlog_row_image=full //If the version of the on-premises MySQL database is later than 5.6, you must set this parameter.
```

4. Run the following commands to restart the MySQL process:

```
$mysql_dir/bin/mysqladmin -u root -p shutdown $mysql_dir/bin/safe_mysqld &
```

The `mysql_dir` parameter indicates the installation directory of MySQL.

Procedure

1. [Log on to the DTS console](#).
2. In the left-side navigation pane, click **Data Migration**. On the **Migration Tasks** page that appears, click **Create Migration Task** in the upper-right corner. In the **Create DTS Instances** dialog box that appears, set the parameters.

The following table describes the required parameters.

Table 5-2: Parameters

Parameter	Description
Feature	The feature specified by the system. In this case, the value is Data Migration.
Region	The region where the source instance resides.
Instances to Create	The number of instances that you want to create.

3. Click **OK**.

- Find the data migration task and click **Configure Migration Task** in the **Actions** column. On the **Create Migration Task** page that appears, configure the source and destination databases.

The following table describes the required parameters.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a name for each task. We recommend that you use an informative name for easy identification.
Source Database	Instance Type	The instance type of the source database. Select User-Created Database with Public IP Address .
	Instance Region	If the instance type is set to User-Created Database with Public IP Address , you do not need to specify the instance region.
	Database Type	The type of the source database. Select MySQL .
	Hostname or IP Address	The endpoint that is used to connect to the source database.
	Port Number	The port number of the source database . The default port number for a MySQL database is 3306.
	Database Account	The account that is used to log on to the source database.
	Database Password	The password for the account that is used to log on to the source database.
Destination Database	Instance Type	The instance type of the destination database. Select User-Created Database with Public IP Address .
	Instance Region	If the instance type is set to User-Created Database with Public IP Address , you do not need to specify the instance region.
	Database Type	The type of the destination database. Select MySQL .
	Hostname or IP Address	The endpoint that is used to connect to the destination database.

Section	Parameter	Description
	Port Number	The port number of the destination database. The default port number for a MySQL database is 3306.
	Database Account	The account that is used to log on to the destination database.
	Database Password	The password for the account that is used to log on to the destination database.



Note:

After the source and destination databases are configured, you can click **Test Connectivity** to verify whether the specified information is valid.

- In the lower-right corner of the page, click **Set Whitelist** and **Next** to go to the **Configure Migration Types and Objects** step.
- Select the migration types and objects to be migrated.



Note:

To modify the name of an object that is migrated to the destination database, move the pointer over the object in the **Selected** section, and then click **Edit**.

Parameter	Description
Migration Types	<ul style="list-style-type: none"> To perform only full data migration, select Schema Migration and Full Data Migration. <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> Note: To ensure data consistency, do not write new data into the source MySQL database during full data migration. </div> <ul style="list-style-type: none"> To migrate data with minimal downtime, select Schema Migration, Full Data Migration, and Incremental Data Migration.

Parameter	Description
Objects to be migrated	<p>In the Available section, select the objects to be migrated and click the right arrow (>) icon to move the objects to the Selected section.</p> <ul style="list-style-type: none"> You can select columns, tables, or databases as the objects to be migrated. After an object is migrated to the destination MySQL database, the name of the object remains the same as that in the source MySQL database. You can change the name of an object in the destination MySQL database by using the object name mapping feature provided by DTS. For more information about how to use this feature, see Database, table, and column name mapping. If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.

7. In the lower-right corner of the page, click Precheck.



Note:

- Before you can start the data migration task, a precheck is performed. You can start the data migration task only after the task passes the precheck.
- If the task fails the precheck, click the info icon next to each failed item to view details. Troubleshoot the issues based on the cause of failure and run the precheck again.
- After troubleshooting, select the task from the task list and run the precheck again.

8. Start the data migration task after the task passes the precheck. After the task is started, you can check the migration status and progress on the Migration Tasks page.

5.3.2 Precheck items

5.3.2.1 Source database connectivity

This check item checks whether the DTS server can connect to the source database for migration. DTS creates a connection to the source database by using the JDBC protocol. If the connection fails, the check item fails.

The source database connectivity check may fail for the following reasons:

- An incorrect account or password is provided when a migration task is created.

Diagnostics:

On any network-ready server that can connect to the source database, use the account and password specified for creating the migration task to connect to the source database through client software. Check whether the connection succeeds. If an error is reported for the connection and the error message contains **Access deny**, the account or password is incorrect.

Troubleshooting:

Modify the migration task in the DTS console. Correct the account and password . Then re-run the precheck.

- The migration account of the source database implements access control based on source IP addresses.

Diagnostics:

- On any network-ready server that can connect to the source database, use the account and password specified for creating the migration task to connect to the source database through client software. Check whether the connection succeeds. If the connection succeeds, the source database has access restrictions based on IP addresses. Only allowed servers can connect to it. The IP address of the DTS server is not included in the whitelist of the source database, so the DTS server cannot connect to the source database.
- If the source database is a MySQL database, you can access the source database by using the MySQL client. Run the `select host from mysql.user where user='Migration account', password='Migration account password'` command. If the query result is not %, the IP address of the DTS server is not included in the whitelist of the source database, which results in the connection failure.

Troubleshooting:

- If the source database is a MySQL database, run the `grant all on . to 'Migration account'@'%' identified by 'Migration account password';` command in the source database to re-authorize the migration account. Replace the migration account and password in this command with the real ones. After the account is authorized, re-run the precheck.

- A firewall is configured on the source database server.

Diagnostics: If the source database is installed on a Linux server, run `iptables -L` in the shell to check whether a firewall has been configured on the server.

If the source database is installed on a Windows server, perform the following operations: Open the Control Panel, click System and Security. On the System and Security window that appears, click Windows Firewall. Check whether a firewall has been configured on the server.

Troubleshooting:

Disable the firewall and perform the precheck again.

- There is no connectivity between the DTS server and the source database.

If none of the preceding cases applies, the check item may fail because there is no connectivity between the DTS server and the source database. In this case, contact the DTS engineers on duty.

5.3.2.2 Check the destination database connectivity

This check item checks whether the DTS server can connect to the destination database for migration. DTS creates a connection to the destination database by using the JDBC protocol. If the connection fails, the check item fails.

The destination database connectivity precheck may fail for the following reasons:

- An incorrect account or password is provided when a migration task is created.

Diagnostics:

On any network-ready server that can connect to the destination database, use the account and password specified for creating the migration task to connect to the destination database through client software. Check whether the connection succeeds. If an error is reported for the connection and the error message contains Access deny, the account or password is incorrect.

Troubleshooting:

Modify the migration task in the DTS console, correct the account and password, and perform the precheck again.

- **There is no connectivity between the DTS server and destination database.**

If you check that the password and account are correct, the check item may fail because there is no connectivity between the DTS server and the destination database. In this case, contact the DTS engineers on duty.

5.3.2.3 Binlog configurations in the source database

Check whether binlogging is enabled for the source database

This check item is run only when incremental data is to be migrated between MySQL instances. This check item checks whether binlogging is enabled for the source database. If this check item fails, binlogging is not enabled for the source database.

Troubleshooting: Set `log_bin=mysql_bin` in the configuration file of the source database to enable binlogging. Restart the source database and re-run the precheck.

Check whether the binlog format is ROW in the source database

This check item is run only when incremental data is to be migrated between MySQL instances. This check item checks whether the binlog format is ROW in the source database. If this check item fails, the binlog format is not ROW in the source database.

Troubleshooting: Run the `set global binlog_format=ROW` command in the source database. Then, re-run the precheck. We recommend that you restart the source MySQL database after the modification. Otherwise, connected sessions may continue to be written in non-ROW mode, resulting in data loss.

Check whether a specified binlog file has been deleted from the source database

This check item is run only when incremental data is to be migrated between MySQL instances. This check item checks whether a specified binlog file has been deleted from the source database. If this check item fails, the binlog file does not exist in the source database.

Troubleshooting: Run the `PURGE BINARY LOGS TO "The name of the binlog file ranking the first place among all binlog files that have not been deleted"` command in the source database. Then, re-run the precheck.

For specific purge file names, see the precheck troubleshooting.

Check whether the `binlog_row_image` value of the MySQL source database is FULL

This check item is run only when incremental data is to be migrated between MySQL instances. This check item checks whether the `binlog_row_image` of the source database is FULL, or whether the full image is recorded. If this check item fails, the binlog file of the source database does not record the full image.

Troubleshooting: Run the `set global binlog_row_image=FULL` command in the source database. Then, re-run the precheck.

5.3.2.4 Referential integrity constraint

This check item checks whether all the parent-child tables with foreign key dependencies among the objects to be migrated have been migrated, to avoid damaging the integrity of foreign key constraints.

If this check item fails, the failure cause is that the "parent table name" parent table on which the "child table name" table to be migrated is dependent has not been migrated.

Troubleshooting:

- **Do not migrate the child tables involved in the failed referential integrity constraint check. Modify the migration task and delete these child tables from the list of objects to be migrated. Then re-run the precheck.**
- **Migrate the parent tables for the child tables involved in the failed referential integrity constraint check. To do so, modify the migration task and add these parent tables to the list of objects to be migrated. Then re-run the precheck.**
- **Delete the foreign key dependencies of the child tables involved in the failed referential integrity constraint check. Modify the source database and delete the foreign key dependencies of these child tables. Then, re-run the precheck.**

5.3.2.5 Existence of Federated tables

This check item is run only when incremental data is to be migrated between MySQL instances. This check item checks whether any storage engines not supported by incremental data migration exist in the source database. Currently, incremental data migration does not support the Federated and the MRG_MyISAM storage engines.

If this check item fails and the error message "The Federated engine is used for the following source tables:" is displayed, the storage engine of some tables in the source database is Federated.

If this check item fails and the error message "The MRG_MyISAM engine is used for the following source tables:" is displayed, the storage engine of some tables in the source database is MRG_MyISAM.

Troubleshooting:

Modify the migration task by deleting the tables with the Federated or MRG_MyISAM storage engine from the list of objects to be migrated. Then create a separate migration task to implement schema migration and full data migration for these tables.

5.3.2.6 Permissions

Check the permissions granted to the migration account of the source database

This check item checks whether the migration account of the source database has the required permissions for data migration. For the migration permissions required by each type of database, see the Data Migration chapter.

Check the permissions granted to the migration account of the destination database

This check item checks whether the migration account of the source database has the required permissions for data migration. For the migration permissions required by each type of database, see the data migration chapter.

5.3.2.7 Object name conflict

This check item checks for duplicate object names in the destination and source database. If this check item fails, an object in the destination RDS instance has the same name as an object to be migrated. This causes the migration to fail.

When this check item fails, an error message is displayed indicating that an object in the destination database has the same name as an object to be migrated from the source database.

Troubleshooting:

- **Use the database and table name mapping feature provided by DTS to migrate the object to be migrated to another object with a different name in the destination database.**

- In the destination database, delete or rename the object that has the same name as the object to be migrated.
- Modify the migration task and delete that object to be migrated from the list of objects to be migrated. Do not migrate this object.

5.3.2.8 Schema existence

This check item checks whether the database to be migrated exists in the destination RDS instance. If no, DTS creates one automatically. However, under the following circumstances, the automatic database creation fails, and this check item prompts a failure:

- The database name contains characters other than lowercase letters, digits, underscores (_), and hyphens (-).

The cause of the precheck failure is that the name of the source database does not comply with the requirements of RDS.

Troubleshooting: On the database management page of the RDS console, create a database that complies with the requirements of RDS and grant the migration account the read and write permissions on the new database. Use the database name mapping feature provided by DTS to map the source database to the new database. Then, perform the precheck again.

- The character set of the database is not UTF8, GBK, Latin1, or UTF-8MB4.

The cause of the precheck failure is that the character set of the source database does not comply with the requirements of RDS.

Troubleshooting: On the database management page of the RDS console, create a database that complies with the requirements of RDS and grant the migration account the read and write permissions on the new database. If the new database and the database to be migrated have different names, you can use the database name mapping feature of DTS to map the database to be migrated to the new database. Then re-run the precheck.

- The migration account of the destination database has no read and write permissions on the database to be migrated.

The cause of the precheck failure is that you are not authorized to operate on the source database.

Troubleshooting: On the database management page of the RDS console, click the Account Management tab. Grant the migration account the read and write permissions on the source database. Then, perform the precheck again.

5.3.2.9 Source database server_id

This check item is run only when incremental data is to be migrated between MySQL instances. This check item checks whether server-id of the source database is set to an integer greater than 1.

If this check item fails, run the `set global server_id='an integer greater than 1'` command in the source database. Then run the precheck again.

5.3.2.10 Source database version

This check item checks whether the version of the source database is supported by DTS. [Table 5-3: Source database types and versions](#) lists the source database versions supported by DTS.

Table 5-3: Source database types and versions

Source database type	Supported version
MySQL	5.0, 5.1, 5.5, and 5.6. Only 5.1, 5.5, and 5.6 are supported for incremental data migration.

When the version check fails, you can only upgrade or downgrade the source database to the versions supported by DTS. Then re-run the precheck.

5.3.3 Database, table, and column name mapping

This topic describes how to use the object name mapping feature when you configure a data migration task.

The data migration feature provided by DTS supports object name mapping. Objects to be migrated, such as databases, tables, and columns, can have different names in the source and destination instances.

Database name mapping

If a database you migrate has different names in the source and destination instances, you can map the database names by using the object name mapping feature of DTS.

You can configure the database name mapping feature in Step 2 "Select migration types and objects to be migrated" when you configure the migration task. Perform the following steps to configure the database name mapping feature:

- 1. In the Selected area, move the pointer over the row of the object that requires database name mapping. The Edit button appears on the right.**
- 2. Modify the database name.**

If you want the database name to change to jiangliutest after the database is migrated to the destination instance, click Edit to open the Edit Database Name dialog box.

In the Edit Database Name dialog box, modify the database name directly. The database is stored under the new name in the destination instance.

Assume that the original database name is amptest.

In the Edit Database Name dialog box, change amptest to jiangliutest, so that the database name changes to jiangliutest after the database is migrated to the destination instance.

The database uses the new name in the destination instance.

Table name mapping

If a table you migrate has different names in the source and destination instances, you can map the table names by using the object name mapping feature of DTS.

If you want to use the table name mapping feature, do not select the entire database as the object to be migrated. Instead, select a specific table.

Besides tables, other schema objects such as views, stored procedures, stored functions, and synonyms are also available for object name mapping in the similar way.

You can configure the table name mapping feature in Step 2 "Select migration types and objects to be migrated" when you configure the migration task. Perform the following steps to configure the table name mapping feature:

1. In the Selected area, move the pointer over the row of the object that requires table name mapping. The Edit button appears on the right.
2. Modify the table name.

If you want the table name to change from amptest to jiangliutest after the table is migrated to the destination instance, click Edit to open the Edit Table Name dialog box.

In the Edit Table Name dialog box, modify the table name directly. The table is stored under the new name in the destination instance.

Assume that the original table name is amptest.

Change amptest to jiangliutest, so that the table name changes to jiangliutest after the table is migrated to the destination instance.

Column name mapping

If columns of a table that you migrate have different names in the source and destination instances, you can use the object name mapping feature provided by DTS.

You can configure the column name mapping feature in Step 2 "Select migration types and objects to be migrated" when you configure the migration task. If you want to modify the name of a column to be migrated, do not select the entire database as the object to be migrated. Instead, select the table that the column belongs to. Perform the following steps to configure the column name mapping feature:

1. Assume that you want to change the name of a column in the sbtest1 table. In the Selected area, move the pointer over the row of the sbtest1 table. The Edit button appears on the right.
2. Click Edit to open the Edit Table Name dialog box.

Modify the column name. After the modification, the column is stored under the new name in the destination database.

Now, you have configured the column name mapping feature.

5.3.4 Configure an SQL filter for filtering the data to be migrated

This section describes how to configure an SQL filter for filtering migration data when you create a migration task.

DTS allows you to configure an SQL filter to filter the table data to be migrated. The SQL filter applies only to the configured table. DTS filters the data in the table of the source database based on this filter. Only data that meets this filter can be migrated to the destination database. This feature is applicable to multiple scenarios such as regular incremental data migration and table partitioning.

Functional restrictions

The SQL filter applies only to full data migration. If you select Incremental Data Migration as the migration type, the SQL filter does not apply.

Configure an SQL filter

You can configure an SQL filter in the Migration Types and Tasks step of migration task configuration.

If you want to configure an SQL filter for table migration, you must select a specific table instead of the entire database as the object to be migrated. The following part describes how to configure an SQL filter.

Configure an SQL filter

1. In the Migration Types and Tasks step, move the pointer over the table for which you want to create an SQL filter in the Selected area. The Edit button appears.
2. Click Edit to configure a filter.

Modify an SQL filter

Filters in DTS are the same as the standard SQL WHERE conditions for databases and support calculation and simple functions.

Enter an SQL filter in the text box as needed.

Now, you have configured an SQL filter.

5.3.5 Troubleshoot migration errors

DTS provides the feature of online troubleshooting in multiple stages to fix migration errors. These stages include:

- **Schema migration**

DTS supports data migration between heterogeneous data sources. If you import data of unsupported types to the destination instance during a schema migration, the migration fails.

- **Full data migration**

During full data migration, the migration task may fail because the destination RDS instance does not have sufficient space or required IP addresses have been deleted from the whitelist. In this case, you can modify the task configurations and then restart the task.

DTS provides the online troubleshooting feature that allows you to resume a failed task when an error occurs during migration. The following sections describe how to troubleshoot errors that occur during schema migration and full data migration.

Troubleshoot errors occurred during schema migration

If a schema migration task fails, the task status changes to Migration Failed and the Rectify button appears.

Click Rectify next to a failed object.

Click Rectify next to each failed object. A troubleshooting dialog box appears.

Modify the schema definition based on the cause of failure. Click Rectify after you complete the modification and re-import the modified definition to the destination instance.

If the error persists after you click Rectify, the cause of failure changes to Troubleshooting Failed and the cause of troubleshooting failure is displayed. You need to continue troubleshooting based on the cause of troubleshooting failure until the troubleshooting is successful.

The details page of the schema migration appears after troubleshooting is successful, and the status of the object changes to Finished.

The task resumes after issues with all objects are rectified. For example, the task resumes by proceeding to the full data migration stage.

Troubleshoot errors occurred during full data migration

DTS provides the troubleshooting and retry feature for the following causes of failures:

- If you fail to connect to the source or destination database, retry the task after you ensure that the network connection is established.
- If a connection to the source or destination database times out, retry the task after you ensure that the network connection is established.
- If the destination RDS instance does not have sufficient space or the instance is locked, retry the task after you scale up the RDS instance or clean up the instance log space.
- If MyISAM of the source database is corrupted, retry the task after troubleshooting.

For other circumstances, if full data migration fails, DTS only offers the Ignore option. You can ignore the failed object and continue the migration of other objects

.

If a full data migration task fails, the status of the task changes to Migration Failed and the Rectify button appears.

When a migration task fails, click Rectify next to a failed object.

If you encounter the preceding failures and the migration tasks can be retried, troubleshoot the errors as prompted. Then, click the Retry button on the full data migration details page to continue the data transfer in the task.

For other causes of failures, DTS only supports the Ignore operation to ignore the full data migration of the object. After you click Ignore, data of this object is not migrated, but data of other objects is migrated to the destination instance.

5.4 Change tracking

5.4.1 Track data changes from a user-created MySQL database

The change tracking feature allows you to track data changes in real time. It applies to scenarios such as lightweight cache updates, business decoupling, asynchronous data processing, and real-time data synchronization of ETL operations. This topic describes how to track data changes from a user-created MySQL database.

Prerequisites

The version of the user-created MySQL database is 5.1, 5.5, 5.6, 5.7, or 8.0.

Supported database type

You can track data changes only from a user-created MySQL database with a public IP address.

**Note:**

If your source database is a user-created MySQL database, you must create a database account and enable binary logging.

Create a change tracking task

1. *Log on to the DTS console.*
2. **In the left-side navigation pane, click Change Tracking.**
3. **On the Change Tracking Tasks page, click Create Change Tracking Task in the upper-right corner.**
4. **In the Create DTS Instances dialog box that appears, specify a region and the number of change tracking instances, and then click Create.**

The following table describes the required parameters.

Table 5-4: Parameters

Parameter	Description
Feature	The feature specified by the system. In this case, the value is Change Tracking.
Region	The region where the source instance resides.
Instance Type	Select MySQL.
Instances to Create	The number of change tracking instances that you want to create.

5. **In the message that appears, click OK.**

Configure a change tracking task

1. **Find the change tracking task and click Configure Channel in the Actions column.**

2. Configure the source database information and network type for the change tracking task.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you use an informative name for easy identification. You do not need to use a unique task name.
Source Database	Version	Select New.
	Instance Type	Select User-Created Database with Public IP Address.
	Database Type	The instance type that you select when creating the change tracking task. You cannot change the value of this parameter.
	Instance Region	The region that you select when creating the change tracking task. You cannot change the value of this parameter.
	Hostname or IP Address	Enter the endpoint that is used to connect to the user-created MySQL database.
	Port Number	Enter the service port number of the user-created MySQL database. The default port number is 3306.
	Database Account	Enter the account for the user-created MySQL database.
	Database Password	Enter the password for the database account.
Consumer Network Type	Network Type	The network type of the change tracking task. Only the classic network is supported.

3. Click Set Whitelist and Next.
4. In the Create Change Tracking Account message that appears, click Next after the account is created.



Note:

In this step, DTS creates a database account for change tracking in the source instance.

5. Select the data change types and objects.

Parameter	Description
Required Data Types	<ul style="list-style-type: none"> • Data Updates: If you select Data Updates, DTS tracks data updates of the selected objects, including INSERT, DELETE, and UPDATE operations. • Schema Updates: If you select Schema Updates, DTS tracks the create, delete, and modify operations that are performed on all object schemas of the source instance. You need to use the change tracking client to filter the required data. <div style="background-color: #f0f0f0; padding: 5px;">  Note: <ul style="list-style-type: none"> • If you select a database as the object, DTS tracks data changes of all objects, including new objects in the database. • If you select a table as the object, DTS tracks only data changes in this table. In this case, if you want to track data changes of a new table, you must add the table to the selected objects. For more information, see Modify objects for change tracking. </div>
Required Objects	<p>Select objects from the Required Objects section and click the right arrow icon to add the objects to the Selected section.</p> <div style="background-color: #f0f0f0; padding: 5px;">  Note: The objects for change tracking include tables and databases. </div>

6. In the lower-right corner of the page, click Save and Precheck.

**Note:**

- A precheck is performed before you can start the change tracking task. You can start the change tracking task only after the task passes the precheck.
- If the task fails the precheck, click the  icon next to each failed item to view details. Fix the issues based on the instructions and run the precheck again.

7. Close the Precheck dialog box after the following message is displayed: The precheck is passed.

After the change tracking task is configured, DTS performs initial change tracking, which takes about one minute. After the initial change tracking is complete, you can create a consumer group or consume the tracked data.

5.4.2 Create consumer groups

The change tracking feature allows you to create multiple consumer groups. Consumers in different consumer groups can track data changes from the same data source. Consumer groups help you reduce the cost for tracking data changes and improve the efficiency of data consumption.

Note

- You can create multiple consumer groups in a change tracking instance to repeatedly consume data.
- A consumer group consumes each message only once, and consumers in the consumer group serve as a backup for each other.
- In a consumer group, only one consumer can consume data at a time, while other consumers serve as disaster recovery nodes.

Procedure

1. [Log on to the DTS console](#).
2. In the left-side navigation pane, click **Change Tracking**.
3. Find the change tracking task and click the task ID.
4. In the left-side navigation pane, click **Consume Data**.
5. On the **Consume Data** page, click **Add Consumer Group** in the upper-right corner.
6. In the **Create Consumer Group** dialog box that appears, set the parameters for the consumer group.

Parameter	Description
Consumer Group Name	Enter a new name for the consumer group. We recommend that you use an informative name for easy identification.
Username	Enter the username of the consumer group. <ul style="list-style-type: none"> • A username must contain one or more of the following character types: uppercase letters, lowercase letters, digits, and underscores (_). • The username must be 1 to 16 characters in length.

Parameter	Description
Password	<p>Enter the password that corresponds to the username of the consumer group.</p> <ul style="list-style-type: none"> • A password must contain two or more of the following character types: uppercase letters, lowercase letters, digits, and special characters. • The password must be 8 to 32 characters in length.
Confirm Password	Enter the new password again.

7. Click Create.

5.4.3 Manage consumer groups

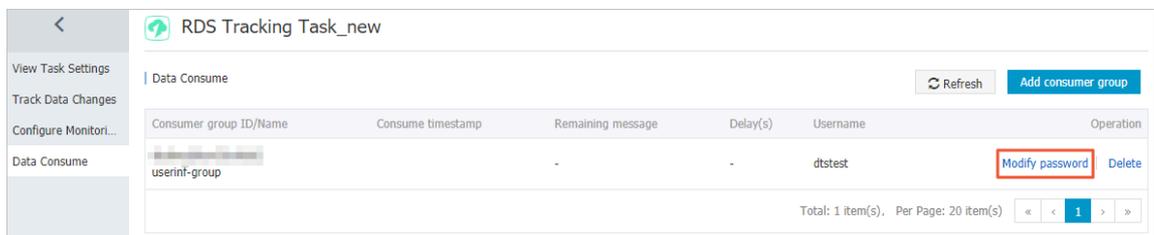
You can manage consumer groups of a change tracking instance in the DTS console. This topic describes how to modify the password of a consumer group and how to delete a consumer group.

Procedure

1. [Log on to the DTS console.](#)
2. In the left-side navigation pane, click Change Tracking.
3. Find the change tracking task and click the task ID.
4. In the left-side navigation pane, click Consume Data.
5. Modify the password of a consumer group or delete a consumer group.

Modify the password of a consumer group

- a) On the Consume Data page, find the target consumer group and click **Modify Password** in the Actions column.



- b) In the Modify Password dialog box that appears, enter the old password and new password, and enter the new password again in the Confirm Password field.



Note:

- A password must contain two or more of the following character types: uppercase letters, lowercase letters, digits, and special characters.
- The password must be 8 to 32 characters in length.

c) Click Modify.

Delete a consumer group



Note:

After a consumer group is deleted, the data in the group will be cleared and cannot be recovered. We recommend that you use caution when performing this operation.

- a) On the Consume Data page, find the target consumer group and click Delete in the Actions column.
- b) In the Delete Consumer Group message that appears, click OK.

5.4.4 Modify objects for change tracking

DTS allows you to add or remove objects for change tracking in the consumption process. This topic describes how to modify objects for change tracking.

Note

- After you add an object, the change tracking task pulls the incremental data of the new object from the time when the modification takes effect.
- After you remove an object, the change tracking client no longer tracks data changes of the removed object.

Procedure

1. [Log on to the DTS console](#).
2. In the left-side navigation pane, click Change Tracking.
3. Find the change tracking task and click Modify Required Objects in the Actions column.

4. In the Select Required Objects step, you can add and remove objects for change tracking.

- **Add objects for change tracking**

In the Required Objects section, select the required objects and click the 

button to add the objects to the Selected section.

- **Remove objects for change tracking**

In the Selected section, select the objects to be removed and click the 

button to move the objects to the Required Objects section.

5. In the lower-right corner of the page, click Save and Precheck.



Note:

- A precheck is performed before you can start the change tracking task. You can start the change tracking task only after the task passes the precheck.
- If the task fails the precheck, click the  icon next to each failed item to view details. Fix the issues based on the instructions and run the precheck again.

6. Close the Precheck dialog box after the following message is displayed: The precheck is passed.

6 KVStore for Redis

6.1 What is KVStore for Redis?

KVStore for Redis is an online key-value storage service compatible with open-source Redis protocols. KVStore for Redis supports various data types such as strings, lists, sets, sorted sets, and hash tables. The service also supports advanced features such as transactions, message subscription, and message publishing.

You can easily deploy and manage KVStore for Redis databases in the KVStore for Redis console.

- **You can create an instance to initialize a database environment.**
- **Before using a KVStore for Redis instance, add IP addresses or CIDR blocks used to access the database to the instance whitelist.**
- **You can manage instances in the KVStore for Redis console.**
- **To secure data, you can periodically or immediately back up or restore databases in the KVStore for Redis console.**
- **You can log on to a database by using a client and then execute SQL statements to perform database operations.**

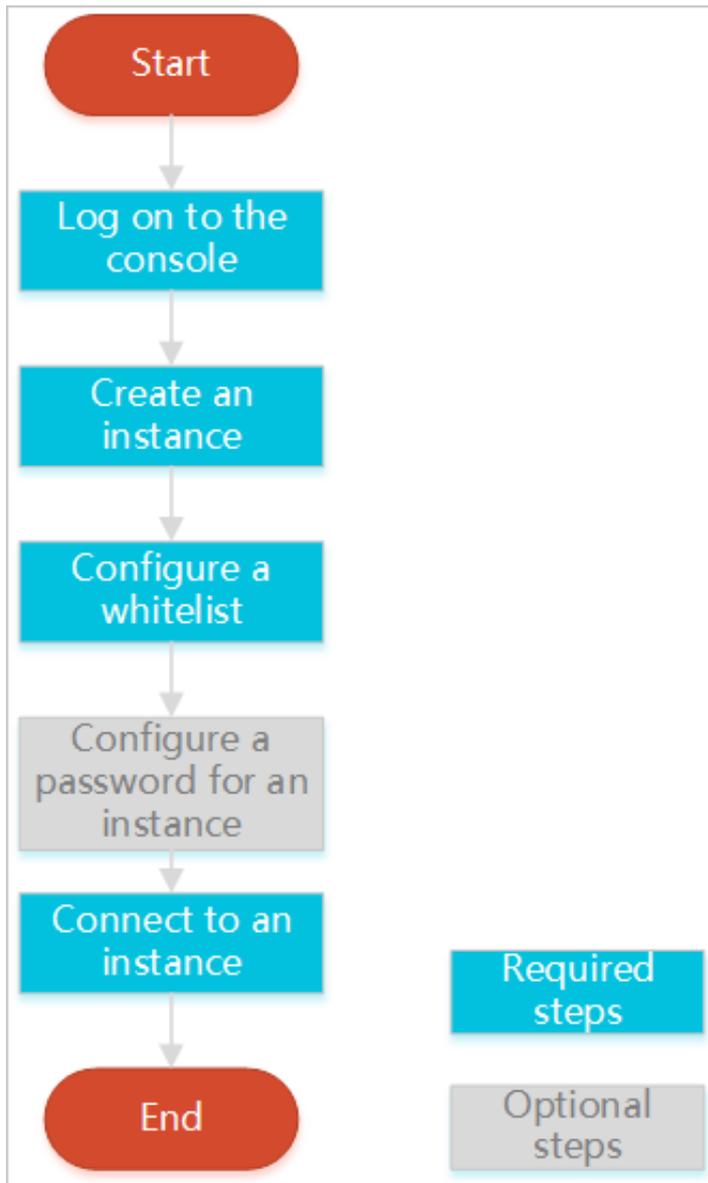
6.2 Quick start

6.2.1 Get started with KVStore for Redis

This topic describes a variety of operations from creating a KVStore for Redis instance to logging on to a database. This helps you understand the procedure of using the KVStore for Redis instance.

The flowchart of using the KVStore for Redis instance is as follows.

Figure 6-1: Flowchart for the KVStore for Redis instance



- [Log on to the KVStore for Redis console](#)

This topic describes how to log on to the KVStore for Redis console.

- [Create an instance](#)

KVStore for Redis supports two types of networks: classic network and Virtual Private Cloud (VPC). You can create KVStore for Redis instances of either network type.

- [Configure a whitelist](#)

To ensure database security and stability, before using a KVStore for Redis instance, you must add one or more IP addresses or CIDR blocks to the whitelist of the instance before you can connect to databases of the instance.

- [Set the password](#)

If you have not specified a password when creating the instance, set the password of the instance on the Instance Information page.

- [Connect to the instance](#)

You can use a client that supports Redis protocols or use the Redis command-line interface (redis-cli) program to connect to the KVStore for Redis instance.

6.2.2 Log on to the KVStore for Redis console

This topic describes how to log on to the KVStore for Redis console.

Prerequisites

- **Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel . The URL used to access the ASCM console is in the following format: `http://IP address or domain name of the ASCM console/manage`.**
- **We recommend that you use the Google Chrome browser.**

Procedure

1. **In the address bar, enter the URL used to access the ASCM console. Press Enter.**
2. **Enter your username and password.**

The system has a default super administrator, whose username is super. The super administrator can create system administrators. A system administrator can create system users and notify the users of the default passwords by SMS or email.



Note:

When you log on to the ASCM console for the first time, you must modify the password of your username as instructed. For security concerns, your password must meet the minimum complexity requirements: The password must be 8 to 20 characters in length and must contain at least two types of the following characters: letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs (\$), and percent signs (%).

3. Click Login to go to the ASCM console homepage.
4. In the top navigation bar, choose Products > Database Services > KVStore for Redis.

6.2.3 Create an instance

This topic describes how to create an instance in the KVStore for Redis console.

Procedure

1. [Log on to the KVStore for Redis console.](#)
2. Click Create Instance in the upper-right corner.
3. Set the following parameters:

Table 6-1: KVStore for Redis instance parameters

Section	Parameter	Description
Basic Settings	Organization	Specifies the project that the target KVStore for Redis instance belongs to.
	Resource Set	Specifies the zone where the target KVStore for Redis instance is located.  Notice: After a project is selected, the KVStore for Redis instance is accessible only to the members of the selected project.
Region	Region	Specifies the region where the target KVStore for Redis instance is located.
	Zone	Specifies the zone where the target KVStore for Redis instance is located.
Specifications	Engine Version	The following Redis versions are supported: <ul style="list-style-type: none"> • Redis 2.8 • Redis 4.0

Section	Parameter	Description
	Architecture Type	<p>Specifies an architecture type for the target KVStore for Redis instance.</p> <p>KVStore for Redis provides cluster and standard architectures. The cluster architecture supports large-capacity or high-performance requirements.</p>
	Node Type	<p>Specifies a node type for the target KVStore for Redis instance.</p> <p>KVStore for Redis supports the dual-copy structure.</p>
	Service Plan	Specifies a standard or premium plan.
	Instance Type	<p>The instance specifications.</p> <p>The maximum connections and maximum internal network bandwidth vary between different instance specifications.</p>
Network	Network Type	<p>KVStore for Redis only supports classic networks.</p> <p>Cloud services in a classic network are not isolated. Unauthorized access to a cloud service is blocked only by a security group or a whitelist policy of the service.</p>
Password	Instance Name	<p>Specifies the name of the target instance.</p> <ul style="list-style-type: none"> • The name must be 2 to 128 characters in length. • The name can contain letters, numbers, underscores (_), and hyphens (-). It must start with an uppercase or lowercase letter.
	Set Password	You can select Now or Later.
	Password	<p>Set a password used to access the instance.</p> <p>The password must follow these rules:</p> <ul style="list-style-type: none"> • It can be 8 to 30 characters in length. • It must contain uppercase letters, lowercase letters, and digits. Special characters are not supported.
	Confirm Password	Enter the password again.

4. After you set the parameters, click Submit.

6.2.4 Configure a whitelist

Before using a KVStore for Redis instance, you need to add the IP addresses or Classless Inter-Domain Routing (CIDR) blocks used to access to database to the instance whitelist to improve database security and stability.

Context



Note:

We recommend that you periodically check and adjust your whitelist to improve the access security protection and secure data in KVStore for Redis.

Procedure

1. *Log on to the KVStore for Redis console.*
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. In the left-side navigation pane of the Instance Information page, click Whitelist Settings.
4. On the Whitelist Settings page, proceed in either of the following ways:
 - To customize the whitelist group name, create a new whitelist group:
 - a. Click Add a Whitelist Group in the upper-right corner.
 - b. In the Add a Whitelist Group dialog box that appears, set Group Name.



Note:

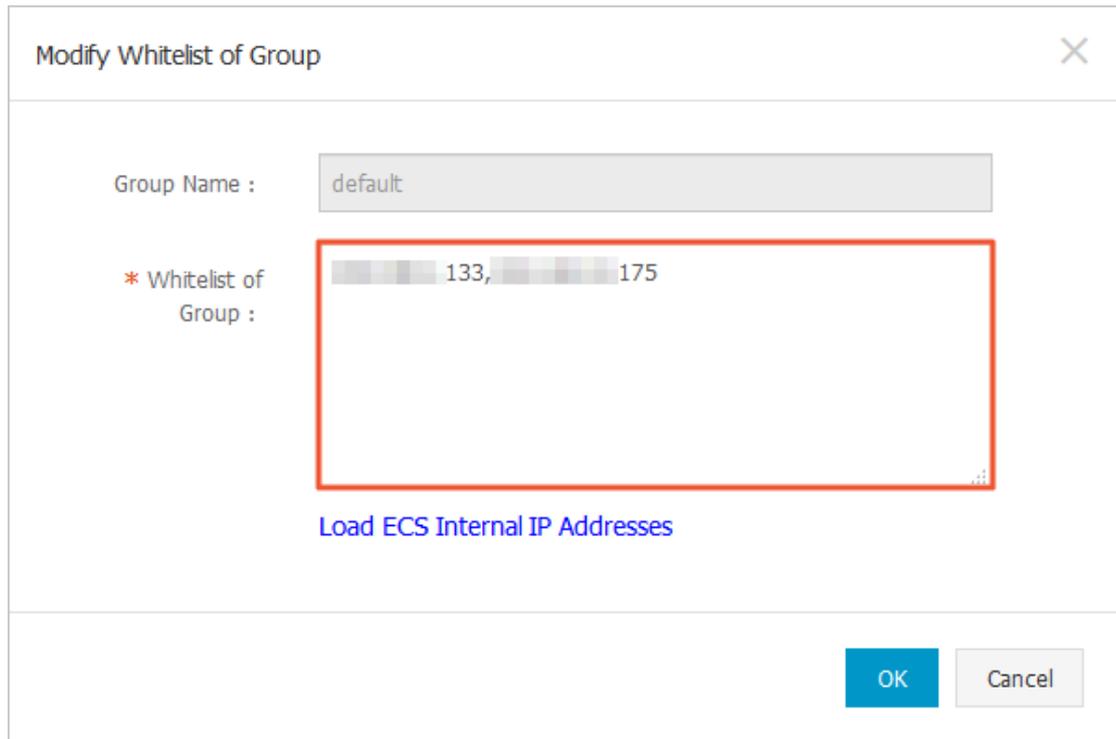
A group name must be 2 to 32 characters in length and contain lowercase letters, digits, or underscores (_). The group name must start with a lowercase letter and end with a letter or digit. You cannot change this name after you create the whitelist group.

- If you do not require a custom whitelist group, click Modify next to the target whitelist group.

5. In the Add a Whitelist Group or Modify Whitelist of Group dialog box that appears, proceed in either of the following ways:

- **Manually modify the Whitelist of Group field:**
 - a. In the Whitelist of Group field, enter the IP addresses or CIDR blocks that you can use to connect to the ApsaraDB for Redis instance.

Figure 6-2: Manually modify the whitelist group



Modify Whitelist of Group

Group Name : default

* Whitelist of Group : 133, 175

[Load ECS Internal IP Addresses](#)

OK Cancel



Note:

- Set the whitelist to `0.0.0.0/0` to allow connections from all IP addresses.
- Set the whitelist to `127.0.0.1` to block connections from all IP addresses.
- Set the whitelist to a CIDR block to allow connections from the IP addresses within the CIDR block, such as `10.10.10.0/24`.
- When you enter multiple IP addresses or CIDR blocks, separate them with commas (,) and leave no space before or after each comma.

- You can add 1,000 or fewer IP addresses or CIDR blocks to each whitelist group.

b. Click OK.

- Load internal IP addresses of target ECS instances under the current Alibaba Cloud account:

a. Click Load ECS Internal IP Addresses.

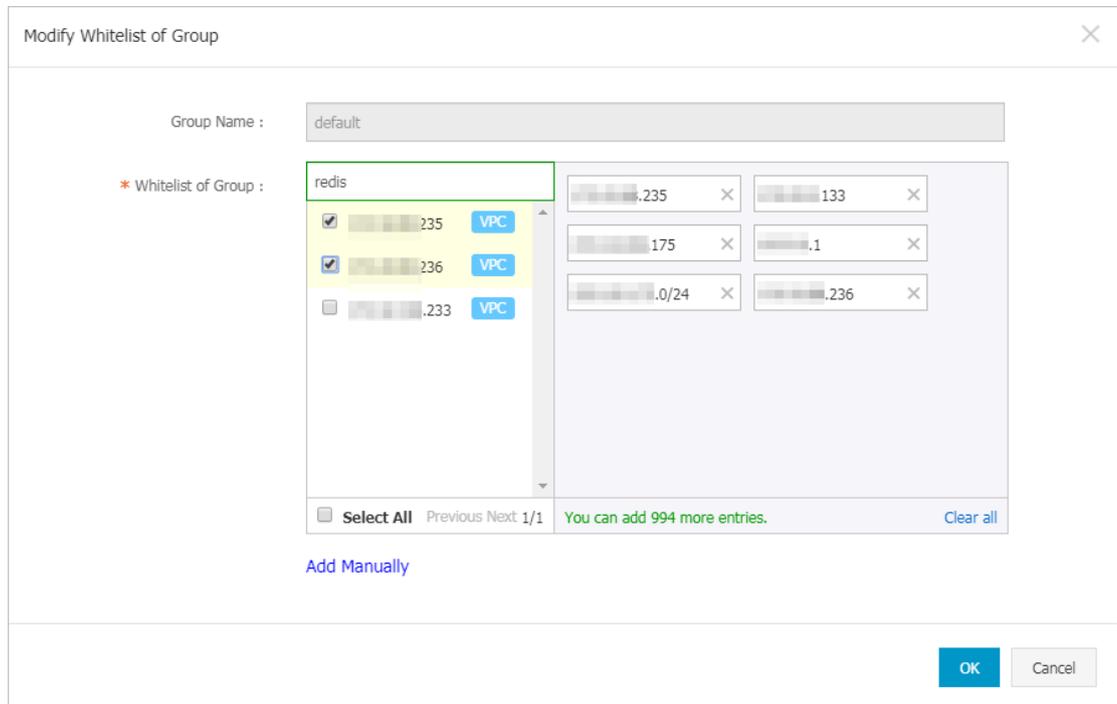
Figure 6-3: Load internal IP addresses of target ECS instances

The screenshot shows a dialog box titled "Modify Whitelist of Group". It contains the following elements:

- Group Name :** A text input field containing the value "default".
- * Whitelist of Group :** A text area containing the value "133, 175".
- Load ECS Internal IP Addresses :** A button with blue text, highlighted by a red rectangular box.
- OK :** A blue button.
- Cancel :** A grey button.

b. Select internal IP addresses of target ECS instances.

Figure 6-4: Select internal IP addresses of target ECS instances



Note:

You can perform a fuzzy search by ECS instance name, ID, or IP address on the search bar above the list of ECS internal IP addresses.

c. Click OK.

6.2.5 Connect to an instance

6.2.5.1 Use a Redis client

You can connect to the KVStore for Redis instance by using clients of several programming languages.

The database service of KVStore for Redis is fully compatible with Redis database service. Therefore, you can connect to both database services in similar ways. All clients that are compatible with Redis protocols support connections to KVStore for Redis. You can use any of these clients according to your application features.

For more information about Redis clients, visit <http://redis.io/clients>.

Prerequisites

- **The internal IP address of the ECS instance or the public IP address of the local host has been added to the whitelist of the KVStore for Redis instance. For more information about how to configure a whitelist, see [Configure a whitelist](#).**
- **If you use a custom account to connect to the KVStore for Redis instance, the connection password must be in the format of `<user>:<password>`. For example, if the username of a custom account is `admin` and the password is `password`, the password used to connect to the KVStore for Redis instance must be `admin:password`.**

Jedis client

You can use a Jedis client to connect to KVStore for Redis in any of the following ways:

- **Single Jedis connection**
- **JedisPool-based connection**

To use a Jedis client to connect to an KVStore for Redis instance, follow these steps:

1. **Download and install the Jedis client. For more information, see [Jedis](#).**
2. **Example of single Jedis connection**

- a. **Open the Eclipse client, create a project, and then enter the following code:**

```
import redis.clients.jedis.Jedis;
public class jedistest {
public static void main(String[] args) {
try {
String host = "xx.kvstore.aliyuncs.com";//You can view the
endpoint in the console.
int port = 6379;
Jedis jedis = new Jedis(host, port);
//Authentication information
jedis.auth("password");//password
String key = "redis";
String value = "aliyun-redis";
//Select a database. Default value: 0.
jedis.select(1);
//Set a key
jedis.set(key, value);
System.out.println("Set Key " + key + " Value: " + value);
//Obtain the configured key value.
String getvalue = jedis.get(key);
System.out.println("Get Key " + key + " ReturnValue: " +
getvalue);
jedis.quit();
jedis.close();
}
catch (Exception e) {
```

```
e.printStackTrace();
}
}
}
```

- b. Run the project. You have connected to KVStore for Redis if you view the following result in the Eclipse console.**

```
Set Key redis Value aliyun-redis
Get Key redis ReturnValue aliyun-redis
```

Afterward, you can use your local Jedis client to manage your KVStore for Redis instance. You can also connect to your KVStore for Redis instance by using JedisPool.

3. Example of JedisPool-based connection

- a. Open the Eclipse client, create a project, and then configure the pom file as follows:**

```
<dependency>
<groupId>redis.clients</groupId>
<artifactId>jedis</artifactId>
<version>2.7.2</version>
<type>jar</type>
<scope>compile</scope>
</dependency>
```

- b. Add the following application to the project:**

```
import org.apache.commons.pool2.PooledObject;
import org.apache.commons.pool2.PooledObjectFactory;
import org.apache.commons.pool2.impl.DefaultPooledObject;
import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
import redis.clients.jedis.HostAndPort;
import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisPool;
import redis.clients.jedis.JedisPoolConfig;
```

- c. If your Jedis client version is Jedis-2.7.2, enter the following code in the project:**

```
JedisPoolConfig config = new JedisPoolConfig();
//Maximum number of idle connections. You can customize this
parameter. Make sure that the specified maximum number of idle
connections does not exceed the maximum number of connections that
the KVStore for Redis instance supports.
config.setMaxIdle(200);
//Maximum number of connections. You can customize this parameter
. Make sure that the specified maximum number of connections does
not exceed the maximum number of connections that the KVStore for
Redis instance supports.
config.setMaxTotal(300);
config.setTestOnBorrow(false);
config.setTestOnReturn(false);
String host = "*.aliyuncs.com";
String password = "Password";
```

```

JedisPool pool = new JedisPool(config, host, 6379, 3000, password
);
Jedis jedis = null;
try {
jedis = pool.getResource();
/// ... do stuff here ... for example
jedis.set("foo", "bar");
String foobar = jedis.get("foo");
jedis.zadd("sose", 0, "car");
jedis.zadd("sose", 0, "bike");
Set<String> sose = jedis.zrange("sose", 0, -1);
} finally {
if (jedis != null) {
jedis.close();
}
}
/// ... when closing your application:
pool.destroy();

```

- d. If your Jedis client version is Jedis-2.6 or Jedis-2.5, enter the following code in the project:**

```

JedisPoolConfig config = new JedisPoolConfig();
//Maximum number of idle connections. You can customize this
parameter. Make sure that the specified maximum number of idle
connections does not exceed the maximum number of connections that
the KVStore for Redis instance supports.
config.setMaxIdle(200);
//Maximum number of connections. You can customize this parameter
. Make sure that the specified maximum number of connections does
not exceed the maximum number of connections that the KVStore for
Redis instance supports.
config.setMaxTotal(300);
config.setTestOnBorrow(false);
config.setTestOnReturn(false);
String host = "*.aliyuncs.com";
String password = "Password";
JedisPool pool = new JedisPool(config, host, 6379, 3000, password
);
Jedis jedis = null;
boolean broken = false;
try {
jedis = pool.getResource();
/// ... do stuff here ... for example
jedis.set("foo", "bar");
String foobar = jedis.get("foo");
jedis.zadd("sose", 0, "car");
jedis.zadd("sose", 0, "bike");
Set<String> sose = jedis.zrange("sose", 0, -1);
}
catch (Exception e)
{
broken = true;
} finally {
if (broken) {
pool.returnBrokenResource(jedis);
} else if (jedis != null) {
pool.returnResource(jedis);
}
}

```

```
}

```

- e. **Run the project. You have connected to KVStore for Redis if you view the following result in the Eclipse console.**

```
Set Key redis Value aliyun-redis
Get Key redis ReturnValue aliyun-redis

```

Afterward, you can use your local Jedis client to manage your KVStore for Redis instance.

phpredis client

To use a phpreidis client to connect to an KVStore for Redis instance, follow these steps:

- 1. Download and install the phpreidis client. For more information, see [phpredis](#).**
- 2. In any editor that supports PHP editing, enter the following code:**

```
<? php
/* Replace the following parameter values with the host name and
port number of the target instance. */
$host = "localhost";
$port = 6379;
/* Replace the following parameter values with the ID and password
of the target instance. */
$user = "test_username";
$pwd = "test_password";
$redis = new Redis();
if ($redis->connect($host, $port) == false) {
    die($redis->getLastError());
}
if ($redis->auth($pwd) == false) {
    die($redis->getLastError());
}
/* You can perform database operations after authentication. For
more information, visit https://github.com/phpredis/phpredis. */
if ($redis->set("foo", "bar") == false) {
    die($redis->getLastError());
}
$value = $redis->get("foo");
echo $value;
? >

```

- 3. Run the code. Afterward, you can use your local phpreidis client to connect to your KVStore for Redis instance. For more information, visit <https://github.com/phpredis/phpredis>.**

redis-py client

To use a redis-py client to connect to an KVStore for Redis instance, follow these steps:

- 1. Download and install the redis-py client. For more information, see [redis-py](#).**

2. In any editor that supports Python editing, enter the following code. Afterward, you can use the local redis-py client to connect to the KVStore for Redis instance and perform database operations.

```
#!/usr/bin/env python
#-*- coding: utf-8 -*-
import redis
#Replace the following parameter values with the host name and port
number of the target instance.
host = 'localhost'
port = 6379
#Replace the following parameter value with the password of the target
instance.
pwd = 'test_password'
r = redis.StrictRedis(host=host, port=port, password=pwd)
#You can perform database operations after you establish a connection
. For more information, visit https://github.com/andymccurdy/redis-py.
r.set('foo', 'bar');
print r.get('foo')
```

C or C++ client

- To use a C or C++ client to connect to an KVStore for Redis instance, follow these steps:**

1. Download, compile, and install the C client by using the following code:

```
git clone https://github.com/redis/hiredis.git
cd hiredis
make
sudo make install
```

2. Enter the following code in the C or C++ editor:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <hiredis.h>
int main(int argc, char **argv) {
    unsigned int j;
    redisContext *c;
    redisReply *reply;
    if (argc < 4) {
        printf("Usage: example xxx.kvstore.aliyuncs.com 6379
instance_id password\n");
        exit(0);
    }
    const char *hostname = argv[1];
    const int port = atoi(argv[2]);
    const char *instance_id = argv[3];
    const char *password = argv[4];
    struct timeval timeout = { 1, 500000 }; // 1.5 seconds
    c = redisConnectWithTimeout(hostname, port, timeout);
    if (c == NULL || c->err) {
        if (c) {
            printf("Connection error: %s\n", c->errstr);
            redisFree(c);
        } else {
```

```

        printf("Connection error: can't allocate redis context\
n");
    }
    exit(1);
}
/* AUTH */
reply = redisCommand(c, "AUTH %s", password);
printf("AUTH: %s\n", reply->str);
freeReplyObject(reply);
/* PING server */
reply = redisCommand(c,"PING");
printf("PING: %s\n", reply->str);
freeReplyObject(reply);
/* Set a key */
reply = redisCommand(c,"SET %s %s", "foo", "hello world");
printf("SET: %s\n", reply->str);
freeReplyObject(reply);
/* Set a key using binary safe API */
reply = redisCommand(c,"SET %b %b", "bar", (size_t) 3, "hello",
(size_t) 5);
printf("SET (binary API): %s\n", reply->str);
freeReplyObject(reply);
/* Try a GET and two INCR */
reply = redisCommand(c,"GET foo");
printf("GET foo: %s\n", reply->str);
freeReplyObject(reply);
reply = redisCommand(c,"INCR counter");
printf("INCR counter: %lld\n", reply->integer);
freeReplyObject(reply);
/* again ... */
reply = redisCommand(c,"INCR counter");
printf("INCR counter: %lld\n", reply->integer);
freeReplyObject(reply);
/* Create a list of numbers, from 0 to 9 */
reply = redisCommand(c,"DEL mylist");
freeReplyObject(reply);
for (j = 0; j < 10; j++) {
    char buf[64];
    snprintf(buf,64,"%d",j);
    reply = redisCommand(c,"LPUSH mylist element-%s", buf);
    freeReplyObject(reply);
}
/* Let's check what we have inside the list */
reply = redisCommand(c,"LRANGE mylist 0 -1");
if (reply->type == REDIS_REPLY_ARRAY) {
    for (j = 0; j < reply->elements; j++) {
        printf("%u) %s\n", j, reply->element[j]->str);
    }
}
freeReplyObject(reply);
/* Disconnects and frees the context */
redisFree(c);
return 0;

```

```
}

```

3. Compile the code.

```
gcc -o example -g example.c -I /usr/local/include/hiredis -lhiredis

```

4. Test the code.

```
example xxx.kvstore.aliyuncs.com 6379 instance_id password

```

Now, the C or C++ client is connected to the KVStore for Redis instance.

.NET client

To use a .NET client to connect to an KVStore for Redis instance, follow these steps:

1. Download and use the .NET client.

```
git clone https://github.com/ServiceStack/ServiceStack.Redis

```

2. Create a .NET project on the .NET client.

3. Add the reference file stored in the library file directory `ServiceStack.Redis/lib/tests` to the client.

4. Enter the following code in the .NET project to connect to the KVStore for Redis instance. For more information about API operations, visit <https://github.com/ServiceStack/ServiceStack.Redis>.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ServiceStack.Redis;
namespace ServiceStack.Redis.Tests
{
    class Program
    {
        public static void RedisClientTest()
        {
            string host = "127.0.0.1"; /*IP address of the host that you
            want to connect to*/
            string password = "password"; /*Password*/
            RedisClient redisClient = new RedisClient(host, 6379,
            password);
            string key = "test-aliyun";
            string value = "test-aliyun-value";
            redisClient.Set(key, value);
            string listKey = "test-aliyun-list";
            System.Console.WriteLine("set key " + key + " value " +
            value);
            string getValue = System.Text.Encoding.Default.GetString(
            redisClient.Get(key));
            System.Console.WriteLine("get key " + getValue);
            System.Console.Read();
        }
        public static void RedisPoolClientTest()
    }
}

```

```

{
    string[] testReadWriteHosts = new[] {
        "redis://password@127.0.0.1:6379"/*redis://Password@IP
address that you want to connect to:Port*/
    };
    RedisConfig.VerifyMasterConnections = false;//You must set the
parameter.
    PooledRedisClientManager redisPoolManager = new PooledRedi
sClientManager(10/*Number of connections in the pool*/, 10/*
Connection pool timeout value*/, testReadWriteHosts);
    for (int i = 0; i < 100; i++){
        IRedisClient redisClient = redisPoolManager.GetClient();//
Obtain the connection.
        RedisNativeClient redisNativeClient = (RedisNativeClient)
redisClient;
        redisNativeClient.Client = null; //KVStore for Redis does
not support the CLIENT SETNAME command. Set Client to null.
        try
        {
            string key = "test-aliyun1111";
            string value = "test-aliyun-value1111";
            redisClient.Set(key, value);
            string listKey = "test-aliyun-list";
            redisClient.AddItemToList(listKey, value);
            System.Console.WriteLine("set key " + key + " value " +
value);
            string getValue = redisClient.GetValue(key);
            System.Console.WriteLine("get key " + getValue);
            redisClient.Dispose();//
        }catch (Exception e)
        {
            System.Console.WriteLine(e.Message);
        }
        System.Console.Read();
    }
}
static void Main(string[] args)
{
    //Single-connection mode
    RedisClientTest();
    //Connection-pool mode
    RedisPoolClientTest();
}
}
}
}

```

node-redis client

To use a node-redis client to connect to an KVStore for Redis instance, follow these steps:

1. Download and install a node-redis client.

```
npm install hiredis redis
```

2. Enter and run the following code on the node-redis client to connect to the KVStore for Redis instance.

```
var redis = require("redis"),
```

```
client = redis.createClient(<port>, <"host">, {detect_buffers: true
});
client.auth("password", redis.print)
```

**Note:**

In the code, the port field specifies the port of the KVStore for Redis instance. Default value: 6379. The host field specifies the endpoint of the KVStore for Redis instance. The following example shows the settings of the port and host fields:

```
client = redis.createClient(6379, "r-abcdefg.redis.rds.aliyuncs.com", {detect_buffers: true});
```

3. Use the KVStore for Redis instance.

```
// Write data to the instance.
client.set("key", "OK");
// Query data on the instance. The instance returns data of String
type.
client.get("key", function (err, reply) {
  console.log(reply.toString()); // print `OK`
});
// If you specify a buffer, the instance returns a buffer.
client.get(new Buffer("key"), function (err, reply) {
  console.log(reply.toString()); // print `<Buffer 4f 4b>`
});
client.quit();
```

C# client StackExchange.Redis

To use the C# client StackExchange.Redis to connect to an KVStore for Redis instance, follow these steps:

1. Download and install [StackExchange.Redis](#).
2. Add a reference.

```
using StackExchange.Redis;
```

3. Initialize ConnectionMultiplexer.

ConnectionMultiplexer is the core of StackExchange.Redis, and shared in the entire application. You must use ConnectionMultiplexer as a singleton. ConnectionMultiplexer is initialized in the following way:

```
// redis config
private static ConfigurationOptions configurationOptions =
ConfigurationOptions.Parse("127.0.0.1:6379,password=xxx,connectTim
eout=2000");
//the lock for singleton
private static readonly object Locker = new object();
//singleton
private static ConnectionMultiplexer redisConn;
```

```
//singleton
public static ConnectionMultiplexer getRedisConn()
{
    if (redisConn == null)
    {
        lock (Locker)
        {
            if (redisConn == null || ! redisConn.IsConnected)
            {
                redisConn = ConnectionMultiplexer.Connect(
configurationOptions);
            }
        }
    }
    return redisConn;
}
}
```

**Note:**

ConfigurationOptions contains multiple options, such as keepAlive, connectRetry, and name. For more information, see StackExchange.Redis.ConfigurationOptions.

- 4. GetDatabase() returns a lightweight object. You can obtain this object from the object of ConnectionMultiplexer.**

```
redisConn = getRedisConn();
var db = redisConn.GetDatabase();
```

- 5. The following examples show five types of data structures. The API operations used in these examples are different from their usage in the native Redis service. These data structures include: string, hash, list, set, and sortedset.**

- **string**

```
//set get
string strKey = "hello";
string strValue = "world";
bool setResult = db.StringSet(strKey, strValue);
Console.WriteLine("set " + strKey + " " + strValue + ", result is "
+ setResult);
//incr
string counterKey = "counter";
long counterValue = db.StringIncrement(counterKey);
Console.WriteLine("incr " + counterKey + ", result is " +
counterValue);
//expire
db.KeyExpire(strKey, new TimeSpan(0, 0, 5));
Thread.Sleep(5 * 1000);
Console.WriteLine("expire " + strKey + ", after 5 seconds, value
is " + db.StringGet(strKey));
//mset mget
KeyValuePair<RedisKey, RedisValue> kv1 = new KeyValuePair<RedisKey
, RedisValue>("key1", "value1");
KeyValuePair<RedisKey, RedisValue> kv2 = new KeyValuePair<RedisKey
, RedisValue>("key2", "value2");
```

```
db.StringSet(new KeyValuePair<RedisKey, RedisValue>[] {kv1,kv2});

RedisValue[] values = db.StringGet(new RedisKey[] {kv1.Key, kv2.
Key});
Console.WriteLine("mget " + kv1.Key.ToString() + " " + kv2.Key.
ToString() + ", result is " + values[0] + "&&" + values[1]);
```

- **hash**

```
string hashKey = "myhash";
//hset
db.HashSet(hashKey,"f1","v1");
db.HashSet(hashKey,"f2", "v2");
HashEntry[] values = db.HashGetAll(hashKey);
//hgetall
Console.Write("hgetall " + hashKey + ", result is");
for (int i = 0; i < values.Length;i++)
{
    HashEntry hashEntry = values[i];
    Console.Write(" " + hashEntry.Name.ToString() + " " + hashEntry.
Value.ToString());
}
Console.WriteLine();
```

- **list**

```
//list key
string listKey = "myList";
//rpush
db.ListRightPush(listKey, "a");
db.ListRightPush(listKey, "b");
db.ListRightPush(listKey, "c");
//lrange
RedisValue[] values = db.ListRange(listKey, 0, -1);
Console.Write("lrange " + listKey + " 0 -1, result is ");
for (int i = 0; i < values.Length; i++)
{
    Console.Write(values[i] + " ");
}
Console.WriteLine();
```

- **set**

```
//set key
string setKey = "mySet";
//sadd
db.SetAdd(setKey, "a");
db.SetAdd(setKey, "b");
db.SetAdd(setKey, "c");
//sismember
bool isContains = db.SetContains(setKey, "a");
Console.WriteLine("set " + setKey + " contains a is " + isContains
);
```

- **sortedset**

```
string sortedSetKey = "myZset";
//sadd
db.SortedSetAdd(sortedSetKey, "xiaoming", 85);
db.SortedSetAdd(sortedSetKey, "xiaohong", 100);
db.SortedSetAdd(sortedSetKey, "xiaofei", 62);
db.SortedSetAdd(sortedSetKey, "xiaotang", 73);
```

```
//zrevrangebyscore
RedisValue[] names = db.SortedSetRangeByRank(sortedSetKey, 0, 2,
Order.Ascending);
Console.WriteLine("zrevrangebyscore " + sortedSetKey + " 0 2, result
is ");
for (int i = 0; i < names.Length; i++)
{
    Console.WriteLine(names[i] + " ");
}
Console.WriteLine();
```

6.2.5.2 Use redis-cli

You can use the Redis command-line interface (**redis-cli**) tool to connect to a KVStore for Redis instance.



Notice:

Because KVStore for Redis only supports connections over an internal network, only the ECS instances that run on the same node as KVStore for Redis and that are installed with **redis-cli** can connect to KVStore for Redis and perform data operations.

Install the **redis-cli** utility

Install the Redis software distribution that includes the **redis-cli** utility in Linux. For more information about the detailed procedure, see [Redis community](#).

Prerequisites

Connection over the internal network

- If the network types for the ECS and KVStore for Redis instances are both classic network, the two instances must reside in the same region.
- You have added the private IP address of an ECS instance to the whitelist of a KVStore for Redis instance.
- The operating system of the local host must be Linux.
- You have installed the Redis software distribution on the ECS instance.
- If you use a custom account to connect to the KVStore for Redis instance, the connection password must be in the format of `<user>:<password>`. For example, if the username of a custom account is `admin` and the password is `password`, the password used to connect to the KVStore for Redis instance must be `admin:password`.

Connection over the Internet

- The KVStore for Redis instance has a public connection string. For more information, see
- You have added the public IP address of the local host to the whitelist of the KVStore for Redis instance.
- The operating system of the local host must be Linux.
- You have installed the Redis software distribution on the local host.
- If you use a custom account to connect to the KVStore for Redis instance, the connection password must be in the format of `<user>:<password>`. For example, if the username of a custom account is `admin` and the password is `password`, the password used to connect to the KVStore for Redis instance must be `admin:password`.

Connect to a KVStore for Redis instance

You can use the following command to connect to a KVStore for Redis instance.

```
redis-cli -h <host> -p <port> -a <password>
```

Table 6-2: Parameters

Parameter	Description
-h	The connection string of the KVStore for Redis instance.
-p	The service port of the KVStore for Redis instance. The default port number is 6379 and cannot be changed.
-a	The password used to connect the KVStore for Redis instance. You can skip this parameter to avoid revealing the password over plaintext and enhance security. After running the preceding command, you can enter <code>auth <password></code> to complete the authentication. The following figure shows an example.

Figure 6-5: Command example

```
[root@ ~]# redis-cli -h r-bpl1xxxxxxxxxxxx.redis.rds.aliyuncs.com -p 6379
r-bpl1xxxxxxxxxxxx.redis.rds.aliyuncs.com:6379> auth admin:password
OK
r-bpl1xxxxxxxxxxxx.redis.rds.aliyuncs.com:6379>
```

6.3 Instance management

6.3.1 Change the password

If you forget your password, need to change your password, or have not set a password for an instance, you can set a new password for the instance.

Procedure

1. [Log on to the KVStore for Redis console.](#)
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. In the upper-right corner of the Basic Information page, click Modify Password.
4. In the Change Password dialog box that appears, set Old Password, New Password, Confirm Password.



Note:

- If you forget your password, you can click **Forgot password?** in the Change Password dialog box. In the Reset Password dialog box that appears, you can set a new password.
- The password must be 8 to 32 characters in length.
- The password must contain characters from at least three of the following categories: uppercase letters, lowercase letters, digits, and special characters. Special characters include ! @ # \$ % ^ & * () _ + - =

6.3.2 Configure a whitelist

Before using a KVStore for Redis instance, you need to add the IP addresses or Classless Inter-Domain Routing (CIDR) blocks used to access to database to the instance whitelist to improve database security and stability.

Context



Note:

We recommend that you periodically check and adjust your whitelist to improve the access security protection and secure data in KVStore for Redis.

Procedure

1. [Log on to the KVStore for Redis console.](#)
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.

3. In the left-side navigation pane of the Instance Information page, click **Whitelist Settings**.
4. On the **Whitelist Settings** page, proceed in either of the following ways:
 - To customize the whitelist group name, create a new whitelist group:
 - a. Click **Add a Whitelist Group** in the upper-right corner.
 - b. In the **Add a Whitelist Group** dialog box that appears, set **Group Name**.

**Note:**

A group name must be 2 to 32 characters in length and contain lowercase letters, digits, or underscores (_). The group name must start with a lowercase letter and end with a letter or digit. You cannot change this name after you create the whitelist group.

- If you do not require a custom whitelist group, click **Modify** next to the target whitelist group.

5. In the Add a Whitelist Group or Modify Whitelist of Group dialog box that appears, proceed in either of the following ways:

- **Manually modify the Whitelist of Group field:**
 - a. In the Whitelist of Group field, enter the IP addresses or CIDR blocks that you can use to connect to the ApsaraDB for Redis instance.

Figure 6-6: Manually modify the whitelist group

The screenshot shows a dialog box titled "Modify Whitelist of Group". It has a close button (X) in the top right corner. The "Group Name" field is a text box containing "default". Below it, the "Whitelist of Group" field is a larger text box containing "133, 175", which is highlighted with a red border. Below this field is a blue link that says "Load ECS Internal IP Addresses". At the bottom right of the dialog box are two buttons: "OK" (in a blue box) and "Cancel" (in a grey box).



Note:

- Set the whitelist to `0.0.0.0/0` to allow connections from all IP addresses.
- Set the whitelist to `127.0.0.1` to block connections from all IP addresses.
- Set the whitelist to a CIDR block to allow connections from the IP addresses within the CIDR block, such as `10.10.10.0/24`.
- When you enter multiple IP addresses or CIDR blocks, separate them with commas (,) and leave no space before or after each comma.

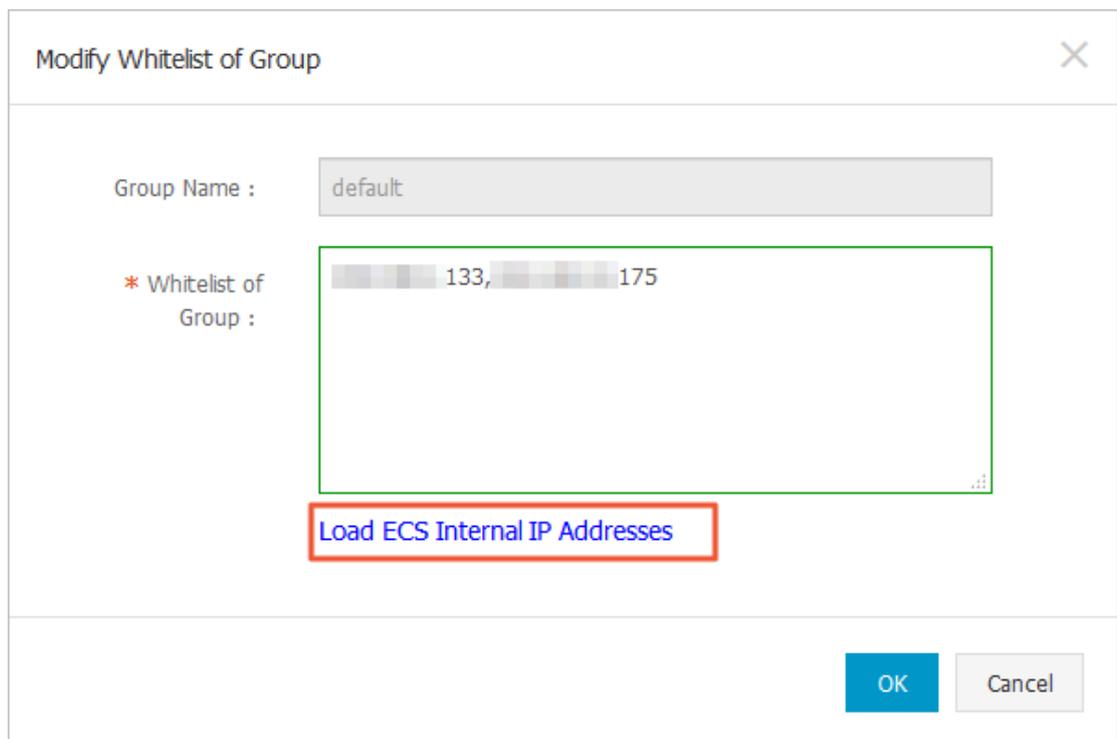
- You can add 1,000 or fewer IP addresses or CIDR blocks to each whitelist group.

b. Click OK.

- Load internal IP addresses of target ECS instances under the current Alibaba Cloud account:

a. Click Load ECS Internal IP Addresses.

Figure 6-7: Load internal IP addresses of target ECS instances



Modify Whitelist of Group

Group Name : default

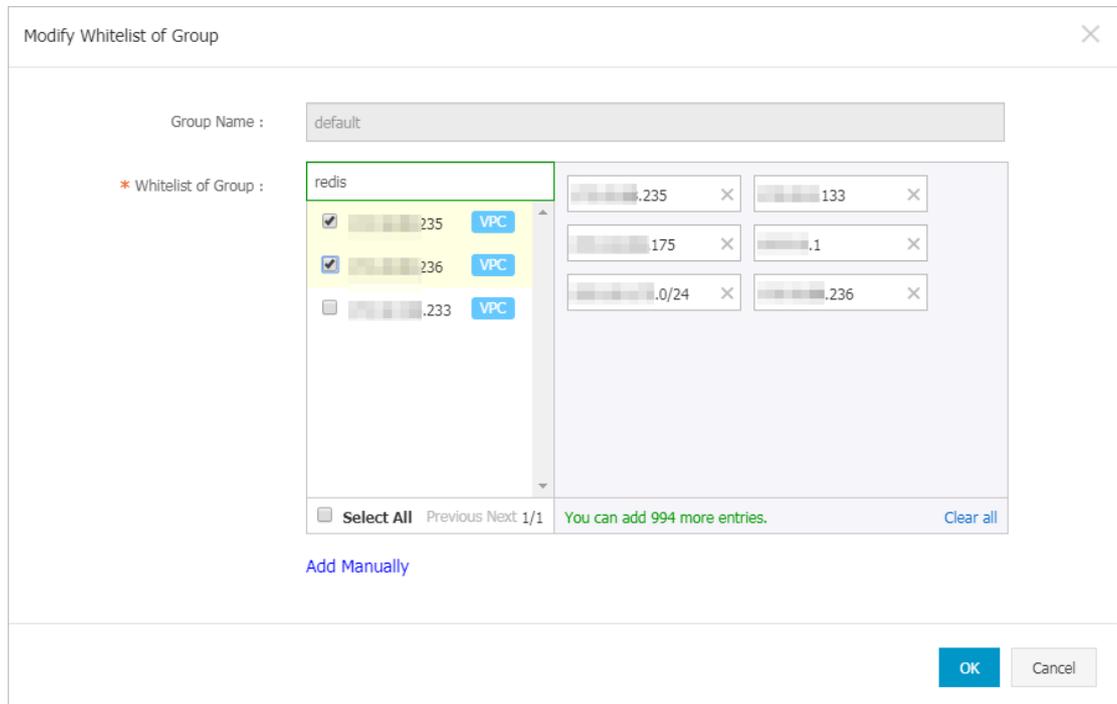
* Whitelist of Group : 133, 175

Load ECS Internal IP Addresses

OK Cancel

b. Select internal IP addresses of target ECS instances.

Figure 6-8: Select internal IP addresses of target ECS instances



Note:

You can perform a fuzzy search by ECS instance name, ID, or IP address on the search bar above the list of ECS internal IP addresses.

c. Click OK.

6.3.3 Change the configuration of an instance

This topic describes how to change the configuration of a KVStore for Redis instance.

Prerequisites

The instance is in the Running state.

Context



Note:

After configuration changes have been completed, the system will migrate data and experience transient disconnection for a few seconds during this process. We recommend that you upgrade or downgrade the instance configuration during off-peak hours.

Procedure

1. [Log on to the KVStore for Redis console](#).
2. On the Instance List page, find the target instance. Then, click the instance ID or click Change Configurations in the Actions column.
3. On the Modify Instance page, change the configuration and click Submit.

6.3.4 Set a maintenance window

You can modify the default maintenance window to perform maintenance on KVStore for Redis during off-peak hours.

Context

To ensure the stability of KVStore for Redis instances on the Alibaba Cloud platform, the backend system performs maintenance on instances and servers occasionally.

To guarantee the stability of the maintenance process, instances will enter the Maintaining Instance status before the preset maintenance window on the day of maintenance. While an instance is in this state, data in the database can still be accessed and query operations such as performance monitoring are still available. However, change operations such as configuration change are temporarily unavailable for this instance in the console.



Note:

During the maintenance process, instances may be disconnected in the process of maintenance. We recommend that you set the maintenance window to a period during off-peak hours.

Procedure

1. [Log on to the KVStore for Redis console](#).
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. On the Instance Information page, click Settings to the right of the Maintenance Window field in the Basic Information section.
4. Select time periods and click Save.



Note:

The time periods are in UTC+8.

6.3.5 Upgrade the minor version

Alibaba Cloud has continuously optimized the kernel of KVStore for Redis to fix security vulnerabilities and provide more stable services. You can upgrade the kernel version (minor version) of a KVStore for Redis instance with one click in the console.

Context



Note:

- We recommend that you upgrade instance versions during off-peak hours and ensure that your application supports automatic reconnection.
- The system automatically checks the kernel version of an instance. If the current version is the latest, the Minor Version Upgrade button in the upper-right corner of the Basic Information section for this instance will appear dimmed.

Procedure

1. [Log on to the KVStore for Redis console](#).
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. On the Instance Information page, click Minor Version Upgrade in the upper-right corner of the Basic Information section.
4. In the Minor Version Upgrade dialog box that appears, click Upgrade Now.

On the Instance Information page, the instance status will become Upgrading a minor version. When the instance status returns to Available, the upgrade has been completed.

6.3.6 Configure SSL encryption

The standard and cluster Instances of Redis 2.8 and the cluster instances of Redis 4.0 support secure sockets layer (SSL) encryption. You can enable SSL encryption to ensure more secure data transmission.

Context



Note:

SSL encryption may increase the network response time of instances. We recommend that you enable this feature only when necessary.

Procedure

1. [Log on to the KVStore for Redis console](#).
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. In the left-side navigation pane of the Instance Information page, click SSL Settings.
4. In the upper-right corner of the SSL Settings page, click Configure SSL.
5. In the Configure SSL dialog box, turn on the Enable switch, which becomes green. Click OK.
 - If an error message indicating that the instance is in an abnormal state is displayed, click OK in the message that appears.
 - If an error message indicates that the version does not support this feature, upgrade the minor version of the instance. For more information, see [Upgrade the minor version](#).
 - After the operation, you must wait a short period of time for the operation result to be displayed.
 - In the upper-right corner of the SSL Settings page, you can also click Update Validity and Download CA Certificate to perform relevant operations.

6.3.7 Clear instance data

You can clear the data of a KVStore for Redis instance in the console.

Context



Warning:

This operation will delete all data contained on an instance. Deleted data cannot be restored. Proceed with caution.

Procedure

1. [Log on to the KVStore for Redis console](#).
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. In the upper-right corner of the Instance Information page, click Clear Data.
4. In the Clear Data message that appears, click OK.

6.3.8 Release an instance

You can release a KVStore for Redis instance at any time based on your business needs. This topic describes how to release a KVStore for Redis instance.

Procedure

1. *Log on to the KVStore for Redis console.*
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. On the Instance List page, find the target instance. Then, click Release in the Actions column.



Warning:

After an instance is released, it cannot be restored. Proceed with caution. We recommend that you back up your data before releasing the instance.

4. In the Release Instance message that appears, click OK.

6.3.9 Manage database accounts

KVStore for Redis allows you to create multiple database accounts for an instance. You can grant permissions to these accounts based on the actual usage to flexibly manage your instance and minimize misoperation.

Prerequisites

The engine version of the instance is Redis 4.0 or later.



Note:

The engine version of the instance is not Redis 4.0, only the default account is available. The default account is created when you create the instance. For more information about how to change the password of the default account, see [Change the password](#).

Context

You can create accounts, delete accounts, reset the password, and change the permissions. After an account is created, you can use this account to log on to the database and use the command-line tool to perform operations on the database with the account and granted permissions.

Create an account

1. [Log on to the KVStore for Redis console.](#)
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. In the left-side navigation pane of the Instance Information page, click Account Management.

**Note:**

If Account Management is not available in the left-side navigation pane of the Instance Information page for an instance of Redis 4.0 or later, you can try to [Upgrade the minor version.](#)

4. In the upper-right corner of the Account Management page, click Create.
5. In the Create Account dialog box that appears, configure the following parameters and click OK.

Table 6-3: Set account parameters

Parameter	Description
Database Account	The name of the account. The account name can be up to 16 characters in length. It must start with a letter and can consist of lowercase letters, digits, and underscores (_).
Privilege	The permission granted to the account. Valid values: Read-only, Read/Write, and Replicate. If you select Replicate, you can run a SYNC or PSYNC command after connecting to the target instance with this account. <div data-bbox="470 1507 537 1574" data-label="Image"> </div> <div data-bbox="550 1534 639 1570" data-label="Section-Header">Note:</div> <div data-bbox="464 1572 1437 1695" data-label="Text"> <p>Currently, you can create accounts that have the Replicate permission only for instances of the standard edition in Redis 4.0 and 5.0.</p> </div>

Parameter	Description
Password Settings	The password of the account. The password must be 8 to 30 characters in length. It must consist of any three types of characters, including uppercase letters, lowercase letters, digits, and special characters. Special characters include exclamation points (!), at signs (@), number signs (#), dollar signs (\$), percent signs (%), carets (^), ampersands (&), asterisks (*), parentheses (()), underscores (_), plus signs (+), hyphens (-), and equal signs (=).
Confirm Password	The same password that you enter to confirm the password.

6.3.10 Use Lua scripts

KVStore for Redis instances of all editions support Lua commands.

Support for Lua commands

Lua scripts improve the performance of KVStore for Redis. With support for the Lua environment, KVStore for Redis is able to perform check-and-set (CAS) operations, allowing you to combine and run multiple commands in an efficient manner.



Note:

If an EVAL command fails to run (for example, the "ERR command eval not support for normal user" message is returned), you can try to [Upgrade the minor version](#). During the upgrade, the instance may be disconnected and become read-only for a few seconds. We recommend that you upgrade instance versions during off-peak hours.

Limits on Lua scripts

To ensure that all operations in a Lua script are performed within the same hash slot, the cluster edition of KVStore for Redis sets the following limits on a Lua script :

- The Lua script uses the `redis.call/redis.pcall` function to call Redis commands. For these commands, all the keys must be passed by using the KEYS

array, which cannot be replaced by Lua variables. Otherwise, the following error message is returned:

```
-ERR bad lua script for redis cluster, all the keys that the script uses should be passed using the KEYS arrayrn
```

- **All the keys that the script uses must be allocated in the same hash slot.**

Otherwise, the following error message is returned:

```
-ERR eval/evalsha command keys must be in same slotrn
```

- **Keys must be passed for all the commands to be called. Otherwise, the following error message is returned:**

```
-ERR for redis cluster, eval/evalsha number of keys can't be negative or zerorn
```



Note:

If you want to break the Lua limits of Redis Cluster and can ensure that all operations are performed in the same hash slot in the code, you can set the `script_check_enable` parameter to 0 in the console to disable the backend script check.

6.3.11 Restart an instance

You can restart an instance from the Instance List page of the console.

Procedure

1. [Log on to the KVStore for Redis console.](#)
2. On the Instance List page, find the target instance. Then, click Restart in the Actions column.



Notice:

During the restart, the instance may be disconnected for a few seconds. We recommend that you restart instances during off-peak hours and ensure that your application supports automatic reconnection.

3. In the dialog box that appears, select the restart time and click OK.
 - **Restart Immediately:** The instance restarts immediately.
 - **Restart Within Maintenance Window:** The instance restarts during the preset [maintenance window](#).

6.3.12 Export the instance list

You can export the list of KVStore for Redis instances from the KVStore for Redis console for offline management.

Procedure

1. *Log on to the KVStore for Redis console.*
2. In the upper-right corner of the Instance List page, click the Export Instances icon.
3. In the Export Instance List dialog box that appears, select the columns to export and click OK.



Note:

After you click OK, the browser begins to download the CSV file. You can use Excel or a text editor to view this file.

6.4 Connection management

6.4.1 View endpoints

You can view the internal and public endpoints of instances in the KVStore for Redis console.

Context



Note:

- The virtual IP address of the KVStore for Redis instance may change when you maintain or modify a service. To ensure connection availability, we recommend that you use an endpoint to access the KVStore for Redis instance.
- For more information about how to apply for a public endpoint, see [Apply for a public endpoint](#).

Procedure

1. *Log on to the KVStore for Redis console.*
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. On the Instance Information page, view Internal Connection Address (Host) and Public Endpoint (Host) in the Connection Information section.

6.4.2 Apply for a public endpoint

This topic describes how to apply for a public endpoint.

Procedure

1. [Log on to the KVStore for Redis console](#).
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. On the Instance Information page, click Apply for External IP Address in the Connection Information section.
4. In the Apply for External IP Address dialog box that appears, enter an endpoint and port number, and click OK.



Note:

- The custom endpoint prefix must be 8 to 64 characters in length and can contain lowercase letters and digits. It must start with a lowercase letter.
- The custom port range is 1024 to 65535. The default value is 6379.
- After applying for a public endpoint, you must add the public IP address to the whitelist before the instance can be accessed over the Internet. For more information about how to configure a whitelist, see [Configure a whitelist](#).

5. On the Instance Information page, view the Public Endpoint in the Connection Information section.



Note:

If a public endpoint is no longer needed, you can click Release Public Endpoint next to the Public Endpoint to release the endpoint.

6.4.3 Modify an endpoint

KVStore for Redis allows you to modify internal and public endpoints for instances. When changing the KVStore for Redis instance, you can change the endpoint of the new instance to the endpoint of the original instance without modifying the application.

Prerequisites

The instance is in the Running state.

Procedure

1. *Log on to the KVStore for Redis console.*
2. **On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.**
3. **On the Instance Information page, click Modify Public Endpoint in the Connection Information section.**
4. **In the Modify Public Endpoint dialog box that appears, set Connection Type, Endpoint, and Port. Click OK.**

**Note:**

- **The custom endpoint prefix must be 8 to 64 characters in length and can contain lowercase letters and digits. It must start with a lowercase letter.**
- **The custom port range is 1024 to 65535. The default value is 6379.**
- **If Connection Type is set to Internal Address, you cannot set Port.**

6.5 System parameters

KVStore for Redis allows you to customize certain instance parameters. This topic describes parameters and the common methods to modify them in the KVStore for Redis console.

Context

KVStore for Redis is completely compatible with the native database services of Redis. The method to set parameters for KVStore for Redis is similar to that of an on-premises Redis database. You can set the parameters described in this topic in the KVStore for Redis console.

Parameters

Table 6-4: Parameters

Parameter	Description
#no_loose_check-whitelist-always	Specifies whether to check whether the client IP address is in the whitelist of the KVStore for Redis instance after password-free access is enabled in Virtual Private Cloud (VPC). Default value: no. If you set this parameter to yes, the whitelist will still take effect in password-free access mode for VPC. Valid values: <ul style="list-style-type: none"> • yes • no
#no_loose_disabled-commands	Specifies the disabled commands. Separate multiple commands with commas (.). You can disable the following commands: FLUSHALL, FLUSHDB, KEYS, HGETALL, EVAL, EVALSHA, and SCRIPT.
#no_loose_ssl-enabled	Specifies whether to enable SSL encryption. Default value: no. Valid values: <ul style="list-style-type: none"> • yes • no
#no_loose_sentinel-enabled	Specifies whether to enable Sentinel-compatible mode. Default value: no. Valid values: <ul style="list-style-type: none"> • yes • no
client-output-buffer-limit pubsub	Limits the size of output buffers for Pub/Sub clients. This parameter can contain options in the following format: <hard limit> <soft limit> <soft seconds>. <ul style="list-style-type: none"> • Hard limit: If the output buffer of a Pub/Sub client reaches or exceeds the number of bytes specified by hard limit, the client is immediately disconnected. • soft limit and soft seconds: If the output buffer of a Pub/Sub client reaches or exceeds the size in bytes specified by soft limit for a period of time in seconds specified by soft seconds, the client will be disconnected.

Parameter	Description
dynamic-hz	<p>Specifies whether to enable dynamic frequency control for background tasks. Default value: yes. Valid values:</p> <ul style="list-style-type: none"> • yes • no
hash-max-ziplist-entries	<p>Specifies the maximum size of each key-value pair stored within a hash in bytes. A hash is encoded using ziplist when it meets the following conditions:</p> <ol style="list-style-type: none"> 1. The maximum size of each key-value pair stored within the hash in bytes must be less than the value of the hash-max-ziplist-value parameter. 2. The number of key-value pairs stored within the hash must be less than the value of the hash-max-ziplist-entries parameter.
hash-max-ziplist-value	<p>Specifies the maximum size of each key-value pair stored within a hash in bytes. A hash is encoded using ziplist when it meets the following conditions:</p> <ol style="list-style-type: none"> 1. The maximum size of each key-value pair stored within the hash in bytes must be less than the value of the hash-max-ziplist-value parameter. 2. The number of key-value pairs stored within the hash must be less than the value of the hash-max-ziplist-entries parameter.
hz	<p>Specifies the execution frequency for background tasks, such as tasks to evict expired keys. Valid values : 1 to 500. Default value: 10. The larger the value of the hz parameter, the more frequently background tasks are performed and the more precisely timeout events are handled, but the more CPU KVStore for Redis consumes. We recommend that you do not set the hz parameter to a value greater than 100.</p>
lazyfree-lazy-eviction	<p>Specifies whether to enable lazyfree for the eviction feature. Default value: no. Valid values:</p> <ul style="list-style-type: none"> • yes • no

Parameter	Description
lazyfree-lazy-expire	<p>Specifies whether to enable lazyfree to delete expired keys. Default value: yes. Valid values:</p> <ul style="list-style-type: none"> • yes • no
lazyfree-lazy-server-del	<p>Specifies whether to enable lazyfree to asynchronously delete data with the DEL command. Default value: yes. Valid values:</p> <ul style="list-style-type: none"> • yes • no
list-compress-depth	<p>Specifies the number of nodes that are not compressed at each side in a list. Default value: 0. Valid values:</p> <ul style="list-style-type: none"> • 0: does not compress any nodes in the list. • 1: does not compress the first node from each side of the list, but compresses all nodes in between. • 2: does not compress the first two nodes from each side of the list, but compresses all nodes in between • • 3: does not compress the first three nodes from each side of the list, but compresses all nodes in between. • And so on up to 65535.

Parameter	Description
list-max-ziplist-size	<ul style="list-style-type: none">• Specifies the maximum size of each ziplist in a quicklist. A positive number indicates the maximum number of elements in each ziplist of a quicklist. For example, if you set this parameter to 5, each ziplist of a quicklist can contain a maximum of five elements.• A negative number indicates the maximum number of bytes in each ziplist of a quicklist. Default value: -2. Valid values:<ul style="list-style-type: none">- -5: indicates that each ziplist of a quicklist cannot exceed 64 KB (1 KB = 1,024 bytes).- -4: indicates that each ziplist of a quicklist cannot exceed 32 KB.- -3: indicates that each ziplist of a quicklist cannot exceed 16 KB.- -2: indicates that each ziplist of a quicklist cannot exceed 8 KB.- -1: indicates that each ziplist of a quicklist cannot exceed 4 KB.

Parameter	Description
maxmemory-policy	<p>Specifies the policy used to evict keys if the memory is fully occupied. Valid values: LRU means least recently used. LFU means least frequently used. LRU, LFU, and TTL are implemented by using approximated randomized algorithms.</p> <ul style="list-style-type: none">• volatile-lru: evicts the approximated least recently used (LRU) keys among keys with a preset expiration time.• allkeys-lru: evicts the approximated LRU keys.• volatile-lfu: evicts the approximated least frequently used (LFU) keys among keys with a preset expiration time.• allkeys-lfu: evicts the approximated LFU keys.• volatile-random: evicts random keys among keys with a preset expiration time.• allkeys-random: evicts random keys.• volatile-ttl: evicts keys with the nearest time to live (TTL) among keys with a preset expiration time.• noeviction: does not evict any keys, but returns an error on write operations.

Parameter	Description
notify-keyspace-events	<p>Specifies the events that the Redis server can notify clients of. The value of this parameter is any combination of the following characters, each of which specifies a type of event to be notified:</p> <ul style="list-style-type: none"> • • K: keyspace events, published with the <code>__keyspace@<db>__</code> prefix. • E: keyevent events, published with the <code>__keyevent@<db>__</code> prefix. • g: generic commands that are non-type specific, such as <code>DEL</code>, <code>EXPIRE</code>, and <code>RENAME</code>. • l: list commands. • s: set commands. • h: hash commands. • z: sorted set commands. • x: expired key events. An expired key event is generated when a key expires. • e: evicted key events. An evicted key event is generated when a key is evicted due to the policy specified by the <code>maxmemory-policy</code> parameter. • A: the alias for <code>g\$lshzxe</code>.
set-max-intset-entries	<p>Specifies the maximum number of data entries in a set. A set is encoded by using intset when it meets the following conditions:</p> <ol style="list-style-type: none"> 1. The set is composed of just strings. The number of strings is less than the value of this parameter. 2. All strings are integers in radix 10 in the range of 64-bit signed integers.
slowlog-log-slower-than	<p>Specifies whether to log slow queries.</p> <ul style="list-style-type: none"> • Negative number: does not log slow queries. • 0: logs all queries. • Positive number: logs queries that exceed an execution time specified by this positive number, in microseconds. <p>Valid values: 0 to 10,000,000. Default value: 10,000.</p>

Parameter	Description
slowlog-max-len	Specifies the maximum number of slow query log entries that can be stored. Valid values: 100 to 10,000. Default value: 1,024.
stream-node-max-bytes	Specifies the maximum memory that can be used by each macro node in streams. Valid values: 0 to 999,999,999,999,999,999. If you set the parameter to 0, each macro node can use an unlimited amount of memory.
stream-node-max-entries	Specifies the maximum number of stream entries that can be stored within each macro node. Valid values: 0 to 999,999,999,999,999. If you set the parameter to 0, each macro node can store unlimited stream entries.
timeout	Specifies a timeout period for client connections. Unit : seconds. Valid values: 0 to 100,000. 0 indicates that client connections never time out.
zset-max-ziplist-entries	<p>Specifies the maximum size of each key-value pair stored within a sorted set in bytes. A sorted set is encoded using ziplist when it meets the following conditions:</p> <ol style="list-style-type: none"> 1. The maximum size of each key-value pair stored within the sorted set in bytes must be less than the value of the <code>zset-max-ziplist-value</code> parameter. 2. The number of key-value pairs stored within the sorted set must be less than the value of the <code>zset-max-ziplist-entries</code> parameter.
zset-max-ziplist-value	<p>Specifies the maximum size of each key-value pair stored within a sorted set in bytes. A sorted set is encoded using ziplist when it meets the following conditions:</p> <ol style="list-style-type: none"> 1. The maximum size of each key-value pair stored within the sorted set in bytes must be less than the value of the <code>zset-max-ziplist-value</code> parameter. 2. The number of key-value pairs stored within the sorted set must be less than the value of the <code>zset-max-ziplist-entries</code> parameter.

Parameter	Description
list-max-ziplist-entries	<p>Specifies the maximum size of each key-value pair stored within a list in bytes. A list is encoded using ziplist when it meets the following conditions:</p> <ol style="list-style-type: none"> 1. The maximum size of each key-value pair stored within the list in bytes must be less than the value of the list-max-ziplist-value parameter. 2. The number of elements stored within the list is less than the value of the list-max-ziplist-entries parameter.
list-max-ziplist-value	<p>Specifies the maximum size of each key-value pair stored within a list in bytes. A list is encoded using ziplist when it meets the following conditions:</p> <ol style="list-style-type: none"> 1. The maximum size of each key-value pair stored within the list in bytes must be less than the value of the list-max-ziplist-value parameter. 2. The number of elements stored within the list is less than the value of the list-max-ziplist-entries parameter.
cluster_compat_enable	<p>Specifies whether to enable compatibility with the syntax of Redis Cluster. Default value: 1. Valid values:</p> <ul style="list-style-type: none"> • 0: no • 1: yes
script_check_enable	<p>Specifies whether to confirm that all the keys used in a Lua script are in the same hash slot. Default value: 1. Valid values:</p> <ul style="list-style-type: none"> • 0: no • 1: yes

**Note:**

The maxclients parameter, which is used to specify the maximum number of connections to Redis data nodes, is fixed to 10,000. You cannot modify the value of this parameter.

Modify parameters in the KVStore for Redis console

1. [Log on to the KVStore for Redis console.](#)

2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. In the left-side navigation pane of the Instance Information page, click System Parameters.
4. Find the target parameter and click Modify in the Action column.
5. In the dialog box that appears, modify the parameter value and click OK.

6.6 Backup and recovery

As more businesses use Redis as their primary persistent storage engine, users will require higher reliability for their data. KVStore for Redis has optimized its data reliability based on Redis backup and recovery solutions.

Back up data automatically through a backup policy

As an increasing number of applications use Redis for persistent storage, conventional backup is required to quickly restore data in the event of misoperation. KVStore for Redis uses RDB snapshots to back up data on secondary nodes. Backup processes will not impact the performance of your instances. You can also conveniently customize backup policies in the console.

1. [Log on to the KVStore for Redis console.](#)
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. In the left-side navigation pane of the Instance Information page, click Backup and Recovery.
4. On the Backup and Recovery page, click the Backup Settings tab.
5. Click Edit to customize the automatic backup cycle and backup time.



Note:

Backup data is retained for seven days. You cannot modify this configuration.

6. Click OK.

Back up data manually

In addition to automatic backup, you can also manually back up data at any time in the console.

1. [Log on to the KVStore for Redis console.](#)
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. In the left-side navigation pane of the Instance Information page, click Backup and Recovery.
4. In the upper-right corner of the Data Backup tab, click Create Backup.
5. In the message that appears, click Confirm.

**Note:**

On the Data Backup tab, you can select a time range to query historical backup data. Backup data is retained for seven days, so you can query historical backup data in the past seven days.

Download data backup files

To archive data backup files for a longer period, you can copy the link in the console and manually download data backup files for local storage.

1. [Log on to the KVStore for Redis console.](#)
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. In the left-side navigation pane of the Instance Information page, click Backup and Recovery.
4. On the Data Backup tab, select the backup set to be archived and click Download.

Restore data

You can restore data from backup files in the console..

**Notice:**

- The data restoration operation is highly risky. Verify the data to be restored before performing this operation. Proceed with caution.
- Data restoration in the console is not applicable to cluster instances.

1. [Log on to the KVStore for Redis console.](#)
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.

3. In the left-side navigation pane of the Instance Information page, click Backup and Recovery.
4. On the Data Backup tab, select the target backup file and click Restore Data.
5. In the Data Recovery dialog box that appears, click Continue.

6.7 Performance monitoring

6.7.1 View monitoring data

You can query the monitoring data of a KVStore for Redis instance for a specified period within the last month.

Procedure

1. *Log on to the KVStore for Redis console.*
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. In the left-side navigation pane of the Instance Information page, click Performance Monitor.
4. On the Historical Monitoring Data page, click the calendar icon next to Query Time.
5. Set a query time range and click OK.



Note:

For more information about metrics, see [Metrics](#).

6.7.2 Customize metrics

You can select the metrics to be displayed on the Historical Monitoring Data page of the KVStore for Redis console as needed.

Context

KVStore for Redis supports more than 10 groups of monitoring metrics. By default, the Performance Monitor page displays the monitoring metrics of the basic monitoring group. You can click the Custom Metrics button to switch to the metrics of other monitoring groups. The following table describes the monitoring groups.

Monitoring group	Description
Basic monitoring group	The basic instance monitoring information, such as the QPS, bandwidth, and memory usage.
Key monitoring group	The monitoring information on the use of key-value related commands, such as the number of times DEL and EXITS are called.
String monitoring group	The monitoring information on the use of string-related commands, such as the number of times APPEND and MGET are called.
Hash monitoring group	The monitoring information on the use of hash-related commands, such as the number of times HGET and HDEL are called.
List monitoring group	The monitoring information on the use of list-related commands, such as the number of times BLPOP and BRPOP are called.
Set monitoring group	The monitoring information on the use of set-related commands, such as the number of times SADD and SCARD are called.
Zset monitoring group	The monitoring information on the use of zset-related commands, such as the number of times ZADD and ZCARD are called.
HyperLog monitoring group	The monitoring information on the use of HyperLogLog-related commands, such as the number of times PFADD and PFCOUNT are called.
Pub/Sub monitoring group	The monitoring information on the use of publication and subscription-related commands, such as the number of times PUBLISH and SUBSCRIBE are called.
Transaction monitoring group	The monitoring information on the use of transaction-related commands, such as the number of times WATCH, MULTI, and EXEC are called.
Lua script monitoring group	The monitoring information on the use of Lua script-related commands, such as the number of times EVAL and SCRIPT are called.

Procedure

1. [Log on to the KVStore for Redis console.](#)

2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. In the left-side navigation pane of the Instance Information page, click Performance Monitor.
4. On the Historical Monitoring Data page, click Customize Metrics in the Data Index section.
5. In the Customize Metrics dialog box that appears, select the new monitoring group and click OK.

6.7.3 Modify the monitoring frequency

KVStore for Redis console allows you to set the frequency at which monitoring data is collected.

Context

You can set the monitoring frequency to either 5 or 60 seconds to specify how often monitoring data to be collected by KVStore for Redis. The default monitoring time of 60 seconds is sufficient to meet common monitoring requirements. If you need to observe certain metrics at a higher frequency and lower latency, you can change the monitoring frequency to 5 seconds as described in the following section. Monitoring data does not occupy instance storage space, and collection of monitoring data does not affect normal running of the instance.

Procedure

1. [Log on to the KVStore for Redis console](#).
2. On the Instance List page, find the target instance. Then, click the instance ID or click Manage in the Actions column.
3. In the left-side navigation pane of the Instance Information page, click Performance Monitor.
4. In the upper-right corner of the Historical Monitoring Data page, click Monitoring Frequency.
5. In the Monitoring Frequency dialog box that appears, select the new monitoring frequency and click OK.

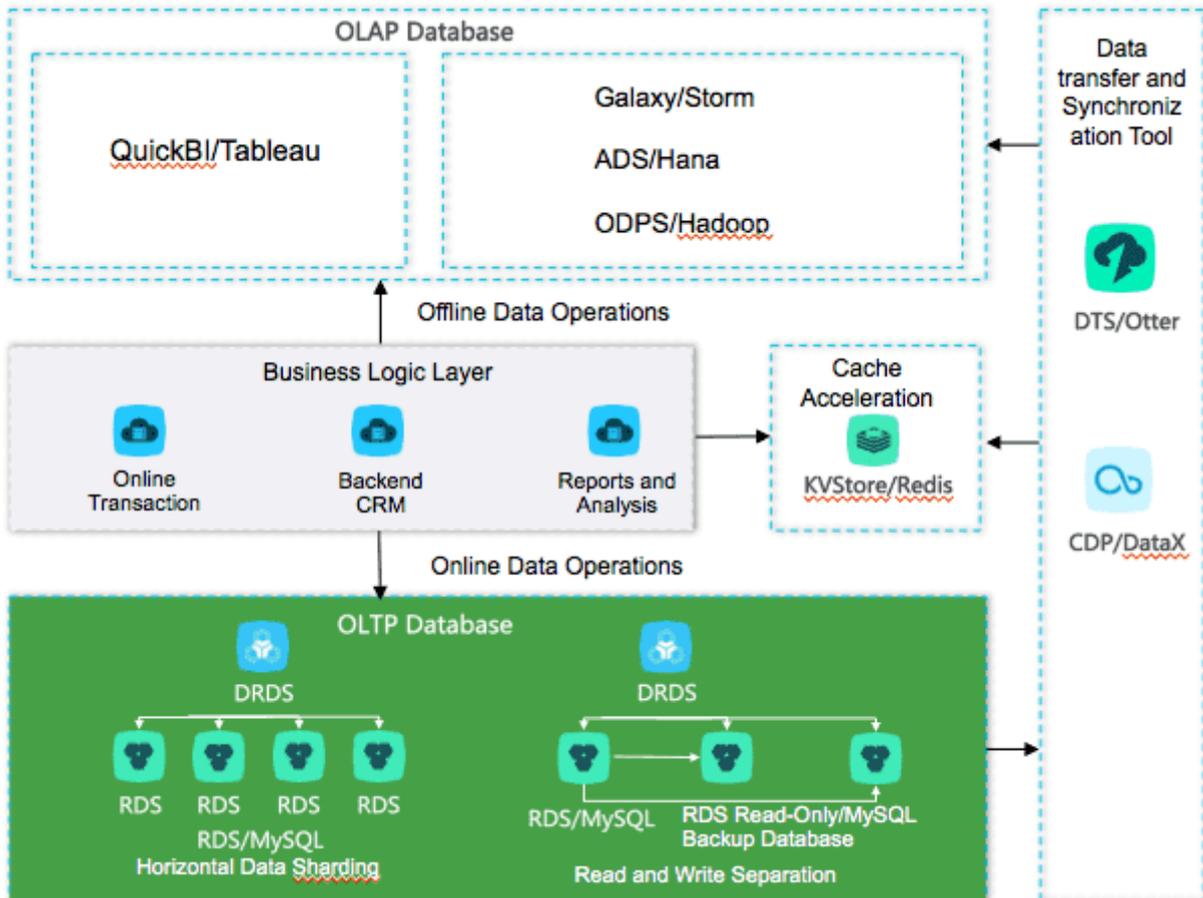
7 Distributed Relational Database Service (DRDS)

7.1 What is DRDS?

Distributed Relational Database Service (DRDS) is a middleware product independently developed by Alibaba Group to solve scaling problems of single-instance relational databases. DRDS is compatible with the MySQL protocol and supports most MySQL data manipulation language (DML) and data definition language (DDL) syntax. It has the core capabilities and features of distributed databases, such as database sharding, table sharding, smooth scale-out, configuration upgrade and downgrade, and transparent read/write splitting. DRDS features lightweight (stateless), flexibility, stability, and efficiency, and provides you with O&M capabilities throughout the lifecycle of distributed databases.

It is mainly used for operations on large-scale online data. By splitting data in specific business scenarios, DRDS maximizes the operation efficiency, meeting the requirements of online businesses on relational databases.

Figure 7-1: DRDS architecture



Problems solved

- **Capacity bottleneck of single-instance databases:** As the data volume and access volume increase, traditional single-instance databases encounter great challenges that cannot be completely solved by hardware upgrades. Distributed solutions use multiple instances to work jointly, effectively resolving the data storage capacity and access volume bottlenecks.
- **Difficult scale-out of relational databases:** Due to the inherent attributes of distributed databases, shard storage nodes can be changed through smooth data migration, supporting the dynamic scale-out of relational databases.

7.2 Quick start

This topic describes how to get started with Distributed Relational Database Service (DRDS).

A DRDS instance is physically a distributed cluster that consists of multiple DRDS servers and underlying storage instances. A DRDS database is a logical concept and only contains metadata. Specific data is stored in the physical database of the underlying storage instance. To get started with DRDS, follow these steps:

1. [Create a DRDS instance.](#)
2. [Create a database.](#)

To create a database in a DRDS instance, you must select one or more ApsaraDB for RDS instances as the data storage nodes. If no ApsaraDB for RDS instance exists, create one first. For more information about how to create and manage ApsaraDB for RDS instances, see *User Guide of ApsaraDB for RDS*.

3. After a DRDS database is created, you also need to create tables in the DRDS database like in a single-instance database. However, the syntax is different, mainly in the expression of data partitioning information in the DRDS table creation statement. For more information about how to create a table, see [Table creation syntax](#).

For more DRDS operations, see the following descriptions.

7.3 Log on to the DRDS console

This topic uses Google Chrome as an example to demonstrate how to log on to the Distribute Relational Database Service (DRDS) console.

Prerequisites

- Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel . The URL used to access the ASCM console is in the following format: `http://IP address or domain name of the ASCM console/manage`.
- We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to access the ASCM console. Press Enter.

2. Enter your username and password.

The system has a default super administrator, whose username is super. The super administrator can create system administrators. A system administrator can create system users and notify the users of the default passwords by SMS or email.



Note:

When you log on to the ASCM console for the first time, you must modify the password of your username as instructed. For security concerns, your password must meet the minimum complexity requirements: The password must be 8 to 20 characters in length and must contain at least two types of the following characters: letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs (\$), and percent signs (%).

3. Click Login to go to the ASCM console homepage.

4. In the top navigation bar, choose Products > Distributed Relational Database Service.

7.4 Instance management

7.4.1 Create a DRDS instance

This topic describes how to create a Distributed Relational Database Service (DRDS) instance in the console.

Procedure

1. *Log on to the DRDS console.*
2. On the Distributed Database Instance Management page, click Create Instance in the upper-right corner.

3. Configure the following parameters.

Parameter	Configuration	Description
Region	Region	The region where the DRDS instance resides. Services in different regions are not interconnected over the internal network. Once a region is selected, it cannot be changed.
	Zone	The zone where the DRDS instance resides.
Basic Settings	Organization	The organization to which the DRDS instance belongs.
	Resource Set	The resource set to which the DRDS instance belongs.
	Instance Type	The type of the DRDS instance. Select the instance type from the options that are available on specific pages.
	Instance Series	The series of the DRDS instance, which is classified into the following categories: <ul style="list-style-type: none"> · Starter Edition · Standard Edition · Enterprise Edition · Ultimate Edition
	Instance Specifications	The specifications of the DRDS instance. The rules vary with instance series. Select the instance specifications from the options available on specific pages.

Parameter	Configuration	Description
Network type	Network Type	<p>The type of the network supported by the DRDS instance. Only classic networks are supported.</p> <div style="background-color: #f0f0f0; padding: 5px;">  Note: Cloud services in a classic network are not isolated. Unauthorized access to cloud services is blocked only by the security group or whitelist policy. </div>
Quantity	Instances	The maximum number of DRDS instances that you can create in batches.

4. After you have configured the preceding parameters, click Submit.

7.4.2 Change configurations

You can modify the configurations of your Distributed Relational Database Service (DRDS) instance if the configurations do not meet the requirements of your application.

Procedure

1. [Log on to the DRDS console](#).
2. Click the name of the target DRDS instance. The Basic Information page appears.
3. In the Common Operations section, click Upgrade or Downgrade.
4. Select the required Instance Series and Instance Specifications.
5. After you have configured the preceding parameters, click Submit.

7.4.3 Restart an instance

This topic describes how to restart a Distributed Relational Database Service (DRDS) instance.

Procedure

1. [Log on to the DRDS console](#).
2. Click the name of the target DRDS instance. The Basic Information page appears.

3. Click **Restart the Instance** in the upper-right corner of the page.
4. In the **Restart the Instance** dialog box, click **Determine**.



Notice:

Restarting an instance will cause service interruption. Exercise caution when performing this operation and make sure that the restart does not affect other services.

7.4.4 Release an instance

You can release a running Distributed Relational Database Service (DRDS) instance in the console.

Prerequisites

Before releasing a DRDS instance, make sure that you have deleted all databases from the DRDS instance.



Note:

The DRDS instance must be running.

Procedure

1. *Log on to the DRDS console.*
2. Find the target DRDS instance, and choose **Operating > Freed**.
3. In the **Release DRDS Instance** dialog box, click **Determine**.



Warning:

You cannot recover the DRDS instances that have been released. Exercise caution when you perform this operation.

7.4.5 Monitor instances

This topic describes the performance monitoring function of Distributed Relational Database Service (DRDS) and how to analyze performance metrics and troubleshoot DRDS performance problems.

7.4.5.1 View the monitoring information

Distributed Relational Database Service (DRDS) provides a wide range of performance metrics. You can view resource monitoring and engine monitoring data in the DRDS console.

Procedure

1. [Log on to the DRDS console.](#)
2. Click the name of the target DRDS instance. The Basic Information page appears.
3. In the left-side navigation pane, choose **Monitoring and Alarm > Instance Monitoring**.

For more information about performance metrics, see [Metrics](#).

7.4.5.2 Metrics

Instance monitoring is divided into resource monitoring and engine monitoring. Engine monitoring metrics are classified into metrics at the Distributed Relational Database Service (DRDS) instance level and metrics at the DRDS database level. When some engine monitoring metrics are abnormal, you can directly check the metrics of each database to locate the database with performance problems. The following table describes the metrics of these two types in details.

Metric	Category	Definition	Data collection cycle	Data retention period	Description
CPU utilization	Resource monitoring	The average CPU utilization of DRDS nodes.	1 minute	3 days	-
Memory usage	Resource monitoring	The memory usage of JVM Old Generation on DRDS nodes.	1 minute	3 days	Memory usage fluctuations are normal.
Inbound network traffic	Resource monitoring	The total inbound network traffic of DRDS nodes.	1 minute	3 days	Inbound network traffic is generated when ApsaraDB for RDS returns data to DRDS.

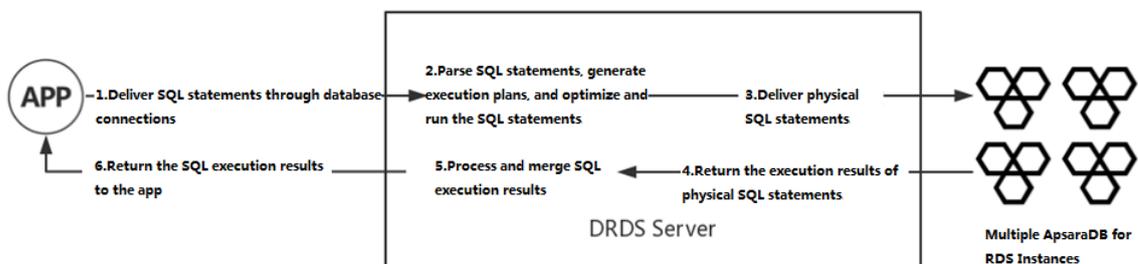
Metric	Category	Definition	Data collection cycle	Data retention period	Description
Outbound network traffic	Resource monitoring	The total outbound network traffic of DRDS nodes.	1 minute	3 days	Outbound network traffic is generated when a DRDS instance sends a physical SQL statement to an ApsaraDB for RDS instance or a DRDS instance returns data to an application.
Logical QPS	Engine monitoring	The total number of SQL statements processed per second on DRDS nodes.	5 seconds	7 days	-
Physical QPS	Engine monitoring	The total number of SQL statements sent from DRDS nodes to ApsaraDB for RDS per second.	5 seconds	7 days	One logical SQL statement can be split into multiple physical SQL statements.
Logical RT	Engine monitoring	The average response time for processing each SQL statement by DRDS.	5 seconds	7 days	If a logical SQL statement is split into physical SQL statements for delivery, the logical RT of the SQL statement contains the RT of the physical SQL statements.

Metric	Category	Definition	Data collection cycle	Data retention period	Description
Physical RT	Engine monitoring	The average response time for sending SQL statements from DRDS to ApsaraDB for RDS.	5 seconds	7 days	-
Number of connections	Engine monitoring	The total number of connections established between an application and DRDS.	5 seconds	7 days	Connections from DRDS to ApsaraDB for RDS are not included.
Number of active threads	Engine monitoring	The number of threads that are used by DRDS to run SQL statements.	5 seconds	7 days	-

7.4.5.3 How metrics work

Before analyzing metrics, you need to understand the execution process of SQL statements on Distributed Relational Database Service (DRDS).

Figure 7-2: SQL execution process in DRDS



In the entire SQL execution process, the execution status of steps 2 through 4 is reflected in various metrics of DRDS.

- In step 2, SQL parsing, optimization, and execution consume CPU resources. A more complex SQL statement (with a complex structure or ultra-long length) consumes more CPU resources. You can run the `TRACE` command to trace the SQL execution process. You can see the time consumed by an SQL statement during optimization. The longer time consumed indicates a higher CPU utilization.
- In step 3, the delivery and execution of physical SQL statements consume I/O resources. You can analyze the execution status of physical SQL statements based on metrics such as logical and physical queries per second (QPS) and response time (RT). For example, if the physical QPS is low and the physical RT is high, the current ApsaraDB for RDS instance is processing SQL statements very slowly. You need to check the performance of the ApsaraDB for RDS instance.
- In step 5, the SQL execution results are processed and integrated, to convert the execution results of physical SQL statements. In most cases, only SQL metadata is converted, which consumes few resources. However, the CPU utilization is high for steps such as `heap sort`. For more information about how to determine the consumption of SQL statements at this stage, see the description of the `TRACE` command in [Troubleshoot slow SQL statements in DRDS](#).

7.4.5.4 Prevent performance problems

7.4.5.4.1 Example 1: DRDS CPU utilization

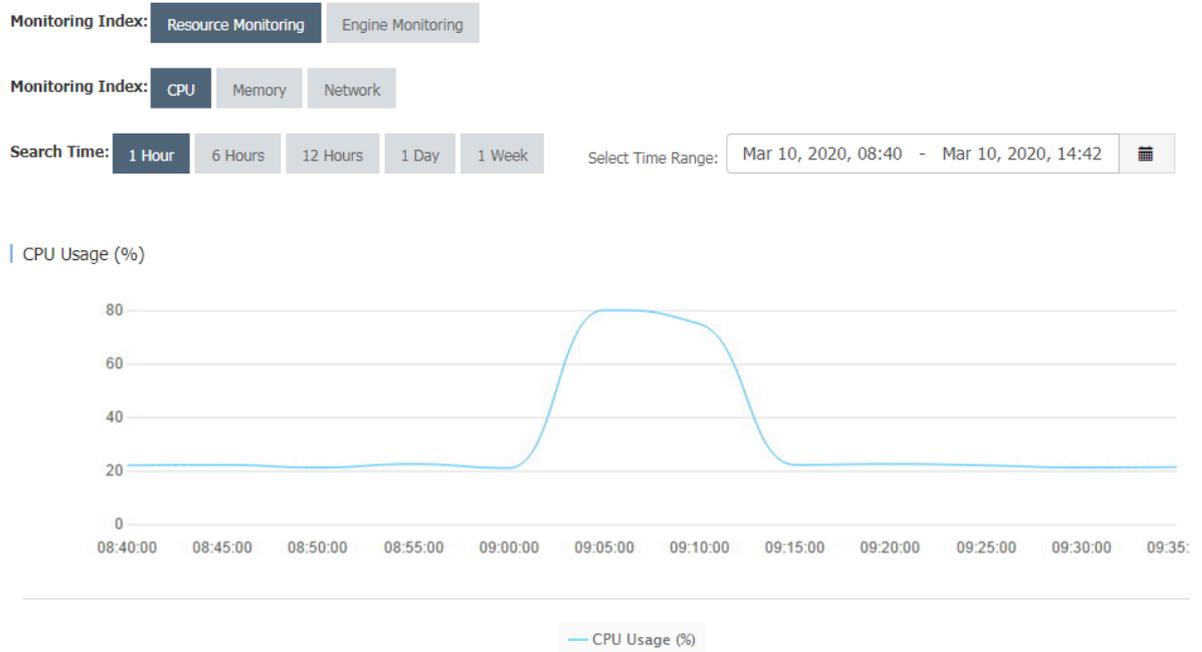
The performance metrics of the Distributed Relational Database Service (DRDS) instance change with the traffic of the system. This topic describes how to determine the running status based on the CPU utilization of the DRDS instance.

The following describes the CPU utilization in two common cases:

- An application has a shopping spree activity at 09:00 every morning, so the traffic of the system increases significantly at this time point. According to the

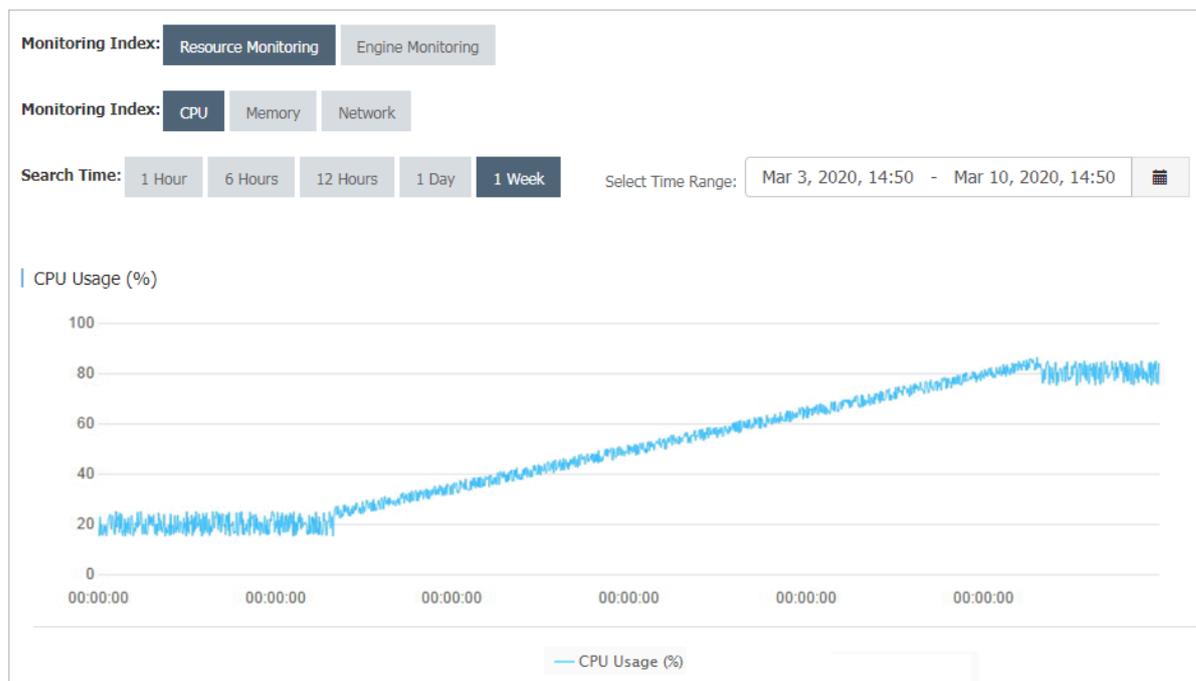
monitoring data, the CPU utilization of the DRDS instance increased from 20% to about 80% from about 09:00, with the traffic peak lasting about 10 minutes.

Figure 7-3: CPU utilization-1



- The system traffic keeps increasing with an application until it reaches a level. The monitored CPU data of the DRDS instance also reflects this change.

Figure 7-4: CPU utilization-2



When the load on the DRDS instance changes with the business, you must pay close attention to the changes in metrics. If the CPU utilization exceeds the threshold, you must upgrade the DRDS configuration to alleviate the performance pressure.

You can set alert rules for instances in the DRDS console. When the average CPU utilization exceeds the preset threshold, the system sends short messages to the corresponding contacts. You can set the CPU utilization threshold as needed. We recommend that you set it to 80%.

7.4.5.4.2 Example 2: Logical RT and physical RT

This topic describes how to determine the running status of a Distributed Relational Database Service (DRDS) instance based on the difference between logical response time (RT) and physical RT.

Logical RT refers to the time from when a DRDS instance receives a logical SQL statement to when it returns data to an application. Physical RT refers to the time from when a DRDS instance sends a physical SQL statement to an ApsaraDB for RDS instance to when it receives the data returned by the ApsaraDB for RDS instance.

If a logical SQL statement is partitioned into one or more physical SQL statements, the logical RT is greater than or equal to the physical RT. Normally, DRDS performs only a few operations on the data returned by ApsaraDB for RDS. Therefore, logical RT is slightly longer than physical RT. Under special circumstances, physical SQL

queries are run fast, while logical SQL queries take a long time to run. In this case, the logical RT and physical RT are as follows:

Figure 7-5: Logical RT

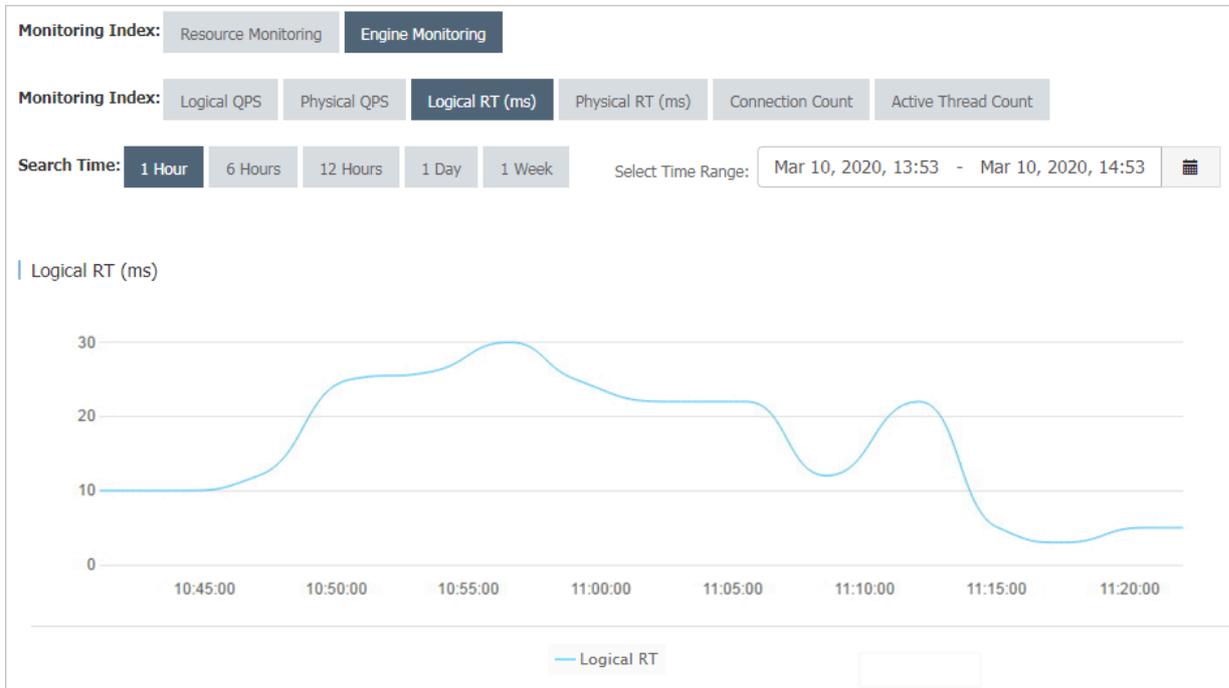


Figure 7-6: Physical RT



As shown in the preceding figures, the change trends of logical RT and physical RT in the two monitoring charts are basically the same, with logical RT fluctuating

between 10 ms and 20 ms and physical RT fluctuating between 2 ms and 5 ms. This means that DRDS has a heavy load, which can be solved by upgrading the DRDS configuration. If both the logical RT and physical RT are high, you can upgrade the ApsaraDB for RDS configuration or optimize SQL statements in the ApsaraDB for RDS instance.

7.4.5.4.3 Example 3: Logical QPS and physical QPS

This topic describes how to determine the running status of a Distributed Relational Database Service (DRDS) instance based on the difference between logical queries per second (QPS) and physical QPS.

The logical QPS and physical QPS have the same trends, but there is a large difference between them.

Figure 7-7: Logical QPS

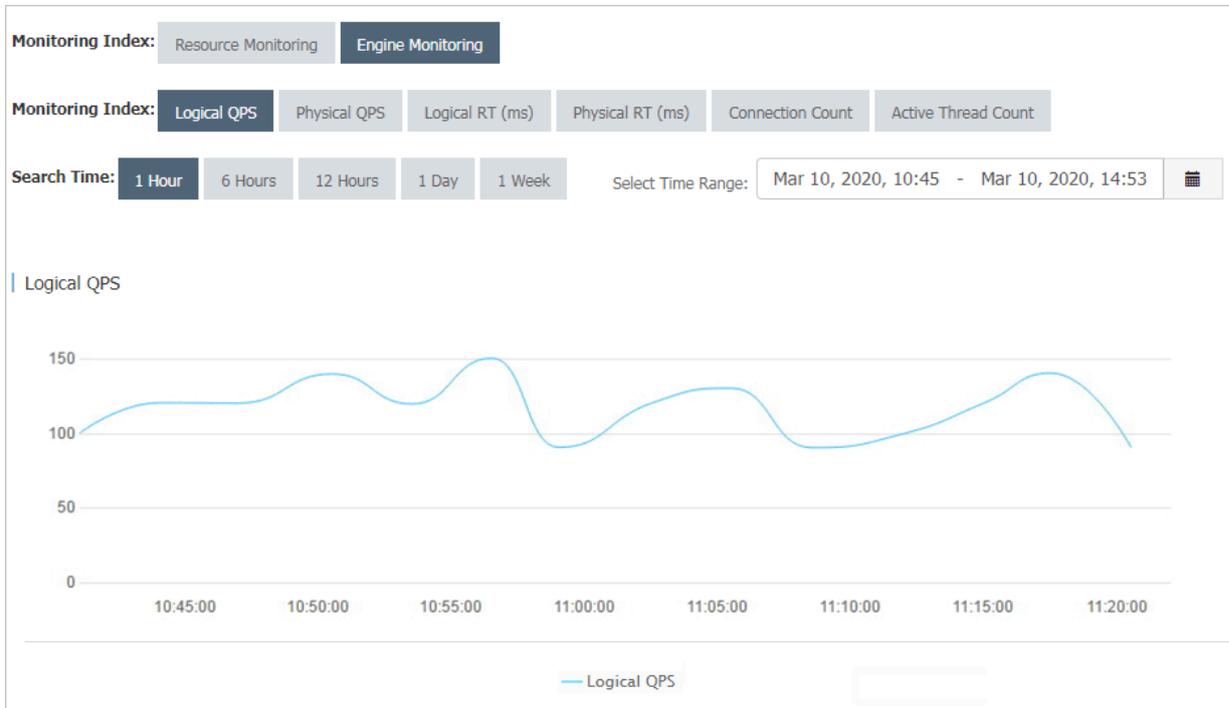
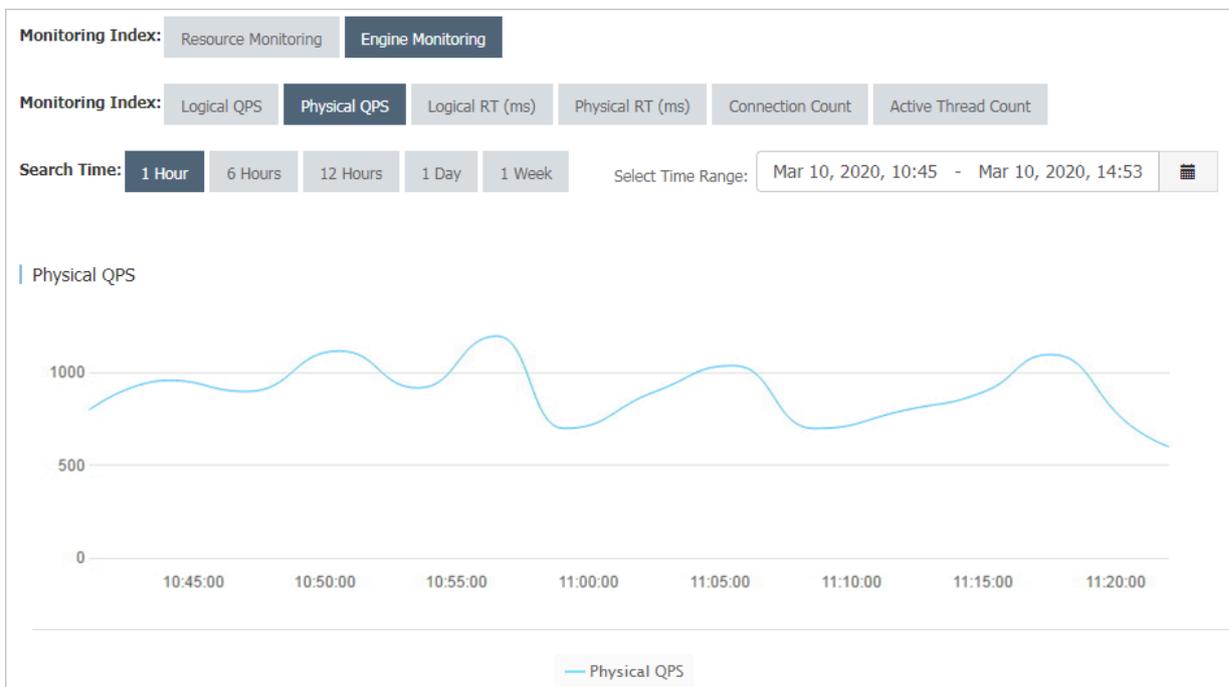


Figure 7-8: Physical QPS



As shown in the preceding figures, logical QPS fluctuates between 80 and 150, and physical QPS fluctuates between 700 and 1,200.

Cause: DRDS generates physical SQL statements based on logical SQL statements. The ratio of logical SQL statements to physical SQL statements is not necessarily 1:1

For example, a DRDS logical table is created by using the following statement:

```
CREATE TABLE drds_user
(id int,
name varchar(30))
dbpartition by hash(id);
```

When the query condition contains the database shard key, DRDS pushes the logical SQL statement down to the ApsaraDB for RDS instance for execution. According to the execution plan, the number of physical SQL statements is 1:

```
mysql> explain select name from drds_user where id = 1;
+-----+
+-----+
+-----+
| GROUP_NAME          | SQL                                |
+-----+-----+
+-----+-----+
| SANGUAN_BSQT_0001_RDS | select `drds_user`.`name` from `drds_user` where (`drds_user`.`id` = 1) | {} |
```

When the query does not contain the database shard key, DRDS splits the logical SQL statement into multiple physical SQL statements. The following execution plan shows that there are eight physical SQL statements.

```
mysql> explain select name from drds_user where name = 'LiLei';
+-----+
+-----+
+-----+
| GROUP_NAME          | SQL                                |
+-----+-----+
+-----+-----+
| SANGUAN_BSQT_0001_RDS | select `drds_user`.`name` from `drds_user` where (`drds_user`.`name` = 'LiLei') | {} |
| SANGUAN_BSQT_0001_RDS | select `drds_user`.`name` from `drds_user` where (`drds_user`.`name` = 'LiLei') | {} |
| SANGUAN_BSQT_0001_RDS | select `drds_user`.`name` from `drds_user` where (`drds_user`.`name` = 'LiLei') | {} |
| SANGUAN_BSQT_0001_RDS | select `drds_user`.`name` from `drds_user` where (`drds_user`.`name` = 'LiLei') | {} |
| SANGUAN_BSQT_0001_RDS | select `drds_user`.`name` from `drds_user` where (`drds_user`.`name` = 'LiLei') | {} |
| SANGUAN_BSQT_0001_RDS | select `drds_user`.`name` from `drds_user` where (`drds_user`.`name` = 'LiLei') | {} |
```

```

| SANGUAN_BSQT_0001_RDS | select `drds_user`.`name` from `
drds_user` where (`drds_user`.`name` = 'LiLei') | {} |
| SANGUAN_BSQT_0001_RDS | select `drds_user`.`name` from `
drds_user` where (`drds_user`.`name` = 'LiLei') | {} |
+-----+
+-----+
8 rows in set (0.06 sec)

```

Logical or physical QPS indicates the total number of logical or physical SQL statements processed per unit of time. When most SQL statements in the system contain the shard key, the ratio of logical QPS to physical QPS is close to 1:1. If the difference between the logical and physical QPS is too large, many SQL statements of the current application do not contain the shard key. In this case, check the SQL statements of the application to improve performance.

7.4.5.4.4 Example 4: High memory usage

The overly high memory usage of the Distributed Relational Database Service (DRDS) instance is mostly caused by the large number of SQL queries in your application and the overlarge result set that is returned. If the memory usage of your DRDS instance remains at about 100%, [Restart an instance](#) to locate and optimize the slow SQL queries of your application.

7.4.6 View the instance version

This topic describes how to view the version of a Distributed Relational Database Service (DRDS) instance in the console or by using the command line.

View the instance version in the console

1. [Log on to the DRDS console](#)
2. **Click the name of the target DRDS instance. The Basic Information page appears.**
3. **In the Configuration Information section, view the value of Current Version.**

Configuration Information	
Instance Type: Dedicated	Instance Edition: Starter Edition
Specification: 12-core 24GB	Current Version: 5.3.9-15653249

View the instance version in the command line

Connect to the DRDS instance through the MySQL command line and run the `SELECT version()` statement to view the version of the DRDS instance.

```

mysql> select version();
+-----+

```

```

| VERSION() |
+-----+
| 5.6.29-TDDL-5.1.28-1320920 |
+-----+
1 row in set (0.00 sec)

```

5.1.28-1320920 indicates the version of the DRDS instance.

For more information about each version, see [DRDS versions](#).

7.5 Account management

Through account management of Distributed Relational Database Service (DRDS), you can use the same account in a DRDS instance to manage multiple databases. With this account, you can create accounts, modify permissions, and reset passwords.

The usage of the account and permission system in DRDS is the same as that in MySQL. DRDS supports the `GRANT`, `REVOKE`, `SHOW GRANTS`, `CREATE USER`, `DROP USER` and `SET PASSWORD` statements. Currently, DRDS allows you to grant permissions at the database and table levels, but does not grant permissions at the global or column level.

For more information about the MySQL account and permission system, see [MySQL official documentation](#).



Notice:

Accounts created through `CREATE USER` in DRDS only exist in the DRDS instance and will not be synchronized to the backend ApsaraDB for RDS instances.

7.5.1 Overview

You can view all accounts and permissions under the current Distributed Relational Database Service (DRDS) instance through account management. An account consists of a user and a host. The user has access permissions on the host, which is the same as MySQL.

Accounts are classified into standard accounts and privileged accounts. Accounts created in the DRDS console are standard accounts, and accounts created by using DRDS SQL commands are privileged accounts. Privileged accounts cannot be modified in the DRDS console.

The database and permission columns display the permissions of the account on the database.

-	SELECT	INSERT	UPDATE	DELETE	INDEX	ALTER	CREATE	DROP	GRANT
DDL	×	×	×	×	√	√	√	√	×
DML	√	√	√	√	×	×	×	×	×
Read-only	√	×	×	×	×	×	×	×	×
Read/Write	√	√	√	√	√	√	√	√	×
ROOT	√	√	√	√	√	√	√	√	√

Custom permissions are complex, typically the permissions that you create and assign by using the SQL statements. By default, an account created with a database and with the same name as the database has the ROOT permission and cannot be deleted in the console.

7.5.2 Create an account

This topic describes how to create a Distributed Relational Database Service (DRDS) account in the console and by using SQL statements.

Create an account in the console

1. [Log on to the DRDS console.](#)
2. Click the name of the target DRDS instance. The Basic Information page appears.
3. In the left-side navigation pane, choose Configuration and Management > Account Management.
4. On the Account Management page, click Create an Account in the upper-right corner.
5. Configure the following parameters.

Parameter	Description
Database Account	<p>Enter the account name. The requirements are as follows:</p> <ul style="list-style-type: none"> • The name must be 4 to 20 characters in length. • It must start with a letter and end with a letter or number. • It can contain lowercase letters, numbers, and underscores (_).

Parameter	Description
New Password	<p>Enter an account password. The requirements are as follows:</p> <ul style="list-style-type: none"> • The password must be 8 to 32 characters in length. • It must contain three of the following character types : uppercase letters, lowercase letters, numbers, and special characters. • It can contain any of the following special characters: ! @#\$%^&*()_+ -=
Confirm the New Password	Enter the password again.
Authorization Database	<p>You can grant permissions on one or multiple databases to the account.</p> <ol style="list-style-type: none"> Select one or more databases in the left-side section, and click Authorization to add them to the right-side section. In the right-side section, select Read and write, Read only, DDL only, or DML only. <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p> Note: You can also grant permissions to multiple authorized databases by clicking Make All Read-only, Set All to DDL Only, Set All to DML Only or Set All to Read and Write in the upper-right corner of the right-side section.</p> <p>The button in the upper-right corner changes after you click it. For example, after you click Make All Read-only, the button is changed to Set All to DDL Only.</p> </div>

Create an account in the command line

Syntax rules:

```
CREATE USER user_specification [, user_specification] ...
  user_specification: user [ auth_option ]
  auth_option: IDENTIFIED BY 'auth_string'
```

Example:

Create an account with the username lily and password 123456, which can be used to log on only from 30.xx.xx.96.

```
CREATE USER lily@30.xx.xx.96 IDENTIFIED BY '123456';
```

Create an account named david with no password that can be used to log on from any host.

```
CREATE USER david@'%';
```

7.5.3 Reset the password

You can reset the password of your database account in the Distributed Relational Database Service (DRDS) console or by using the command line.



Note:

- Accounts with ROOT permissions cannot be deleted or modified.
- For data security, we recommend that you change your password periodically.

Reset the password in the console

1. [Log on to the DRDS console](#).
2. Click the name of the target DRDS instance. The Basic Information page appears.
3. In the left-side navigation pane, choose Configuration and Management > Account Management.
4. Find the target account, and click Reset Password.
5. In the Reset Account Password dialog box, enter a new password and confirm the new password.



Note:

The password must meet the following requirements:

- The password must be 8 to 32 characters in length.
- It must contain three of the following character types: uppercase letters, lowercase letters, numbers, and special characters.
- It can contain any of the following special characters: !@#\$%^&*()_+ -=

6. After confirming that the password is correct, click Determine.

Reset the password in the command line

Syntax rules:

```
SET PASSWORD FOR user = password_option
password_option: {
    PASSWORD('auth_string')
}
```

Example:

Change the password of the account `lily@30.xx.xx.96` to 123456.

```
SET PASSWORD FOR lily@30.xx.xx.96 = PASSWORD('123456')
```

7.5.4 Modify account permissions

You can modify the account permissions on your instances at any time when using Distributed Relational Database Service (DRDS).

Notes

- **Privileged accounts cannot be modified.**
- **In the console, you can only grant DML, DDL, read-only, and read/write permissions to standard accounts. To grant more permissions, use the command line.**

Modify account permissions in the console

1. *Log on to the DRDS console.*
2. **Click the name of the target DRDS instance. The Basic Information page appears.**
3. **In the left-side navigation pane, choose Configuration and Management > Account Management.**
4. **Find the target account and click Modify Permissions.**

5. In the Modify Permissions dialog box, grant or remove the permissions on one or more databases to or from the account.

- **Add an authorized database:**

Select one or more databases in the left-side section, and click **Authorization** to add them to the right-side section.

- **Remove an authorized database:**

Select one or more databases in the right-side section, and click **Remove** to remove them. The removed databases are displayed in the left-side section.

- **Modify permissions of an authorized database:**

In the right-side section, select **Read and Write**, **Read-only**, **DDL Only** or **DML Only**.



Note:

You can also grant permissions to multiple authorized databases by clicking **Make All Read-only**, **Set All to DDL Only**, **Set All to DML Only** or **Set All to Read and Write** in the upper-right corner of the right-side section.

The button in the upper-right corner changes after you click it. For example, after you click **Make All Read-only**, the button is changed to **Set All to DDL Only**.

6. After the configuration is complete, click Confirm.

GRANT statements

Syntax rules:

```
GRANT
    priv_type[, priv_type] ...
    ON priv_level
    TO user_specification [, user_specification] ...
    [WITH GRANT OPTION]
priv_level: {
    | db_name.*
    | db_name.tbl_name
    | tbl_name
}
user_specification:
    user [ auth_option ]
auth_option: {
    IDENTIFIED BY 'auth_string'
}
```



Notice:

- If the account in the GRANT statement does not exist and no IDENTIFIED BY information is provided, an error message indicating that the account does not exist is returned.
- If the account specified in the GRANT statement does not exist but the IDENTIFIED BY information is provided, the account is created and granted with the specified permission.

For example, in the easydb database, create an account named david, which can be used to log on from any host and has all the permissions on easydb.

Method 1: Create an account and then grant permissions to the account.

```
CREATE USER david@%' IDENTIFIED BY 'your#password';  
GRANT ALL PRIVILEGES ON easydb.* to david@%';
```

Method 2: Create an account and grant permissions to the account by using one statement.

```
GRANT ALL PRIVILEGES ON easydb.* to david@%' IDENTIFIED BY 'your#  
password';
```

In the easydb database, create an account named hanson, which can be used to log on from any host and has all the permissions on the easydb.employees table.

```
GRANT ALL PRIVILEGES ON easydb.employees to hanson@%' IDENTIFIED BY '  
your#password';
```

In the easydb database, create an account named hanson, which can be used to log on only from 192.xx.xx.10 and has the INSERT and SELECT permissions on the easydb.emp table.

```
GRANT INSERT,SELECT ON easydb.emp to hanson@'192.xx.xx.10' identified  
by 'your#password';
```

In the easydb database, create a read-only account named actro, which can be used to log on from any host.

```
GRANT SELECT ON easydb.* to actro@%' IDENTIFIED BY 'your#password';
```

REVOKE statements

Syntax rules:

- Delete the permissions at a certain level of an account. The permission level is specified by priv_level.

```
REVOKE
```

```
priv_type
[, priv_type] ...
ON priv_level
FROM user [, user] ...
```

- **Delete all permissions of the account at the database and table levels.**

```
REVOKE ALL PRIVILEGES, GRANT OPTION
FROM user [, user] ...
```

Example:

Delete the CREATE, DROP, and INDEX permissions of hanson@'%' on the easydb.emp table.

```
REVOKE CREATE,DROP,INDEX ON easydb.emp FROM hanson@'%';
```

Delete all permissions of the account lily@30.xx.xx.96.

```
REVOKE ALL PRIVILEGES,GRANT OPTION FROM lily@30.xx.xx.96;
```



Notice:

GRANT OPTION must be added to the statement for compatibility with MySQL.

SHOW GRANTS statements

Syntax rules:

```
SHOW GRANTS[ FOR user@host];
```

Query all permissions:

```
SHOW GRANTS
```

Query the permissions of an account:

```
SHOW GRANTS FOR user@host;
```

7.5.5 Delete an account

You can delete an account in the Distributed Relational Database Service (DRDS) console or by using the command line.

Delete an account in the console



Note:

You can only delete standard accounts that you created in the console.

1. *Log on to the DRDS console.*

2. Click the name of the target DRDS instance. The Basic Information page appears.
3. In the left-side navigation pane, choose Configuration and Management > Account Management.
4. Find the target account and click Delete.
5. In the Delete Account dialog box, click Determine.

Delete an account by using the command line

Syntax rules:

```
DROP USER user [, user] ...
```

Example:

Delete the account `lily@30.xx.xx.96`:

```
DROP USER lily@30.xx.xx.96;
```

7.6 Database management

7.6.1 Create a database

When you create a database in the Distributed Relational Database Service (DRDS) console, you must select an ApsaraDB RDS for MySQL instance as the storage. The procedure is as follows:

Open the Create Database page

1. [Log on to the DRDS console](#).
2. Click the name of the target DRDS instance. The Basic Information page appears.
3. In the left-side navigation pane, choose Configuration and Management > Database Management.
4. On the Database Management page, click Create Database in the upper-right corner.

On the Create Database page, select [Horizontal partitioning](#) mode.

Horizontal partitioning

1. On the Fill in the Basic Information page, enter the following information.

Parameter	Description
Split Mode	Select Split Horizontally.
Database Name	Specify a name for the DRDS database. The name must meet the following requirements: <ul style="list-style-type: none"> • It must be 2 to 24 characters in length. • It must start with a letter and end with a letter or number. • It can contain lowercase letters, numbers, and underscores (_). • The database name must be unique in the DRDS instance.
Character Set	Select UTF8, GBK, Latin1, or utf8mb4.
DRDS Link Password	Set the DRDS connection password. The rules are as follows: <ul style="list-style-type: none"> • The password must be 8 to 30 characters in length • The password must contain three of the following character types: uppercase letters, lowercase letters, numbers, and underscores (_).
Confirm Password	Enter the password again.

2. Click Next Step.

3. On the Select MySQL page, select one or more ApsaraDB for RDS instances and move them to RDS Selected on the right, and then click Next Step.

4. After pre-check has passed, click Next Step.

**Note:**

If pre-check fails, rectify the configuration as prompted.

5. On the Library Build Preview page, click Next Step.

6. Wait until the database is created.

7.6.2 View a database

After the database is created, you can view the basic information of the database in the console.

Procedure

1. *Log on to the DRDS console.*
2. Click the name of the target DRDS instance. The Basic Information page appears.
3. In the left-side navigation pane, choose Configuration and Management > Database Management.
4. Find the target database, and click Management. The Basic Information page of the database appears.

On the Basic Information page, you can perform management of the database, such as running DDL statements, scaling out the database, importing data, and deleting the database.

What's next

DRDS is fully compatible with the MySQL protocol. You can use Command Line Link Address on the MySQL client to connect to the DRDS instance and enter the username and password to log on to the DRDS database. When using the MySQL client, note the following points:



Note:

- Some MySQL clients of earlier versions have limits on the username length, which cannot be more than 16 characters. The DRDS database name and username are the same. If the database name exceeds 16 characters, an error is reported.
- When using the MySQL client, you must add the `-c` parameter to the hint command. In DRDS, HINT is implemented by using annotations. If the `-c` parameter is not added, the annotation is lost and the DRDS hint is lost.

7.6.3 Perform smooth scale-out

When the underlying storage of the logical database reaches the physical bottleneck, for example, when the remaining disk space is 30%, you can smoothly

scale it out to improve the performance. The smooth scale-out process is divided into four steps: configuration > migration > switchover > cleaning.

Configuration



Note:

In smooth scale-out, ApsaraDB for RDS instances are added, and some source database shards are migrated to the new ApsaraDB for RDS instance. In this way, the overall data storage capacity is increased and the number of requests that a single ApsaraDB for RDS instance needs to process is reduced.

1. [Log on to the DRDS console](#).
2. Click the name of the target DRDS instance. The Basic Information page appears.
3. In the left-side navigation pane, choose Configuration and Management > Database Management.
4. Find the target database, and click Management. The Basic Information page of the database appears.
5. In the left-side navigation pane, choose Configuration and Management > Expansion Management.
6. In the upper-right corner of the Expansion Management page, click Expansion.
7. Select Smooth Expansion, and then click Next Step.
8. After pre-check has passed, click Next Step.



Note:

If pre-check fails, rectify the configuration as prompted.

9. On the Select MySQL page, select the ApsaraDB for RDS instances you want to add, and click Next Step.
10. On the Library Build Preview page, click Next Step.
11. On the Results Preview page, click Start Expansion.



Note:

By default, the console evenly distributes the physical database shards to the ApsaraDB for RDS instance you added. You can also manually add or delete physical database shards to or from the new ApsaraDB for RDS instance.

12. Click the icon in the upper-right corner to view the details of the scale-out task.

Migration

Some physical database shards are migrated during smooth scale-out.

The migration does not change the data in the source database, and therefore it does not affect online services. Before switchover, you can cancel the smooth scale-out operation through rollback.



Note:

- **This is because before switchover, the current scale-out operation does not have a real impact on the data in the source database.**
- **During scale-out, the Binlog files of the source ApsaraDB for RDS instance are not cleaned, which may result in insufficient disk space. Therefore, you must reserve sufficient disk space on the source ApsaraDB for RDS instance. Generally, the remaining disk space should be more than 30%. If the disk space cannot be guaranteed, you can submit a ticket to expand the ApsaraDB for RDS storage space.**
- **To reduce the pressure of read operations on the source ApsaraDB for RDS instance, perform scale-out when the load on the source ApsaraDB for RDS instance is low.**
- **During the scale-out, do not submit DDL tasks in the console or connect to the DRDS instance to directly run DDL statements. Otherwise, the scale-out task may fail.**
- **Make sure that all tables in the source database have primary keys before scale-out.**

After historical data and incremental data are migrated, the migration progress reaches 100%. Then, you can switch the read and write traffic to the new ApsaraDB for RDS instance or roll back to cancel this scale-out.

Switchover

The switchover task switches the read and write traffic to the new ApsaraDB for RDS instance. The whole process takes 3 to 5 minutes. During the switchover process, the service is not affected except for one or two transient disconnections. Perform switchover during off-peak hours.

1. **In the upper-right corner of the Basic Information page, click the icon. The Task Progress dialog box appears.**

2. In the Task Progress dialog box, click **Switch** and then **Determine**.

During the switchover process, a switchover task is generated and displayed in the Task Progress dialog box.

3. After the switchover is complete, the **Clean Up** button appears in the Task Progress dialog box, which means that the switchover task is complete.

Cleanup

In this step, the migrated database shards are deleted from the source ApsaraDB for RDS instance.

1. After switchover is complete, click **Clean Up** next to the target task.

2. Click **Determine**. A cleanup task is displayed in the Task Progress dialog box.

The cleanup task is an asynchronous task. You can view the execution status in the Task Progress dialog box.

After the cleanup task is complete, the smooth scale-out process ends. The new ApsaraDB for RDS instance becomes the storage node of the DRDS logical database.

Currently, smooth scale-out is implemented by migrating physical database shards. If no further scale-out is allowed after the number of database shards exceeds the capacity of a single ApsaraDB for RDS instance, you can submit a ticket to apply for increasing the number of database shards and scaling out the database. In this case, Hash calculation is performed again to reallocate data.



Note:

- The cleanup task deletes database shards that are no longer used after the current scale-out. You can back up the database shards before running the cleanup task.
- The cleanup operation brings pressure to databases. We recommend that you perform this operation during off-peak hours.

7.6.4 View the monitoring information

Distributed Relational Database Service (DRDS) displays the historical monitoring information of a DRDS database in two dimensions: data metrics and query time.

Procedure

1. *Log on to the DRDS console.*

2. Click the name of the target DRDS instance. The Basic Information page appears.
3. In the left-side navigation pane, choose Monitoring and Alarm > Database Monitoring.
4. Select Data Indicators or Query Time. Then, the corresponding monitoring information appears.

7.6.5 Configure a whitelist

The whitelist feature is used to set the IP address of the machine that can access the Distributed Relational Database Service (DRDS) database.

Prerequisites

DRDS provides the access control function. Only IP addresses in the whitelist of a database can access the database.

Procedure

1. *Log on to the DRDS console.*
2. Click the name of the target DRDS instance. The Basic Information page appears.
3. In the left-side navigation pane, choose Configuration and Management > Database Management.
4. Find the target database, and click Management. The Basic Information page of the database appears.
5. On the Basic Information page of the database, click Whitelist Settings in the left-side navigation pane.
6. On the Whitelist Settings page, click Modify Manually.
7. Enter the IP address that is allowed to access the database, and click Determine.



Note:

- The whitelist supports the following formats:
 - IP address, for example, 192.168.1.1.
 - CIDR IP address, for example, 192.168.1.1/24.
 - Asterisk (*) as a placeholder, for example, 192.168.1. *.
 - A CIDR block, for example, 192.168.1.1-192.168.1.254.
- If you want to add multiple IP addresses or CIDR blocks, separate them with commas (,) without spaces. For example, 192.168.0.1,172.16.213.9.

7.6.6 Delete a database

This topic describes how to delete a database in the Distributed Relational Database Service (DRDS) console.

Procedure

1. *Log on to the DRDS console.*
2. Click the name of the target DRDS instance. The Basic Information page appears.
3. In the left-side navigation pane, choose Configuration and Management > Database Management.
4. Find the target database and click Delete.
5. In the Delete Database dialog box, click Determine.



Warning:

You cannot recover databases that have been deleted. Exercise caution when you perform this operation.

7.6.7 Fix database shard connections

Context

When using a Distributed Relational Database Service (DRDS) instance, you need to access ApsaraDB for RDS. If the network configuration of the connected ApsaraDB for RDS instance changes, for example, if the zone is switched or the network type is changed from classic to Virtual Private Cloud (VPC), the network connection between the DRDS instance and the ApsaraDB for RDS instance is broken. This way, the DRDS instance cannot access the ApsaraDB for RDS instance. In this case, you must manually fix the database shard link in the DRDS console to restore the network connection from the DRDS instance to the ApsaraDB for RDS instance.

Procedure

1. *Log on to the DRDS console.*
2. Click the name of the target DRDS instance. The Basic Information page appears.
3. In the left-side navigation pane, choose Configuration and Management > Database Management.
4. Find the target database, and click Management. The Basic Information page of the database appears.
5. In the Quick Operation section, click Fix Sub-library Connection.

6. In the pop-up window, click **Determine**.

7.7 Custom control commands

Distributed Relational Database Service (DRDS) provides a series of auxiliary SQL commands to help you conveniently use DRDS.

7.7.1 Help statement

This topic describes all the auxiliary SQL commands of Distributed Relational Database Service (DRDS) and their descriptions.

SHOW HELP statement:

```
mysql> show help;
+-----+
+-----+
+-----+
| STATEMENT                                | EXAMPLE          | DESCRIPTION
+-----+-----+-----+
| show rule                                |                  | Report all table rule
| show rule from TABLE                    | show rule from user | Report table rule
| show full rule from TABLE               | show full rule from user | Report table full rule
| show topology from TABLE                | show topology from user | Report table physical
topology
| show partitions from TABLE              | show partitions from user | Report table dbPartition
or tbPartition columns
| show broadcasts                          |                  | Report all broadcast
tables
| show datasources                         |                  | Report all partition db
threadPool info
| show node status                         |                  | Report master/slave read
| show slow                                |                  | Report top 100 slow sql
| show physical_slow                       |                  | Report top 100 physical
slow sql
| clear slow                               |                  | Clear slow data
+-----+-----+-----+
```

```

| trace SQL | Start trace sql, use show
trace to print profiling data | trace select count(*) from user; show
trace |
| show trace | Report sql execute
profiling info |
| explain SQL | Report sql plan info
| explain select count(*) from user
| explain detail SQL | Report sql detail plan
info | explain detail select count(*) from
user |
| explain execute SQL | Report sql on physical db
plan info | explain execute select count(*) from
user |
| show sequences | Report all sequences
status |
| create sequence NAME [start with COUNT] | Create sequence
| create sequence test start with 0
| alter sequence NAME [start with COUNT] | Alter sequence
| alter sequence test start with 100000
| drop sequence NAME | Drop sequence
| drop sequence test
+-----+
+-----+
+-----+
20 rows in set (0.00 sec)

```

7.7.2 Statements for viewing rules and topology

SHOW RULE [FROM tablename]

Usage notes:

- show rule: **shows the partitioning status of each logical table in a database.**
- show rule from tablename: **shows the partitioning status of a specified logical table in a database.**

The following describes the meanings of important columns:

- **BROADCAST:** indicates whether the table is a broadcast table. 0 indicates "No" and 1 indicates "Yes".
- **DB_PARTITION_KEY:** indicates the database shard key. If no database shards exist, this parameter is NULL.
- **DB_PARTITION_POLICY:** indicates the database sharding policy. Options are Hash and date policies such as YYYYMM, YYYYDD, and YYYYWEEK.
- **DB_PARTITION_COUNT:** indicates the number of database shards.

- **TB_PARTITION_KEY:** indicates the table shard key. If no table shards exist, this parameter is NULL.
- **TB_PARTITION_POLICY:** indicates the table sharding policy. Options are Hash and date policies such as MM, DD, MMDD, and WEEK.
- **TB_PARTITION_COUNT:** indicates the number of table shards.

```
mysql> show rule;
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| ID    | TABLE_NAME | BROADCAST | DB_PARTITION_KEY | DB_PARTITI
ON_POLICY | DB_PARTITION_COUNT | TB_PARTITION_KEY | TB_PARTITI
ON_POLICY | TB_PARTITION_COUNT |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
|      0 | dept_manager |          0 |                | NULL      | 1
|      1 | emp          |          0 | emp_no         | hash     | 2
|      2 | example     |          0 | shard_key     | hash     | 1
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

SHOW FULL RULE [FROM tablename]

You can run this SQL statement to view the sharding rules of logical tables in a database. It displays more detailed information than the SHOW RULE command.

The following describes the meanings of important columns:

- **BROADCAST:** indicates whether the table is a broadcast table. 0 indicates "No" and 1 indicates "Yes".
- **JOIN_GROUP:** a reserved field.
- **ALLOW_FULL_TABLE_SCAN:** indicates whether to allow data querying when no table shard key is specified for database or table sharding. If this parameter is set to True, each physical table is scanned to find data that meets the condition, which is a full table scan.
- **DB_NAME_PATTERN:** The value 0 between {} in DB_NAME_PATTERN is a placeholder. When the SQL statement is run, this value is replaced by the value of DB_RULES_STR, with the number of digits unchanged. For example, if the value of DB_NAME_PATTERN is SEQ_{0000}_RDS and the value of DB_RULES_STR


```
+-----+
+-----+
+-----+
+-----+
3 rows in set (0.01 sec)
```

SHOW TOPOLOGY FROM tablename

You can run this SQL statement to view the topology of a specified logical table, that is, the database shards to which data in the logical table is partitioned and the table shards in each database shard.

```
mysql> show topology from emp;
+-----+-----+-----+
| ID    | GROUP_NAME                                     | TABLE_NAME |
+-----+-----+-----+
| 0     | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0000_RDS | emp_0       |
| 1     | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0000_RDS | emp_1       |
| 2     | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0001_RDS | emp_0       |
| 3     | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0001_RDS | emp_1       |
| 4     | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0002_RDS | emp_0       |
| 5     | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0002_RDS | emp_1       |
| 6     | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0003_RDS | emp_0       |
| 7     | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0003_RDS | emp_1       |
| 8     | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0004_RDS | emp_0       |
| 9     | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0004_RDS | emp_1       |
| 10    | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0005_RDS | emp_0       |
| 11    | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0005_RDS | emp_1       |
| 12    | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0006_RDS | emp_0       |
| 13    | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0006_RDS | emp_1       |
| 14    | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0007_RDS | emp_0       |
| 15    | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0007_RDS | emp_1       |
+-----+-----+-----+
16 rows in set (0.01 sec)
```

SHOW PARTITIONS FROM tablename

You can run this SQL statement to view the set of database shard keys and table shard keys, which are separated by commas (.). If the final result contains two

values, both database sharding and table sharding are performed. The first value is the database shard key and the second value is the table shard key. If only one value is returned, only database sharding is performed. This value is the database shard key.

```
mysql> show partitions from emp;
+-----+
| KEYS  |
+-----+
| emp_no,id |
+-----+
1 row in set (0.00 sec)
```

SHOW BROADCASTS

You can run this SQL statement to view the list of broadcast tables.

```
mysql> show broadcasts;
+-----+-----+
| ID  | TABLE_NAME |
+-----+-----+
| 0   | brd2        |
| 1   | brd_tbl     |
+-----+-----+
2 rows in set (0.01 sec)
```

SHOW DATASOURCES

You can run this SQL statement to view the information about the underlying storage, including the database name, database group name, connection URL, username, storage type, read/write weight, and connection pool information.

The following describes the meanings of important columns:

- **SCHEMA:** indicates the database name.
- **GROUP:** indicates the database group name. Grouping aims to manage multiple groups of databases that have identical data, such as the primary and secondary databases after data replication through ApsaraDB RDS for MySQL. It is mainly used for read/write splitting and primary/secondary switchover.
- **URL:** indicates the connection information of the underlying ApsaraDB RDS for MySQL instance.
- **TYPE:** indicates the type of the underlying storage. Currently, only ApsaraDB RDS for MySQL instances are supported.
- **READ_WEIGHT:** indicates the read weight of the database. When the primary ApsaraDB for RDS instance is under a heavy load of many read requests, you can use the read/write splitting function of Distributed Relational Database Service

- **Read-only queries in transactions are sent to the primary ApsaraDB for RDS instance.**
- **The MASTER_READ_PERCENT and SLAVE_READ_PERCENT fields indicate the accumulative historical data. After the read/write weight ratio has been changed, these values do not immediately reflect the latest read/write weight ratio, which is displayed after a long period of time has passed.**

```
mysql> show node;
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+
| ID      | NAME                                                                 | MASTER_READ_COUNT | SLAVE_READ_COUNT | MASTER_READ_PERCENT | SLAVE_READ_PERCENT |
+-----+-----+-----+-----+-----+
| 0       | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0000_RDS                | 12                | 0                | 100%                | 0%                 |
| 1       | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0001_RDS                | 0                 | 0                | 0%                  | 0%                 |
| 2       | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0002_RDS                | 0                 | 0                | 0%                  | 0%                 |
| 3       | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0003_RDS                | 0                 | 0                | 0%                  | 0%                 |
| 4       | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0004_RDS                | 0                 | 0                | 0%                  | 0%                 |
| 5       | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0005_RDS                | 0                 | 0                | 0%                  | 0%                 |
| 6       | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0006_RDS                | 0                 | 0                | 0%                  | 0%                 |
| 7       | SEQ_TEST_1487767780814RGKKSEQ_TEST_WNJG_0007_RDS                | 0                 | 0                | 0%                  | 0%                 |
+-----+-----+-----+-----+-----+
+-----+
8 rows in set (0.01 sec)
```

7.7.3 SQL tuning statements

SHOW [FULL] SLOW [WHERE expr] [limit expr]

The SQL statements that take more than 1 second to execute are slow SQL statements. Slow logical SQL statements are the slow SQL statements sent from an application to a Distributed Relational Database Service (DRDS) instance.

- **SHOW SLOW:** You can run this SQL statement to view the 100 slowest logical SQL queries that are recorded since the DRDS instance is started up or the last time when `CLEAR SLOW` is executed.

**Note:**

The recorded 100 slowest SQL queries are stored in the DRDS system. When the DRDS instance is restarted or executes `CLEAR SLOW`, these queries will be discarded.

- **SHOW FULL SLOW:** You can run this SQL statement to view all the slow logical SQL queries that are recorded and persisted to the built-in database of the DRDS instance since the DRDS instance is started up. The upper limit for the number of records is specified in the specifications of the DRDS instance. The DRDS instance scrolls to delete the earliest slow SQL statements. If the specifications of the DRDS instance is 4-core 4 GB, a maximum of 10,000 slow SQL statements can be recorded, including slow logical and physical SQL statements. If the specifications of the DRDS instance is 8-core 8 GB, a maximum of 20,000 slow SQL statements can be recorded, including slow logical and physical SQL statements. The same rule applies to other specifications.

The following describes the meanings of important columns:

- **HOST:** the IP address of the host from which the SQL statement is sent.
- **START_TIME:** the time when the SQL statement started to be executed.
- **EXECUTE_TIME:** the time that the DRDS instance takes to run the SQL statement.
- **AFFECT_ROW:** for data manipulation language (DML) statements, this parameter indicates the number of affected rows. For query statements, this parameter indicates the number of returned records.

```
mysql> show slow where execute_time > 1000 limit 1;
+-----+-----+-----+-----+
+-----+
| HOST      | START_TIME          | EXECUTE_TIME | AFFECT_ROW | SQL
|          |                    |              |            |
+-----+-----+-----+-----+
+-----+
| 127.0.0.1 | 2016-03-16 13:02:57 |          2785 |           7 | show
rule |
+-----+-----+-----+-----+
+-----+
```

```
1 row in set (0.02 sec)
```

SHOW [FULL] PHYSICAL_SLOW [WHERE expr] [limit expr]

The SQL statements that take more than 1 second to execute are slow SQL statements. Slow logical SQL statements are the slow SQL statements sent from an application to a Distributed Relational Database Service (DRDS) instance.

- **SHOW SLOW:** You can run this SQL statement to view the 100 slowest logical SQL queries that are recorded since the DRDS instance is started up or the last time when `CLEAR SLOW` is executed.



Note:

The recorded 100 slowest SQL queries are stored in the DRDS system. When the DRDS instance is restarted or executes `CLEAR SLOW`, these queries will be discarded.

- **SHOW FULL SLOW:** You can run this SQL statement to view all the slow logical SQL queries that are recorded and persisted to the built-in database of the DRDS instance since the DRDS instance is started up. The upper limit for the number of records is specified in the specifications of the DRDS instance. The DRDS instance scrolls to delete the earliest slow SQL statements. If the specifications of the DRDS instance is 4-core 4 GB, a maximum of 10,000 slow SQL statements can be recorded, including slow logical and physical SQL statements. If the specifications of the DRDS instance is 8-core 8 GB, a maximum of 20,000 slow SQL statements can be recorded, including slow logical and physical SQL statements. The same rule applies to other specifications.

The following describes the meanings of important columns:

- **GROUP_NAME:** the name of the group to which the database that executes the SQL statement belongs.
- **START_TIME:** the time when the SQL statement started to be executed.
- **EXECUTE_TIME:** the time that the DRDS instance takes to run the SQL statement.
- **AFFECT_ROW:** for data manipulation language (DML) statements, this parameter indicates the number of affected rows. For query statements, this parameter indicates the number of returned records.

```
mysql> show physical_slow;
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

```

+-----+-----+-----+-----+
+-----+
| GROUP_NAME      | DBKEY_NAME      | START_TIME      |
| EXECUTE_TIME    | SQL_EXECUTE_TIME | GETLOCK_CONNECTION_TIME |
CREATE_CONNECTION_TIME | AFFECT_ROW | SQL
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| TDDL5_00_GROUP  | db218249098_sqa_zmf_tddl5_00_3309 | 2016-03-16 13:
05:38 |          1057 |          1011 |          0 |
          0 |          1 | select sleep(1) |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
+-----+
+-----+
+-----+
1 row in set (0.01 sec)

```

CLEAR SLOW

You can run this SQL statement to clear the 100 slowest logical SQL queries and the 100 slowest physical SQL queries that are recorded since the DRDS instance is started up or the last time when `CLEAR SLOW` is executed.



Note:

Both `SHOW SLOW` and `SHOW PHYSICAL_SLOW` can be executed to display the 100 slowest SQL statements. If `CLEAR SLOW` has not been executed for a long time, these SQL statements may have been recorded a long time ago. Therefore, after SQL tuning statements are executed, we recommend that you execute `CLEAR SLOW`, and after the system runs for a while, check the tuning results of slow SQL statements.

```
mysql> clear slow;
Query OK, 0 rows affected (0.00 sec)
```

EXPLAIN SQL

You can run this SQL statement to view the execution plan of a specified SQL statement in the DRDS. Note that this SQL statement is not truly executed.

Examples

You can run this SQL statement to view the execution plan of the SQL `select * from doctest` statement. The `doctest` table is stored in database shards according to values in the `id` column. According to the execution plan, the SQL statement will be routed to each database shard for execution, and the execution results will be aggregated.

```
mysql> explain select * from doctest;
```

```

+-----+
+-----+-----+-----+
| GROUP_NAME          | PARAMS | SQL
+-----+-----+-----+
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0000_RDS | select `doctest`.`id` from `doctest` | {}
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0001_RDS | select `doctest`.`id` from `doctest` | {}
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0002_RDS | select `doctest`.`id` from `doctest` | {}
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0003_RDS | select `doctest`.`id` from `doctest` | {}
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0004_RDS | select `doctest`.`id` from `doctest` | {}
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0005_RDS | select `doctest`.`id` from `doctest` | {}
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0006_RDS | select `doctest`.`id` from `doctest` | {}
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0007_RDS | select `doctest`.`id` from `doctest` | {}
+-----+-----+-----+
8 rows in set (0.00 sec)

```

You can run this SQL statement to view the execution plan of the SQL `select * from doctest where id = 1` statement. The `doctest` table is stored in database shards according to values in the `id` column. According to the execution plan, the DRDS instance will calculate a specified database shard based on the shard key, which is `id`, directly route the SQL statement to the database shard, and aggregate the execution results.

```

mysql> explain select * from doctest where id = 1;
+-----+
+-----+-----+-----+
| GROUP_NAME          | PARAMS | SQL
+-----+-----+-----+
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0001_RDS | select `doctest`.`id` from `doctest` where (`doctest`.`id` = 1) | {}
+-----+-----+-----+
1 row in set (0.01 sec)

```

EXPLAIN DETAIL SQL

You can run this SQL statement to view the execution plan of a specified SQL statement in the DRDS. Note that this SQL statement is not truly executed.

```
mysql> explain detail select * from doctest where id = 1;
```

```

+-----+
+-----+
+-----+
| GROUP_NAME          | SQL
+-----+
|                     | PARAMS |
+-----+
+-----+
| DOCTEST_1488704345426RCUPDOCTEST_CAET_0001_RDS | Query from doctest
as doctest
  keyFilter:doctest.id = 1
  queryConcurrency:SEQUENTIAL
  columns:[doctest.id]
  tableName:doctest
  executeOn:DOCTEST_1488704345426RCUPDOCTEST_CAET_0001_RDS
| NULL |
+-----+
+-----+
+-----+
1 row in set (0.02 sec)

```

EXPLAIN EXECUTE SQL

You can run this SQL statement to view the execution plan of underlying storage. This statement is equivalent to the MYSQL EXPLAIN statement.

```

mysql> explain execute select * from tddl_mgr_log limit 1;
+----+-----+-----+-----+-----+-----+-----+
| id | select_type | table      | type | possible_keys | key  |
key_len | ref  | rows | Extra |
+----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE      | tddl_mgr_log | ALL | NULL          | NULL | NULL
| NULL |      | 1 | NULL |
+----+-----+-----+-----+-----+-----+
+----+-----+-----+-----+
1 row in set (0.07 sec)

```

TRACE SQL and SHOW TRACE

You can run these SQL statements to view the execution results of an SQL statement . Note that you must use TRACE [SQL] and SHOW TRACE together. The difference between TRACE SQL and EXPLAIN SQL is that TRACE SQL is truly executed.

For example, you can run these statements to view the execution results of the select 1 statement.

```

mysql> trace select 1;
+----+
| 1 |
+----+
| 1 |
+----+
1 row in set (0.03 sec)
mysql> show trace;

```

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ID    | TYPE      | GROUP_NAME      | DBKEY_NAME
|      | TIME_COST(MS) | CONNECTION_TIME_COST(MS) | ROWS | STATEMENT |
PARAMS |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|      0 | Optimize  | DRDS            | DRDS
|      3 |           | 0.00           |      0 | select 1 | NULL
|      1 | Query     | TDDL5_00_GROUP | db218249098_sqa_zmf_tddl5_00_3309
|      7 |           | 0.15           |      1 | select 1 | NULL
|
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
2 rows in set (0.01 sec)

```

CHECK TABLE tablename

You can run this SQL statement to check a data table. This SQL statement is mainly used when a table failed to be created by using a DDL statement.

- **If the data table is a table shard, this SQL statement allows you to check whether any underlying physical table is missing and whether the column and index of the underlying physical table are consistent.**
- **If the data table is a single-database non-partition table, this SQL statement allows you to check whether this table exists.**

```

mysql> check table tddl_mgr_log;
+-----+-----+-----+-----+
| TABLE                | OP    | MSG_TYPE | MSG_TEXT |
+-----+-----+-----+-----+
| TDDL5_APP.tddl_mgr_log | check | status   | OK       |
+-----+-----+-----+-----+
1 row in set (0.56 sec)
mysql> check table tddl_mg;
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| TABLE                | OP    | MSG_TYPE | MSG_TEXT |
|                       |       |          |          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| TDDL5_APP.tddl_mg     | check | Error    | Table 'tddl5_00.tddl_mg'
|                       |       |          | doesn't exist |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
1 row in set (0.02 sec)

```

SHOW TABLE STATUS [LIKE 'pattern' | WHERE expr]

You can run this SQL statement to obtain the information about a table. This command aggregates the data of all underlying physical tables.

The following describes the meanings of important columns:

- **NAME:** The name of the table.
- **ENGINE:** The storage engine of the table.
- **VERSION:** The version of the storage engine of the table.
- **ROW_FORMAT:** The format of the rows in the table. Valid values include Dynamic, Fixed, and Compressed. The value Dynamic indicates that the row length is variable, for example, is a VARCHAR or BLOB field. The value Fixed indicates that the row length is constant, for example, is a CHAR or INTEGER field.
- **ROWS:** the number of rows in the table.
- **AVG_ROW_LENGTH:** the average number of bytes in each row.
- **DATA_LENGTH:** the data volume of the entire table. Unit: bytes
- **MAX_DATA_LENGTH:** the maximum volume of data that can be stored in the table.
- **INDEX_LENGTH:** the size of the disk space occupied by indexes.
- **CREATE_TIME:** the time when the table was created.
- **UPDATE_TIME:** the time when the table was last updated.
- **COLLATION:** the default character set and character sorting rule of the table.
- **CREATE_OPTIONS:** all the other options specified when the table was created.

```
mysql> show table status like 'multi_db_multi_tbl';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| NAME                | ENGINE | VERSION | ROW_FORMAT | ROWS |
AVG_ROW_LENGTH | DATA_LENGTH | MAX_DATA_LENGTH | INDEX_LENGTH |
DATA_FREE | AUTO_INCREMENT | CREATE_TIME          | UPDATE_TIME |
CHECK_TIME | COLLATION          | CHECKSUM | CREATE_OPTIONS | COMMENT |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| multi_db_multi_tbl | InnoDB | 10      | Compact    | 2    |
16384 | 16384 | 0      | 16384 | 0 |
100000 | 2017-03-27 17:43:57.0 | NULL | NULL |
utf8_general_ci | NULL | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

1 row in set (0.03 sec)

The combination of the SHOW TABLE STATUS statement and the DRDS SCAN hint allow you to view the data volume of each physical table shard.

```
mysql> /*! TDDL:SCAN='multi_db_multi_tbl'*/show table status like '
multi_db_multi_tbl';
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+
| Name          | Engine | Version | Row_format | Rows |
Avg_row_length | Data_length | Max_data_length | Index_length |
Data_free | Auto_increment | Create_time      | Update_time |
Check_time | Collation      | Checksum | Create_options | Comment |
Block_format |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+
| multi_db_multi_tbl_1 | InnoDB | 10 | Compact | 0 |
| 0 | 16384 | 0 | 16384 | 0 |
| 1 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | | Original |
| multi_db_multi_tbl_0 | InnoDB | 10 | Compact | 0 |
| 0 | 16384 | 0 | 16384 | 0 |
| 1 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | | Original |
| multi_db_multi_tbl_1 | InnoDB | 10 | Compact | 0 |
| 0 | 16384 | 0 | 16384 | 0 |
| 1 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | | Original |
| multi_db_multi_tbl_0 | InnoDB | 10 | Compact | 1 |
| 16384 | 16384 | 0 | 16384 | 0 |
| 2 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | | Original |
| multi_db_multi_tbl_1 | InnoDB | 10 | Compact | 0 |
| 0 | 16384 | 0 | 16384 | 0 |
| 1 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | | Original |
| multi_db_multi_tbl_0 | InnoDB | 10 | Compact | 0 |
| 0 | 16384 | 0 | 16384 | 0 |
| 1 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | | Original |
| multi_db_multi_tbl_1 | InnoDB | 10 | Compact | 0 |
| 0 | 16384 | 0 | 16384 | 0 |
| 1 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | | Original |
| multi_db_multi_tbl_0 | InnoDB | 10 | Compact | 0 |
| 0 | 16384 | 0 | 16384 | 0 |
| 1 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | | Original |
| multi_db_multi_tbl_0 | InnoDB | 10 | Compact | 0 |
| 0 | 16384 | 0 | 16384 | 0 |
| 1 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | | Original |
```

```

| multi_db_multi_tbl_1 | InnoDB | 10 | Compact | 0 |
| 0 | 16384 | 0 | 16384 | 0 |
| 1 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | Original |
| multi_db_multi_tbl_0 | InnoDB | 10 | Compact | 0 |
| 0 | 16384 | 0 | 16384 | 0 |
| 1 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | Original |
| multi_db_multi_tbl_1 | InnoDB | 10 | Compact | 0 |
| 0 | 16384 | 0 | 16384 | 0 |
| 1 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | Original |
| multi_db_multi_tbl_0 | InnoDB | 10 | Compact | 0 |
| 0 | 16384 | 0 | 16384 | 0 |
| 1 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | Original |
| multi_db_multi_tbl_1 | InnoDB | 10 | Compact | 0 |
| 0 | 16384 | 0 | 16384 | 0 |
| 1 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | Original |
| multi_db_multi_tbl_0 | InnoDB | 10 | Compact | 1 |
| 16384 | 16384 | 0 | 16384 | 0 |
| 3 | 2017-03-27 17:43:57 | NULL | NULL |
utf8_general_ci | NULL | Original |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+
16 rows in set (0.04 sec)

```

7.7.4 Statistics query statements

SHOW [FULL] STATS

You can run this SQL statement to view the overall statistics of a Distributed Relational Database Service (DRDS) instance. The statistics are instantaneous values. Note that the results of DRDS SHOW FULL STATS vary with the version of the DRDS instance.

The following describes the meanings of important columns:

- **QPS:** the number of queries per second (QPS) sent from an application to the DRDS instance. These queries are usually called logical QPS.
- **RDS_QPS:** the number of QPS sent from the DRDS instance to an ApsaraDB RDS for MySQL instance. These queries are usually called physical QPS.
- **ERROR_PER_SECOND:** the total number of errors that occur on the DRDS instance per second. These errors include various errors such as SQL syntax errors, primary key conflicts, system errors, and connectivity errors.
- **VIOLATION_PER_SECOND:** the number of conflicts that occur on primary keys or unique keys per second.

- **MERGE_QUERY_PER_SECOND**: the number of queries processed on multiple tables through database sharding and table sharding per second.
- **ACTIVE_CONNECTIONS**: the number of active connections to the DRDS instance.
- **CONNECTION_CREATE_PER_SECOND**: the number of connections that are created for the DRDS instance per second.
- **RT(MS)**: the time to respond to an SQL query sent from an application to the DRDS instance. This response time (RT) is usually called logical RT.
- **RDS_RT(MS)**: the time to respond to an SQL query sent from the DRDS instance to an ApsaraDB RDS for MySQL instance. This RT is usually called physical RT.
- **NET_IN(KB/S)**: the amount of inbound traffic of the DRDS instance per second.
- **NET_OUT(KB/S)**: the amount of outbound traffic of the DRDS instance per second
-
- **THREAD_RUNNING**: the number of threads that are running in the DRDS instance.
- **HINT_USED_PER_SECOND**: the number of SQL queries that contain hints and are processed by the DRDS instance per second.
- **HINT_USED_COUNT**: the total number of SQL queries that contain hints and have been processed by the DRDS instance since startup.
- **AGGREGATE_QUERY_PER_SECOND**: the number of aggregate SQL queries processed by the DRDS instance per second.
- **AGGREGATE_QUERY_COUNT**: the total number of aggregate SQL queries that have been processed by the DRDS instance.
- **TEMP_TABLE_CREATE_PER_SECOND**: the number of temporary tables created in the DRDS instance per second.
- **TEMP_TABLE_CREATE_COUNT**: the total number of temporary tables that have been created in the DRDS instance since startup.
- **MULTI_DB_JOIN_PER_SECOND**: the number of multi-database JOIN queries processed by the DRDS instance per second.
- **MULTI_DB_JOIN_COUNT**: the number of multi-database JOIN queries that have been processed by the DRDS instance since startup.

```
mysql> show stats;
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| QPS | RDS_QPS | SLOW_QPS | PHYSICAL_SLOW_QPS | ERROR_PER_SECOND |
MERGE_QUERY_PER_SECOND | ACTIVE_CONNECTIONS | RT(MS) | RDS_RT(MS) |
NET_IN(KB/S) | NET_OUT(KB/S) | THREAD_RUNNING |
```

```

+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
| 1.77 | 1.68 | 0.03 | 0.03 | 0.02 |
| 134.49 | 0.00 | 1.48 | 7 | 157.13 | 51.14 |
| 1 |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)
mysql> show full stats;
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
| QPS | RDS_QPS | SLOW_QPS | PHYSICAL_SLOW_QPS | ERROR_PER_SECOND
| VIOLATION_PER_SECOND | MERGE_QUERY_PER_SECOND | ACTIVE_CONNECTIONS
| CONNECTION_CREATE_PER_SECOND | RT(MS) | RDS_RT(MS) | NET_IN(
KB/S) | NET_OUT(KB/S) | THREAD_RUNNING | HINT_USED_PER_SECOND |
HINT_USED_COUNT | AGGREGATE_QUERY_PER_SECOND | AGGREGATE_QUERY_COUNT
| TEMP_TABLE_CREATE_PER_SECOND | TEMP_TABLE_CREATE_COUNT | MULTI_DB_J
OIN_PER_SECOND | MULTI_DB_JOIN_COUNT | CPU | FREEMEM | FULLGCCOUNT
| FULLGCTIME |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
| 1.63 | 1.68 | 0.03 | 0.03 | 0.02 | |
| 0.00 | 0.01 | 157.13 | 0.00 | 51.14 | 134.33 |
| 1.21 | 1 | 0.00 | 6 |
| 0.00 | 0.00 | 663 | 54 |
| 0.00 | 512 | 0.00 |
| 516 | 0.09% | 6.96% | 76446 | 21326906 |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

```

SHOW DB STATUS

You can run this SQL statement to view the capacity and performance information of a physical database, which is also called a database shard. All the returned values indicate the real-time information. The capacity information is obtained from the ApsaraDB RDS for MySQL system table, and therefore may be different from the actual capacity information.

The following describes the meanings of important columns:

- **NAME:** the internal tag that represents a logical database corresponding to the database shard. The value is different from the name of the logical database.
- **CONNECTION_STRING:** the information about a connection from the DRDS instance to the database shard.
- **PHYSICAL_DB:** the name of the database shard. The **TOTAL** row indicates the total amount of capacity of all the database shards corresponding to the logical database.
- **SIZE_IN_MB:** the size of the space occupied by the data in the database shard.
Unit: MB
- **RATIO:** the ratio of the data volume of the database shard to the total data volume of the current logical database.
- **THREAD_RUNNING:** the number of threads that are running in the ApsaraDB RDS for MySQL instance to which the physical database belongs. The meaning of this parameter is the same as that of the **THREAD_RUNNING** parameter returned by the MySQL `SHOW GLOBAL STATUS` command. For more information, see [MySQL Documentation](#).

```
mysql> show db status;
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| ID    | NAME                                     | CONNECTION_STRING | PHYSICAL_DB
|      | SIZE_IN_MB | RATIO | THREAD_RUNNING |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| 1    | drds_db_1516187088365dau | 100.100.64.1:59077 | TOTAL
|      | 13.109375 | 100%  | 3              |
| 2    | drds_db_1516187088365dau | 100.100.64.1:59077 | drds_db_xz
ip_0000 | 1.578125 | 12.04% |                |
| 3    | drds_db_1516187088365dau | 100.100.64.1:59077 | drds_db_xz
ip_0001 | 1.4375  | 10.97% |                |
| 4    | drds_db_1516187088365dau | 100.100.64.1:59077 | drds_db_xz
ip_0002 | 1.4375  | 10.97% |                |
| 5    | drds_db_1516187088365dau | 100.100.64.1:59077 | drds_db_xz
ip_0003 | 1.4375  | 10.97% |                |
| 6    | drds_db_1516187088365dau | 100.100.64.1:59077 | drds_db_xz
ip_0004 | 1.734375 | 13.23% |                |
| 7    | drds_db_1516187088365dau | 100.100.64.1:59077 | drds_db_xz
ip_0005 | 1.734375 | 13.23% |                |
| 8    | drds_db_1516187088365dau | 100.100.64.1:59077 | drds_db_xz
ip_0006 | 2.015625 | 15.38% |                |
| 9    | drds_db_1516187088365dau | 100.100.64.1:59077 | drds_db_xz
ip_0007 | 1.734375 | 13.23% |                |
```

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

SHOW FULL DB STATUS [LIKE {tablename}]

You can run this SQL statement to view the capacity and performance information of a table shard in a physical database, which is also called a database shard. All the returned values indicate the real-time information. The capacity information is obtained from the ApsaraDB RDS for MySQL system table, and therefore may be different from the actual capacity information.

The following describes the meanings of important columns:

- **NAME:** the internal tag that represents a logical database corresponding to the database shard. The value is different from the name of the logical database.
- **CONNECTION_STRING:** the information about a connection from the DRDS instance to the database shard.
- **PHYSICAL_DB:** the name of the database shard. If the LIKE keyword is used for filtering in the statement, the TOTAL row indicates the total amount of capacity of the database shard. If the LIKE keyword is not used for filtering in the statement, the TOTAL row indicates the total amount of capacity of all database shards.
- **PHYSICAL_TABLE:** the name of the table shard in the database shard. If the LIKE keyword is used for filtering in the statement, the TOTAL row indicates the total amount of capacity of the table shard. If the LIKE keyword is not used for filtering in the statement, the TOTAL row indicates the total amount of capacity of all table shards in the database shard.
- **SIZE_IN_MB:** the size of the space occupied by the data in the database shard.
Unit: MB
- **RATIO:** the ratio of the data volume of the table shard to the total data volume of all the table shards obtained through filtering.
- **THREAD_RUNNING:** the number of threads that are running in the ApsaraDB RDS for MySQL instance to which the physical database belongs. The meaning of this parameter is the same as that of the THREAD_RUNNING parameter returned by the MySQL SHOW GLOBAL STATUS command. For more information, see [MySQL Documentation](#).

```
mysql> show full db status like hash_tb;
```

ID	NAME	CONNECTION_STRING	PHYSICAL_DB
	PHYSICAL_TABLE	SIZE_IN_MB	RATIO
		THREAD_RUNNING	
1	drds_db_1516187088365dau	100.100.64.1:59077	TOTAL
		19.875	100%
		3	
2	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0000	TOTAL	3.03125	15.25%
3	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0000	hash_tb_00	1.515625	7.63%
4	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0000	hash_tb_01	1.515625	7.63%
5	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0001	TOTAL	2.0	10.06%
6	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0001	hash_tb_02	1.515625	7.63%
7	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0001	hash_tb_03	0.484375	2.44%
8	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0002	TOTAL	3.03125	15.25%
9	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0002	hash_tb_04	1.515625	7.63%
10	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0002	hash_tb_05	1.515625	7.63%
11	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0003	TOTAL	1.953125	9.83%
12	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0003	hash_tb_06	1.515625	7.63%
13	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0003	hash_tb_07	0.4375	2.2%
14	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0004	TOTAL	3.03125	15.25%
15	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0004	hash_tb_08	1.515625	7.63%
16	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0004	hash_tb_09	1.515625	7.63%
17	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0005	TOTAL	1.921875	9.67%
18	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0005	hash_tb_11	1.515625	7.63%
19	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0005	hash_tb_10	0.40625	2.04%
20	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0006	TOTAL	3.03125	15.25%
21	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0006	hash_tb_12	1.515625	7.63%
22	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0006	hash_tb_13	1.515625	7.63%
23	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0007	TOTAL	1.875	9.43%
24	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0007	hash_tb_14	1.515625	7.63%
25	drds_db_1516187088365dau	100.100.64.1:59077	drds_db_xz
ip_0007	hash_tb_15	0.359375	1.81%

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

7.7.5 SHOW PROCESSLIST and KILL commands



Note:

- **Distributed Relational Database Service (DRDS) 5.1.28-1408022 and later support the SHOW PROCESSLIST and KILL commands for both logical and physical connections. For more information, see this topic.**
- **DRDS instances earlier than 5.1.28-1408022 only support the SHOW PROCESSLIST and KILL commands for physical connections. For more information, see [SHOW PROCESSLIST and KILL commands in earlier versions](#).**

SHOW PROCESSLIST command

In a DRDS instance, you can run the SHOW PROCESSLIST command to view the information such as connections to the DRDS instance and SQL statements that are being executed in the DRDS instance.

Syntax

```
SHOW [FULL] PROCESSLIST
```

Examples

```
mysql> SHOW PROCESSLIST\G
  ID: 1971050
  USER: admin
  HOST: 111.111.111.111:4303
  DB: drds_test
  COMMAND: Query
  TIME: 0
  STATE:
  INFO: show processlist
1 row in set (0.01 sec)
```

The following describes the meanings of the fields in the result set:

- **ID:** the ID of the connection. The value is a long-type number.
- **USER:** the name of the user who sets up the connection.
- **HOST:** the IP address and port number of the host that sets up the connection.
- **DB:** the name of the database accessed by the connection.

- **COMMAND:** the usage state of the connection. Currently, this field can be set to the following values:
 - **Query:** this value indicates that the current connection is executing an SQL statement.
 - **Sleep:** this value indicates that the current connection is idle.
- **TIME:** the duration when the connection is in the current state:
 - When the value of **COMMAND** is **Query**, this field indicates how long the SQL statement has been being executed on the connection.
 - When the value of **COMMAND** is **Sleep**, this parameter indicates how long the connection has been in idle state.
- **STATE:** currently, no meaning has been assigned for this field. The value is constantly empty.
- **INFO:**
 - When the value of **COMMAND** is **Query**, this field indicates the content of the SQL statement that is being executed on the connection. If the **FULL** parameter is not specified, a maximum of the first 30 characters of the SQL statement are returned. If the **FULL** parameter is specified, a maximum of the first 1000 characters of the SQL statement are returned.
 - When the value of **COMMAND** is other values, this field is meaningless and left empty.

SHOW PHYSICAL_PROCESSLIST command

In a DRDS instance, you can run the **SHOW PHYSICAL_PROCESSLIST** command to view the information about all the SQL statements that are being executed on underlying ApsaraDB RDS for MySQL instances.

Syntax

```
SHOW [FULL] PHYSICAL_PROCESSLIST
```

When an SQL statement is excessively long, the responses of the **SHOW PHYSICAL_PROCESSLIST** command may be truncated. In this case, you can run the **SHOW FULL PHYSICAL_PROCESSLIST** command to obtain the complete SQL statement.

The meaning of each column in the responses is equivalent to that in the responses of the **SHOW PROCESSLIST** command. For more information, see [SHOW PROCESSLIST Statement](#).

**Note:**

Different from ApsaraDB RDS for MySQL, the DRDS instance returns a string instead of a number in the ID column of a physical connection.

```
mysql> SHOW PHYSICAL_PROCESSLIST\G
***** 1. row *****
      ID: 0-0-521414
      USER: tddl5
      DB: tddl5_00
      COMMAND: Query
      TIME: 0
      STATE: init
      INFO: show processlist
***** 2. row *****
      ID: 0-0-521570
      USER: tddl5
      DB: tddl5_00
      COMMAND: Query
      TIME: 0
      STATE: User sleep
      INFO: /*DRDS /88.88.88.88/b67a0e4d8800000/ */ select sleep(
1000)
2 rows in set (0.01 sec)
```

KILL command

The KILL command is used to terminate an SQL statement that is being executed.

The DRDS instance connects to an ApsaraDB RDS for MySQL instance by using the username created by the DRDS instance on the ApsaraDB RDS for MySQL instance. Therefore, if you directly connect to the ApsaraDB RDS for MySQL instance, you do not have the permission to run the KILL command on a request initiated by the DRDS instance.

To terminate an SQL statement that is being executed on the DRDS instance, you must use tools such as the MySQL command line and to connect to the DRDS instance, and then run the KILL command on the DRDS instance.

Syntax

```
KILL PROCESS_ID | 'PHYSICAL_PROCESS_ID' | 'ALL'
```

The KILL command can be used in the following ways:

- Run `KILL PROCESS_ID` to terminate a specified logical SQL statement.

The `PROCESS_ID` parameter is obtained from the `ID` column in the responses of the `SHOW [FULL] PROCESSLIST` command.

Running the `KILL PROCESS_ID` command in the DRDS instance will terminate both logical and physical SQL statements that are being executed for this connection, and disconnect this connection.

The DRDS instance does not support the `KILL QUERY` command.

- Run `KILL 'PHYSICAL_PROCESS_ID'` to terminate a specified physical SQL statement.

The `PHYSICAL_PROCESS_ID` parameter is obtained from the `ID` column in the responses of the `SHOW PHYSICAL_PROCESS_ID` command.



Note:

The `PHYSICAL_PROCESS_ID` column is a string instead of a number. Therefore, the `PHYSICAL_PROCESS_ID` parameter must be enclosed in single quotation marks (") in the `KILL` command.

Examples

```
mysql> KILL '0-0-521570';
Query OK, 0 rows affected (0.01 sec)
```

- Run `KILL 'ALL'` to terminate all the physical SQL statements that are executed by the DRDS instance in the current logical database.

When the underlying ApsaraDB RDS for MySQL instance is overloaded due to some SQL statements, you can use the `KILL 'ALL'` command to terminate all the physical SQL statements that are being executed in the current logical database.

All physical SQL statements indicated by `PROCESS` that meet the following conditions can be terminated by running `KILL 'ALL'`:

- The value of the `User` parameter for the physical SQL statement indicated by `PROCESS` is a username created by the DRDS instance in the ApsaraDB RDS for MySQL instance.
- The physical SQL statement indicated by `PROCESS` is executing a query. In other words, the value of `COMMAND` is `Query`.

7.7.6 SHOW PROCESSLIST and KILL commands in earlier versions



Note:

- Distributed Relational Database Service (DRDS) 5.1.28-1408022 and later support the `SHOW PROCESSLIST` and `KILL` commands for both logical and physical connections. For more information, see [SHOW PROCESSLIST and KILL commands](#).
- DRDS instances earlier than 5.1.28-1408022 only support the `SHOW PROCESSLIST` and `KILL` commands for physical connections. For more information, see [this topic](#).

SHOW PROCESSLIST command

In a DRDS instance, you can run the `SHOW PROCESSLIST` command to view the information about all the SQL statements that are being executed on the ApsaraDB RDS for MySQL instances.

Syntax

```
SHOW [FULL] PROCESSLIST
```

When an SQL statement is excessively long, the responses of the `SHOW PROCESSLIST` command may be truncated. In this case, you can run the `SHOW FULL PROCESSLIST` command to obtain the complete SQL statement.

The meaning of each column in the responses is equivalent to that in the responses of the `SHOW PROCESSLIST` command. For more information, see [SHOW PROCESSLIST Statement](#).

```
mysql> SHOW PROCESSLIST\G
***** 1. row *****
      ID: 0-0-521414
      USER: tddl5
      DB: tddl5_00
      COMMAND: Query
      TIME: 0
      STATE: init
      INFO: show processlist
      ROWS_SENT: NULL
      ROWS_EXAMINED: NULL
      ROWS_READ: NULL
***** 2. row *****
      ID: 0-0-521570
      USER: tddl5
      DB: tddl5_00
```

```
COMMAND: Query
TIME: 0
STATE: User sleep
INFO: /*DRDS /88.88.88.88/b67a0e4d8800000/ */ select sleep(
1000)
ROWS_SENT: NULL
ROWS_EXAMINED: NULL
ROWS_READ: NULL
2 rows in set (0.01 sec)
```

KILL command

The **KILL** command is used to terminate an SQL statement that is being executed.

The DRDS instance connects to an ApsaraDB RDS for MySQL instance by using the username created by the DRDS instance on the ApsaraDB RDS for MySQL instance. Therefore, if you directly connect to the ApsaraDB RDS for MySQL instance, you do not have the permission to run the **KILL** command on a request initiated by the DRDS instance.

To terminate an SQL statement that is being executed on the DRDS instance, you must use tools such as the MySQL command line and to connect to the DRDS instance, and then run the **KILL** command on the DRDS instance.

Syntax

```
KILL 'PROCESS_ID' | 'ALL'
```

The **KILL** command can be used in the following ways::

- Run **KILL 'PROCESS_ID'** to terminate a specified SQL statement.

The **PROCESS_ID** parameter is obtained from the **ID** column in the responses of the **SHOW PROCESSLIST** command.



Note:

Different from ApsaraDB RDS for MySQL, the DRDS instance returns a string instead of a number in the **ID** column. Therefore, the **PROCESS_ID** parameter must be enclosed in single quotation marks (") in the **KILL** command.

Examples

```
mysql> KILL '0-0-521570';
```

```
Query OK, 0 rows affected (0.01 sec)
```

- Run `KILL 'ALL'` to terminate all the SQL statements executed by the DRDS instance in the current logical database.

When the underlying ApsaraDB RDS for MySQL instance is overloaded due to several SQL statements, you can use the `KILL 'ALL'` command to terminate all the SQL statements that are being executed in the current logical database.

All SQL statements indicated by `PROCESS` that meet the following conditions can be terminated by running `KILL 'ALL'`:

- The value of the `User` parameter for the physical SQL statement indicated by `PROCESS` is a username created by the DRDS instance in the ApsaraDB RDS for MySQL instance.
- The physical SQL statement indicated by `PROCESS` is executing a query. In other words, the value of `COMMAND` is `Query`.

DRDS instances in earlier versions do not support the `KILL 'ALL'` command. An error will be reported if this command is being executed in these instances. To resolve this problem, you can upgrade the version of the DRDS instance.

7.8 Custom hints



Note:

This topic is applicable to Distributed Relational Database Service (DRDS) 5.3 and later. For other versions of DRDS instances, see [Custom hints for DRDS 5.2](#).

7.8.1 Introduction to hints

As a supplement to the SQL syntax, hints play a critical role in relational databases. They allow you to influence execution plans of SQL statements by using relevant

syntax, to specially optimize the SQL statements. Distributed Relational Database Service (DRDS) also provides special hint syntax.

For example, if you know the target data is stored in table shards in certain database shards and you need to route the SQL statement directly to the database shards for execution, you can use custom hints provided by DRDS.

```
SELECT /*+TDDL:node('node_name')*/ * FROM table_name;
```

In the preceding SQL statement, the part between `/*` and `*/`, namely, `+TDDL:node('node_name')`, is a DRDS hint. The hint specifies the ApsaraDB RDS for MySQL database shard where the SQL statement is to be executed.



Note:

- DRDS hints can be in the formats of `/*+TDDL:hint_command*/` and `/*! +TDDL:hint_command*/`.
- In the MySQL command line, if you need to run an SQL statement that contains a DRDS hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because DRDS hints are based on the [MySQL Comment Syntax](#). Otherwise, the client deletes the DRDS hint and then sends the SQL statement to the server for execution, which causes the DRDS hint to fail to take effect. For more information, see [MySQL Client Options](#).

Syntax of DRDS hints

Basic syntax

```
/*+TDDL: hint_command [hint_command ...] */
/*! +TDDL: hint_command [hint_command ...] */
```

DRDS hints are based on the [MySQL Comment Syntax](#). The hint statements are located between `/*` and `*/` or between `/*!` and `*/`, and must begin with `+TDDL:.` The `hint_command` parameter indicates a DRDS hint command related to the specific operation. Multiple `hint_command` parameters are separated by spaces.

Examples

```
# Query the names of physical tables in each database shard
/*+TDDL:scan()*/SHOW TABLES;
# Route the query to database shard 0000 of a read-only ApsaraDB RDS
for MySQL instance
```

```
/*+TDDL:node(0) slave()*/SELECT * FROM t1;
```

In the example, `/*+TDDL:scan()*/` and `/*+TDDL:node(0) slave()*/` are DRDS hints that begin with `+TDDL:.` The `scan()`, `node(0)`, and `slave()` functions are DRDS hint commands. Hint commands are separated by spaces.

- Use one hint in an SQL statement:

DRDS allows you to use hints in data manipulation language (DML), data definition language (DDL), and data access language (DAL) statements. The following describes the syntax in detail.

- For all statements that support hints, you can specify a hint at the beginning of the statements, for example:

```
/*+TDDL: ... */ SELECT ...
/*+TDDL: ... */ INSERT ...
/*+TDDL: ... */ REPLACE ...
/*+TDDL: ... */ UPDATE ...
/*+TDDL: ... */ DELETE ...
/*+TDDL: ... */ CREATE TABLE ...
/*+TDDL: ... */ ALTER TABLE ...
/*+TDDL: ... */ DROP TABLE ...
/*+TDDL: ... */ SHOW ...
...
```

- For DML statements, you can specify a hint behind the first keyword of the statements, for example:

```
SELECT /*+TDDL: ... */ ...
INSERT /*+TDDL: ... */ ...
REPLACE /*+TDDL: ... */ ...
UPDATE /*+TDDL: ... */ ...
DELETE /*+TDDL: ... */ ...
...
```



Note:

Different hints may be applicable to different syntaxes. For more information about the applicable syntaxes, see the documentation of hint commands.

- Use multiple hint commands in an SQL statement:

DRDS allows you to use multiple hint commands in an SQL statement that contains a hint.

```
SELECT /*+TDDL:node(0) slave()*/ ... ;
```

DRDS has the following limitations on the use of multiple hint commands:

```
# A single SQL statement cannot contain multiple hint statements.
```

```
SELECT /*+TDDL:node(0)*/ /*+TDDL:slave()*/ ... ;
# An SQL statement that contains a hint cannot contain duplicate
hint commands.
SELECT /*+TDDL:node(0) node(1)*/ ... ;
```

Classification of DRDS hints

DRDS hints are classified into the following major categories according to operation types:

- *Read/write splitting*
- *Specify a timeout period for an SQL statement*
- *Specify a database shard to run an SQL statement*
- *Scan all or some of database shards and table shards*

7.8.2 Read/write splitting

Distributed Relational Database Service (DRDS) provides transparent read/write splitting at the application layer. However, data synchronization between primary and read-only ApsaraDB RDS for MySQL instances has a delay of several milliseconds. If you need to read changed data immediately after the primary ApsaraDB RDS for MySQL instance is changed, you must ensure that the SQL statement for reading data is routed to the primary ApsaraDB RDS for MySQL instance. To meet this demand, DRDS provides custom hints for read/write splitting, to route SQL statements to a specified primary or read-only ApsaraDB RDS for MySQL instance.



Note:

This topic is applicable to DRDS 5.3 and later. For earlier versions, see [Read/write splitting](#).

Syntax

```
/*+TDDL:
    master()
    | slave()
*/
```

With this custom hint, you can specify whether to run an SQL statement on a primary or read-only ApsaraDB RDS for MySQL instance. With the custom hint `/*+TDDL:slave()*/`, if a primary ApsaraDB RDS for MySQL instance is configured with multiple read-only ApsaraDB RDS for MySQL instances, the DRDS instance randomly selects a read-only ApsaraDB RDS for MySQL instance based on its weight, to run the SQL statement.

**Note:**

- DRDS hints can be in the formats of `/*+TDDL:hint_command*/` and `/*! +TDDL:hint_command*/`.
- In the MySQL command line, if you need to run an SQL statement that contains a DRDS hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because DRDS hints are based on the *MySQL Comment Syntax*. Otherwise, the client deletes the DRDS hint and then sends the SQL statement to the server for execution, which causes the DRDS hint to fail to take effect. For more information, see *MySQL Client Options*.

Examples

- Specify a primary ApsaraDB RDS for MySQL instance to run an SQL statement:

```
SELECT /*+TDDL:master()*/ * FROM table_name;
```

After the custom hint `/*+TDDL:master()*/` is added behind the first key word in the SQL statement, this SQL statement is routed to the primary ApsaraDB RDS for MySQL instance for execution.

- Specify a read-only ApsaraDB RDS for MySQL instance to run an SQL statement:

```
SELECT /*+TDDL:slave()*/ * FROM table_name;
```

After the custom hint `/*+TDDL:slave()*/` is added behind the first key word in the SQL statement, this SQL statement is randomly routed to a read-only ApsaraDB RDS for MySQL instance based on the allocated weight.

Note

- The custom hints for read-write splitting are only applicable to read SQL statements for non-transactional data. SQL statements for transactional data and write SQL statements are still routed to the primary ApsaraDB RDS for MySQL instance for execution.
- The DRDS hint `/*+TDDL:slave()*/` allows you to randomly select a read-only ApsaraDB RDS for MySQL instance based on the allocated weight, to route the SQL statement to the read-only RDS instance for execution. If no read-only ApsaraDB RDS for MySQL instance is available, no error is reported. Instead, the primary ApsaraDB RDS for MySQL instance is selected to run the SQL statement.

7.8.3 Specify a timeout period for an SQL statement

In Distributed Relational Database Service (DRDS), the SQL statements for DRDS instances and ApsaraDB RDS for MySQL instances are timed out after 900 seconds (which can be adjusted) by default. However, for some slow SQL statements, the timeout period may exceed 900 seconds. For these slow SQL statements, DRDS provides a custom hint to adjust their timeout periods. You can use this custom hint to adjust the SQL execution duration as needed.



Note:

This topic is applicable to DRDS 5.3 and later. For earlier versions, see [Specify a timeout period for an SQL statement](#).

Syntax

The syntax of the DRDS hint for specifying a timeout period for an SQL statement is as follows:

```
/*+TDDL:SOCKET_TIMEOUT(time)*/
```

The `SOCKET_TIMEOUT` parameter is measured in milliseconds. With this custom hint, you can adjust the timeout period for the SQL statement based on business requirements.



Note:

- DRDS hints can be in the formats of `/*+TDDL:hint_command*/` and `/*!+TDDL:hint_command*/`.
- In the MySQL command line, if you need to run an SQL statement that contains a DRDS hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because DRDS hints are based on the [MySQL Comment Syntax](#). Otherwise, the client deletes the DRDS hint and then sends the SQL statement to the server for execution, which causes the DRDS hint to fail to take effect. For more information, see [MySQL Client Options](#).

Examples

Set the timeout period of an SQL statement to 40 seconds:

```
/*+TDDL:SOCKET_TIMEOUT(40000)*/SELECT * FROM t_item;
```

**Note:**

A longer timeout period causes database resources to be occupied for a longer period of time. If excessive SQL statements are executed over a long time within the same period, a large amount of database resources may be consumed. This will make users unable to use DRDS properly. In this case, we need to use this custom hint to optimize the SQL statements that take a long time to execute.

7.8.4 Specify a database shard to run an SQL statement

When running SQL commands in a Distributed Relational Database Service (DRDS) instance, you may find that some SQL statements are not supported by the DRDS instance. In this case, you can use the `NODE` hint provided by DRDS, to route the SQL statements to one or more database shards for execution. In addition, if you need to query the data in a specified database shard or the data in a specified table shard in a known database shard, you can use the `NODE` hint to directly route the SQL statement to the database shard for execution.

**Note:**

This topic is applicable to DRDS 5.3 and later. For earlier versions, see [Specify a database shard to run an SQL statement](#).

Syntax

The `NODE` hint allows you to specify a database shard by using a shard name, to run the SQL statement in the database shard. A shard name uniquely identifies a database shard in a DRDS instance. You can run the `SHOW NODE` statement to obtain the shard name.

Specify a database shard by using a shard name, to run an SQL statement

This custom hint allows you to specify one or more database shards to run an SQL statement.

**Note:**

If the hint for specifying a database shard is used in an INSERT statement that contains a DRDS sequence for the target table, the DRDS sequence will not take effect. For more information, see [Limits and precautions for DRDS sequence](#).

- Specify one database shard to run an SQL statement:

```
/*+TDDL:node('node_name')*/
```

Specifically, `node_name` indicates the shard name. This DRDS hint enables you to route the SQL statement to the database shard specified by `node_name`.

- Specify multiple database shards to run an SQL statement:

```
/*+TDDL:node('node_name' [, 'node_name1' , 'node_name2'])*/
```

You can specify multiple shard names in the parameters and route the SQL statement to multiple database shards for execution. The shard names are separated by commas (,).



Note:

- When this custom hint is used, the DRDS instance directly routes the SQL statement to the specified database shards for execution. Therefore, the specified shard names in the SQL statement must correspond to existing database shards.
- The `NODE` hint can be used in data manipulation language (DML), data definition language (DDL), and data access language (DAL) statements.
- DRDS hints can be in the formats of `/*+TDDL:hint_command*/` and `/*! +TDDL:hint_command*/`.
- In the MySQL command line, if you need to run an SQL statement that contains a DRDS hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because DRDS hints are based on the [MySQL Comment Syntax](#). Otherwise, the client deletes the DRDS hint and then sends the SQL statement to the server for execution, which causes the DRDS hint to fail to take effect. For more information, see [MySQL Client Options](#).

Examples

The following shows the responses of the SHOW NODE statement for a logical database named `drds_test` in a DRDS instance.

```
mysql> SHOW NODE\G
***** 1. row *****
      ID: 0
      NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0000_RDS
  MASTER_READ_COUNT: 212
  SLAVE_READ_COUNT: 0
  MASTER_READ_PERCENT: 100%
  SLAVE_READ_PERCENT: 0%
***** 2. row *****
      ID: 1
      NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0001_RDS
  MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
  MASTER_READ_PERCENT: 100%
  SLAVE_READ_PERCENT: 0%
***** 3. row *****
      ID: 2
      NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0002_RDS
  MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
  MASTER_READ_PERCENT: 100%
  SLAVE_READ_PERCENT: 0%
***** 4. row *****
      ID: 3
      NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0003_RDS
  MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
  MASTER_READ_PERCENT: 100%
  SLAVE_READ_PERCENT: 0%
***** 5. row *****
      ID: 4
      NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0004_RDS
  MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
  MASTER_READ_PERCENT: 100%
  SLAVE_READ_PERCENT: 0%
***** 6. row *****
      ID: 5
      NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0005_RDS
  MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
  MASTER_READ_PERCENT: 100%
  SLAVE_READ_PERCENT: 0%
***** 7. row *****
      ID: 6
      NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0006_RDS
  MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
  MASTER_READ_PERCENT: 100%
  SLAVE_READ_PERCENT: 0%
```

```

***** 8. row *****
          ID: 7
          NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0007_RDS
  MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
  SLAVE_READ_PERCENT: 0%
8 rows in set (0.02 sec)

```

As you can see, each database shard has the `NAME` attribute, which indicates the shard name corresponding to the database shard. Each shard name uniquely corresponds to one database shard name. For example, the shard name `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0003_RDS` corresponds to the database shard name `drds_test_vtla_0003`. Therefore, after obtaining the shard name, you can use the DRDS hint to specify the corresponding database shard to run the SQL statement.

- Specify database shard 0 to run an SQL statement:

```

SELECT /*TDDL:node('DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0000_RDS')*/ * FROM table_name;

```

- Specify multiple database shards to run an SQL statement:

```

SELECT /*TDDL:node('DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0000_RDS', 'DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0006_RDS')*/
* FROM table_name;

```

This SQL statement will be executed in the database shards corresponding to the shard names `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS` and `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0006_RDS`.

- View the execution plan of an SQL statement in database shard 0:

```

/*TDDL:node('DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS')*/
EXPLAIN SELECT * FROM table_name;

```

7.8.5 Scan all or some of database shards and table shards

In addition to routing an SQL statement to one or more database shards for execution, Distributed Relational Database Service (DRDS) provides the `SCAN` hint to allow you to scan all or some of database shards and table shards. With the `SCAN` hint, you can route an SQL statement to each database shard at a time. For example, you can view all the table shards in a specified database shard or view the data volume of each physical table of a specified logical table.

**Note:**

This topic is applicable to DRDS 5.3 and later. For earlier versions, see [Scan all database shards and table shards](#).

With the SCAN hint, you can specify the following SQL execution manners:

- Run an SQL statement in all table shards in all database shards.
- Run an SQL statement in all table shards in a specified database shard.
- Run an SQL statement in the specified table shard in the specified database shard by calculating the name of the physical table based on conditions.
- Run an SQL statement in the specified table shard in the specified database shard by explicitly specifying the name of the physical table.

The SCAN hint can be used in data manipulation language (DML) statements, data definition language (DDL) statements, and some data access language (DAL) statements.

**Note:**

- DRDS hints can be in the formats of `/*+TDDL:hint_command*/` and `/*! +TDDL: hint_command*/`.
- In the MySQL command line, if you need to run an SQL statement that contains a DRDS hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because DRDS hints are based on the [MySQL Comment Syntax](#). Otherwise, the client deletes the DRDS hint and then sends the SQL statement to the server for execution, which causes the DRDS hint to fail to take effect. For more information, see [MySQL Client Options](#).

Syntax

```
# SCAN hint
# Route an SQL statement to all table shards in all database shards.
SCAN()
# Route an SQL statement to all table shards in a specified database
shard.
SCAN(NODE="node_list")           # Specify the database shard.
# Route an SQL statement to the specified table shard in the specified
database shard by calculating the name of the physical table based on
conditions.
SCAN(
  [TABLE="]table_name_list"      # Specify the name of the logical
table.
  , CONDITION="condition_string" # Calculate the names of physical
databases based on the content of TABLE and CONDITION.
```

```
[, NODE="node_list" ] # Filter the results obtained
based on the content of CONDITION, to retain only the results of the
specified physical database.
# Route an SQL statement to the specified table shard in the specified
database shard by explicitly specifying the name of the physical
table.
SCAN(
  [TABLE="table_name_list" # Specify the name of the logical
table.
  , REAL_TABLE=("table_name_list") # Specify the name of the
physical table. The same physical table names are applied to all
physical databases.
  [, NODE="node_list" ] # Filter the results obtained
based on the content of CONDITION, to retain only the results of the
specified physical database.
# Specify physical table names or logical table names.
table_name_list:
  table_name [, table_name]...
# Specify physical databases by using GROUP_KEY and GROUP_INDEX, which
can be obtained by running the `SHOW NODE` statement.
node_list:
  {group_key | group_index} [, {group_key | group_index}]...
# Run an SQL WHERE statement. When using this syntax, you must specify
conditions for each table, for example, t1.id = 2 and t2.id = 2.
condition_string:
  where_condition
```

Examples

- **Run the following SQL statement in all table shards in all database shards:**

```
SELECT /*+TDDL:scan()*/ COUNT(1) FROM t1
```

After this statement is executed, the SQL statement is routed to all the physical tables corresponding to the logical table `t1`, and the result sets are merged and returned.

- **Run the following SQL statement in all table shards in specified database shards:**

```
SELECT /*+TDDL:scan(node='0,1,2')*/ COUNT(1) FROM t1
```

After this statement is executed, all physical tables corresponding to the logical table `t1` in database shards 0000, 0001, and 0002 are calculated, the SQL statement is routed to the physical tables, and the result sets are merged and returned.

- **Run the following SQL statement in specified table shards based on conditions:**

```
SELECT /*+TDDL:scan('t1', condition='t1.id = 2')*/ COUNT(1) FROM t1
```

After this statement is executed, all physical tables that correspond to the logical table `t1` and meet the conditions are calculated, the SQL statement is routed to the physical tables, and the result sets are merged and returned.

- Run the following SQL JOIN statement in the specified table shards based on conditions:

```
SELECT /*+TDDL:scan('t1, t2', condition='t1.id = 2 and t2.id = 2')*/
 * FROM t1 a JOIN t2 b ON a.id = b.id WHERE b.name = "test"
```

After this statement is executed, all physical tables that correspond to the logical tables `t1` and `t2` and meet the conditions are calculated, the SQL statement is routed to the physical tables, and the result sets are merged and returned.



Notice:

Before using this custom hint, you must ensure that the logical tables `t1` and `t2` are split into the same number of database shards and the same number of table shards. Otherwise, the database shards calculated by the DRDS instance based on the conditions are different, and an error will be returned.

- Run the following SQL statement in the specified table shards in database shards by explicitly specifying the names of the physical tables:

```
SELECT /*+TDDL:scan('t1', real_table=('t1_00', 't1_01'))*/ COUNT(1)
FROM t1
```

After this statement is executed, the SQL statement is routed to the table shards `t1_00` `t1_01` in all database shards, and the result sets are merged and returned.

- Run the following SQL JOIN statement in the specified table shards in database shards by explicitly specifying the names of the physical tables:

```
SELECT /*+TDDL:scan('t1, t2', real_table=('t1_00,t2_00', 't1_01,
t2_01'))*/ * FROM t1 a JOIN t2 b ON a.id = b.id WHERE b.name = "test
";
```

After this statement is executed, the SQL statement is routed to the table shards `t1_00, t2_00, t1_01, and t2_01` in all database shards, and the result sets are merged and returned.

7.8.6 INDEX hint

- Distributed Relational Database Service (DRDS) supports global secondary indexes (GSIs). The INDEX hint allows you to obtain query results from a specified GSI.
- The INDEX hint takes effect only for SQL SELECT statements.

**Note:**

This custom hint is applicable only to ApsaraDB RDS for MySQL 5.7 and later and DRDS 5.4.1 and later.

Syntax

```
# FORCE INDEX
tbl_name [[AS] alias] [index_hint]
index_hint:
    FORCE INDEX({index_name})
# INDEX()
/*+TDDL:
    INDEX({table_name | table_alias}, {index_name})
*/
```

DRDS INDEX hint can be used in two ways:

- **FORCE INDEX():** This syntax is the same as that of *MySQL FORCE INDEX*.
- **INDEX():** In this syntax, a GSI is specified using a table name (or alias) and an index name. This hint does not take effect in the following cases:
 - The query does not contain the specified table name or alias.
 - The specified GSI is not in the specified table.

**Note:**

- **DRDS hints can be in the formats of `/*+TDDL:hint_command*/` and `/*! +TDDL:hint_command*/`.**
- **In the MySQL command line, if you need to run an SQL statement that contains a DRDS hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because DRDS hints are based on the *MySQL Comment Syntax*. Otherwise, the client deletes the DRDS hint and then sends the SQL statement to the server for execution, which causes the DRDS hint to fail to take effect. For more information, see *MySQL Client Options*.**

Examples

```
CREATE TABLE t_order (
  `id` bigint(11) NOT NULL AUTO_INCREMENT,
  `order_id` varchar(20) DEFAULT NULL,
  `buyer_id` varchar(20) DEFAULT NULL,
  `seller_id` varchar(20) DEFAULT NULL,
  `order_snapshot` longtext DEFAULT NULL,
  `order_detail` longtext DEFAULT NULL,
  PRIMARY KEY (`id`),
  GLOBAL INDEX `g_i_seller`(`seller_id`) dbpartition by hash(`seller_id`
),
```

```
UNIQUE GLOBAL INDEX `g_i_buyer` (`buyer_id`) COVERING(`seller_id`, `order_snapshot`)
  dbpartition by hash(`buyer_id`) tpartition by hash(`buyer_id`)
  tpartitions 3
) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by hash(`order_id`);
```

Specify the GSI `g_i_seller` by using `FORCE INDEX` in the `FROM` clause:

```
SELECT a.*, b.order_id
  FROM t_seller a
      JOIN t_order b FORCE INDEX(g_i_seller) ON a.seller_id = b.seller_id
 WHERE a.seller_nick="abc";
```

Specify the GSI `g_i_buyer` by using `INDEX+table alias`:

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

7.9 Custom hints for DRDS 5.2

7.9.1 Introduction to hints

As a supplement to the SQL syntax, hints play a critical role in relational databases. They allow you to affect execution plans of SQL statements by using relevant syntax, to specially optimize the SQL statements.

Overview of DRDS hints

Distributed relational database service (DRDS) provides special hint syntax.

For example, if you know the target data is stored in table shards in certain database shards and you need to route the SQL statement directly to the database shards for execution, you can use custom hints provided by DRDS.

```
/*+TDDL:NODE IN('node_name', ...) */SELECT * FROM table_name;
```

In the preceding SQL statement, the part between `/*+TDDL:` and `*/`, namely, `TDDL:node in('node_name', ...)`, is a DRDS hint. The hint specifies the ApsaraDB RDS for MySQL database shard where the SQL statement is to be executed.



Note:

- **DRDS hints can be in the formats of `/*+TDDL:hint_command*/` and `/*+TDDL:hint_command*/`.**

- In the MySQL command line, if you need to run an SQL statement that contains a DRDS hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because DRDS hints are based on the [MySQL Comment Syntax](#). Otherwise, the client deletes the DRDS hint and then sends the SQL statement to the server for execution, which causes the DRDS hint to fail to take effect. For more information, see [MySQL Client Options](#).

Syntax of DRDS hints

Basic syntax:

```
/*! TDDL:hint command*/
```

DRDS hints are based on [MySQL Comment Syntax](#). Therefore, an SQL statement that contains a DRDS hint is located between `/*!` and `*/`, and must begin with `TDDL:`. The `hint command` indicates a DRDS hint command related to the specific operation. For example, a DRDS hint is added to the following SQL statement to display the name of each database shard.

```
/*! TDDL:SCAN*/SHOW TABLES;
```

In this SQL statement, `/*! TDDL:SCAN*/` is the DRDS hint that begins with `TDDL:`, and `SCAN` is a DRDS hint command.

7.9.2 Read/write splitting

Distributed Relational Database Service (DRDS) provides transparent read/write splitting at the application layer. However, data synchronization between primary and read-only ApsaraDB RDS for MySQL instances has a delay of several milliseconds. If you need to read changed data immediately after the primary ApsaraDB RDS for MySQL instance is changed, you must ensure that the SQL statement for reading data is routed to the primary ApsaraDB RDS for MySQL instance. To meet this demand, DRDS provides custom hints for read/write splitting, to route SQL statements to a specified primary or read-only ApsaraDB RDS for MySQL instance.

Syntax

```
/*! TDDL:MASTER|SLAVE*/
```

With this custom hint, you can specify whether to run an SQL statement on a primary or read-only ApsaraDB RDS for MySQL instance. With the custom hint

`/*!TDDL:SLAVE*/`, if a primary ApsaraDB RDS for MySQL instance is configured with multiple read-only ApsaraDB RDS for MySQL instances, the DRDS instance randomly selects a read-only ApsaraDB RDS for MySQL instance based on its weight, to run the SQL statement.



Note:

- DRDS hints can be in the formats of `/*+TDDL:hint_command*/` and `/*! +TDDL:hint_command*/`.
- In the MySQL command line, if you need to run an SQL statement that contains a DRDS hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because DRDS hints are based on the [MySQL Comment Syntax](#). Otherwise, the client deletes the DRDS hint and then sends the SQL statement to the server for execution, which causes the DRDS hint to fail to take effect. For more information, see [MySQL Client Options](#).

Examples

- Specify a primary ApsaraDB RDS for MySQL instance to run an SQL statement:

```
/*! TDDL:MASTER*/SELECT * FROM table_name;
```

After the custom hint `/*! TDDL:MASTER*/` is added at the beginning of the SQL statement, this SQL statement is routed to the primary ApsaraDB RDS for MySQL instance for execution.

- Specify a read-only ApsaraDB RDS for MySQL instance to run an SQL statement:

```
/*! TDDL:SLAVE*/SELECT * FROM table_name;
```

After the custom hint `/*! TDDL:SLAVE*/` is added at the beginning of the SQL statement, this SQL statement is randomly routed to a read-only ApsaraDB RDS for MySQL instance based on the allocated weight.

Note

- The custom hints for read-write splitting are only applicable to read SQL statements for non-transactional data. SQL statements for transactional data and write SQL statements are still routed to the primary ApsaraDB RDS for MySQL instance for execution.

- The DRDS hint `/*+TDDL:slave()*/` allows you to randomly select a read-only ApsaraDB RDS for MySQL instance based on the allocated weight, to route the SQL statement to the read-only RDS instance for execution. If no read-only ApsaraDB RDS for MySQL instance is available, no error is reported. Instead, the primary ApsaraDB RDS for MySQL instance is selected to run the SQL statement.

7.9.3 Prevent the delay from a read-only ApsaraDB for RDS instance

Normally, if you have configured a read-only ApsaraDB for RDS instance for the primary ApsaraDB RDS for MySQL instance of a logical database in a (DRDS) instance and set read traffic for both the primary and read-only ApsaraDB RDS for MySQL instances, DRDS routes SQL statements to the primary and read-only ApsaraDB RDS for MySQL instances based on the read/write ratio. However, if asynchronous data replication between the primary and read-only ApsaraDB for RDS instances has a high delay, an error is reported or error results are returned when DRDS routes the SQL statements to the read-only ApsaraDB RDS for MySQL instance.

To address this issue, the DRDS instance provides a custom hint to cut off the delay of the read-only instance. Specifically, based on the maximum delay of primary/secondary replication, DRDS determines whether to route the SQL statement to the primary or the read-only ApsaraDB RDS for MySQL instance

Syntax

```
/*! TDDL:SQL_DELAY_CUTOFF=time*/
```

With this custom hint, you can specify the value of `SQL_DELAY_CUTOFF`. When the value of `SQL_DELAY` (primary/secondary replication delay of ApsaraDB RDS for MySQL) for the read-only ApsaraDB RDS for MySQL instance reaches or exceeds the value of `time` (which is measured in seconds), the SQL statement is routed to the primary ApsaraDB RDS for MySQL instance.



Note:

- DRDS hints can be in the formats of `/*+TDDL:hint_command*/` and `/*! +TDDL:hint_command*/`.

- In the MySQL command line, if you need to run an SQL statement that contains a DRDS hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because DRDS hints are based on the [MySQL Comment Syntax](#). Otherwise, the client deletes the DRDS hint and then sends the SQL statement to the server for execution, which causes the DRDS hint to fail to take effect. For more information, see [MySQL Client Options](#).

Examples

- Set the primary/secondary replication delay to 5 seconds:

```
/*! TDDL:SQL_DELAY_CUTOFF=5*/SELECT * FROM table_name;
```

In this SQL statement, the value of `SQL_DELAY_CUTOFF` is set to 5. Therefore, when the value of `SQL_DELAY` for the read-only ApsaraDB RDS for MySQL instance reaches or exceeds 5 seconds, the SQL statement is routed to the primary ApsaraDB RDS for MySQL instance.

- Use the custom hint for delay cutoff with other custom hints:

```
/*! TDDL:SLAVE AND SQL_DELAY_CUTOFF=5*/SELECT * FROM table_name;
```

The custom hint for cutting off the delay of the read-only ApsaraDB RDS for MySQL instance can be used with other hints. By default, the SQL query request is routed to a read-only ApsaraDB RDS for MySQL instance. However, when the primary/secondary replication delay reaches or exceeds 5 seconds, the SQL query request is routed to the primary ApsaraDB RDS for MySQL instance.

7.9.4 Specify a timeout period for an SQL statement

In Distributed Relational Database Service (DRDS), the SQL statements for DRDS instances and ApsaraDB RDS for MySQL instances are timed out after 900 seconds (which can be adjusted) by default. However, for some slow SQL statements, the timeout period may exceed 900 seconds. For these slow SQL statements, DRDS provides a custom hint to adjust their timeout periods. You can use this custom hint to adjust the SQL execution duration as needed.

Syntax

The syntax of the DRDS hint for specifying a timeout period for an SQL statement is as follows:

```
/*! TDDL:SOCKET_TIMEOUT=time*/
```

The `SOCKET_TIMEOUT` parameter is measured in milliseconds. With this custom hint, you can adjust the timeout period for the SQL statement based on business requirements.

**Note:**

- DRDS hints can be in the formats of `/*+TDDL:hint_command*/` and `/*! +TDDL:hint_command*/`.
- In the MySQL command line, if you need to run an SQL statement that contains a DRDS hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because DRDS hints are based on the *MySQL Comment Syntax*. Otherwise, the client deletes the DRDS hint and then sends the SQL statement to the server for execution, which causes the DRDS hint to fail to take effect. For more information, see *MySQL Client Options*.

Examples

Set the timeout period of an SQL statement to 40 seconds:

```
/*! TDDL:SOCKET_TIMEOUT=40000*/SELECT * FROM t_item;
```

**Note:**

A longer timeout period causes database resources to be occupied for a longer period of time. If excessive SQL statements are executed over a long time within the same period, a large amount of database resources may be consumed. This will make users unable to use DRDS properly. In this case, we need to use this custom hint to optimize the SQL statements that take a long time to execute.

7.9.5 Specify a database shard to run an SQL statement

When running SQL statements in a Distributed Relational Database Service (DRDS) instance, you may find that some SQL statements are not supported by the DRDS instance. In this case, you can use the custom hint provided by DRDS, to route the SQL statements to one or more database shards for execution. In addition, if you

need to query the data in a specified database shard or the data in a specified table shard in a known database shard, you can use the custom hint to directly route the SQL statement to the database shard for execution.

Syntax

This custom hint allows you to specify a database shard by using a shard name or the value of the database shard key, to run an SQL statement in the database shard. A shard name uniquely identifies a database shard in a DRDS instance. You can run the `SHOW NODE` command to obtain the shard name.



Note:

If the hint for specifying a database shard is used in an `INSERT` statement that contains a DRDS sequence for the target table, the DRDS sequence will not take effect. For more information, see [Limits and precautions for DRDS sequence](#).

- Specify a database shard by using a shard name, to run an SQL statement

This custom hint allows you to specify one or more database shards to run an SQL statement.

- Specify one database shard to run an SQL statement:

```
/*! TDDL:NODE='node_name'*/
```

Specifically, `node_name` indicates the shard name. This DRDS hint enables you to route the SQL statement to the database shard specified by `node_name`.

- Specify multiple database shards to run an SQL statement:

```
/*! TDDL:NODE IN ('node_name'[, 'node_name1', 'node_name2'])*/
```

The `IN` keyword is used to specify multiple shard names. This DRDS hint allows you to route the SQL statement to multiple database shards. The shard names in the square brackets ([]) are separated by commas (,).



Note:

When this custom hint is used, the DRDS instance directly routes the SQL statement to the specified database shards for execution. Therefore, the specified shard names in the SQL statement must correspond to existing database shards.

- Specify a database shard by using the value of the database shard key, to run an SQL statement

```
/*! TDDL:table_name.partition_key=value [and table_name1.partition_key=value1]*/
```

In this DRDS hint, `table_name` indicates the name of a logical table, and this table is a partitioned table. In addition, `partition_key` indicates a shard key, and `value` indicates the value specified for the shard key. In this custom hint, you can use the `and` keyword to specify the shard keys of multiple partitioned tables. When this DRDS hint is used, the DRDS instance calculates the database shards and table shards where the SQL statement is to be executed, and routes the SQL statement to the corresponding database shards.



Note:

- DRDS hints can be in the formats of `/*+TDDL:hint_command*/` and `/*! +TDDL:hint_command*/`.
- In the MySQL command line, if you need to run an SQL statement that contains a DRDS hint in the format of `/*+TDDL:hint_command*/`, add the `-c` parameter to the logon command, because DRDS hints are based on the *MySQL Comment Syntax*. Otherwise, the client deletes the DRDS hint and then sends the SQL statement to the server for execution, which causes the DRDS hint to fail to take effect. For more information, see *MySQL Client Options*.

Examples

The following shows the responses of the `SHOW NODE` statement for a logical database named `drds_test` in a DRDS instance.

```
mysql> SHOW NODE\G
***** 1. row *****
          ID: 0
          NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0000_RDS
  MASTER_READ_COUNT: 212
  SLAVE_READ_COUNT: 0
  MASTER_READ_PERCENT: 100%
  SLAVE_READ_PERCENT: 0%
***** 2. row *****
          ID: 1
          NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0001_RDS
  MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
  MASTER_READ_PERCENT: 100%
```

```

SLAVE_READ_PERCENT: 0%
***** 3. row *****
      ID: 2
      NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0002_RDS
  MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
SLAVE_READ_PERCENT: 0%
***** 4. row *****
      ID: 3
      NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0003_RDS
  MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
SLAVE_READ_PERCENT: 0%
***** 5. row *****
      ID: 4
      NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0004_RDS
  MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
SLAVE_READ_PERCENT: 0%
***** 6. row *****
      ID: 5
      NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0005_RDS
  MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
SLAVE_READ_PERCENT: 0%
***** 7. row *****
      ID: 6
      NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0006_RDS
  MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
SLAVE_READ_PERCENT: 0%
***** 8. row *****
      ID: 7
      NAME: DRDS_TEST_1473471355140LRPRDRDS_TEST_VTL
A_0007_RDS
  MASTER_READ_COUNT: 29
  SLAVE_READ_COUNT: 0
MASTER_READ_PERCENT: 100%
SLAVE_READ_PERCENT: 0%
8 rows in set (0.02 sec)

```

As you can see, each database shard has the NAME attribute, which indicates the shard name corresponding to the database shard. Each shard name uniquely corresponds to one database shard name. For example, the shard name `DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0003_RDS` corresponds to the database shard name `drds_test_vtla_0003`. Therefore, after obtaining the shard name, you can use the DRDS hint to specify the corresponding database shard to run the SQL statement.

- **Specify database shard 0 to run an SQL statement:**

```

/! TDDL:NODE='DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS'*/
SELECT * FROM table_name;

```

- **Specify multiple database shards to run an SQL statement:**

```

/! TDDL:NODE IN('DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS',
'DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0006_RDS')*/SELECT *
FROM table_name;

```

This SQL statement will be executed in the database shards corresponding to the shard names DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS and DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0006_RDS.

- **View the execution plan of an SQL statement in a specified database shard:**

```

/! TDDL:NODE='DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS'*/
EXPLAIN SELECT * FROM table_name;

```

After this SQL statement is executed, the execution plan of the `SELECT` statement in the database shard corresponding to the shard name DRDS_TEST_1473471355140LRPRDRDS_TEST_VTLA_0000_RDS will be returned.

- **Specify a database shard by using the value of the database shard key, to run an SQL statement:**

DRDS does not support subqueries in the `SET` clause of an `UPDATE` statement, because a shard key must be specified for `UPDATE` statements in DRDS. To address this issue, DRDS provides a custom hint to allow you to route the statement to a database shard for execution.

For example, the following shows the `CREATE TABLE` statement for creating two logical tables `t1` and `t2`, which are split into table shards in database shards:

```

CREATE TABLE `t1` (
  `id` bigint(20) NOT NULL,
  `name` varchar(20) NOT NULL,
  `val` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by hash(`id`)
tpartition by hash(`name`) tpartitions 3
CREATE TABLE `t2` (
  `id` bigint(20) NOT NULL,
  `name` varchar(20) NOT NULL,
  `val` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`id`)
)

```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by hash(`id`)
tbpartition by hash(`name`) tpartitions 3
```

The following SQL statement is to be executed for the two tables:

```
UPDATE t1 SET val=(SELECT val FROM t2 WHERE id=1) WHERE id=1;
```

If this statement is directly executed in a DRDS instance, an error will be returned indicating that this statement is not supported. In this case, you can add the DRDS hint to this SQL statement before submitting it to the DRDS instance for execution. The final SQL statement is as follows:

```
/*! TDDL:t1.id=1 and t2.id=1*/UPDATE t1 SET val=(SELECT val FROM t2
WHERE id=1) WHERE id=1;
```

This statement will be routed to database shards corresponding to `t1`, with the `id` of the database shards being 1. You can run the following EXPLAIN command to view the execution plan of this SQL statement:

```
mysql> explain /*! TDDL:t1.id=1 and t2.id=1*/UPDATE t1 SET val=(
SELECT val FROM t2 WHERE id=1) WHERE id=1\G
***** 1. row *****
GROUP_NAME: TEST_DRDS_1485327111630IXLWTEST_DRDS_IGHF_0001_RDS
SQL: UPDATE `t1_2` AS `t1` SET `val` = (SELECT val FROM `
t2_2` AS `t2` WHERE `id` = 1) WHERE `id` = 1
PARAMS: {}
***** 2. row *****
GROUP_NAME: TEST_DRDS_1485327111630IXLWTEST_DRDS_IGHF_0001_RDS
SQL: UPDATE `t1_1` AS `t1` SET `val` = (SELECT val FROM `
t2_1` AS `t2` WHERE `id` = 1) WHERE `id` = 1
PARAMS: {}
***** 3. row *****
GROUP_NAME: TEST_DRDS_1485327111630IXLWTEST_DRDS_IGHF_0001_RDS
SQL: UPDATE `t1_0` AS `t1` SET `val` = (SELECT val FROM `
t2_0` AS `t2` WHERE `id` = 1) WHERE `id` = 1
PARAMS: {}
3 rows in set (0.00 sec)
```

According to the result set of the EXPLAIN command, the SQL statement is rewritten into three statements, which are then routed to the database shards for execution. You can further specify a table shard by using the value of the table shard key, to narrow the execution scope of the SQL statement to a specified table shard.

```
mysql> explain /*! TDDL:t1.id=1 and t2.id=1 and t1.name='1'*/UPDATE
t1 SET val=(SELECT val FROM t2 WHERE id=1) WHERE id=1\G
***** 1. row *****
GROUP_NAME: TEST_DRDS_1485327111630IXLWTEST_DRDS_IGHF_0001_RDS
SQL: UPDATE `t1_1` AS `t1` SET `val` = (SELECT val FROM `
t2_1` AS `t2` WHERE `id` = 1) WHERE `id` = 1
PARAMS: {}
```

```
1 row in set (0.00 sec)
```

**Note:**

Before using this custom hint, you must ensure that the logical tables t1 and t2 are split into the same number of database shards and the same number of table shards. Otherwise, the database shards calculated by the DRDS instance based on the values of the two shard keys are different, and an error will be returned.

7.9.6 Scan all database shards and table shards

In addition to routing an SQL statement to one or more database shards for execution, Distributed Relational Database Service (DRDS) provides a custom hint to allow you to scan all database shards and table shards. With this custom hint, you can route an SQL statement to each database shard at a time. For example, you can use this custom hint to view all the table shards in a specified database shard. In addition, you can use this custom hint to view the data volume of table shards in each database shard corresponding to a specified logical table.

Syntax

With this hint, you can route an SQL statement to all database shards for execution and route an SQL statement to all database shards to perform an operation on a specified logical table.

- Route an SQL statement to all database shards for execution:

```
/*! TDDL:SCAN*/
```

- Perform an operation on a specified logical table:

```
/*! TDDL:SCAN='table_name'*/
```

The `table_name` parameter indicates the name of a logical table in the logical database of a DRDS instance. This custom hint is provided for table shards in database shards. Ensure that the value of `table_name` is the name of a table shard in database shards.

**Note:**

- DRDS hints can be in the formats of `/*! TDDL:hint command*/` and `/*TDDL:hint command*/`.

- In the MySQL command line, if you need to run an SQL statement that contains a DRDS hint in the format of `/*TDDL:hint command*/`, add the `-c` parameter to the logon command, because DRDS hints are based on the [MySQL Comment Syntax](#). Otherwise, the client deletes the DRDS hint and then sends the SQL statement to the server for execution, which causes the DRDS hint to fail to take effect. For more information, see [MySQL Client Options](#).

Examples

- View the data volume of a specified broadcast table in each database shard:

```
/*! TDDL:SCAN*/SELECT COUNT(1) FROM table_name
```

In this SQL statement, `table_name` indicates a broadcast table. This hint causes the DRDS instance to route the SQL statement to each database shard for execution. Therefore, the result sets include the total data volume of the broadcast table `table_name` in all database shards. This statement allows you to conveniently check whether the data of the broadcast table is normal.

- Scan a single-database non-partition logical table:

```
/*! TDDL:SCAN*/SELECT COUNT(1) FROM table_name
```

This hint causes the DRDS instance to route the SQL `select count(1) from table_name` statement to each database shard for execution. The `table_name` parameter indicates a logical table in a logical database of a DRDS instance. Before using this hint, ensure that each database shard contains the table shard `table_name`. In other words, the table shard `table_name` is a logical table that is only split into database shards, but not split into table shards. Otherwise, an error that indicates that the table is not found will be returned.

- Scan a partitioned logical table in database shards:

```
/*! TDDL:SCAN='table_name'*/SELECT COUNT(1) FROM table_name
```

When executing this statement, the DRDS instance first calculates all the database shards and table shards corresponding to the logical table `table_name`, and then generates a COUNT clause for each table shard in each database shard.

- **View the execution plans of all database shards:**

```
/*! TDDL:SCAN='table_name'*/EXPLAIN SELECT * FROM table_name;
```

7.10 Distributed transactions

7.10.1 Distributed transactions based on MySQL 5.7



Note:

- **When you use MySQL 5.7 or later and a Distributed Relational Database Service (DRDS) instance of 5.3.4 or later, XA distributed transactions are automatically enabled. The user experience of the XA distributed transactions is the same as that of single-database transactions in MySQL. No special commands are required to enable XA distributed transactions.**
- **When you use MySQL and a DRDS instance in other versions, see [Distributed transactions based on MySQL 5.6](#).**

How it works

When you use MySQL 5.7 or later, the DRDS instance processes distributed transactions based on the XA protocol by default.

Use method

The user experience of distributed transactions in a DRDS instance is the same as that of single-database transactions in MySQL, for example, in terms of the following commands:

- **SET AUTOCOMMIT=0: Start a transaction.**
- **COMMIT: Commit the current transaction.**
- **ROLLBACK: Roll back the current transaction.**

If the SQL statement in a transaction involves only a single shard, the DRDS instance routes the transaction directly to the ApsaraDB RDS for MySQL instance as a single-database transaction. If the SQL statement in the transaction is to modify the data of multiple shards, the DRDS instance automatically upgrades the current transaction to a distributed transaction.

7.10.2 Distributed transactions based on MySQL 5.6

How it works

The XA protocol for MySQL 5.6 is not mature. Therefore, the Distributed Relational Database Service (DRDS) instance independently implements two-phase commit (2PC) transaction policies for distributed transactions. When you use MySQL 5.7 or later, we recommend that you use XA transaction policies.



Note:

The distributed transactions described in this topic are intended for users who use MySQL 5.6 or DRDS earlier than 5.3.4. When you use MySQL 5.6 or later and DRDS 5.3.4 or later, see [Distributed transactions based on MySQL 5.7](#).

Use method

If a transaction involves multiple database shards, you must declare the current transaction as a distributed transaction. If a transaction involves only a single database shard, you do not need to enable distributed transactions, but can process the transaction as a single-database transaction in MySQL. No additional operations are required.

To enable distributed transactions, do as follows:

After transactions are enabled, run `SET drds_transaction_policy = '...'`.

To enable 2PC transactions in the MySQL command-line client, run the following statements:

```
SET AUTOCOMMIT=0;
SET drds_transaction_policy = '2PC'; -- We recommend that users who
use MySQL 5.6 run this command.
.... -- Here, you can run your business SQL statement.
COMMIT; -- You can alternatively write ROLLBACK.
```

To enable 2PC transactions by using the Java database connectivity (JDBC) API, write the code as follows:

```
conn.setAutoCommit(false);
try (Statement stmt = conn.createStatement()) {
    stmt.execute("SET drds_transaction_policy = '2PC'");
}
// ... Here, you can run your service SQL statement.
```

```
conn.commit(); // You can alternatively write rollback().
```

FAQ

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

A: If you enable transactions by using the Spring `@Transactional` annotation, you can enable DRDS distributed transactions by extending the transaction manager.

Sample code:

```
import org.springframework.jdbc.datasource.DataSourceTransactionManager;
import org.springframework.transaction.TransactionDefinition;
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
public class DrdsTransactionManager extends DataSourceTransactionManager {
    public DrdsTransactionManager(DataSource dataSource) {
        super(dataSource);
    }
    @Override
    protected void prepareTransactionalConnection(Connection con,
TransactionDefinition definition) throws SQLException {
        try (Statement stmt = con.createStatement()) {
            stmt.executeUpdate("SET drds_transaction_policy = '2PC
'"); // In this case, a 2PC transaction is used as an example.
        }
    }
}
```

After that, instantiate the preceding class in the Spring configuration. For example, you can write the code as follows:

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

To enable DRDS distributed transactions for a class, you can add the `@Transactional("drdsTransactionManager")` annotation.

7.11 DDL operations

7.11.1 DDL statements

The data definition language (DDL) statement CREATE TABLE in a Distributed Relational Database Service (DRDS) instance is similar to that in a MySQL database, and is extended based on the syntax in a MySQL database. To create a table shard

in a DRDS instance, you must specify the table sharding manner and the database sharding manner in the `drds_partition_options` parameter. The valid values include `DBPARTITION BY`, `TBPARTITION BY`, `TBPARTITIONS`, and `BROADCAST`.

Currently, you can run a DDL statement in the following ways:

- Run the DDL statement through the MySQL command-line client, for example, by using MySQL command lines, Navicat, or MySQL Workbench.
- Connect to the specified DRDS instance by using program code and then call the DDL statement for execution.

For the syntax of the CREATE TABLE statement in a MySQL database, see [MySQL CREATE TABLE Statement](#).

7.11.2 CREATE TABLE statement

7.11.2.1 Overview

This topic describes the syntax, clauses, parameters, and basic methods for creating a table by using a data definition language (DDL) statement.



Note:

Distributed Relational Database Service (DRDS) instances do not allow you to directly create a database by using a data definition language (DDL) statement. To create a database, you can [Log on to the DRDS console](#). For the information about how to create a database, see [Create a database](#).

Syntax

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...)
    [table_options]
    [drds_partition_options]
    [partition_options]
drds_partition_options:
    DBPARTITION BY
        HASH([column])
    | TBPARTITION BY
        { HASH(column)
        | {MM|DD|WEEK|MMDD}(column)}
    [TBPARTITIONS num]
```

]

Clauses and parameters for database and table sharding

- `DBPARTITION BY hash(partition_key)`: **This parameter specifies the shard key and the sharding algorithm for database sharding. Database sharding by time is not supported.**
- `TBPARTITION BY { HASH(column) | {MM|DD|WEEK|MMDD}(column)`: **(Optional) This parameter specifies the method of mapping the data to a physical table. The value is the same as that of DBPARTITION BY by default.**
- `TBPARTITIONS num`: **(Optional) This parameter specifies the number of physical tables to be created in each database shard. The value is 1 by default. If no table sharding is required, you do not need to specify this parameter.**

7.11.2.2 Create a single-database non-partition table

This topic describes how to create a single-database non-partition table.

Create a single-database non-partition table

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

According to the node topology of the logical table, you can see that a single-database non-partition logical table is created in database 0.

```
mysql> show topology from single_tbl;
+-----+
+-----+
+-----+
| ID    | GROUP_NAME
| TABLE_NAME |
+-----+
+-----+
|      0 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000
_RDS | single_tbl |
+-----+
+-----+
+-----+
```

```
1 row in set (0.01 sec)
```

Specify parameters

You can also specify the `select_statement` parameter when creating a a single-database non-partition table. If you need to create table shards, you cannot specify this parameter.

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

For example, you can run the following statement to create a single-database non-partition table `single_tbl2` to store the data from the `single_tbl` table. In this case, no sharding is required.

```
CREATE TABLE single_tbl2(
  id int,
  name varchar(30),
  primary key(id)
) select * from single_tbl;
```

7.11.2.3 Create a non-partition table in database shards

This topic describes how to create a non-partition table in database shards.

Assume that eight database shards have been created. You can run the following command to create a non-partition table in the database shards by calculating the hash function based on the `userId` shard key.

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

According to the node topology of the logical table, you can see that a table shard is created in each database shard. In other words, the table is only distributed to database shards.

```
mysql> show topology from multi_db_single_tbl;
+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID    | GROUP_NAME
|      | TABLE_NAME      |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
```

```

|      0 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000
_RDS | multi_db_single_tbl |
|      1 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001
_RDS | multi_db_single_tbl |
|      2 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0002
_RDS | multi_db_single_tbl |
|      3 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0003
_RDS | multi_db_single_tbl |
|      4 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0004
_RDS | multi_db_single_tbl |
|      5 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005
_RDS | multi_db_single_tbl |
|      6 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0006
_RDS | multi_db_single_tbl |
|      7 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007
_RDS | multi_db_single_tbl |
+-----+
+-----+
+-----+
8 rows in set (0.01 sec)

```

7.11.2.4 Create table shards in database shards

This topic describes how to create table shards in database shards in different sharding manners.

- Use **HASH** for sharding
- Use **RANGE_HASH** for sharding
- Use **date functions** for sharding

In the following examples, it is assumed that eight database shards have been created.

Use HASH for sharding

Create a table that is split into table shards in database shards, with each database shard containing three physical tables. The database sharding process is calculating the hash function by using `id` as the shard key, and the table sharding process is calculating the hash function by using `bid` as the shard key. Specifically, a hash operation is performed on the data of the table based on the `id` column, to distribute the data to multiple database shards. Then, a hash operation is performed on the data in each database shard based on the `bid` column, to distribute the data to the three physical tables.

```

CREATE TABLE multi_db_multi_tbl(
  id int auto_increment,
  bid int,
  name varchar(30),
  primary key(id)

```

```
) dbpartition by hash(id) tbpartition by hash(bid) tbpartitions 3;
```

According to the node topology of the logical table, you can see that three table shards are created in each database shard.

```
mysql> show topology from multi_db_multi_tbl;
+-----+
+-----+
+-----+
| ID   | GROUP_NAME
|     | TABLE_NAME
+-----+
+-----+
+-----+
| 0   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000
_RDS | multi_db_multi_tbl_00 |
| 1   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000
_RDS | multi_db_multi_tbl_01 |
| 2   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0000
_RDS | multi_db_multi_tbl_02 |
| 3   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001
_RDS | multi_db_multi_tbl_03 |
| 4   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001
_RDS | multi_db_multi_tbl_04 |
| 5   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0001
_RDS | multi_db_multi_tbl_05 |
| 6   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0002
_RDS | multi_db_multi_tbl_06 |
| 7   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0002
_RDS | multi_db_multi_tbl_07 |
| 8   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0002
_RDS | multi_db_multi_tbl_08 |
| 9   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0003
_RDS | multi_db_multi_tbl_09 |
| 10  | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0003
_RDS | multi_db_multi_tbl_10 |
| 11  | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0003
_RDS | multi_db_multi_tbl_11 |
| 12  | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0004
_RDS | multi_db_multi_tbl_12 |
| 13  | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0004
_RDS | multi_db_multi_tbl_13 |
| 14  | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0004
_RDS | multi_db_multi_tbl_14 |
| 15  | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005
_RDS | multi_db_multi_tbl_15 |
| 16  | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005
_RDS | multi_db_multi_tbl_16 |
| 17  | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005
_RDS | multi_db_multi_tbl_17 |
| 18  | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0006
_RDS | multi_db_multi_tbl_18 |
| 19  | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0006
_RDS | multi_db_multi_tbl_19 |
| 20  | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0006
_RDS | multi_db_multi_tbl_20 |
| 21  | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007
_RDS | multi_db_multi_tbl_21 |
| 22  | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007
_RDS | multi_db_multi_tbl_22 |
| 23  | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0007
_RDS | multi_db_multi_tbl_23 |
```

```
+-----+
+-----+
+-----+
24 rows in set (0.01 sec)
```

According to the sharding rule of the logical table, you can see that both database sharding and table sharding are running the hash function, except that the database shard key is id and the table shard key is bid.

```
mysql> show rule from multi_db_multi_tbl;
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ID      | TABLE_NAME          | BROADCAST | DB_PARTITION_KEY |
DB_PARTITION_POLICY | DB_PARTITION_COUNT | TB_PARTITION_KEY |
TB_PARTITION_POLICY | TB_PARTITION_COUNT |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|      0 | multi_db_multi_tbl |          0 | id                |
      |      8                |      bid                | hash              |
+-----+-----+-----+-----+
|      3 |                    |                    |                    |
+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

Use RANGE_HASH for sharding

- **Requirements**

The shard key must be a character or a number.

- **Routing method**

Calculate a hash value based on the last N digits of any shard key, and then calculate the route by using RANGE_HASH. The number N is the third parameter in the function. For example, during calculation of the RANGE_HASH(COL1, COL2, N) function, COL1 is preferentially selected and then truncated to obtain the last N digits for calculation. If COL1 does not exist, COL2 is selected and truncated for calculation.

- **Scenarios**

RANGE_HASH is applicable to scenarios where two shard keys are used for sharding but only the values of one shard is used for SQL query. Assume that a

DRDS database is partitioned into eight physical databases. Our customer has the following requirements:

1. The order table of each service needs to be split into database shards by buyer ID and order ID.
2. The query is executed based on either the buyer ID or order ID as the condition.

In this case, you can run the following DDL statement to create the order table:

```
create table test_order_tb (  
    id int,  
    seller_id varchar(30) DEFAULT NULL,  
    order_id varchar(30) DEFAULT NULL,  
    buyer_id varchar(30) DEFAULT NULL,  
    create_time datetime DEFAULT NULL,  
    primary key(id)  
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by RANGE_HASH(  
buyer_id, order_id, 10) tpartition by RANGE_HASH(buyer_id, order_id  
, 10) tpartitions 3;
```



Note:

- Neither of the two shard keys can be modified.
- Data insertion fails if the two shard keys point to different database shards or table shards.

Use date functions for sharding

In addition to using the hash function as the sharding algorithm, you can also use the date functions MM, DD, WEEK, and MMDD as the table sharding algorithms. For more information, see the following examples.

- Create a table and then split the table into table shards in database shards. The database sharding process is calculating the hash function by using `userId` as the shard key, and the table sharding process is calculating `DAY_OF_WEEK` through the `WEEK(actionDate)` function and then splitting the table into table shards based on the `actionDate` column, with one week counted as seven days.

For example, if the value in the `actionDate` column is 2017-02-27, which is on Monday, the value obtained by calculating the `WEEK(actionDate)` function is 2. In this case, the record is stored in table shard 2, because $2 \% 7 = 2$. This table shard is located in a database shard and is named `user_log_2`. For another example, if the value in the `actionDate` column is 2017-02-26, which is on Sunday, the value obtained by calculating the `WEEK(actionDate)` function is 1. In this

case, the record is stored in table shard 1, because $1 \% 7 = 1$. This table shard is located in a database shard and is named `user_log_1`.

```
CREATE TABLE user_log(
  userId int,
  name varchar(30),
  operation varchar(30),
  actionDate DATE
) dbpartition by hash(userId) tpartition by WEEK(actionDate)
tpartitions 7;
```

According to the node topology of the logical table, you can see that seven table shards are created in each database shard, because one week is counted as seven days in the function. The responses are very long, and therefore are omitted by using an ellipsis (...).

```
mysql> show topology from user_log;
+-----+
+-----+
+-----+
| ID    | GROUP_NAME
      | TABLE_NAME |
+-----+
+-----+
+-----+
|      0 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log_0 |
|      1 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log_1 |
|      2 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log_2 |
|      3 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log_3 |
|      4 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log_4 |
|      5 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log_5 |
|      6 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log_6 |
|      7 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log_0 |
|      8 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log_1 |
|      9 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log_2 |
|     10 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log_3 |
|     11 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log_4 |
|     12 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log_5 |
|     13 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log_6 |
|     ...
|     49 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log_0 |
|     50 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log_1 |
```

```

| 51 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log_2 |
| 52 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log_3 |
| 53 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log_4 |
| 54 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log_5 |
| 55 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log_6 |
+-----
+-----+
+-----+
56 rows in set (0.01 sec)

```

According to the sharding rule of the logical table, you can see that the database sharding process is calculating the hash function by using `userId` as the shard key, and the table sharding process is calculating the WEEK function by using `actionDate` as the shard key.

```

mysql> show rule from user_log;
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| ID | TABLE_NAME | BROADCAST | DB_PARTITION_KEY | DB_PARTITI
ON_POLICY | DB_PARTITION_COUNT | TB_PARTITION_KEY | TB_PARTITI
ON_POLICY | TB_PARTITION_COUNT |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| 0 | user_log | 0 | userId | hash
| 7 | 8 | | actionDate | week
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

According to the specified database and table shard key parameters, you can see the specific physical table in the specific physical database to which the SQL statement is routed.

Figure 7-9: View the route of the SQL statement

```

mysql> explain select name from user_log where userId = 1 and actionDate = '2017-02-27'\G
***** 1 row *****
GROUP_NAME: SANGUAN_1490167540907XWDVSANGUAN_BSOT_0001_RDS
SQL: select `user_log`.`name` from `user_log_2` `user_log` where ((`user_log`.`userId` = 1) AND (`user_log`.`actionDate` = '2017-02-27'))
PARAMS: {}
1 row in set (0.01 sec)

```

- Create a table that is split into table shards in database shards. The database sharding process is calculating the hash function by using `userId` as the shard key, and the table sharding process is calculating MONTH_OF_YEAR through the

MM(actionDate) function and then splitting the table into table shards based on the actionDate column, with one year counted as 12 months.

For example, if the value in the actionDate column is 2017-02-27, the value obtained by calculating the MM(actionDate) function is 02. In this case, the record is stored in table shard 02, because $02 \% 12 = 02$. This table shard is located in a database shard and is named user_log_02. For another example, if the value in the actionDate column is 2016-12-27, the value obtained by calculating the MM(actionDate) function is 12. In this case, the record is stored in table shard 00, because $12 \% 12 = 00$. This table shard is located in a database shard and is named user_log_00.

```
CREATE TABLE user_log2(
  userId int,
  name varchar(30),
  operation varchar(30),
  actionDate DATE
) dbpartition by hash(userId) tpartition by MM(actionDate)
tpartitions 12;
```

According to the node topology of the logical table, you can see that 12 table shards are created in each database shard, because one year is counted as 12 months in the function. The responses are very long, and therefore are omitted by using an ellipsis (...).

```
mysql> show topology from user_log2;
+-----+
+-----+
+-----+
| ID    | GROUP_NAME
|      | TABLE_NAME |
+-----+
+-----+
+-----+
|      0 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log2_00 |
|      1 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log2_01 |
|      2 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log2_02 |
|      3 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log2_03 |
|      4 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log2_04 |
|      5 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log2_05 |
|      6 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log2_06 |
|      7 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log2_07 |
|      8 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log2_08 |
```

```

|    9 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log2_09 |
|   10 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log2_10 |
|   11 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log2_11 |
|   12 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log2_00 |
|   13 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log2_01 |
|   14 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log2_02 |
|   15 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log2_03 |
|   16 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log2_04 |
|   17 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log2_05 |
|   18 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log2_06 |
|   19 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log2_07 |
|   20 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log2_08 |
|   21 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log2_09 |
|   22 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log2_10 |
|   23 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0001_RDS | user_log2_11 |
...
|   84 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log2_00 |
|   85 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log2_01 |
|   86 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log2_02 |
|   87 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log2_03 |
|   88 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log2_04 |
|   89 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log2_05 |
|   90 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log2_06 |
|   91 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log2_07 |
|   92 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log2_08 |
|   93 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log2_09 |
|   94 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log2_10 |
|   95 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log2_11 |
+-----
+-----+
96 rows in set (0.02 sec)

```

According to the sharding rule of the logical table, you can see that the database sharding process is calculating the hash function by using `userId` as the shard key, and the table sharding process is calculating the MM function by using `actionDate` as the shard key.

```
mysql> show rule from user_log2;
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ID    | TABLE_NAME | BROADCAST | DB_PARTITION_KEY | DB_PARTITI
ON_POLICY | DB_PARTITION_COUNT | TB_PARTITION_KEY | TB_PARTITI
ON_POLICY | TB_PARTITION_COUNT |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|      0 | user_log2   |          0 | userId          | hash
|      8 |             |          | actionDate      | mm
|     12 |             |          |                 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

- Create a table that is split into table shards in database shards. The database sharding process is calculating the hash function by using `userId` as the shard key, and the table sharding process is calculating `DAY_OF_MONTH` through the `DD(actionDate)` function and then splitting the table into table shards, with one month counted as 31 days.

For example, if the value in the `actionDate` column is 2017-02-27, the value obtained by calculating the `DD(actionDate)` function is 27. In this case, the record is stored in table shard 27, because $27 \% 31 = 27$. This table shard is located in a database shard and is named `user_log_27`.

```
CREATE TABLE user_log3(
  userId int,
  name varchar(30),
  operation varchar(30),
  actionDate DATE
) dbpartition by hash(userId) tpartition by DD(actionDate)
tpartitions 31;
```

According to the node topology of the logical table, you can see that 31 table shards are created in each database shard, because one month is counted as 31 days in the function. The responses are very long, and therefore are omitted by using an ellipsis (...).

```
mysql> show topology from user_log3;
```

```

+-----+
+-----+
+-----+
| ID    | GROUP_NAME
|      | TABLE_NAME |
+-----+
+-----+
| 0    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_00 |
| 1    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_01 |
| 2    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_02 |
| 3    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_03 |
| 4    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_04 |
| 5    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_05 |
| 6    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_06 |
| 7    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_07 |
| 8    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_08 |
| 9    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_09 |
| 10   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_10 |
| 11   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_11 |
| 12   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_12 |
| 13   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_13 |
| 14   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_14 |
| 15   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_15 |
| 16   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_16 |
| 17   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_17 |
| 18   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_18 |
| 19   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_19 |
| 20   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_20 |
| 21   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_21 |
| 22   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_22 |
| 23   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_23 |
| 24   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_24 |
| 25   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_25 |
| 26   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_26 |
| 27   | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_27 |

```

```

| 28 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_28 |
| 29 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_29 |
| 30 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log3_30 |
...
| 237 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log3_20 |
| 238 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log3_21 |
| 239 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log3_22 |
| 240 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log3_23 |
| 241 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log3_24 |
| 242 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log3_25 |
| 243 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log3_26 |
| 244 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log3_27 |
| 245 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log3_28 |
| 246 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log3_29 |
| 247 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log3_30 |
+-----+
+-----+
248 rows in set (0.01 sec)

```

According to the sharding rule of the logical table, you can see that the database sharding process is calculating the hash function by using `userId` as the shard key, and the table sharding process is calculating the DD function by using `actionDate` as the shard key.

```

mysql> show rule from user_log3;
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| ID | TABLE_NAME | BROADCAST | DB_PARTITION_KEY | DB_PARTITI
ON_POLICY | DB_PARTITION_COUNT | TB_PARTITION_KEY | TB_PARTITI
ON_POLICY | TB_PARTITION_COUNT |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| 0 | user_log3 | 0 | userId | hash
| 31 | 8 | | actionDate | dd
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

```

- Create a table that is split into table shards in database shards. The database sharding process is calculating the hash function by using `userId` as the shard

key, and the table sharding process is calculating `DAY_OF_YEAR % 365` through the `MMDD(actionDate) tbpartitions 365` function and then splitting the table into 365 physical tables, with one year counted as 365 days.

For example, if the value in the `actionDate` column is `2017-02-27`, the value obtained by calculating the `MMDD(actionDate)` function is 58. In this case, the record is stored in table shard 58. This table shard is located in a database shard and is named `user_log_58`.

```
CREATE TABLE user_log4(
  userId int,
  name varchar(30),
  operation varchar(30),
  actionDate DATE
) dbpartition by hash(userId) tbpartition by MMDD(actionDate)
tbpartitions 365;
```

According to the node topology of the logical table, you can see that 365 table shards are created in each database shard, because one year is counted as 365 days in the function. The responses are very long, and therefore are omitted by using an ellipsis (...).

```
mysql> show topology from user_log4;
+-----+
+-----+
+-----+
| ID    | GROUP_NAME
|       | TABLE_NAME |
+-----+
+-----+
+-----+
...
| 2896 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
|_WVVP_0007_RDS | user_log4_341 |
| 2897 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
|_WVVP_0007_RDS | user_log4_342 |
| 2898 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
|_WVVP_0007_RDS | user_log4_343 |
| 2899 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
|_WVVP_0007_RDS | user_log4_344 |
| 2900 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
|_WVVP_0007_RDS | user_log4_345 |
| 2901 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
|_WVVP_0007_RDS | user_log4_346 |
| 2902 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
|_WVVP_0007_RDS | user_log4_347 |
| 2903 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
|_WVVP_0007_RDS | user_log4_348 |
| 2904 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
|_WVVP_0007_RDS | user_log4_349 |
| 2905 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
|_WVVP_0007_RDS | user_log4_350 |
| 2906 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
|_WVVP_0007_RDS | user_log4_351 |
```

```
| 2907 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log4_352 |
| 2908 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log4_353 |
| 2909 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log4_354 |
| 2910 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log4_355 |
| 2911 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log4_356 |
| 2912 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log4_357 |
| 2913 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log4_358 |
| 2914 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log4_359 |
| 2915 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log4_360 |
| 2916 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log4_361 |
| 2917 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log4_362 |
| 2918 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log4_363 |
| 2919 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log4_364 |
+-----+
+-----+
+-----+
2920 rows in set (0.07 sec)
```

According to the sharding rule of the logical table, you can see that the database sharding process is calculating the hash function by using `userId` as the shard key, and the table sharding process is calculating the MMDD function by using `actionDate` as the shard key.

```
mysql> show rule from user_log4;
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ID    | TABLE_NAME | BROADCAST | DB_PARTITION_KEY | DB_PARTITI
ON_POLICY | DB_PARTITION_COUNT | TB_PARTITION_KEY | TB_PARTITI
ON_POLICY | TB_PARTITION_COUNT |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|      0 | user_log4   |          0 | userId           | hash
|      8 |             |          | actionDate       | mmdd
|    365 |             |          |                  |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
1 row in set (0.02 sec)
```

- Create a table that is split into table shards in database shards. The database sharding process is calculating the hash function by using `userId` as the shard key, and the table sharding process is calculating `DAY_OF_YEAR % 10` through the

MMDD(actionDate) tpartitions 10 function and then splitting the table into 10 physical tables, with one year counted as 365 days.

```
CREATE TABLE user_log5(
  userId int,
  name varchar(30),
  operation varchar(30),
  actionDate DATE
) dbpartition by hash(userId) tpartition by MMDD(actionDate)
tpartitions 10;
```

According to the node topology of the logical table, you can see that 10 table shards are created in each database shard, because one year is counted as 365 days in the function and the table data is routed to 10 physical tables. The responses are very long, and therefore are omitted by using an ellipsis (...).

```
mysql> show topology from user_log5;
+-----+
+-----+
+-----+
| ID    | GROUP_NAME
|       | TABLE_NAME |
+-----+
+-----+
+-----+
| 0     | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log5_00 |
| 1     | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log5_01 |
| 2     | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log5_02 |
| 3     | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log5_03 |
| 4     | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log5_04 |
| 5     | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log5_05 |
| 6     | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log5_06 |
| 7     | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log5_07 |
| 8     | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log5_08 |
| 9     | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0000_RDS | user_log5_09 |
...
| 70    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log5_00 |
| 71    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log5_01 |
| 72    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log5_02 |
| 73    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log5_03 |
| 74    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log5_04 |
| 75    | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log5_05 |
```

```
| 76 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log5_06 |
| 77 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log5_07 |
| 78 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log5_08 |
| 79 | SANGUAN_TEST_123_1488766060743ACTJSANGUAN_TEST_123
_WVVP_0007_RDS | user_log5_09 |
+-----
+-----+
+-----+
80 rows in set (0.02 sec)
```

According to the sharding rule of the logical table, you can see that the database sharding process is calculating the hash function by using `userId` as the shard key, and the table sharding process is calculating the MMDD function by using `actionDate` as the shard key, and then routing the table data to 10 physical tables.

```
mysql> show rule from user_log5;
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| ID    | TABLE_NAME | BROADCAST | DB_PARTITION_KEY | DB_PARTITI
ON_POLICY | DB_PARTITION_COUNT | TB_PARTITION_KEY | TB_PARTITI
ON_POLICY | TB_PARTITION_COUNT |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| 0    | user_log5  | 0    | userId      | hash
| 8    |            |      | actionDate  | mmdd
| 10   |            |      |             |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

7.11.2.5 Use the primary key as the shard key

When no shard key is specified for the sharding algorithm, the system uses the primary key as the shard key by default. The following illustrates how to use the primary key as the database shard key and the table shard key.

Use the primary key as the database shard key

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

Use the primary key as the database shard key and the table shard key

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

```
) dbpartition by hash() tpartition by hash() tpartitions 3;
```

7.11.2.6 Create a broadcast table

The **BROADCAST** clause is used to specify a broadcast table to be created. A broadcast table is replicated to each database shard and data consistency is ensured between the database shards by using a synchronization mechanism with a delay of several seconds. This feature allows you to route a JOIN operation from a Distributed Relational Database Service (DRDS) instance to an underlying ApsaraDB RDS for MySQL instance to prevent the JOIN operation from being performed in multiple databases. [Overview](#) describes how to optimize SQL statements by using broadcast tables.

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

```
CREATE TABLE brd_tbl(
  id int,
  name varchar(30),
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 BROADCAST;
```

7.11.2.7 Other attributes of the MySQL CREATE TABLE statement

When creating table shards in database shards, you can also specify other attributes of the table shards in the MySQL CREATE TABLE statement. For example, you can specify other attributes as follows:

```
CREATE TABLE multi_db_multi_tbl(
  id int,
  name varchar(30),
  primary key(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 dbpartition by hash(id)
tpartition by hash(id) tpartitions 3;
```

7.11.3 ALTER TABLE statement

The syntax of the ALTER TABLE statement used to modify a table is as follows:

```
ALTER [ONLINE|OFFLINE] [IGNORE] TABLE tbl_name
  [alter_specification [, alter_specification] ...]
  [partition_options]
```

In a Distributed Relational Database Service (DRDS) instance, you can use this data definition language (DDL) statement to perform routine DDL operations, such

as adding a column, adding an index, and modifying a data definition. For more information about the syntax, see [MySQL CREATE TABLE Statement](#).



Note:

If you need to modify a table shard, you are not allowed to modify the shard key.

- **Add a column:**

```
ALTER TABLE user_log
  ADD COLUMN idcard varchar(30);
```

- **Add an index:**

```
ALTER TABLE user_log
  ADD INDEX idcard_idx (idcard);
```

- **Delete an index:**

```
ALTER TABLE user_log
  DROP INDEX idcard_idx;
```

- **Modify a field:**

```
ALTER TABLE user_log
  MODIFY COLUMN idcard varchar(40);
```

7.11.4 DROP TABLE statement

The syntax of the DROP TABLE statement used to delete a table is as follows:

```
DROP [TEMPORARY] TABLE [IF EXISTS]
  tbl_name [, tbl_name] ...
  [RESTRICT | CASCADE]
```

The DROP TABLE statement in a Distributed Relational Database Service (DRDS) instance is the same as the DROP TABLE statement in a MySQL database. After the statement is executed, the system automatically deletes the corresponding physical table. For more information about the syntax, see [MySQL DROP TABLE Statement](#).

For example, you can run the following statement to delete the user_log table:

```
DROP TABLE user_log;
```

7.11.5 FAQ about DDL statements

What can I do if an error occurs during table creation?

In a Distributed Relational Database Service (DRDS) instance, the execution of a data definition language (DDL) statement is a distributed processing process. The

error may cause structural inconsistency between table shards. Therefore, you must manually clear the error.

To clear this error, perform the following steps:

1. Check the basic error descriptions provided by the DRDS instance, such as syntactic errors. If the error message is too long, the system will prompt you to call the SQL `SHOW WARNINGS` command to view the failure cause of each database shard.
2. Run the SQL `SHOW TOPOLOGY` command to view the topology of physical tables.

```
SHOW TOPOLOGY FROM multi_db_multi_tbl;
+-----+-----+-----+
| ID    | GROUP_NAME      | TABLE_NAME      |
+-----+-----+-----+
| 0     | corona_qatest_0 | multi_db_multi_tbl_00 |
| 1     | corona_qatest_0 | multi_db_multi_tbl_01 |
| 2     | corona_qatest_0 | multi_db_multi_tbl_02 |
| 3     | corona_qatest_1 | multi_db_multi_tbl_03 |
| 4     | corona_qatest_1 | multi_db_multi_tbl_04 |
| 5     | corona_qatest_1 | multi_db_multi_tbl_05 |
| 6     | corona_qatest_2 | multi_db_multi_tbl_06 |
| 7     | corona_qatest_2 | multi_db_multi_tbl_07 |
| 8     | corona_qatest_2 | multi_db_multi_tbl_08 |
| 9     | corona_qatest_3 | multi_db_multi_tbl_09 |
| 10    | corona_qatest_3 | multi_db_multi_tbl_10 |
| 11    | corona_qatest_3 | multi_db_multi_tbl_11 |
+-----+-----+-----+
12 rows in set (0.21 sec)
```

3. Run the `CHECK TABLE tablename` command to check whether the logical table has been created. For example, the following response indicates that a physical table corresponding to the logical table `multi_db_multi_tbl` failed to be created.

```
mysql> check table multi_db_multi_tbl;
+-----+-----+-----+
+-----+
+-----+-----+-----+
| TABLE                                     | OP    | MSG_TYPE |
| MSG_TEXT                                   |       |          |
|-----|-----|-----|
+-----+-----+-----+
+-----+
+-----+-----+-----+
| andor_mysql_qatest. multi_db_multi_tbl | check | Error    | Table
| 'corona_qatest_0. multi_db_multi_tbl_02' doesn't exist |      |          |
+-----+-----+-----+
+-----+
+-----+-----+-----+
+-----+
```

```
1 row in set (0.16 sec)
```

4. Continue to create or delete the table in idempotent mode to create or delete the remaining physical tables.

```
CREATE TABLE IF NOT EXISTS table1
(id int, name varchar(30), primary key(id))
dbpartition by hash(id);
DROP TABLE IF EXISTS table1;
```

What can I do if I failed to create an index or add a column?

The method for handling the failure in creating an index or adding a column is similar to that for the failure in creating a table. For more information, see [Handle DDL exceptions](#).

7.11.6 DDL sharding functions

7.11.6.1 Overview

Distributed Relational Database Service (DRDS) is a database service that supports both database sharding and table sharding.

Support for database sharding and table sharding

The following table lists the support for database sharding and table sharding in DRDS data definition language (DDL) sharding functions.

Sharding function	Description	Support for database sharding	Support for table sharding
HASH	Performs a simple modulus operation.	Yes	Yes
UNI_HASH	Performs a simple modulus operation.	Yes	Yes
RIGHT_SHIFT	Shifts the value to the right.	Yes	Yes
RANGE_HASH	Performs double hashing.	Yes	Yes
MM	Performs hashing by month.	No	Yes
DD	Performs hashing by date.	No	Yes

Sharding function	Description	Support for database sharding	Support for table sharding
WEEK	Performs hashing by week.	No	Yes
MMDD	Performs hashing by month and date.	No	Yes
YYYYMM	Performs hashing by year and month.	Yes	Yes
YYYYWEEK	Performs hashing by year and week.	Yes	Yes
YYYYDD	Performs hashing by year and date.	Yes	Yes
YYYYMM_OPT	Performs optimized hashing by year and month.	Yes	Yes
YYYYWEEK_OPT	Performs optimized hashing by year and week.	Yes	Yes
YYYYDD_OPT	Performs optimized hashing by year and date.	Yes	Yes

**Note:**

- In a DRDS instance, the sharding method of a logical table is defined jointly by a sharding function and a shard key. The sharding function contains the number of shards to be created and the routing algorithm. The shard key also specifies the MySQL data type of the shard key.
- When the database sharding function is the same as the table sharding function and the database shard key is the same as the table shard key in a DRDS instance, the same sharding method is used for database sharding and table sharding. This allows the DRDS instance to uniquely locate one physical database and one physical table based on the value of the shard key.
- If the database sharding method and the table sharding method of a logical table are different and an SQL query does not contain both database sharding

conditions and table sharding conditions, the DRDS instance scans all database shards or all table shards when processing the SQL query.

Support for data types

Different DRDS DDL sharding functions support different data types. The following table lists the support for various data types in DRDS sharding functions (✓ indicates supported and × indicates not supported):

Figure 7-10: Support for data types in DRDS DDL sharding functions

Sharding function	BIGINT	INT	MEDIUMINT	SMALLINT	TINYINT	VARCHAR	CHAR	DATE	DATETIME	TIMESTAMP	Other types
HASH	✓	✓	✓	✓	✓	✓	✓	×	×	×	×
UNI_HASH	✓	✓	✓	✓	✓	✓	✓	×	×	×	×
RANGE_HASH	✓	✓	✓	✓	✓	✓	✓	×	×	×	×
RIGHT_SHIFT	✓	✓	✓	✓	✓	×	×	×	×	×	×
MM	×	×	×	×	×	×	×	✓	✓	✓	×
DD	×	×	×	×	×	×	×	✓	✓	✓	×
WEEK	×	×	×	×	×	×	×	✓	✓	✓	×
MMDD	×	×	×	×	×	×	×	✓	✓	✓	×
YYYYMM	×	×	×	×	×	×	×	✓	✓	✓	×
YYYYWEEK	×	×	×	×	×	×	×	✓	✓	✓	×
YYYYDD	×	×	×	×	×	×	×	✓	✓	✓	×
YYYYMM_OPT	×	×	×	×	×	×	×	✓	✓	✓	×
YYYYWEEK_OPT	×	×	×	×	×	×	×	✓	✓	✓	×
YYYYDD_OPT	×	×	×	×	×	×	×	✓	✓	✓	×

Syntax of DRDS DDL sharding functions

DRDS is compatible with the CREATE TABLE statement in MySQL, and additionally provides the `drds_partition_options` keyword to support database sharding and table sharding:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...)
    [table_options]
    [drds_partition_options]
    [partition_options]
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)]
    [table_options]
    [drds_partition_options]
    [partition_options]
    select_statement
drds_partition_options:
    DBPARTITION BY
        { {HASH|YYYYMM|YYYYWEEK|YYYYDD|YYYYMM_OPT|YYYYWEEK_OPT|
        YYYYDD_OPT} ([column]) }
```

```
[TBPARTITION BY
  { {HASH|MM|DD|WEEK|MMDD|YYYYMM|YYYYWEEK|YYYYDD|YYYYMM_OPT|
  YYYYWEEK_OPT|YYYYDD_OPT} (column) }
  [TBPARTITIONS num]
]
```

7.11.6.2 HASH

Requirements

- **The shard key must be an integer or a string.**
- **This sharding function has no requirements on the version of a Distributed Relational Database Service (DRDS) instance. It supports all DRDS instances by default.**

Routing method

When the HASH function is run by using different shard keys for database sharding and table sharding, perform the remainder operation on the value of the database shard key based on the number of database shards. If the value of the shard key is a string, the string is converted to a hash value before route calculation. For example, HASH('8') is equivalent to $8 \% D$, where D indicates the number of database shards.

When the UNI_HASH function is run by using the same shard key for both database sharding and table sharding, perform the remainder operation on the value of the shard key based on the total number of table shards. For example, assume that two database shards are created, each database shard contains four table shards, table shards 0 to 3 are stored in database shard 0, and table shards 4 to 7 are stored in database shard 1. If a key value is 15, the key value 15 is distributed to table shard 7 in database shard 1, because $15 \% (2 \times 4) = 7$.

Scenarios

- **HASH is applicable when database sharding is implemented by user ID or order ID.**
- **HASH is also applicable when the shard key is a string.**

Examples

If you need to create a non-partition table in database shards by using the HASH function based on the ID column, you can use the following CREATE TABLE statement:

```
create table test_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);
```

Notes

The HASH is a simple modulus operation. The output distribution of the HASH function can be even only when the values in the shard column are evenly distributed.

7.11.6.3 UNI_HASH

Requirements

- **The shard key must be an integer or a string.**
- **The version of the Distributed Relational Database Service (DRDS) instance must be 5.1.28-1508068 or later. For more information about DRDS versions, see [View the instance version](#).**

Routing method

When the UNI_HASH function is used for database sharding, perform a remainder operation on the value of the database shard key based on the number of database shards. If the value of the shard key is a string, the string is converted to a hash value before route calculation. For example, HASH('8') is equivalent to $8 \% D$, where D indicates the number of database shards.

When the UNI_HASH function is run by using the same shard key for both database sharding and table sharding, perform the remainder operation on the value of the database shard key based on the number of database shards first (this step is different from that in the HASH function). Then, the data is evenly distributed to the table shards in the database shard.

Scenarios

- **UNI_HASH is applicable when database sharding is implemented by user ID or order ID.**
- **UNI_HASH is also applicable when the shard key is an integer or a string.**
- **UNI_HASH can be used when the following conditions are met: two logical tables need to be split into different numbers of table shards in database shards based on the same shard key. In addition, the two tables are frequently joined by using a JOIN statement based on the shard key.**

Comparison with HASH

When you use the UNI_HASH function to create a non-partition table in database shards, the routing method is the same as that used in the HASH function.

Specifically, the route is calculated by performing the remainder operation on the key value of the database shard key based on the number of database shards.

When the UNI_HASH function is run by using the same shard key for both database sharding and table sharding, as the number of table shards changes, the database shard route calculated based on the same key value may also change.

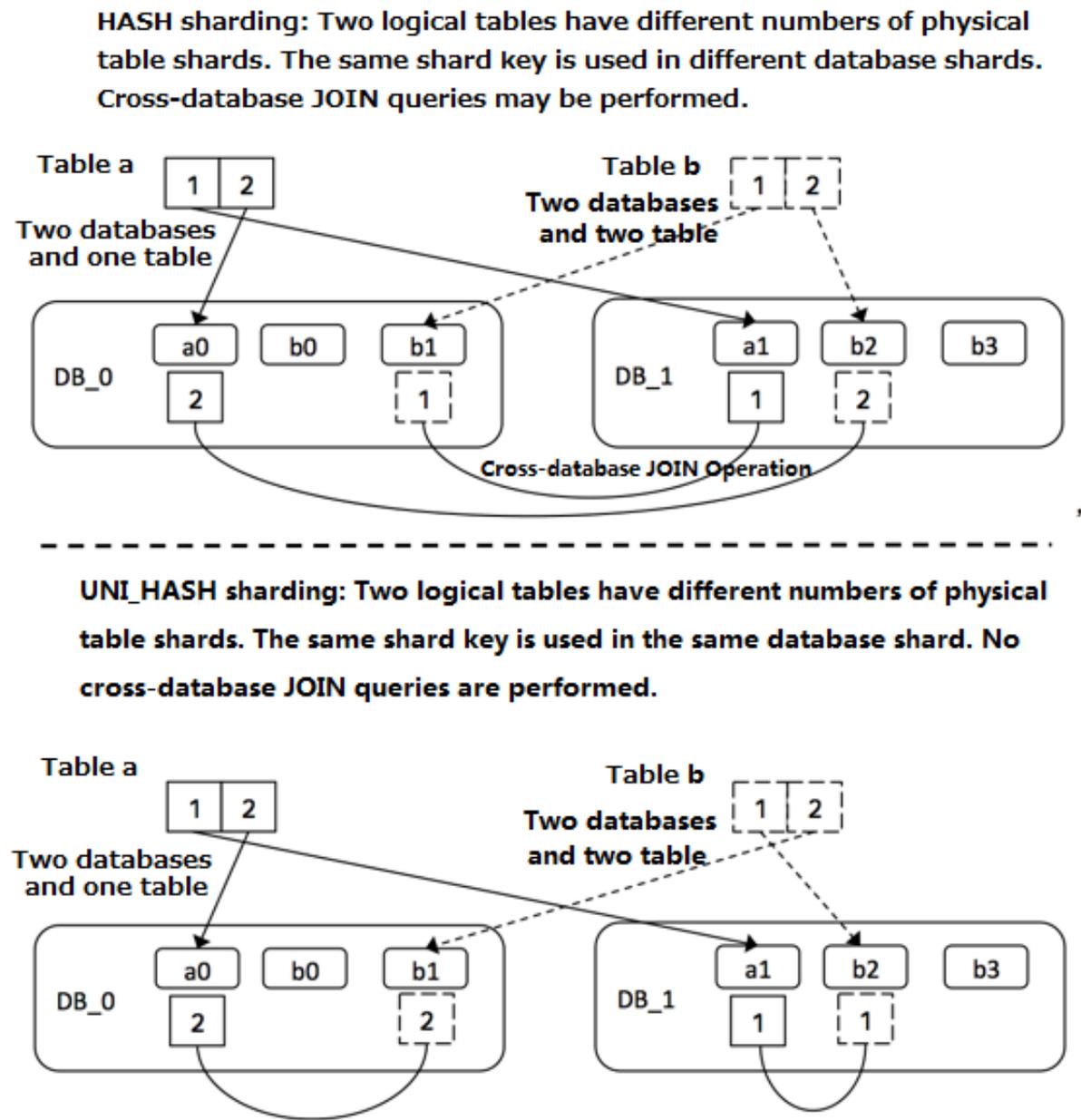
When the UNI_HASH function is run by using the same shard key for both database sharding and table sharding, the database shard route calculated based on the same key value is always the same regardless of the number of table shards.

If two logical tables need to be split into different table shards in database shards based on the same shard key, when the two tables are joined by using the HASH function based on the shard key, multi-database join may occur. However, when the two tables are joined by using the UNI_HASH function based on the shard key, multi-database join does not occur.

Assume that you have two database shards and two logical tables, and each database shard stores one table shard corresponding to logical table a and two table shards corresponding to logical table b. The following figures separately show the results of a JOIN query for logical tables a and b after the HASH function is used for

sharding and the results of a JOIN query for logical tables a and b after the HASH function is used for sharding.

Figure 7-11: Comparison between HASH and UNI_HASH



Examples

If you need to create four table shards in each database shard by using the UNI_HASH function based on the ID column, you can run the following CREATE TABLE statement:

```
create table test_hash_tb (
```

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

Notes

The UNI_HASH is a simple modulus operation. The output distribution of the UNI_HASH function can be even only when the values in the shard column are evenly distributed.

7.11.6.4 RIGHT_SHIFT

Requirements

- **The shard key must be an integer.**
- **The version of the Distributed Relational Database Service (DRDS) instance must be 5.1.28-1320920 or later. For more information about DRDS versions, see [View the instance version](#).**

Routing method

Shift the value of the database shard key to the right by a specified number of binary digits, and then then perform the remainder operation on the obtained integer based on the number of database shards or table shards. In particular, you can specify the number of shifted digits by running a data definition language (DDL) statement.

Scenarios

RIGHT_SHIFT is applicable to improve the evenness of the hash results when the lower-digit parts of most shard key values are very similar to each other but the higher-digit parts vary greatly.

Assume that four shard key values are available: 12340000, 12350000, 12460000, and 12330000. The four lower digits of the four values are all 0000. Directly hashing the values of the shard keys outputs poor results. However, if you run the RIGHT_SHIFT(shardKey, 4) statement to shift the values of the shard keys to the right by four digits, to obtain 1234, 1235, 1246, and 1233, the hashing results are improved.

Examples

If you need to use the ID column as a shard key and shift the values of the ID column to the right by four binary digits to obtain hash values, you can run the following **CREATE TABLE** statement:

```
create table test_hash_tb (  
  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) tpartitions 2;
```

Notes

The number of shifted digits cannot exceed the number of digits occupied by the integer.

7.11.6.5 RANGE_HASH

Requirements

- **The shard key must be a character or a number.**
- **The version of the Distributed Relational Database Service (DRDS) instance must be 5.1.28-1320920 or later. For more information about DRDS versions, see [View the instance version](#).**

Routing method

Calculate the value of the hash value based on the last N digits of any shard key and then perform the remainder operation on the hash value based on the number of database shards. This completes the route computing. The number N is the third parameter in the function.

For example, during calculation of the RANGE_HASH(COL1, COL2, N) function, COL1 is preferentially selected and then truncated to obtain the last N digits for calculation. If COL1 does not exist, COL2 is selected and truncated for calculation.

Scenarios

RANGE_HASH is applicable to scenarios where a table needs to be partitioned by two shard keys but query is performed only based on the value of one shard key.

Examples

Assume that a DRDS database is partitioned into eight physical databases. Our customer has the following requirements:

The order table of a business needs to be partitioned into database shards by buyer ID and order ID. The query is executed based on either the buyer ID or order ID as the condition.

In this case, you can run the following DDL statement to create the order table:

```
create table test_order_tb (  
    id int,  
    buyer_id varchar(30) DEFAULT NULL,  
    order_id varchar(30) DEFAULT NULL,  
    create_time datetime DEFAULT NULL,  
    primary key(id)  
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8  
    dbpartition by RANGE_HASH(buyer_id,order_id, 10)  
    tbpartition by RANGE_HASH (buyer_id,order_id, 10) tbpartitions 3;
```

Notes

- **Neither of the two shard keys can be modified.**
- **Data insertion fails if the two shard keys point to different database shards or table shards.**

7.11.6.6 MM

Requirements

- **The shard key must be of the DATE, DATETIME, or TIMESTAMP type.**
- **MM is only applicable to table sharding.**
- **The version of the Distributed Relational Database Service (DRDS) instance must be 5.1.28-1320920 or later. For more information, see [View the instance version](#).**

Routing method

Perform the remainder operation based on the month that corresponds to the time value of the database shard key to obtain the table shard subscript.

Scenarios

MM is applicable to table sharding by month. The table shard name indicates a specific month.

Examples

Assume that we need to perform database sharding by ID, perform table sharding for the create_time column by month, and map every month to a physical table. The data definition language (DDL) statement is as follows.

```
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 HASH(id)  
tbpartition by MM(create_time) tpartitions 12;
```

Notes

When performing table sharing with MM, ensure that each database shard has no more than 12 table shards because a year has 12 months.

7.11.6.7 DD

Requirements

- **The shard key must be of the DATE, DATETIME, or TIMESTAMP type.**
- **DD is only applicable to table sharding.**
- **The version of the Distributed Relational Database Service (DRDS) instance must be 5.1.28-1320920 or later. For more information, see [View the instance version](#).**

Routing method

Perform the remainder operation based on the day of a month that corresponds to the time value of the database shard key to obtain the table shard subscript.

Scenarios

DD is applicable to table sharding based on a specified number of days in a month, that is, a date. The subscript of the table shard name indicates the day in a month. A month has 31 days at most.

Examples

Assume that we need to perform database sharding by ID, perform table sharding for the create_time column by day, and map every day to a physical table. The data definition language (DDL) statement is as follows.

```
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 HASH(id)  
tbpartition by DD(create_time) tpartitions 31;
```

Notes

When performing table sharing with DD, ensure that each database shard has no more than 31 table shards because a month has 31 days at most.

7.11.6.8 WEEK

Requirements

- **The shard key must be of the DATE, DATETIME, or TIMESTAMP type.**
- **WEEK is only applicable to table sharding.**
- **The version of the Distributed Relational Database Service (DRDS) instance must be 5.1.28-1320920 or later. For more information, see [View the instance version](#).**

Routing method

Perform the remainder operation based on the day of a week that corresponds to the time value of the database shard key to obtain the table shard subscript.

Scenarios

WEEK is applicable to table sharding based on days in a week. The subscript of the table shard name corresponds to each day of a week, from Monday to Sunday.

Examples

Assume that we need to perform database sharding by ID, perform table sharding for the create_time column by week, and map every day of a week (from Monday to Sunday) to a physical table. The data definition language (DDL) statement is as follows.

```
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) tpartitions 7;
```

Notes

When performing table sharding with WEEK, ensure that each database shard has no more than seven table shards because a week has seven days.

7.11.6.9 MMDD

Requirements

- **The shard key must be of the DATE, DATETIME, or TIMESTAMP type.**
- **MMDD is only applicable to table sharding.**
- **The version of the Distributed Relational Database Service (DRDS) instance must be 5.1.28-1320920 or later. For more information, see [View the instance version](#).**

Routing method

Perform the remainder operation based on the number of days in a year that correspond to the time value of the database shard key to obtain the table sharding subscript.

Scenarios

MMDD is applicable to table sharding based on the number of days in a year that correspond to a date in that year. The subscript of the table shard name indicates the day in that year, with a maximum of 366 days in a year.

Examples

Assume that we need to perform database sharding by ID, create tables for the create_time column by date (month-day), and map every day of a year to a physical table. The data definition language (DDL) statement is as follows.

```
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 HASH(name)  
tbpartition by MMDD(create_time) tpartitions 365;
```

Notes

When performing table sharing with MMDD, ensure that each database shard has no more than 366 table shards because a year has 366 days at most.

7.11.6.10 YYYYMM

Requirements

- **The shard key must be of the DATE, DATETIME, or TIMESTAMP type.**
- **The version of the Distributed Relational Database Service (DRDS) instance must be 5.1.28-1320920 or later. For more information, see [View the instance version](#).**

Routing method

Calculate the hash value based on the year and months of the year in the time value of the database shard key and then perform the remainder operation on the hash value based on the number of database shards. This completes route computing.

For example, YYYYMM('2012-12-31 12:12:12') is equivalent to $(2012 * 12 + 12) \% D$, where D indicates the number of database shards.

Scenarios

YYYYMM is applicable to database sharing by year and month. We recommend that you use YYYYMM with `tbpartition YYYYMM(ShardKey)`.

Assume that a DRDS database is partitioned into eight physical databases. Our customer has the following requirements:

- **Perform database sharding for a service by year and month.**
- **Distribute data from the same month to the same table shard and data from every month within two years to a separate table shard.**
- **Distribute a query with the database and table shard key to a physical table shard of a physical database shard.**

The preceding requirements can be met by using YYYYMM. For the requirement of distributing data from every month within two years to a table shard (that is, one table stores the data of one month), create at least 24 physical table shards because a year has 12 months. Create three physical table shards for each database shard because the DRDS instance contains eight database shards. The data definition language (DDL) statement is as follows.

```
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) tpartitions 3;
```

Notes

- **YYYYMM does not support distributing data from every month in every year to a separate table shard. Instead, the number of table shards must be fixed.**
- **After a cycle over months (for example, a cycle exists between 2012-03 and 2013-03), data from the same month may be routed to the same database or table shard, depending on the actual number of table shards.**

7.11.6.11 YYYYWEEK

Requirements

- **The shard key must be of the DATE, DATETIME, or TIMESTAMP type.**
- **The version of the Distributed Relational Database Service (DRDS) instance must be 5.1.28-1320920 or later. For more information, see [View the instance version](#).**

Routing method

Calculate the hash value based on the year and weeks of the year in the time value of the database shard key and then perform the remainder operation on the hash value based on the number of database shards. This completes route computing.

For example, YYYYWEEK('2012-12-31 12:12:12') is equivalent to $(2013 \times 52 + 1) \% D$, with the date 2012-12-31 falling on the first week of 2013, where D indicates the number of database shards.

Scenarios

YYYYWEEK is applicable to database sharing by year and the number of weeks in a year. We recommend that you use YYYYWEEK with tbpartition YYYYWEEK(ShardKey).

Assume that a DRDS database is partitioned into eight physical databases. Our customer has the following requirements:

- **Perform database sharding for a service by year and by week.**
- **Distribute data from the same week to the same table shard and data from every week within two years to a separate table shard.**
- **Distribute a query with the database and table shard key to a physical table shard of a physical database shard.**

The preceding requirements can be met by using **YYYYWEEK**. For the requirement of distributing data from every week within two years to a table shard (that is, one table stores the data of one week), create at least 106 physical table shards because a year has roughly 53 weeks (rounded). Create 14 physical table shards for each database shard because the DRDS instance contains eight database shards ($14 \times 8 = 112 > 106$). We recommend that the number of table shards be an integer multiple of the number of database shards. The data definition language (DDL) statement is as follows.

```
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;
```

Notes

- **YYYYWEEK does not support distributing data from every week in every year to a separate table shard. Instead, the number of table shards must be fixed.**
- **After a cycle over weeks (for example, a cycle exists between the first week of 2012 and the first week of 2013), data from the same week after a cycle may be routed to the same database shard or table shard, depending on the actual number of table shards.**

7.11.6.12 YYYYDD

Requirements

- **The shard key must be of the DATE, DATETIME, or TIMESTAMP type.**
- **The version of the Distributed Relational Database Service (DRDS) instance must be 5.1.28-1320920 or later. For more information, see [View the instance version](#).**

Routing method

Calculate the hash value based on the year and days of the year in the time value of the database shard key and then perform the remainder operation on the hash value based on the number of database shards. This completes route computing.

For example, `YYYYDD('2012-12-31 12:12:12')` is equivalent to $(2012 \times 366 + 365) \% D$, with 2012-12-31 as the 365th day of 2012, where D indicates the number of database shards.

Scenarios

Database sharding is performed by year and the number of days in a year. We recommend that you use YYYYDD with `tbpartition YYYYDD(ShardKey)`.

Assume that a DRDS database is partitioned into eight physical databases. Our customer has the following requirements:

- Perform database sharding for a service by year and day.
- Distribute data from the same week to the same table shard and data from every day within two years to a separate table shard.
- Distribute a query with the database and table shard key to a physical table shard of a physical database shard.

The preceding requirements can be met by using YYYYDD. For the requirement of distributing data from every day within two years to a table shard (that is, one table stores the data of one day), create at least 732 physical table shards because a year has up to 366 days. Create 92 physical table shards for each database shard because the DRDS instance contains eight database shards ($732/8 = 91.5$, rounded to 92). We recommend that the number of table shards be an integer multiple of the number of database shards. The data definition language (DDL) statement is as follows.

```
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;
```

Notes

- YYYYDD does not support distributing data from every day in every year to a separate table shard. Instead, the number of table shards must be fixed.
- After a cycle of a specific date (for example, a cycle exists between 2012-03-01 and 2013-03-01), data from the same date may be routed to the same database shard or table shard, depending on the actual number of table shards.

7.11.6.13 YYYYMM_OPT

Requirements

- The shard key must be of the DATE, DATETIME, or TIMESTAMP type.

- **The year and month of user data increase naturally over time, rather than randomly.**
- **The version of the Distributed Relational Database Service (DRDS) instance must be 5.1.28-1320920 or later. For more information, see [View the instance version](#).**

Optimizations

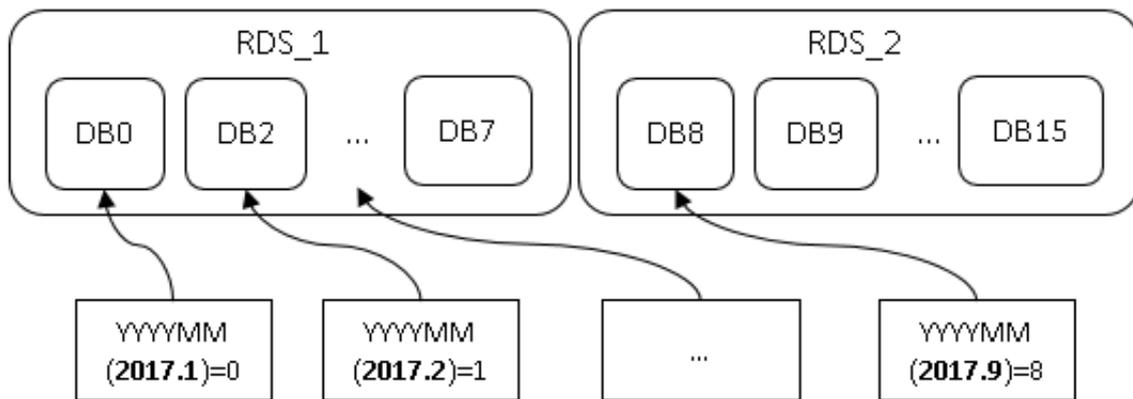
Compared with YYYYMM, YYYYMM_OPT maintains the even distribution of data among ApsaraDB for RDS instances as the timeline increases.

For example, assume that two RDS instances are attached to a DRDS instance, with 16 database shards. DB0 to DB7 shards are located on one ApsaraDB for RDS instance, and DB8 to DB15 shards are located on the other ApsaraDB for RDS instance.

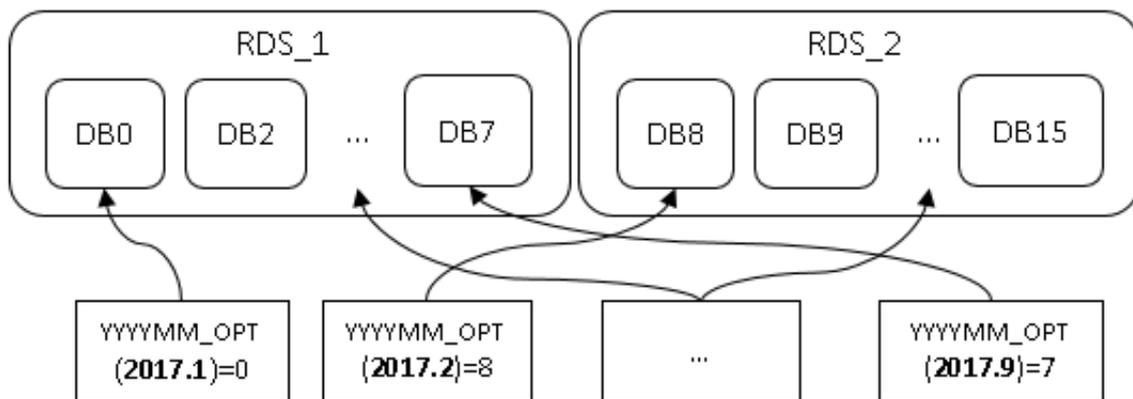
The following figure shows the mappings when YYYYMM and YYYYMM_OPT are used for database sharding, respectively.

Figure 7-12: Comparison between YYYYMM and YYYYMM_OPT

As the time goes on linearly, YYYYMM fills data in ApsaraDB for RDS instances in sequence (data is first distributed to the database shards of RDS_1, then to the database shards of RDS_2, and then to the database shards of RDS_1 again).



YYYYMM_OPT evenly distributes data between ApsaraDB for RDS instances as the time goes on (data is alternately distributed between RDS_1 and RDS_2, so that the data size of the two RDS instances is balanced).



- YYYYMM_OPT distributes data evenly to each ApsaraDB for RDS instance, helping to maximize the performance of each ApsaraDB for RDS instance.

- **How to choose between YYYYMM and YYYYMM_OPT:**
 - **YYYYMM_OPT can be used to distribute data evenly to each ApsaraDB for RDS instance if the time of service data increases sequentially and the data volume does not differ much between the time points.**
 - **YYYYMM is applicable if the time of service data is not consecutive and data appears at random time points.**

Routing method

- **Calculate the hash value based on the year and months of the year in the time value of the database shard key and then perform the remainder operation on the hash value based on the number of database shards. This completes route computing.**
- **The hash calculation based on the database and table shard key considers the data distribution among the ApsaraDB for RDS instances that connect to the DRDS instances.**

Scenarios

- **Database and table sharding is performed by year and month.**
- **Data must be evenly distributed to each ApsaraDB for RDS instance that connects to the DRDS instance.**
- **The time of the shard key increases sequentially rather than randomly, and the data volume is relatively average from month to month. For example, the number of monthly journal logs increases every month, and the log data is not concentrated on the same ApsaraDB for RDS instance.**

Notes

- **YYYYMM_OPT does not support distributing data from every month in every year to a separate table shard. Instead, the number of table shards must be fixed.**
- **After a cycle over months (for example, a cycle exists between 2012-03 and 2013-03), data from the same month may be routed to the same database or table shard, depending on the actual number of table shards.**

7.11.6.14 YYYYWEEK_OPT

Requirements

- **The shard key must be of the DATE, DATETIME, or TIMESTAMP type.**

- **The version of the Distributed Relational Database Service (DRDS) instance must be 5.1.28-1320920 or later. For more information, see [View the instance version](#).**

Optimizations

- **Compared with YYYYWEEK, YYYYWEEK_OPT maintains the even distribution of data among ApsaraDB for RDS instances as the timeline increases. The effect is similar to [YYYYMM_OPT](#).**
- **YYYYWEEK_OPT distributes data evenly to each ApsaraDB for RDS instance, helping to maximize the performance of each ApsaraDB for RDS instance.**
- **How to choose between YYYYWEEK and YYYYWEEK_OPT:**
 - **YYYYWEEK_OPT can be used to distribute data evenly to each ApsaraDB for RDS instance if the time of service data increases sequentially and the data volume does not differ much between time points.**
 - **YYYYWEEK is applicable if the time of service data is not consecutive and data appears at random time points.**

Routing method

- **Calculate the hash value based on the year and weeks of the year in the time value of the database shard key and then perform the remainder operation on the hash value based on the number of database shards. This completes route computing.**
- **The hash calculation based on the database shard and table shard key considers the data distribution among the ApsaraDB for RDS instances that connect to the DRDS instance.**

Scenarios

- **Database sharding and table sharding are performed by year and week, respectively.**
- **Data must be evenly distributed to each ApsaraDB for RDS instance that connects to the DRDS instance.**
- **The time of the shard key increases sequentially rather than randomly, and the data volume is relatively average from week to week. For example, the number of weekly journal logs increases every week, and the log data is not concentrated on the same ApsaraDB for RDS instance.**

Notes

- **YYYYWEEK_OPT does not support distributing data from every week in every year to a separate table shard. Instead, the number of table shards must be fixed.**
- **After a cycle over weeks (for example, a cycle exists between the first week of 2012 and the first week of 2013), data from the same week of the previous cycle may be routed to the same database shard or table shard, depending on the actual number of table shards.**

7.11.6.15 YYYYDD_OPT

Requirements

- **The shard key must be of the DATE, DATETIME, or TIMESTAMP type.**
- **The version of the Distributed Relational Database Service (DRDS) instance must be 5.1.28-1320920 or later. For more information, see [View the instance version](#).**

Optimizations

- **Compared with YYYYDD, YYYYDD_OPT maintains the even distribution of data among ApsaraDB for RDS instances as the timeline increases. The effect is similar to [YYYYMM_OPT](#).**
- **YYYYDD_OPT distributes data evenly to each ApsaraDB for RDS instance, helping to maximize the performance of each RDS instance.**
- **How to choose between YYYYDD and YYYYDD_OPT:**
 - **YYYYDD_OPT can be used to distribute data evenly to each ApsaraDB for RDS instance if the time of service data increases sequentially and the data volume does not differ much between time points.**
 - **YYYYDD is applicable if the time of service data is not consecutive and data appears at random time points.**

Routing method

- **Calculate the hash value based on the year and days of the year in the time value of the database shard key and then perform the remainder operation on the hash value based on the number of database shards. This completes route computing.**
- **The hash calculation based on the database shard and table shard key considers the data distribution among the ApsaraDB for RDS instances that connect to the DRDS instance.**

Scenarios

- **Database sharding and table sharding are performed by year and by day, respectively.**
- **Data must be evenly distributed to each ApsaraDB for RDS instance that connects to the DRDS instance.**
- **The time of the shard key increases sequentially rather than randomly, and the data volume is relatively average from day to day. For example, the number of daily journal logs increases every day, and the log data is not concentrated on the same ApsaraDB for RDS instance.**

Notes

- **YYYYDD_OPT does not support distributing data from every day in every year to a separate table shard. Instead, the number of table shards must be fixed.**
- **After a cycle of a specific date (for example, a cycle exists between 2012-03-01 and 2013-03-01), data from the same date may be routed to the same database shard or table shard, depending on the actual number of table shards.**

7.12 Automatic protection of important SQL statements

In Distributed Relational Database Service (DRDS), the data manipulation language (DML) statements are the same as MySQL statements.

We recommend that you include the shard key in the `SELECT` and `UPDATE` statements of DRDS. The `INSERT` statement of DRDS must include the shard key and a non-empty key value.

By default, DRDS disables full-table deletion and updating to avoid misoperation.

The following statements are prohibited by default:

- **A `DELETE` statement without the `WHERE` or `LIMIT` condition**
- **An `UPDATE` statement without the `WHERE` or `LIMIT` condition**

If you need to perform full-table deletion or update, you can temporarily skip this limit by using the following hint:

```
HINT: /*! TDDL:FORBID_EXECUTE_DML_ALL=false*/
```

Examples

- **Full-table deletion is intercepted by default.**

```
mysql> delete from tt;  
ERR-CODE: [TDDL-4620][ERR_FORBID_EXECUTE_DML_ALL] Forbid execute  
DELETE ALL or UPDATE ALL sql. More: [http://middleware.alibaba-inc.  
com/faq/faqByFaqCode.html?faqCode=TDDL-4620]
```

The operation is successful if the following hint is added:

```
mysql> /*! TDDL:FORBID_EXECUTE_DML_ALL=false*/delete from tt;  
Query OK, 10 rows affected (0.21 sec)
```

- **Full-table update is intercepted by default.**

```
mysql> update tt set id = 1;  
ERR-CODE: [TDDL-4620][ERR_FORBID_EXECUTE_DML_ALL] Forbid execute  
DELETE ALL or UPDATE ALL sql. More: [http://middleware.alibaba-inc.  
com/faq/faqByFaqCode.html?faqCode=TDDL-4620]
```

The operation is successful if the following HINT is added:

```
mysql> /*! TDDL:FORBID_EXECUTE_DML_ALL=false*/update tt set id = 1;  
Query OK, 10 rows affected (0.21 sec)
```

- **This limit does not apply to DELETE or UPDATE statements that contain the WHERE or LIMIT condition.**

```
mysql> delete from tt where id = 1;  
Query OK, 1 row affected (0.21 sec)
```

7.13 DRDS sequence

7.13.1 Overview

A Distributed Relational Database Service (DRDS) sequence is a 64-digit number that corresponds to the signed BIGINT type in MySQL. It is used to create a globally unique and sequentially incremental numeric sequence, such as the values of primary key columns and unique index columns.

DRDS sequences are used in the following two ways:

- **Explicit sequences are created and maintained by using sequence-specific data definition language (DDL) syntax and can be used independently. The sequence**

value can be acquired by using `select seq.nextval;`, in which `seq` indicates the sequence name.

- Implicit sequences are used to automatically fill in primary keys with `AUTO_INCREMENT` defined and are automatically maintained by DRDS.



Notice:

DRDS creates implicit sequences only after `AUTO_INCREMENT` is defined for partitioned tables and broadcast tables. This is not the case for non-partition tables. The `AUTO_INCREMENT` value of a non-partition table is created by ApsaraDB RDS for MySQL.

Types and features of DRDS sequences

Currently, three types of DRDS sequences are supported.

Type (abbreviation)	Globally unique	Consecutive	Monotonically increasing	Monotonically increasing within the same connection	Non-single point	Data type	Readability
Group sequence (GROUP)	Yes	No	No	Yes	Yes	All integer types	High
Time-based sequence (TIME)	Yes	No	Monotonically increasing at the macro level and non-monotonically increasing at the micro level	Yes	Yes	Only BIGINT is supported	Low

Type (abbreviation)	Globally unique	Consecutive	Monotonically increasing	Monotonically increasing within the same connection	Non-single point	Data type	Readability
Simple sequence (SIMPLE)	Yes	Yes	Yes	Yes	No	All integer types	High

Concepts:

- **Consecutive:** If the current value is n , the next value must be $n + 1$. If the next value is not $n + 1$, it is nonconsecutive.
- **Monotonically increasing:** If the current value is n , the next value must be a number greater than n .
- **Single point:** The risk of single point of failure exists.
- **Monotonically increasing at the macro level and non-monotonically increasing at the micro level:** An example of this is 1, 3, 2, 4, 5, 7, 6, 8, ...

Group sequence (GROUP, used by default)

Features

A group sequence is a globally unique sequence with natural numeric values, which are not necessarily consecutive or monotonically increasing. If the sequence type is not specified, DRDS uses the group sequence type by default.

- **Advantages:** A group sequence is globally unique and provides excellent performance, preventing single point of failure.
- **Disadvantages:** A group sequence may contain nonconsecutive values, which may not necessarily start from the initial value and do not cycle.

Implementation

The values of a group sequence are created by multiple nodes to ensure high availability. The values in a segment are nonconsecutive if the values are not all used, such as in the case of disconnection.

Time-based sequence

Features

A time-based sequence consists of a timestamp, node ID, and serial number. It is globally unique and automatically increments at the macro level. Value updates are database-independent and not persistently stored in databases. Only names and types are stored in databases. This delivers good performance to time-based sequences, which create values like 776668092129345536, 776668098018148352, 776668111578333184, and 776668114812141568.



Notice:

Sequence values must be of the BIGINT type when used in the auto-increment columns of tables.

- **Advantages:** Time-based sequences are globally unique with good performance.
- **Disadvantages:** The values of a time-based sequence are nonconsecutive. The **START WITH**, **INCREMENT BY**, **MAXVALUE**, and **CYCLE** or **NOCYCLE** parameters are invalid for time-based sequences.

Simple sequence

Features

Only simple sequences support the **START WITH**, **INCREMENT BY**, **MAXVALUE**, and **CYCLE** or **NOCYCLE** parameters.

- **Advantages:** Simple sequences are globally unique and monotonically increasing with consecutive values.
- **Disadvantages:** Simple sequences are prone to single point of failure, poor performance, and bottlenecks. Use them with caution.

Implementation

Each sequence value must be persistently stored.

Scenarios

Group sequences, time-based sequences, and simple sequences are globally unique and can be used in primary key columns and unique index columns.

- We recommend that you use group sequences.

- Use only simple sequences for services that strongly depend on consecutive sequence values. Pay attention to sequence performance.
- We recommend that you use time-based sequences if you have high requirements for sequence performance, the amount of data inserted to tables is small, and large sequence values are acceptable. Time-based sequences are CPU-bound with no requirements on computing lock, database dependence, or persistent storage.

The following example shows how to create a sequence with an initial value of 100000 and a step of 1.

- A simple sequence creates globally unique, consecutive, and monotonically increasing values, such as 100000, 100001, 100002, 100003, 100004, ..., 200000, 200001, 200002, 200003... The values of a simple sequence are persistently stored. Even after services are restarted upon a single point of failure, values are still created consecutively from the breakpoint. However, simple sequences have poor performance because each value is persistently stored once it is created.
- A group sequence may create values like 200001, 200002, 200003, 200004, 100001, 100002, 100003...



Notice:

- The initial value of a group sequence is not necessarily the same as the **START WITH** value (which is 100000 in this example) but is invariably greater than this value. In this example, the initial value is 200001.
 - A group sequence is globally unique but may contain nonconsecutive values, which may occur when a node is faulty or the connection that only uses partial values is closed. The group sequence in this example contains nonconsecutive values because the values between 200004 and 100001 are missing.
- A time-based sequence may create values like 776668092129345536, 776668098018148352, 776668111578333184, 776668114812141568...

7.13.2 Explicit sequence usage

This topic describes how to use data definition language (DDL) statements to create, modify, delete, and query sequences and how to acquire the values of explicit sequences.

Create a sequence

Syntax:

```
CREATE [ GROUP | SIMPLE | TIME ] SEQUENCE <name>
[ START WITH <numeric value> ] [ INCREMENT BY <numeric value> ]
[ MAXVALUE <numeric value> ] [ CYCLE | NOCYCLE ]
```

Parameters:

Parameter	Description	Applicable To
START WITH	The initial sequence value. If it is not set, the default value is 1.	Simple sequence and group sequence
INCREMENT BY	The increment (or interval value or step) of each sequence increase. If it is not set, the default value is 1.	Simple sequence
MAXVALUE	The maximum sequence value. If it is not specified, the default value is the maximum value of the signed BIGINT type.	Simple sequence
CYCLE or NOCYCLE	Indicates whether to repeat the sequence value which starts from the value specified by START WITH after the sequence value reaches the maximum value. If it is not specified, the default value is NOCYCLE.	Simple sequence



Note:

- If the sequence type is not specified, the group sequence type is used by default.
- The INCREMENT BY, MAXVALUE, and CYCLE or NOCYCLE parameters are invalid for group sequences.
- The START WITH, INCREMENT BY, MAXVALUE, and CYCLE or NOCYCLE parameters are invalid for time-based sequences.

- **Group sequences are nonconsecutive. The `START WITH` parameter only provides reference for group sequences. The initial group sequence value is not necessarily the same as but is greater than the value of `START WITH`.**

Example 1: Create a group sequence.

- **Method 1:**

```
mysql> CREATE SEQUENCE seq1;
Query OK, 1 row affected (0.27 sec)
```

- **Method 2:**

```
mysql> CREATE GROUP SEQUENCE seq1;
Query OK, 1 row affected (0.27 sec)
```

Example 2: Create a time-based sequence.

```
mysql> CREATE TIME SEQUENCE seq1;
Query OK, 1 row affected (0.27 sec)
```

Example 3: Create a simple sequence with an initial value of 1000, step of 2, and maximum value of 9999999999, which does not repeat after increasing to the maximum value.

```
mysql> CREATE SIMPLE SEQUENCE seq2 START WITH 1000 INCREMENT BY 2
MAXVALUE 9999999999 NOCYCLE;
Query OK, 1 row affected (0.03 sec)
```

Modify a sequence

Distributed Relational Database Service (DRDS) allows you to modify sequences in the following ways:

- **For simple sequences, change the values of `START WITH`, `INCREMENT BY`, `MAXVALUE`, and `CYCLE` or `NOCYCLE`.**
- **For group sequences, change the value of `START WITH`.**
- **Convert the sequence type to another.**

Syntax:

```
ALTER SEQUENCE <name> [ CHANGE TO GROUP | SIMPLE | TIME ]
START WITH <numeric value> [ INCREMENT BY <numeric value> ]
[ MAXVALUE <numeric value> ] [ CYCLE | NOCYCLE ]
```

Parameters:

Parameter	Description	Applicable To
START WITH	The initial sequence value. If it is not set, the default value is 1.	Simple sequence and group sequence
INCREMENT BY	The increment (or interval value or step) of each sequence increase. If it is not set, the default value is 1.	Simple sequence
MAXVALUE	The maximum sequence value. If it is not specified, the default value is the maximum value of the signed BIGINT type.	Simple sequence
CYCLE or NOCYCLE	Indicates whether to repeat the sequence value which starts from the value specified by START WITH after the sequence value reaches the maximum value. If it is not specified, the default value is NOCYCLE .	Simple sequence

**Note:**

- **Group sequences are nonconsecutive. The **START WITH** parameter only provides reference for group sequences. The initial group sequence value is not necessarily the same as but is greater than the value of **START WITH**.**
- **If you set **START WITH** when modifying a simple sequence, the **START WITH** value takes effect immediately. The following sequence value starts from the new **START WITH** value. For example, if you change the **START WITH** value to 200 when the sequence value increases to 100, the following sequence value starts from 200.**
- **Before changing the **START WITH** value, you need to analyze the existing sequence values and the speed of creating sequence values to avoid conflicts. Do not change the **START WITH** value unless necessary.**

Example: Change the initial value, step, and maximum value of the simple sequence named seq2 to 3000, 5, and 1000000 respectively, and set **CYCLE.**

```
mysql> ALTER SEQUENCE seq2 START WITH 3000 INCREMENT BY 5 MAXVALUE
1000000 CYCLE;
```

Query OK, 1 row affected (0.01 sec)

Convert the sequence type to another

- Use the `CHANGE TO <sequence_type>` clause of `ALTER SEQUENCE`.
- If the `CHANGE TO` clause of `ALTER SEQUENCE` is specified, add the `START WITH` parameter to prevent duplicate values. This parameter is optional if `CHANGE TO` is not specified.

Example: Convert a group sequence to a simple sequence.

```
mysql> ALTER SEQUENCE seq1 CHANGE TO SIMPLE START WITH 1000000;
Query OK, 1 row affected (0.02 sec)
```

Delete a sequence

Syntax:

```
DROP SEQUENCE <name>
```

Example:

```
mysql> DROP SEQUENCE seq3;
Query OK, 1 row affected (0.02 sec)
```

Query sequences

Syntax:

```
SHOW SEQUENCES
```

Example: The TYPE column lists the sequence types in the abbreviated form.

```
mysql> SHOW SEQUENCES;
+-----+-----+-----+-----+-----+
| NAME          | VALUE          | CYCLE | TYPE  | INCREMENT_BY | START_WITH |
+-----+-----+-----+-----+-----+
| AUTO_SEQ_1   | 91820513      | N     | SIMPLE | 1             | 91820200  |
| AUTO_SEQ_4   | 91820200      | Y     | SIMPLE | 2             | 1000      |
| seq_test     | N/A           |       |       | N/A           | N/A       |
| AUTO_SEQ_2   | 100000        |       | TIME  | N/A           | N/A       |
| AUTO_SEQ_3   | 200000        |       | GROUP | N/A           | N/A       |
+-----+-----+-----+-----+-----+
```

```
5 rows in set (0.01 sec)
```

Get the sequence value

Syntax:

```
< sequence name >.NEXTVAL
```

Example:

```
SELECT sample_seq.nextVal FROM dual;
+-----+
| SAMPLE_SEQ.NEXTVAL |
+-----+
|           101001 |
+-----+
1 row in set (0.04 sec)
```

You can also write `SAMPLE_SEQ.nextVal` as a value to the SQL statement:

```
mysql> INSERT INTO some_users (name,address,gmt_create,gmt_modified,
intro) VALUES ('sun',SAMPLE_SEQ.nextVal,now(),now(),'aa');
Query OK, 1 row affected (0.01 sec)
```



Note:

If you set the `AUTO_INCREMENT` parameter when creating a table, you do not need to specify an auto-increment column when running the `INSERT` statement. The auto-increment column is automatically maintained by DRDS.

Acquire the values of sequences in batches

Syntax:

```
SELECT < sequence name >.NEXTVAL FROM DUAL WHERE COUNT = < numeric
value >
```

Example:

```
SELECT sample_seq.nextVal FROM dual WHERE count = 10;
+-----+
| SAMPLE_SEQ.NEXTVAL |
+-----+
|           101002 |
|           101003 |
|           101004 |
|           101005 |
|           101006 |
|           101007 |
|           101008 |
|           101009 |
|           101010 |
|           101011 |
+-----+
```

```
10 rows in set (0.04 sec)
```

7.13.3 Implicit sequence usage

After `AUTO_INCREMENT` is set for a primary key, the primary key is automatically filled in by using a sequence which is maintained by Distributed Relational Database Service (DRDS).

CREATE TABLE

The standard CREATE TABLE syntax is extended to add the sequence type for auto-increment columns. If the type keyword is not specified, the default type is GROUP. The sequence names that are automatically created by DRDS and associated with tables are prefixed with `AUTO_SEQ_` and suffixed with the table name.

```
CREATE TABLE <name> (  
    <column> ... AUTO_INCREMENT [ BY GROUP | SIMPLE | TIME ],  
    <column definition>,  
    ...  
) ... AUTO_INCREMENT=<start value>
```

SHOW CREATE TABLE

The sequence type is displayed for the auto-increment column of a table shard or broadcast table.

```
SHOW CREATE TABLE <name>
```

Examples

- If `AUTO_INCREMENT` is set but the sequence type is not specified when a table is created, the group sequence type is used by default.

Figure 7-13: Example 1

```
mysql> CREATE TABLE `xkv_shard` (  
-> `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT COMMENT ' ',  
-> `gmt_create` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' ON UPDATE CURRENT_TIMESTAMP COMMENT ' ',  
-> `uid` bigint(20) unsigned DEFAULT '10' COMMENT 'uid',  
-> `msg` varchar(40) DEFAULT '127.0.0.1' COMMENT 'desc',  
-> `val` float DEFAULT '0' COMMENT 'val',  
-> `time` time DEFAULT NULL COMMENT 'time',  
-> PRIMARY KEY (`id`),  
-> UNIQUE KEY `msg` (`msg`)  
->) ENGINE=InnoDB AUTO_INCREMENT=100009 DEFAULT CHARSET=utf8 dbpartition by hash(`id`);  
Query OK, 0 rows affected (1.24 sec)  
  
mysql> show create table xkv_shard;  
+-----+-----+  
| Table | Create Table  
+-----+-----+  
| xkv_shard | CREATE TABLE `xkv_shard` (  
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT BY GROUP COMMENT ' ',  
  `gmt_create` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' ON UPDATE CURRENT_TIMESTAMP COMMENT ' ',  
  `uid` bigint(20) unsigned DEFAULT '10' COMMENT 'uid',  
  `msg` varchar(40) DEFAULT '127.0.0.1' COMMENT 'desc',  
  `val` float DEFAULT '0' COMMENT 'val',  
  `time` time DEFAULT NULL COMMENT 'time',  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `msg` (`msg`)  
) ENGINE=InnoDB AUTO_INCREMENT=100009 DEFAULT CHARSET=utf8 dbpartition by hash(`id`) |  
+-----+-----+  
1 row in set (0.02 sec)  
  
mysql> drop table xkv_shard;
```

- When creating a table, set `AUTO_INCREMENT` and specify a time-based sequence as the primary key value.

Figure 7-14: Example 2

```
mysql> CREATE TABLE `timeseq_test` (  
-> `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT BY TIME COMMENT ' ',  
-> `gmt_create` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' ON UPDATE CURRENT_TIMESTAMP COMMENT ' ',  
-> `uid` bigint(20) unsigned DEFAULT '10' COMMENT 'uid',  
-> `msg` varchar(40) DEFAULT '127.0.0.1' COMMENT 'desc',  
-> `val` float DEFAULT '0' COMMENT 'val',  
-> `time` time DEFAULT NULL COMMENT 'time',  
-> PRIMARY KEY (`id`),  
-> UNIQUE KEY `msg` (`msg`)  
->) ENGINE=InnoDB AUTO_INCREMENT=100009 DEFAULT CHARSET=utf8 dbpartition by hash(`id`);  
Query OK, 0 rows affected (1.27 sec)  
  
mysql> show create table timeseq_test;  
+-----+-----+  
| Table | Create Table  
+-----+-----+  
| timeseq_test | CREATE TABLE `timeseq_test` (  
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT BY TIME COMMENT ' ',  
  `gmt_create` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' ON UPDATE CURRENT_TIMESTAMP COMMENT ' ',  
  `uid` bigint(20) unsigned DEFAULT '10' COMMENT 'uid',  
  `msg` varchar(40) DEFAULT '127.0.0.1' COMMENT 'desc',  
  `val` float DEFAULT '0' COMMENT 'val',  
  `time` time DEFAULT NULL COMMENT 'time',  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `msg` (`msg`)  
) ENGINE=InnoDB AUTO_INCREMENT=100009 DEFAULT CHARSET=utf8 dbpartition by hash(`id`) |  
+-----+-----+  
1 row in set (0.04 sec)
```

ALTER TABLE

Currently, ALTER TABLE cannot be used to change the sequence type but can be used to change the initial value. If you want to modify the data of the implicit sequence type in a table, run SHOW SEQUENCES to find the names and types of sequences and run ALTER SEQUENCE to modify the data.

```
ALTER TABLE <name> ... AUTO_INCREMENT=<start value>
```

**Notice:**

Exercise caution when changing the initial value of AUTO_INCREMENT after DRDS sequences are used. You need to analyze the existing sequence values and the speed of creating sequence values to avoid conflicts.

7.13.4 Limits and precautions for DRDS sequence

This topic describes the limits and precautions for Distributed Relational Database Service (DRDS) sequence.

Limits and precautions

- **When a time-based sequence is used in the auto-increment column of a table, the column must be of the BIGINT type.**
- **START WITH must be set when the sequence is changed to another type.**
- **When the INSERT statement is executed on a DRDS database in non-partition mode or on a DRDS database that has only one table but no broadcast table, DRDS automatically optimizes and sends the statement, and bypasses the part of the optimizer that allocates the sequence value. The non-partition mode indicates that only one ApsaraDB for RDS database is bound to the DRDS database. In this case, INSERT INTO ... VALUES (seq.nextval, ...) is not supported. We recommend that you use the ApsaraDB RDS for MySQL auto-increment column feature instead.**
- **If the hint for a specific database shard is used by the INSERT statement such as INSERT INTO ... VALUES ... or INSERT INTO ... SELECT ... and the target table uses a sequence, DRDS bypasses the optimizer and directly pushes the statement so that the sequence does not take effect. The target table creates an ID by using the auto-increment feature of the ApsaraDB RDS for MySQL table.**
- **The auto-increment ID allocation method for the same table must be kept consistent, which may be based on DRDS sequences or the auto-increment**

column of the ApsaraDB RDS for MySQL table. If both of the two allocation methods are used for the same table, duplicate IDs may be created and making location difficult.

Troubleshoot primary key conflicts

Assume that data is directly written to ApsaraDB RDS for MySQL and that the related primary key value is not the DRDS-created sequence value. If DRDS automatically creates a primary key and writes it to the database, this primary key may conflict with that of the directly written data. This problem can be resolved as follows.

- 1. View the existing sequences by using the DRDS-specified SQL statement. The sequence prefixed with `AUTO_SEQ_` is an implicit sequence. This is the sequence that is created when a table is created with the `AUTO_INCREMENT` parameter.**

```
mysql> SHOW SEQUENCES;
+-----+-----+-----+-----+-----+
+-----+-----+-----+
| NAME          | VALUE | INCREMENT_BY | START_WITH |
MAX_VALUE | CYCLE | TYPE |
+-----+-----+-----+-----+-----+
| AUTO_SEQ_timeseq_test | N/A | N/A | N/A | N/A
| N/A | TIME |
| AUTO_SEQ_xkv_shard_tbl1 | 0 | N/A | N/A | N/A
| N/A | GROUP |
| AUTO_SEQ_xkv_shard | 0 | N/A | N/A | N/A
| N/A | GROUP |
+-----+-----+-----+-----+-----+
3 rows in set (0.04 sec)
```

- 2. For example, if the `t_item` table contains conflicts and its primary key is `ID`, then acquire the maximum primary key value of this table from DRDS.**

```
mysql> SELECT MAX(id) FROM t_item;
+-----+
| max(id) |
+-----+
| 8231 |
+-----+
1 row in set (0.01 sec)
```

- 3. Update the related value in the DRDS sequence table to a value greater than 8231, such as 9000. Then, no error is returned for the auto-increment primary key created by the subsequent `INSERT` statement.**

```
mysql> ALTER SEQUENCE AUTO_SEQ_USERS START WITH 9000;
```

Query OK, 1 row affected (0.01 sec)

7.14 Best practices

7.14.1 Select a shard key

A shard key is a field for database sharding and table sharding, which is used to create sharding rules during horizontal partitioning. Distributed Relational Database Service (DRDS) partitions a logical table horizontally into the physical database shards on each ApsaraDB for RDS instance based on the shard key.

The primary principle of sharding is to identify the business logic-specific subject of data in a table as much as possible and confirm that most (or core) database operations are performed based on this subject. Then, use the subject-related field as the shard key to perform database sharding and table sharding.

The business logic-specific subject is related to business scenarios. The following typical scenarios include business logic-specific subjects that can be used as shard keys:

- **User-oriented Internet applications are operated to meet user requirements. Users are the business logic-specific subject and the user-related field can be used as the shard key.**
- **Seller-oriented e-commerce applications are operated to meet seller requirements. Sellers are the business logic-specific subject and the seller-related field can be used as the shard key.**
- **Game-oriented applications are operated to meet gamer requirements. Gamers are the business logic-specific subject and the gamer-related field can be used as the shard key.**
- **Internet of Vehicles (IoV) applications are operated based on vehicle information. Vehicles are the business logic-specific subject and the vehicle-related field can be used as the shard key.**
- **Tax-oriented applications are operated based on taxpayer information to provide frontend services. Taxpayers are the business logic-specific subject and the taxpayer-related field can be used as the shard key.**

In other scenarios, you can also use the appropriate subject of business logic as the shard key.

For example, in a seller-oriented e-commerce application, the following single table must be horizontally partitioned:

```
CREATE TABLE sample_order (  
  id INT(11) NOT NULL,  
  sellerId INT(11) NOT NULL,  
  trade_id INT(11) NOT NULL,  
  buyer_id INT(11) NOT NULL,  
  buyer_nick VARCHAR(64) DEFAULT NULL,  
  PRIMARY KEY (id)  
)
```

The sellerId field is used as the shard key because seller is the business logic-specific subject. In the case of database sharding but no table sharding, the distributed data definition language (DDL) statement for table creation is as follows:

```
CREATE TABLE sample_order (  
  id INT(11) NOT NULL,  
  sellerId INT(11) NOT NULL,  
  trade_id INT(11) NOT NULL,  
  buyer_id INT(11) NOT NULL,  
  buyer_nick VARCHAR(64) DEFAULT NULL,  
  PRIMARY KEY (id)  
) DBPARTITION BY HASH(sellerId)
```

If no business logic-specific subject can be used as the shard key, use the following methods to select an appropriate shard key:

- **Determine the shard key based on the distribution of data and access requests**
 - **Distribute the data in a table to different physical database shards and table shards as evenly as possible. This method is applicable to scenarios with massive analytical queries, in which query concurrency stays at 1.**
- **Determine the shard key for database sharding and table sharding by combining fields of the numeric (string) type and time type. This method is applicable to log retrieval.**

For example, a log system records all user operations and needs to horizontally partition the following single log table:

```
CREATE TABLE user_log (  
  userId INT(11) NOT NULL,  
  name VARCHAR(64) NOT NULL,  
  operation VARCHAR(128) DEFAULT NULL,  
  actionDate DATE DEFAULT NULL
```

)

You can combine the user identifier with the time field to create a shard key for partitioning the table by week. The distributed DDL statement for table creation is as follows:

```
CREATE TABLE user_log (
  userId INT(11) NOT NULL,
  name VARCHAR(64) NOT NULL,
  operation VARCHAR(128) DEFAULT NULL,
  actionDate DATE DEFAULT NULL
) DBPARTITION BY HASH(userId) TBPARTITION BY WEEK(actionDate)
TBPARTITIONS 7
```

For more information about shard key selection and table shard forms, see [DDL statements](#).



Notice:

Avoid using hotspot data as the shard key if possible.

7.14.2 Select the number of shards

Distributed Relational Database Service (DRDS) supports horizontal partitioning of databases and tables. By default, eight physical database shards are created on an ApsaraDB for RDS instance, and one or more physical table shards can be created on each physical database shard. The number of table shards is also called the number of shards.

Generally, we recommend that each physical table shard contain no more than 5 million rows of data. Generally, you can estimate the data growth in one to two years. Divide the estimated total data size by the total number of physical database shards, and then divide the result by the recommended maximum data size of 5 million, to obtain the number of physical table shards to be created on each physical database shard:

```
Number of physical table shards in each physical database shard
= CEILING(Estimated total data size/(Number of ApsaraDB for RDS
instances x 8)/5,000,000)
```

Therefore, when the calculated number of physical table shards is equal to 1, only database sharding needs to be performed, that is, a physical table shard is created in each physical database shard. If the calculation result is greater than 1, we recommend that you perform both database sharding and table sharding, that is, there are multiple physical table shards in each physical database shard.

For example, if a user estimates that the total data size of a table will be about 0.1 billion rows two years later and the user has four ApsaraDB for RDS instances, then according to the preceding formula:

```
Number of physical table shards in each physical database shard =
CEILING(100,000,000/(4 x 8)/5,000,000) = CEILING(0.625) = 1
```

If the result is 1, only database sharding is needed, that is, one physical table shard is created in each physical database shard.

If only one ApsaraDB for RDS instance is used in the preceding example, the formula is as follows:

```
Number of physical table shards in each physical database shard =
CEILING(100,000,000/(1 x 8)/5,000,000) = CEILING(2.5) = 3
```

If the result is 3, we recommend that you create three physical table shards in each physical database shard.

7.14.3 Basic concepts of SQL optimization

Distributed Relational Database Service (DRDS) is an efficient and stable distributed relational database service that processes distributed relational computing. DRDS optimizes SQL statements differently from single-instance relational databases such as MySQL. DRDS focuses on the network I/O overheads in a distributed environment and pushes SQL operations down to the underlying database shards (such as databases on ApsaraDB for RDS instances) for execution, thereby reducing the network I/O overheads and improving the SQL execution efficiency.

DRDS provides commands for obtaining the SQL execution information to help SQL optimization, for example, EXPLAIN commands for obtaining SQL execution plans and TRACE commands for obtaining SQL execution processes and overheads. This topic describes the basic concepts and common commands related to SQL optimization in DRDS.

Execution plan

An SQL execution plan (or execution plan) is a set of ordered operation steps generated to access data. In DRDS, the execution plan is divided into the execution plan at the DRDS layer and the execution plan at the ApsaraDB for RDS layer. Execution plan analysis is an effective way to optimize SQL statements. Through execution plan analysis, you can know whether DRDS or ApsaraDB for RDS has

generated optimal execution plans for SQL statements and whether further optimization can be made.

During SQL statement execution, based on the basic information of the SQL statement and related tables, the DRDS optimizer determines on which the database shards the SQL statement should be executed, and the specific SQL statement form, execution policy, and data merging and computing policy for each database shard. This process optimizes SQL statement execution and generates execution plans at the DRDS layer. The execution plan at the ApsaraDB for RDS layer is the native MySQL execution plan.

DRDS provides a set of EXPLAIN commands to display execution plans at different levels or with different levels of detail.

The following table briefly describes the EXPLAIN commands in DRDS.

Table 7-1: EXPLAIN command description

Command	Description	Example
EXPLAIN { SQL }	Displays the summary execution plan of SQL statements at the DRDS layer, including the database shards on which the SQL statement is run, physical statements, and general parameters.	EXPLAIN SELECT * FROM test
EXPLAIN DETAIL { SQL }	Displays the detailed execution plans of SQL statements at the DRDS layer, including the statement type, concurrency, returned field information, physical tables, and database groups.	EXPLAIN DETAIL SELECT * FROM test
EXPLAIN EXECUTE { SQL }	Displays the execution plan of the underlying ApsaraDB for RDS instance, which is equivalent to the EXPLAIN statement of MySQL.	EXPLAIN EXECUTE SELECT * FROM test

Execution plans at the DRDS layer

The following table describes the fields in the results returned for a DRDS-layer execution plan.

Table 7-2: Description of fields in DRDS-layer execution plans

Field	Description
GROUP_NAME	The name of the DRDS database shard. The suffix identifies the specific database shard. The value is consistent with the result of the SHOW NODE command.
SQL	The SQL statement run on this database shard.
PARAMS	The SQL statement parameters used when DRDS communicates with ApsaraDB for RDS over the Prepare protocol.

The SQL field can be in two forms:

1. If an SQL statement does not contain the following parts, the execution plan is displayed as an SQL statement:

- Aggregate function involving multiple database shards.
- Distributed JOIN queries involving multiple shards.
- Complex subqueries.

Example:

```
mysql> EXPLAIN SELECT * FROM test;
+-----+
+-----+-----+-----+
| GROUP_NAME                                | SQL
+-----+-----+-----+
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0000_RDS | select `test`.`c1`
, `test`.`c2` from `test` | {}
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0001_RDS | select `test`.`c1`
, `test`.`c2` from `test` | {}
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0002_RDS | select `test`.`c1`
, `test`.`c2` from `test` | {}
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0003_RDS | select `test`.`c1`
, `test`.`c2` from `test` | {}
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0004_RDS | select `test`.`c1`
, `test`.`c2` from `test` | {}
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0005_RDS | select `test`.`c1`
, `test`.`c2` from `test` | {}
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0006_RDS | select `test`.`c1`
, `test`.`c2` from `test` | {}
```

```
| TESTDB_1478746391548CDTCTESTDB_OXGJ_0007_RDS | select `test`.`c1`
| , `test`.`c2` from `test` | {} |
+-----+
+-----+
8 rows in set (0.04 sec)
```

The group names displayed in the GROUP_NAME field can be found in the returned result of SHOW NODE:

```
mysql> SHOW NODE;
+-----+
+-----+
+-----+
| ID | NAME | MASTER_READ_COUNT | MASTER_READ_PERCENT | SLAVE_READ_COUNT | SLAVE_READ_PERCENT |
+-----+
+-----+
+-----+
| 0 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0000_RDS | 69 | 100% | 0 | 0% |
| 1 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0001_RDS | 45 | 100% | 0 | 0% |
| 2 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0002_RDS | 30 | 100% | 0 | 0% |
| 3 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0003_RDS | 29 | 100% | 0 | 0% |
| 4 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0004_RDS | 11 | 100% | 0 | 0% |
| 5 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0005_RDS | 1 | 100% | 0 | 0% |
| 6 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0006_RDS | 8 | 100% | 0 | 0% |
| 7 | TESTDB_1478746391548CDTCTESTDB_OXGJ_0007_RDS | 50 | 100% | 0 | 0% |
+-----+
+-----+
+-----+
8 rows in set (0.10 sec)
```

2. Execution plans that cannot be expressed by SQL statements can be expressed by DRDS in custom format.

Example:

```
mysql> EXPLAIN DETAIL SELECT COUNT(*) FROM test;
+-----+
+-----+
| GROUP_NAME | PARAMS | SQL |
+-----+
+-----+
| TEST_DB_1478746391548CDTCTEST_DB_OXGJ_0000_RDS | Merge as test
queryConcurrency:GROUP_CONCURRENT
columns:[count(*)]
executeOn: TEST_DB_1478746391548CDTCTEST_DB_OXGJ_0000_RDS
Query from test as test
queryConcurrency:SEQUENTIAL
columns:[count(*)]
tableName:test
```

```

executeOn: TEST_DB_1478746391548CDTCTEST_DB_OXGJ_00
00_RDS
  Query from test as test
  queryConcurrency:SEQUENTIAL
  columns:[count(*)]
  tableName:test
  executeOn: TEST_DB_1478746391548CDTCTEST_DB_OXGJ_00
01_RDS
  ... ..
  Query from test as test
  queryConcurrency:SEQUENTIAL
  columns:[count(*)]
  tableName:test
  executeOn: TEST_DB_1478746391548CDTCTEST_DB_OXGJ_00
07_RDS
| NULL |
+-----+
+-----+-----+
1 row in set (0.00 sec)

```

executeOn in the SQL statement field indicates the database shard on which the SQL statement is run. DRDS finally merges the results returned by the database shards.

Execution plans at the ApsaraDB for RDS layer

The execution plans at the ApsaraDB for RDS layer are the same as the native MySQL execution plan. For more information, see [official MySQL documentation](#).

One DRDS logical table may consist of multiple shards distributed in different database shards. Therefore, you can view the execution plans at the ApsaraDB for RDS layer in multiple ways.

1. View the execution plan of an ApsaraDB for RDS database shard.

If the query condition contains a shard key, directly run the EXPLAIN EXECUTE command to display the execution plan on the corresponding database shard.

Example:

```

mysql> EXPLAIN EXECUTE SELECT * FROM test WHERE c1 = 1;
+----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | test | const | PRIMARY | PRIMARY | 4 | | 1 | NULL |
+----+-----+-----+-----+-----+-----+-----+
1 row in set (0.04 sec)

```



If an SQL statement involves multiple shards (for example, its condition does not contain a shard key), the EXPLAIN EXECUTE command returns an execution plan on a random ApsaraDB for RDS database shard.

To view the execution plan of an SQL statement on a specified database shard, you can use the Hint method. Example:

```
mysql> /*! TDDL:node='TESTDB_1478746391548CDTCTESTDB_OXGJ_0000_RDS'*/
EXPLAIN SELECT * FROM test;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len |
| ref | rows | Extra |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | test | ALL | NULL | NULL | NULL |
| NULL | 2 | NULL |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.04 sec)
```

2. View the execution plans of all ApsaraDB for RDS database shards.

You can run SCAN Hint to display the execution plans of SQL statements on all database shards:

```
mysql> /*! TDDL:scan='test'*/EXPLAIN SELECT * FROM test;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len |
| ref | rows | Extra |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | test | ALL | NULL | NULL | NULL |
| NULL | 2 | NULL |
| 1 | SIMPLE | test | ALL | NULL | NULL | NULL |
| NULL | 3 | NULL |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.08 sec)
```



Notice:

- a. In Hint mode, DRDS only replaces the table names in case of database or table sharding, and then directly sends the logical SQL statement to ApsaraDB for RDS for execution. It will not process the result.
- b. Execution plans obtained by using an EXPLAIN command are generated by static analysis and are not actually executed in databases.

TRACE command

The TRACE command in DRDS can track the SQL execution process and the overheads in each stage. It can be used together with the execution plan to facilitate SQL statement optimization.

The TRACE command contains two related commands: TRACE and SHOW TRACE, which must be used together.

7.14.4 SQL optimization methods

7.14.4.1 Overview

This topic describes the SQL optimization principles and methods for optimizing different types of SQL statements in Distributed Relational Database Service (DRDS).

Basic principles of SQL optimization

In DRDS, SQL computing that can be performed by ApsaraDB for RDS instances is called push-down computing. Push-down computing reduces data transmission, decreases overheads at the network layer and DRDS layer, and improves the SQL statement execution efficiency.

Therefore, the basic principle for SQL statement optimization in DRDS is as follows: Push down as many computations as possible to ApsaraDB for RDS instances.

Push-down computations include:

- JOIN connections.
- Filter conditions, such as WHERE or HAVING conditions.
- Aggregate computing, such as COUNT and GROUP BY.
- Sorting, such as ORDER BY.
- Deduplication, such as DISTINCT.
- Function computing, such as the NOW() function.
- Subqueries.



Notice:

The preceding list only lists possible forms of push-down computations. It does not mean that all clauses or conditions or combinations of clauses or conditions can be pushed down for computing.

SQL statements of different types and containing different conditions can be optimized in different ways. The following uses some examples to describe how to optimize SQL statements:

- **Single-table SQL optimization**
 - **Filter condition optimization**
 - **Optimization of the number of returned rows for a query**
 - **Grouping and sorting optimization**
- **JOIN query optimization**
 - **Optimization of push-down JOIN queries**
 - **Optimization of distributed JOIN queries**
- **Subquery optimization**

7.14.4.2 Single-table SQL optimization

Single-table SQL optimization must follow the following principles:

- **Make sure that the SQL statements contain the shard key.**
- **Use an equivalence condition for the shard key whenever possible.**
- **If the shard key is an IN condition, the number of values after IN should be as small as possible (far fewer than the number of shards, and remain unchanged as the business grows).**
- **If SQL statements do not contain a shard key, use only one of DISTINCT, GROUP BY, and ORDER BY in the same SQL statement.**

Filter condition optimization

DRDS partitions data horizontally by the shard key. Therefore, the filter condition must contain the shard key as much as possible so that DRDS can push queries down to specific database shards based on the shard key value, to avoid scanning all tables in the DRDS instance.

For example, the shard key of the test table is c1. If the filter condition does not contain this shard key, full table scan is performed:

```
mysql> SELECT * FROM test WHERE c2 = 2;
+-----+-----+
| c1 | c2 |
+-----+-----+
| 2 | 2 |
+-----+-----+
```

1 row in set (0.05 sec)

The corresponding execution plan is as follows:

```
mysql> EXPLAIN SELECT * FROM test WHERE c2 = 2;
+-----+
+-----+
+-----+
| GROUP_NAME                                     | SQL
|                                               | PARAMS |
+-----+-----+
+-----+
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0004_RDS | select `test`.`c1
| `test`.`c2` from `test` where (`test`.`c2` = 2) | {}      |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0007_RDS | select `test`.`c1
| `test`.`c2` from `test` where (`test`.`c2` = 2) | {}      |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0005_RDS | select `test`.`c1
| `test`.`c2` from `test` where (`test`.`c2` = 2) | {}      |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0002_RDS | select `test`.`c1
| `test`.`c2` from `test` where (`test`.`c2` = 2) | {}      |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0003_RDS | select `test`.`c1
| `test`.`c2` from `test` where (`test`.`c2` = 2) | {}      |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0006_RDS | select `test`.`c1
| `test`.`c2` from `test` where (`test`.`c2` = 2) | {}      |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0000_RDS | select `test`.`c1
| `test`.`c2` from `test` where (`test`.`c2` = 2) | {}      |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0001_RDS | select `test`.`c1
| `test`.`c2` from `test` where (`test`.`c2` = 2) | {}      |
+-----+
+-----+
8 rows in set (0.00 sec)
```

The smaller the value range of the filter condition containing the shard key, the faster the DRDS query speed.

For example, in the query on the test table, the filter condition contains the value range of the shard key c1:

```
mysql> SELECT * FROM test WHERE c1 > 1 AND c1 < 4;
+-----+
| c1 | c2 |
+-----+
| 2 | 2 |
| 3 | 3 |
+-----+
2 rows in set (0.04 sec)
```

The corresponding execution plan is as follows:

```
mysql> EXPLAIN SELECT * FROM test WHERE c1 > 1 AND c1 < 4;
+-----+
+-----+
+-----+
| GROUP_NAME                                     | SQL
|                                               | PARAMS |
```

```

+-----+
+-----+
+-----+
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0002_RDS | select `test`.`c1`
,`test`.`c2` from `test` where ((`test`.`c1` > 1) AND (`test`.`c1` <
4)) | {} |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0003_RDS | select `test`.`c1`
,`test`.`c2` from `test` where ((`test`.`c1` > 1) AND (`test`.`c1` <
4)) | {} |
+-----+
+-----+
+-----+
2 rows in set (0.00 sec)

```

The equivalence condition is executed faster than the range condition. For example:

```

mysql> SELECT * FROM test WHERE c1 = 2;
+----+----+
| c1 | c2 |
+----+----+
|  2 |  2 |
+----+----+
1 row in set (0.03 sec)

```

The corresponding execution plan is as follows:

```

mysql> EXPLAIN SELECT * FROM test WHERE c1 = 2;
+-----+
+-----+
+-----+
| GROUP_NAME                                     | SQL
                                                | PARAMS |
+-----+
+-----+
+-----+
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0002_RDS | select `test`.`c1`
,`test`.`c2` from `test` where (`test`.`c1` = 2) | {} |
+-----+
+-----+
+-----+
1 row in set (0.00 sec)

```

In addition, if you want to insert data into a table shard, the inserted field must contain a shard key.

For example, data inserted into the test table contains the shard key c1:

```

mysql> INSERT INTO test(c1,c2) VALUES(8,8);
Query OK, 1 row affected (0.07 sec)

```

Optimization of the number of returned rows for a query

When DRDS runs a query containing LIMIT [*offset*,] *row_count*, DRDS actually reads records before *offset* in order and directly discards them. In this way, when the value

of *offset* is large, the query is slow even if the value of *row_count* is small. Take the following SQL statement as an example:

```
SELECT *
FROM sample_order
ORDER BY sample_order.id
LIMIT 10000, 2
```

Although only the 10000th and 10001st records are returned, it takes about 12 seconds to run the SQL statement because DRDS actually reads 10,002 records.

```
mysql> SELECT * FROM sample_order ORDER BY sample_order.id LIMIT 10000,2;
+-----+-----+-----+-----+
+-----+
| id          | sellerId   | trade_id   | buyer_id   | buyer_nick |
+-----+-----+-----+-----+
| 242012755468 | 1711939506 | 242012755467 | 244148116334 | zhangsan   |
| 242012759093 | 1711939506 | 242012759092 | 244148138304 | wangwu     |
+-----+-----+-----+-----+
2 rows in set (11.93 sec)
```

The corresponding execution plan is as follows:

```
mysql> EXPLAIN SELECT * FROM sample_order ORDER BY sample_order.id
LIMIT 10000,2;
+-----+-----+-----+
| GROUP_NAME          | SQL                                               | PARAMS |
+-----+-----+-----+
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0004_RDS | select `sample_order`.`id`,`sample_order`.`sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `sample_order` order by `sample_order`.`id` asc limit 0,10002 | {}      |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0007_RDS | select `sample_order`.`id`,`sample_order`.`sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `sample_order` order by `sample_order`.`id` asc limit 0,10002 | {}      |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0005_RDS | select `sample_order`.`id`,`sample_order`.`sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `sample_order` order by `sample_order`.`id` asc limit 0,10002 | {}      |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0002_RDS | select `sample_order`.`id`,`sample_order`.`sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `sample_order` order by `sample_order`.`id` asc limit 0,10002 | {}      |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0003_RDS | select `sample_order`.`id`,`sample_order`.`sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `sample_order` order by `sample_order`.`id` asc limit 0,10002 | {}      |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0006_RDS | select `sample_order`.`id`,`sample_order`.`sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `sample_order` order by `sample_order`.`id` asc limit 0,10002 | {}      |
```

```
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0000_RDS | select `sample_order`.`id`,`sample_order`.`sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `sample_order` order by `sample_order`.`id` asc limit 0,10002 | {} |
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0001_RDS | select `sample_order`.`id`,`sample_order`.`sellerId`,`sample_order`.`trade_id`,`sample_order`.`buyer_id`,`sample_order`.`buyer_nick` from `sample_order` order by `sample_order`.`id` asc limit 0,10002 | {} |
+-----+-----+-----+-----+
8 rows in set (0.01 sec)
```

To optimize the preceding SQL statement, find the ID set, and use IN to match the actual records. The modified SQL query is as follows:

```
SELECT *
FROM sample_order o
WHERE o.id IN (
    SELECT id
    FROM sample_order
    ORDER BY id
    LIMIT 10000, 2 )
```

The purpose is to cache IDs in the memory first (on the premise that the number of IDs is small). If the shard key of the sample_order table is an ID, DRDS can also push down such an IN query to different database shards through rule-based calculation, avoiding full table scan and unnecessary network I/O. Check the result of the rewritten SQL query:

```
mysql> SELECT *
-> FROM sample_order o
-> WHERE o.id IN ( SELECT id FROM sample_order ORDER BY id LIMIT
10000,2 );
+-----+-----+-----+-----+
+-----+
| id          | sellerId   | trade_id   | buyer_id   | buyer_nick
|
+-----+-----+-----+-----+
| 242012755468 | 1711939506 | 242012755467 | 244148116334 | zhangsan
|
| 242012759093 | 1711939506 | 242012759092 | 244148138304 | wangwu
|
+-----+-----+-----+-----+
+-----+
2 rows in set (1.08 sec)
```

The execution time is significantly reduced from 12 seconds to 1.08 seconds.

The corresponding execution plan is as follows:

```
mysql> EXPLAIN SELECT *
-> FROM sample_order o
-> WHERE o.id IN ( SELECT id FROM sample_order ORDER BY id LIMIT
10000,2 );
```

```

+-----+
+-----+
+-----+
| GROUP_NAME          | SQL
+-----+
|                     | PARAMS |
+-----+
+-----+
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0002_RDS | select `o`.`id`,`
o`.`sellerId`,`o`.`trade_id`,`o`.`buyer_id`,`o`.`buyer_nick` from `
sample_order`,`o` where (`o`.`id` IN (10002)) | {}
| SEQPERF_1478746391548CDTCSEQPERF_OXGJ_0001_RDS | select `o`.`id`,`
o`.`sellerId`,`o`.`trade_id`,`o`.`buyer_id`,`o`.`buyer_nick` from `
sample_order`,`o` where (`o`.`id` IN (10001)) | {}
+-----+
+-----+
2 rows in set (0.03 sec)

```

Grouping and sorting optimization

In DRDS, if an SQL query must use DISTINCT, GROUP BY, and ORDER BY at the same time, try to ensure that the fields after DISTINCT, GROUP BY, and ORDER BY are the same and the fields are shard keys. In this way, only a small amount of data is returned for the SQL query. This minimizes the network bandwidth consumed by distributed queries and removes the need to retrieve a large amount of data and sort the data in a temporary table, thereby maximizing the system performance.

7.14.4.3 JOIN query optimization

JOIN queries in DRDS are classified into push-down JOIN queries and non-push-down JOIN queries (distributed JOIN queries). The optimization policies for these two types of JOIN queries are different.

Optimize push-down JOIN queries

Push-down JOIN queries are classified into the following types:

- **JOIN queries between single tables (non-partition tables).**
- **The tables involved in the JOIN query contain the shard key in the filter condition and use the same sharding algorithm (that is, the data calculated by the sharding algorithm is distributed to the same shard).**
- **Tables involved in the JOIN query use the shard key as the JOIN condition and use the same sharding algorithm.**
- **JOIN query between broadcast tables (or small table broadcast) and table shards.**

In DRDS, optimize JOIN queries to push-down JOIN queries that can be executed on database shards.

Take a JOIN query between a broadcast table and table shards as an example. The broadcast table is used as the JOIN driving table (the left table in the JOIN query is called the driving table). The DRDS broadcast table stores the same data in each database shard. When the broadcast table is used as the JOIN driving table, the JOIN query between this broadcast table and table shards can be converted into single-database JOIN queries and combined for computing to improve query performance.

For example, a JOIN query is performed on the following three tables, among which the sample_area table is the broadcast table, and the sample_item and sample_buyer tables are table shards. The query execution time is about 15s:

```
mysql> SELECT sample_area.name
-> FROM sample_item i JOIN sample_buyer b ON i.sellerId = b.
sellerId JOIN sample_area a ON b.province = a.id
-> WHERE a.id < 110107
-> LIMIT 0, 10;
```

name
BJ

10 rows in set (14.88 sec)

If you adjust the JOIN query order and move the broadcast table to the farthest left as the JOIN driving table, the JOIN query is pushed down to a single database shard in the DRDS instance:

```
mysql> SELECT sample_area.name
-> FROM sample_area a JOIN sample_buyer b ON b.province = a.id
JOIN sample_item i ON i.sellerId = b.sellerId
-> WHERE a.id < 110107
-> LIMIT 0, 10;
```

name
BJ

```
| BJ |
+-----+
10 rows in set (0.04 sec)
```

The query execution time decreases from 15 seconds to 0.04 seconds, which is a significant improvement to the query performance.



Notice:

The broadcast table achieves data consistency through the synchronization mechanism on database shards, with a latency of several seconds.

Optimize distributed JOIN queries

If a JOIN query cannot be pushed down (that is, the JOIN condition and filter condition do not contain the shard key), DRDS must complete part of the computing in the query. Such a query is a distributed JOIN query.

Tables in a distributed JOIN query are classified into two types based on the data size:

- **Small table:** A table that contains a small amount of data (less than 100 data records or less data than other tables) that is involved in JOIN computing after filtering.
- **Large table:** A table that contains a large amount of data (more than 100 data records or more data than other tables) that is involved in JOIN computing after filtering.

In most cases, Nested Loop and its derived algorithms are used in JOIN computing at the DRDS layer. If sorting is required for JOIN queries, the Sort Merge algorithm is used. When the Nested Loop algorithm is used, a smaller data size in the left table in a JOIN query indicates a smaller number of queries performed by DRDS on the right table. If the right table has indexes or contains a small amount of data, the JOIN query is even faster. Therefore, in DRDS, the left table of a distributed JOIN query is called the driving table. To optimize a distributed JOIN query, use a small table as the driving table and set as many filter conditions as possible for the driving table.

Take the following distributed JOIN query as an example. The query takes about 24 seconds:

```
mysql> SELECT t.title, t.price
-> FROM sample_order o,
```

```

->      ( SELECT * FROM sample_item i WHERE i.id = 242002396687 )
t
-> WHERE t.source_id = o.source_item_id AND o.sellerId <
1733635660;
+-----+-----+
| title                                     | price |
+-----+-----+
| Sample Item for Distributed JOIN         | 239.00 |
| Sample Item for Distributed JOIN         | 239.00 |
| Sample Item for Distributed JOIN         | 239.00 |
| Sample Item for Distributed JOIN         | 239.00 |
| Sample Item for Distributed JOIN         | 239.00 |
| Sample Item for Distributed JOIN         | 239.00 |
| Sample Item for Distributed JOIN         | 239.00 |
| Sample Item for Distributed JOIN         | 239.00 |
| Sample Item for Distributed JOIN         | 239.00 |
| Sample Item for Distributed JOIN         | 239.00 |
+-----+-----+
10 rows in set (23.79 sec)

```

The preceding JOIN query is an INNER JOIN query, with the actual size of the intermediate data involved in JOIN computing unknown. In this case, perform COUNT() on the o table and t table respectively to obtain the actual data size.

For the o table, o.sellerId < 1733635660 in the WHERE condition is only related to the o table. Then, extract and add it to the COUNT() condition of the o table. The following query result is returned:

```

mysql> SELECT COUNT(*) FROM sample_order o WHERE o.sellerId <
1733635660;
+-----+
| count(*) |
+-----+
|      504018 |
+-----+
1 row in set (0.10 sec)

```

The intermediate result of the o table contains about 500,000 records. Similarly, the t table is a subquery, which can be extracted directly for the COUNT() query:

```

mysql> SELECT COUNT(*) FROM sample_item i WHERE i.id = 242002396687;
+-----+
| count(*) |
+-----+
|          1 |
+-----+
1 row in set (0.01 sec)

```

The intermediate result of the t table contains only one record. Therefore, the o table is a large table and the t table is a small table. Use the small table as the driving table of the distributed JOIN query. The result of the adjusted JOIN query is as follows:

```

mysql> SELECT t.title, t.price

```

```
-> FROM ( SELECT * FROM sample_item i WHERE i.id = 242002396687 )
t,
->     sample_order o
-> WHERE t.source_id = o.source_item_id AND o.sellerId <
1733635660;
```

title	price
Sample Item for Distributed JOIN	239.00
Sample Item for Distributed JOIN	239.00
Sample Item for Distributed JOIN	239.00
Sample Item for Distributed JOIN	239.00
Sample Item for Distributed JOIN	239.00
Sample Item for Distributed JOIN	239.00
Sample Item for Distributed JOIN	239.00
Sample Item for Distributed JOIN	239.00
Sample Item for Distributed JOIN	239.00
Sample Item for Distributed JOIN	239.00

```
10 rows in set (0.15 sec)
```

The query execution time decreases from about 24 seconds to 0.15 seconds, with the query performance significantly improved.

7.14.4.4 Subquery optimization

When optimizing an SQL query that contains subqueries, push the subqueries down to database shards as much as possible to reduce the computing workload at the DRDS layer.

For this purpose, you can try two optimization methods:

- Rewrite subqueries into multi-table JOIN queries, and optimize the JOIN queries.
- Use the shard key in the JOIN condition or filter condition so that DRDS can push the query down to a specific database shard to avoid full table scan.

The following subquery is used as an example:

```
SELECT o. *
FROM sample_order o
WHERE NOT EXISTS
      (SELECT sellerId FROM sample_seller s WHERE o.sellerId = s.id)
```

Rewrite the subquery into a JOIN query query:

```
SELECT o. *
FROM sample_order o LEFT JOIN sample_seller s ON o.sellerId = s.id
```

```
WHERE s.id IS NULL
```

7.14.5 Select connection pools for an application

A database connection pool is used to manage database connections in a centralized manner, so as to improve application performance and reduce database loads.

- **Reuse resources:** Connections can be reused to avoid high performance overheads caused by frequent connection creation and release. Resource reuse can also improve the system stability.
- **Improve the system response efficiency:** After the connection initialization is complete, all requests can directly use the existing connections, which avoids the overheads of connection initialization and release and improves the system response efficiency.
- **Avoid connection leakage:** The connection pool forcibly revokes connections based on the preset de-allocation policy to avoid connection resource leakage.

We recommend that you use a connection pool to connect applications and databases for service operations. For Java programs, we recommend that you use the [Druid connection pool](#).

The following is an example of standard Druid Spring configuration:

```
<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataS
ource" init-method="init" destroy-method="close">
  <property name="driverClassName" value="com.mysql.jdbc.Driver
" />
  <!-- Basic attributes URL, user, and password -->
  <property name="url" value="jdbc:mysql://ip:port/db?
autoReconnect=true&rewriteBatchedStatements=true&socketTimeout
=30000&connectTimeout=3000" />
  <property name="username" value="root" />
  <property name="password" value="123456" />
  <!-- Configure the initial size, minimum value, and maximum
value -->
  <property name="maxActive" value="20" />
  <property name="initialSize" value="3" />
  <property name="minIdle" value="3" />
  <!-- maxWait indicates the time-out period for obtaining the
connection -->
  <property name="maxWait" value="60000" />
  <!-- timeBetweenEvictionRunsMillis indicates the interval for
detecting idle connections to be closed, in milliseconds -->
  <property name="timeBetweenEvictionRunsMillis" value="60000" /
>
  <!-- minEvictableIdleTimeMillis indicates the minimum idle
time of a connection in the connection pool, in milliseconds-->
  <property name="minEvictableIdleTimeMillis" value="300000" />
  <!-- SQL statement used to check whether connections are
available -->
```

```

<property name="validationQuery" value="SELECT 'z'" />
<!-- Whether to enable idle connection check -->
<property name="testWhileIdle" value="true" />
<!-- Whether to check the connection status before obtaining
a connection -->
<property name="testOnBorrow" value="false" />
<!-- Whether to check the connection status before releasing
a connection -->
<property name="testOnReturn" value="false" />
</bean>

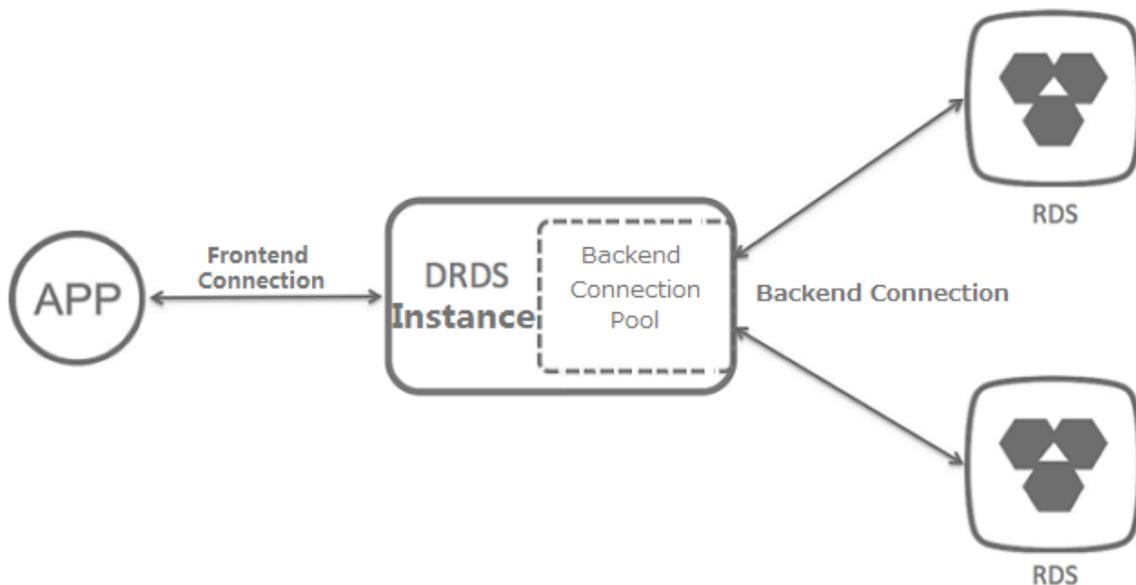
```

7.14.6 Connections in a DRDS instance

When an application connects to a DRDS instance for operation, there are two types of connections from the perspective of the DRDS instance:

- **Frontend connection:** a connection established by an application to the logical database in the DRDS instance.
- **Backend connection:** a connection established by a node in a DRDS instance to a physical database in a backend ApsaraDB for RDS instance.

Figure 7-15: DRDS instance connections



Frontend connection

Theoretically, the number of frontend connections is limited by the available memory size and the number of network connections to nodes of the DRDS instance. However, in actual application scenarios, when an application connects to a DRDS instance, the nodes of the DRDS instance usually manage a limited number of connections to perform requested operations, and do not maintain a large number

of concurrent persistent connections (for example, tens of thousands of concurrent persistent connections). Therefore, the number of frontend connections that a DRDS instance can accept can be considered as unlimited.

Considering that the number of frontend connections is unlimited and a large number of idle connections are allowed, this method applies to scenarios where a large number of servers are deployed and need to maintain their connections to the DRDS instance.



Note:

Although the number of frontend connections is considered as unlimited, operation requests obtained from frontend connections are actually executed by internal threads of the DRDS instance through backend connections. Due to the limited number of internal threads and backend connections, the total number of concurrent requests that can be processed by the DRDS instance is limited.

Backend connection

Each node of a DRDS instance creates a backend connection pool to automatically manage and maintain the backend connections to the physical databases in the ApsaraDB for RDS instance. Therefore, the maximum number of connections in the backend connection pool of a DRDS instance is directly related to the maximum number of connections supported by the ApsaraDB for RDS instance. You can use the following formula to calculate the maximum number of connections in the backend connection pool of a DRDS instance:

```
Maximum number of connections in a backend connection pool of a DRDS instance = FLOOR(Maximum number of connections in an ApsaraDB for RDS instance/Number of physical database shards in the ApsaraDB for RDS instance/Number of nodes in the DRDS instance)
```

For example, a user has purchased an ApsaraDB for RDS instance and a DRDS instance of the following types:

- The ApsaraDB for RDS instance has eight physical database shards, four cores, and a 16 GB memory, supporting a maximum number of 4,000 connections.
- The DRDS dedicated instance has 32 cores and a 32 GB memory, with each DRDS node having two cores and one 2 GB memory (that is, the instance has 16 DRDS nodes).

According to the preceding formula, the maximum number of connections in the backend connection pool of the DRDS instance is as follows:

Maximum number of connections in the backend connection pool of the DRDS instance = $\text{FLOOR}(4000/8/16) = \text{FLOOR}(31.25) = 31$



Note:

- **The calculation result of the preceding formula is the maximum number of connections in the backend connection pool of the DRDS instance. In actual use, to reduce the connection pressure on the ApsaraDB for RDS instance, the DRDS instance adjusts the maximum number of connections in the backend connection pool to make it smaller than the upper limit.**
- **We recommend that you create databases in a DRDS instance in a dedicated ApsaraDB for RDS instance. Do not create databases for other applications or DRDS instances in the dedicated ApsaraDB for RDS instance.**

Adjust the number of backend connections

If you want to adjust the maximum number of connections in the backend connection pool of a DRDS instance after increasing the maximum number connections in an ApsaraDB for RDS instance, follow these steps:

- 1. Log on to the DRDS console and select a database in the database list.**
- 2. On the Basic Information page of the database, click Adjust Database Connection Pool Information. The DRDS console automatically adjusts the number.**

Relationship between frontend and backend connections

After an application establishes frontend connections to a DRDS instance and sends SQL statement execution requests, the DRDS nodes process the requests asynchronously and obtain backend connections through the internal backend connection pool, and then run optimized SQL statements on one or more physical databases.

DRDS nodes process requests asynchronously and frontend connections are not bound to backend connections. Therefore, a small number of backend connections can process a large number of requests for short transactions and simple queries from many concurrent frontend connections. This is why you need to focus on the QPS in DRDS, rather than the number of concurrent connections.

Although the number of frontend connections is considered to be unlimited, the maximum number of connections maintained in the backend connection pool of a DRDS instance is limited. For more information, see "Backend connections." Therefore, note the following points in actual application scenarios:

- **Avoid long or large transactions in applications.** These transactions occupy many or even all backend connections when they are not committed or rolled back for a long time, which reduces the overall concurrent processing capability and increases the response time.
- **Monitor and optimize or remove slow SQL queries run in the DRDS instance, to prevent them from occupying too many or even all backend connections.** Otherwise, the DRDS instance or the ApsaraDB for RDS instance is under greater processing pressure, which may lead to reduced overall concurrent processing capability, increased response time, or higher SQL execution failure rate due to execution timeout. For troubleshooting and optimization of slow SQL queries, see [Troubleshoot slow SQL statements in DRDS](#) and [Overview](#).
- **If the maximum number of connections in the backend connection pool of the DRDS instance is occupied in normal SQL queries, contact customer services.**

7.14.7 Perform instance upgrade

Database performance can be measured by the response time (RT) and queries per second (QPS). RT reflects the performance of a single SQL statement, which can be solved through SQL optimization. Distributed Relational Database Service (DRDS) upgrade expands the capacity to improve performance, and is suitable for database access services with low latency and high QPS.

The performance of a DRDS instance depends on the performance of DRDS and ApsaraDB for RDS. Insufficient performance of any DRDS or ApsaraDB for RDS node can create a bottleneck in the overall performance. This topic describes how to check the performance metrics of a DRDS instance and upgrade the DRDS instance to solve the performance bottleneck. For more information about how to determine the performance of an ApsaraDB for RDS instance and upgrade the ApsaraDB for RDS instance, see the ApsaraDB for RDS documentation.

Determine the performance bottleneck of a DRDS instance

The QPS and CPU utilization of a DRDS instance are positively correlated. When the DRDS instance has performance bottlenecks, the CPU utilization of the DRDS instance remains high.

View the CPU utilization

- 1. On the Basic Information page of the DRDS instance, click Monitoring Information in the left-side navigation pane.**
- 2. On the Monitoring Details page, select a monitoring dimension and the corresponding metrics to view details.**

If the CPU utilization exceeds 90% or remains above 80%, the DRDS instance has a performance bottleneck. If the ApsaraDB for RDS instance has no performance bottleneck, the current DRDS instance specifications cannot meet the QPS performance requirements of the business. In this case, upgrade the DRDS instance.

For more performance-related service monitoring scenarios and methods for configuring the DRDS CPU utilization alert, see [View the monitoring information](#).

Perform instance upgrade

QPS is an important metric for determining whether the DRDS instance specifications can meet the business requirements. Each type of instance specifications corresponds to a reference QPS value.

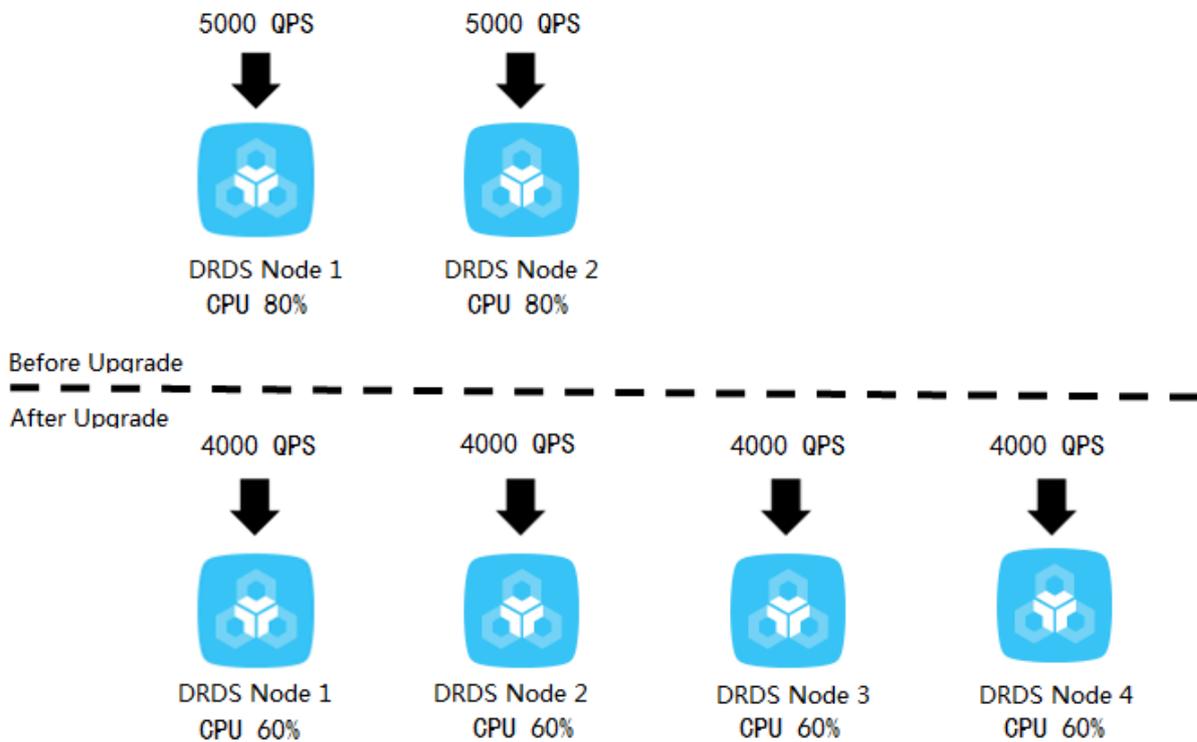
Some special SQL statements require more computing (such as temporary table sorting and aggregate computing) in DRDS. In this case, the QPS supported by each DRDS instance is lower than the standard value in its type.

DRDS upgrade improves the processing performance of a DRDS instance by adding nodes to share the QPS. As DRDS nodes are stateless, this upgrade method linearly improves the performance of DRDS instances.

For example, service A requires a QPS of about 15 thousand. The current DRDS instance has a 4-core vCPU and 4 GB memory and two nodes, supporting a QPS of only 10 thousand. After finding that the CPU utilization of the DRDS instance remains high, we upgraded the instance to 8-core vCPU and 8 GB memory, with each node handling about 4,000 QPS. Then, the performance meets the needs of

the user, and the CPU utilization also drops to a reasonable level, as shown in the following figure.

Figure 7-16: DRDS upgrade



For more information about how to upgrade a DRDS instance, see [Change configurations](#).

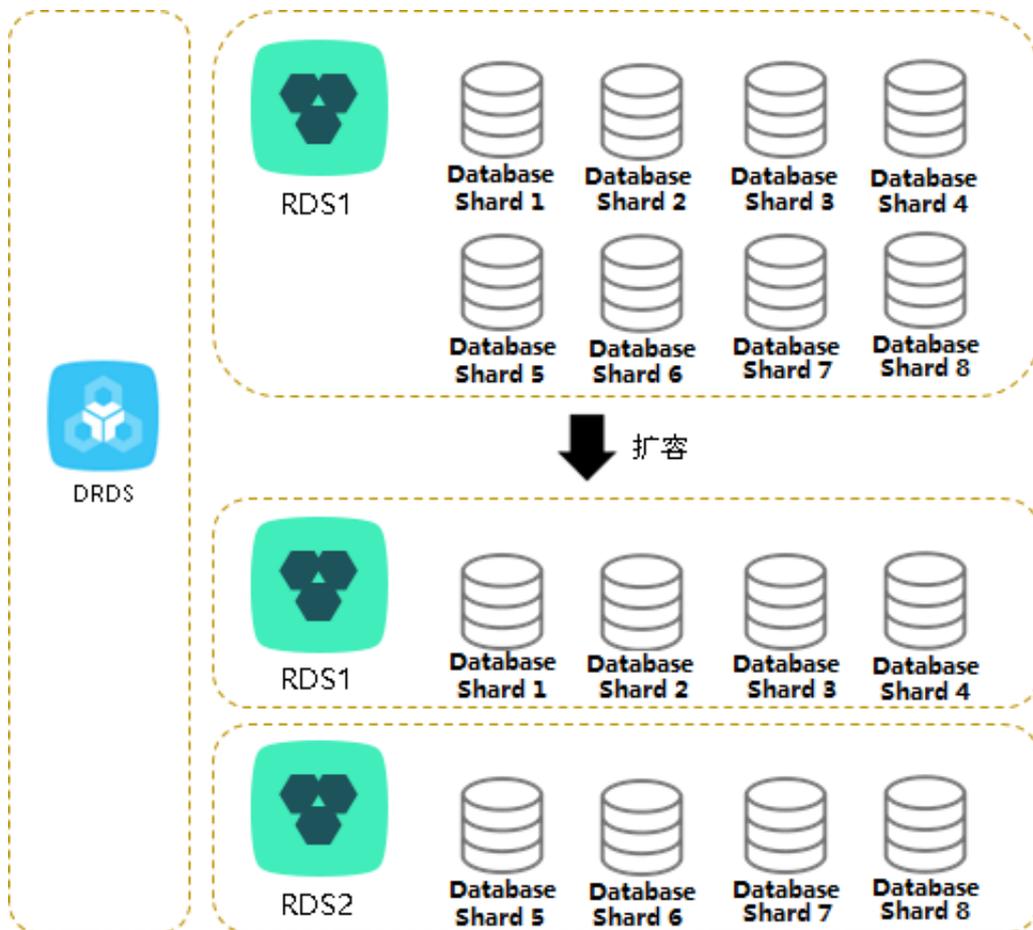
7.14.8 Perform scale-out

In Distributed Relational Database Service (DRDS), smooth scale-out improves the overall performance by increasing the number of ApsaraDB for RDS instances. You can increase the number of ApsaraDB for RDS instances to increase the DRDS database capacity when the following conditions are met: 1) The input/output operations per second (IOPS), CPU utilization, disk space, and other metrics of the ApsaraDB for RDS instance reach their bottlenecks. 2) The bottlenecks cannot be removed through SQL optimization or ApsaraDB for RDS upgrade (for example, the disk has been upgraded to the top configuration).

DRDS smooth scale-out reduces the pressure on the original ApsaraDB for RDS instance by migrating database shards to the new ApsaraDB for RDS instance. For example, before scale-out, all the eight databases are deployed in one ApsaraDB for RDS instance. After scale-out, the eight databases are deployed in two ApsaraDB

for RDS instances, and the pressure on a single ApsaraDB for RDS instance is significantly reduced, as shown in the following figure.

Figure 7-17: DRDS scale-out



After multiple scale-out operations, if the number of ApsaraDB for RDS instances is equal to the number of database shards, you need to create another DRDS instance and RDS databases with the expected capacity, and then migrate data to further increase the data capacity. This process is complex. We recommend that you consider the data growth expected in the next two to three years and plan the number of ApsaraDB for RDS instances properly when creating a DRDS database.

Determine whether scale-out is required

You can determine whether DRDS smooth scale-out is required based on three ApsaraDB for RDS metrics: IOPS, CPU utilization, and disk space. You can view these metrics in the ApsaraDB for RDS console. For more information, see the ApsaraDB for RDS documentation.

IOPS and CPU utilization

If you find that the IOPS or CPU utilization remains above 80% for a long time or you frequently receive alerts, follow these steps:

1. Optimize the SQL statement. Generally, you can solve the high CPU utilization by this method.
2. If the problem persists, upgrade the ApsaraDB for RDS instance. For more information, see the ApsaraDB for RDS documentation.
3. When the CPU utilization or IOPS exceeds the threshold, you can set read-only databases to share the load on the primary database. However, read/write splitting affects read consistency. For more information, see the [Read/write splitting](#) documentation.
4. If the problem persists, scale out the DRDS instance.

Disk space

ApsaraDB for RDS has the following types of disk space:

1. **Data space:** the space occupied by data. The space usage continues increasing as more data is inserted. We recommend that you keep the remaining disk space above 30%.
2. **System file space:** the space occupied by shared tables and error log files.
3. **Binary log file space:** the space occupied by binary logs generated during database operation. The more update transactions there are, the larger the occupied space is.

Whether scale-out is required depends on the data space. When the data space is about to or expected to exceed the disk capacity, you can distribute the data to databases on multiple ApsaraDB for RDS instances through scale-out.

Scale-out risks and precautions

DRDS scale-out consists of four steps: configuration > migration > switchover > cleanup. For more information, see the [Perform smooth scale-out](#) documentation.

Note the following points before scale-out:

- To reduce the pressure of read operations on the source ApsaraDB for RDS instance, perform scale-out when the load on the source ApsaraDB for RDS instance is low.

- During scale-out, do not submit DDL tasks in the console or connect to the DRDS instance to directly run DDL SQL statements. Otherwise, the scale-out task may fail.
- Scale-out requires that the source database table have a primary key. If the source database does not have a primary key, add one first.
- During scale-out, the read and write traffic is switched to the new ApsaraDB for RDS instance. The switchover process takes three to five minutes. We recommend that you perform switchover during off-peak hours.
- Scale-out does not affect the DRDS instance before the switchover. Therefore, you can cancel the scale-out through rollback before the switchover.
- Scale-out creates pressure on databases. We recommend that you perform this operation during off-peak hours.

7.14.9 Troubleshoot slow SQL statements in DRDS

7.14.9.1 Details about a low SQL statement

Distributed Relational Database Service (DRDS) defines an SQL statement that takes more than 1 second to run as a slow SQL statement. Slow SQL statements in DRDS are classified into logical slow SQL statements and physical slow SQL statements. In DRDS, an SQL statement is run step by step on DRDS and ApsaraDB RDS for MySQL nodes. Large execution loss on any node will result in slow SQL statements.

- Slow logical SQL statements are slow SQL statements sent by an application to DRDS.
- Slow physical SQL statements are slow SQL statements sent by DRDS to ApsaraDB RDS for MySQL.

Syntax

```
SHOW FULL {SLOW | PHYSICAL_SLOW} [WHERE where_condition]
        [ORDER BY col_name [ASC | DESC], ...]
row_count OFFSET offset}]
```

Description

`SHOW FULL SLOW` shows slow logical SQL statements, that is, SQL statements that an application sends to DRDS.

The result set of `SHOW FULL SLOW` contains the following columns:

- **TRACE_ID:** the unique identifier of the SQL statement. A logical SQL statement and the physical SQL statements generated by the logical SQL statement have the same TRACE_ID. The TRACE_ID is also sent as a comment to ApsaraDB RDS for MySQL. You can find this SQL statement based on TRACE_ID in SQL statement details of ApsaraDB RDS for MySQL.
- **HOST:** the IP address of the client that sends the SQL statement.

**Notice:**

The client IP address may not be obtained when the network type is Virtual Private Cloud (VPC).

- **START_TIME:** the time when DRDS starts running the SQL statement.
- **EXECUTE_TIME:** the time consumed by DRDS to run the SQL statement.
- **AFFECT_ROW:** the number of records returned or the number of rows affected by the SQL statement.
- **SQL:** the statement that is run.

`SHOW FULL PHYSICAL_SLOW` shows the physical slow SQL statements, that is, SQL statements that DRDS sends to ApsaraDB RDS for MySQL.

The result set of `SHOW FULL PHYSICAL_SLOW` contains the following columns:

- **TRACE_ID:** the unique identifier of the SQL statement. A logical SQL statement and the physical SQL statements generated by the logical SQL statement have the same TRACE_ID. The TRACE_ID is also sent as a comment to ApsaraDB RDS for MySQL. You can find this SQL statement based on TRACE_ID in SQL statement details of ApsaraDB RDS for MySQL.
- **GROUP_NAME:** the name of a database group. Grouping aims to manage multiple groups of databases with identical data, such as the primary and secondary databases after data replication through ApsaraDB RDS for MySQL, which are mainly used for read/write splitting and primary/secondary switchover.
- **DBKEY_NAME:** the name of the database shard on which the SQL statement is run
-
- **START_TIME:** the time when DRDS starts running the SQL statement.
- **EXECUTE_TIME:** the time consumed by DRDS to run the SQL statement.
- **SQL_EXECUTE_TIME:** the time consumed by DRDS to call ApsaraDB RDS for MySQL to run this SQL statement.

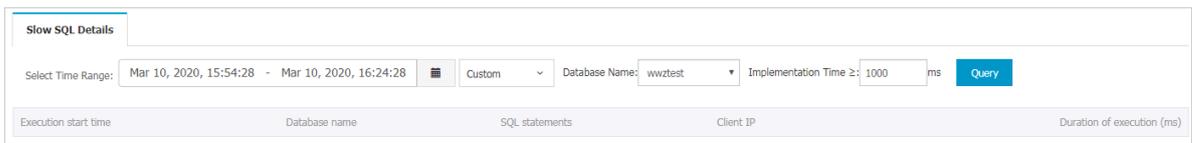

```

+-----+
+-----+
+-----+
+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| ae0e565b8c00000 | PRIV_TEST_1489167306631PJAFPRIV_TEST_APK
K_0000_RDS | rdso6g5b6206sdq832ow_priv_test_apk_0000_nfup | 2017-03
-29 19:27:53.02 |          50001 |          50001 |
          0 |          0 |          1 | select sleep(50) |
+-----+
+-----+
+-----+
+-----+-----+-----+
+-----+-----+-----+
+-----+
1 row in set (0.01 sec)

```

3. In the SQL statement details and slow SQL statement records of the ApsaraDB RDS for MySQL instance, you can query the execution information of this SQL statement on the ApsaraDB RDS for MySQL instance based on TRACE_ID.

Figure 7-18: Slow query logs



Example 2

This example describes how to locate the original SQL statement in DRDS based on the slow SQL statement located in ApsaraDB RDS for MySQL.

1. Based on the slow SQL query log in ApsaraDB RDS for MySQL, TRACE_ID of the slow SQL statement is ae0e55660c00000.
2. Based on the TRACE_ID obtained in Step 1, run SHOW FULL PHYSICAL_SLOW to obtain the physical execution information of this SQL statement.

```

mysql> show full physical_slow where trace_id = 'ae0e55660c00000';
+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| TRACE_ID          | GROUP_NAME
| DBKEY_NAME
| EXECUTE_TIME     | SQL_EXECUTE_TIME | GETLOCK_CONNECTION_TIME
| CREATE_CONNECTION_TIME | AFFECT_ROW | SQL
+-----+
+-----+
+-----+
+-----+-----+-----+

```

```

+-----+-----+-----+-----+
+-----+
| ae0e55660c00000 | PRIV_TEST_1489167306631PJAFPRIV_TEST_APK
K_0000_RDS | rdso6g5b6206sdq832ow_priv_test_apkk_0000_nfup | 2017-03
-29 19:27:37.308 | 10003 | 10001 |
0 | 0 | 1 | select sleep(10) |
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
1 row in set (0.02 sec)

```

7.14.9.2 Locate slow SQL statements

Generally, you can locate a slow SQL statement through two scenarios: you can either query historical information from slow SQL statement records, or can run SHOW PROCESSLIST to display the real-time execution information of slow SQL statements.

You can troubleshoot slow SQL statements as follows:

1. Locate slow SQL statements.
2. Locate nodes with performance loss.
3. Troubleshoot the performance loss.



Note:

During troubleshooting, we recommend that you use the MySQL command line `mysql -hIP -PPORT -uUSER -pPASSWORD -c` to create the connection. Be sure to add `-c` to prevent the MySQL client from filtering out the comments (default operation) and thus affecting the execution of HINT.

- View slow SQL statement records

Run the following command to query top 10 slow SQL statements. This command can query logical SQL statements in Distributed Relational Database Service (DRDS). One logic SQL statement corresponds to SQL statements of one or more databases or tables of the MySQL or ApsaraDB RDS for MySQL instance. For more information, see [Details about a low SQL statement](#).

```

mysql> SHOW SLOW limit 10;
+-----+-----+-----+-----+
+-----+
| TRACE_ID          | HOST          | START_TIME        |
EXECUTE_TIME | AFFECT_ROW | SQL
|

```

```

+-----+-----+-----+
+-----+-----+
+-----+
| ac3133132801001 | 42.120.74.97 | 2017-03-06 15:48:32.330 |
  900392 |          -1 | select detail_url, sum(price) from t_item
group by detail_url; |
.....
+-----+-----+-----+
+-----+-----+
+-----+
10 rows in set (0.01 sec)

```

- **View real-time SQL execution information**

If the execution of an in-progress SQL statement is slow in the current server, run `SHOW PROCESSLIST` to view the real-time SQL execution information in the current DRDS database. The value in the `TIME` column indicates how long the current SQL statement has been run.

```

mysql> SHOW PROCESSLIST WHERE COMMAND != 'Sleep';
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
+-----+
+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID          | USER          | DB          | COMMAND          |
TIME        | STATE        | INFO       | ROWS_SENT       |
ROWS_EXAMINED | ROWS_READ   |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
| 0-0-352724126 | ifisibhk0    | test_123_wvvp_0000 | Query          |
  13 | Sending data
      | /*DRDS /42.120.74.88/ac47e5a72801000/ */select `t_item
`.`detail_url`,SUM(`t_item`.`price`) from `t_i | NULL |
NULL | NULL |
| 0-0-352864311 | cowxhthg0    | NULL        | Binlog Dump    |
  17 | Master has sent all binlog to slave; waiting for binlog to
be updated | NULL
      | NULL |
NULL | NULL |
| 0-0-402714795 | ifisibhk0    | test_123_wvvp_0005 | Alter          |
  114 | Sending data
      | /*DRDS /42.120.74.88/ac47e5a72801000/ */ALTER TABLE `
Persons` ADD `Birthday` date | NULL |
NULL | NULL |
.....
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
+-----+
+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+

```

```
12 rows in set (0.03 sec)
```

The following describes each column:

- **ID:** The ID of the connection.
- **USER:** The username of the database shard in which this SQL statement is run.
- **DB:** The specified database. If no database is specified, the value is NULL.
- **COMMAND:** The type of the in-progress command. SLEEP indicates an idle connection. For more information about other commands, see [MySQL thread information documentation](#).
- **TIME:** The execution time of the SQL statement, in seconds.
- **STATE:** The current execution status. For more information, see [MySQL thread status documentation](#).
- **INFO:** The in-progress SQL statement. The SQL statement may be too long to be displayed completely. You can derive the complete SQL statement based on service parameters and other information.

In the current example, locate the following slow SQL statement:

```
ALTER TABLE `Persons` ADD `Birthday` date
```

7.14.9.3 Locate nodes with performance loss

When you locate a slow SQL statement in slow SQL statement records or real-time SQL execution information, you can run the TRACE command to trace the running time of the SQL statement in Distributed Relational Database Service (DRDS) and ApsaraDB RDS for MySQL to locate the bottleneck.

The TRACE command actually runs the SQL statement, records the time consumed by all nodes, and returns the execution result. For more information about TRACE and other control commands, see [Help statement](#).



Note:

The DRDS TRACE command needs to maintain the context information of the connection. Some GUI clients may use connection pools, which results in command exceptions. Therefore, we recommend that you use the MySQL command line to run the TRACE command.

Run the following command for the located slow SQL statement:

```
mysql> trace select detail_url, sum(distinct price) from t_item group by detail_url;
```

```

+-----+-----+
| detail_url | sum(price) |
+-----+-----+
| www.xxx.com | 1084326800.00 |
| www.xx1.com | 1084326800.00 |
| www.xx2.com | 1084326800.00 |
| www.xx3.com | 1084326800.00 |
| www.xx4.com | 1084326800.00 |
| www.xx5.com | 1084326800.00 |
| ..... |
+-----+-----+
1 row in set (7 min 2.72 sec)

```

After the TRACE command is run, run SHOW TRACE to view the result. You can identify the bottleneck of the slow SQL statement based on the time consumption of each component.

```

mysql> SHOW TRACE;
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| ID | TIMESTAMP | TYPE | GROUP_NAME
| DBKEY_NAME
| TIME_COST(MS) | CONNECTION_TIME_COST(MS) | ROWS | STATEMENT
| PARAMS |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| 0 | 0.000 | Optimize | DRDS
| DRDS
| 2 | 0.00 | 0 | select
detail_url, sum(price) from t_item group by detail_url
| NULL
|
| 1 | 423507.342 | Merge Sorted | DRDS
| DRDS
| 411307 | 0.00 | 8 | Using Merge
Sorted, Order By (`t_item`.`detail_url` asc )
| NULL |
| 2 | 2.378 | Query | TEST_123_14887660607
43ACTJSANGUAN_TEST_123_WVVP_0003_RDS | rdso6g5b6206sdq832ow_test_123_
wvvp_0003_hbpz | 15 | 1.59 | 1 |
select `t_item`.`detail_url`,SUM(distinct `t_item`.`price`) from
t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url`
asc | NULL |
| 3 | 2.731 | Query | TEST_123_14887660607
43ACTJSANGUAN_TEST_123_WVVP_0000_RDS | rdso6g5b6206sdq832ow_test_123_
wvvp_0000_hbpz | 11 | 1.78 | 1 |
select `t_item`.`detail_url`,SUM(distinct `t_item`.`price`) from
t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url`
asc | NULL |
| 4 | 2.933 | Query | TEST_123_14887660607
43ACTJSANGUAN_TEST_123_WVVP_0004_RDS | rdso6g5b6206sdq832ow_test_123_
wvvp_0004_hbpz | 15 | 1.48 | 1 |

```

```

select `t_item`.`detail_url`,SUM(distinct `t_item`.`price`) from `
t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url`
asc | NULL |
| 5 | 3.111 | Query | TEST_123_14887660607
43ACTJSANGUAN_TEST_123_WVVP_0001_RDS | rdso6g5b6206sdq832ow_test_123_
wvvp_0001_hbpz | 15 | 1.56 | 1 |
select `t_item`.`detail_url`,SUM(distinct `t_item`.`price`) from `
t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url`
asc | NULL |
| 6 | 3.323 | Query | TEST_123_14887660607
43ACTJSANGUAN_TEST_123_WVVP_0007_RDS | rdso6g5b6206sdq832ow_test_123_
wvvp_0007_hbpz | 15 | 1.54 | 1 |
select `t_item`.`detail_url`,SUM(distinct `t_item`.`price`) from `
t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url`
asc | NULL |
| 7 | 3.496 | Query | TEST_123_14887660607
43ACTJSANGUAN_TEST_123_WVVP_0006_RDS | rdso6g5b6206sdq832ow_test_123_
wvvp_0006_hbpz | 18 | 1.30 | 1 |
select `t_item`.`detail_url`,SUM(distinct `t_item`.`price`) from `
t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url`
asc | NULL |
| 8 | 3.505 | Query | TEST_123_14887660607
43ACTJSANGUAN_TEST_123_WVVP_0005_RDS | rdso6g5b6206sdq832ow_test_123_
wvvp_0005_hbpz | 423507 | 1.97 | 1 |
select `t_item`.`detail_url`,SUM(distinct `t_item`.`price`) from `
t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url`
asc | NULL |
| 9 | 3.686 | Query | TEST_123_14887660607
43ACTJSANGUAN_TEST_123_WVVP_0002_RDS | rdso6g5b6206sdq832ow_test_123_
wvvp_0002_hbpz | 14 | 1.47 | 1 |
select `t_item`.`detail_url`,SUM(distinct `t_item`.`price`) from `
t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url`
asc | NULL |
| 10 | 423807.906 | Aggregate | DRDS
| DRDS
| 1413 | 0.00 | 1 | Aggregate
Function (SUM(`t_item`.`price`)), Group By (`t_item`.`detail_url` asc
) | NULL
|
+-----+-----+-----+
+-----+
+-----+
+-----+-----+-----+
+-----+
+-----+
11 rows in set (0.01 sec)

```

In the returned results of SHOW TRACE, you can determine which node has a long execution time based on the values (in milliseconds) in the TIME_COST column. You can also see the corresponding GROUP_NAME (that is, the DRDS or ApsaraDB RDS for MySQL node) and the STATEMENT column information (that is, the SQL statement being executed). Based on GROUP_NAME, you can determine whether the slow node exists in DRDS or ApsaraDB RDS for MySQL.

According to the preceding results, the Merge Sorted action on the DRDS node and the TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005_RDS node of ApsaraDB RDS for MySQL consume a lot of time.

7.14.9.4 Troubleshoot performance loss

Slow nodes may exist in the Distributed Relational Database Service (DRDS) instance or its storage instance, namely, the ApsaraDB RDS for MySQL instance. Troubleshoot the fault accordingly after the cause is determined.

Solution for slow DRDS nodes

When the GROUP_NAME of a slow node is in the DRDS instance, check whether time-consuming computing operations such as Merge Sorted, Temp Table Merge, and Aggregate exist during execution. If yes, see [Overview](#) for optimization.

Solution for slow ApsaraDB RDS for MySQL nodes

When the slow node is in the ApsaraDB RDS for MySQL instance, check the execution plan of this SQL statement in the ApsaraDB RDS for MySQL instance.

In DRDS, you can run `/*! TDDL:node={GROUP_NAME}*/ EXPLAIN` to check the SQL execution plan of an ApsaraDB RDS for MySQL instance. The execution plan displays the SQL execution process information, including inter-table association and index information.

The detailed process is as follows:

1. Based on GROUP_NAME, assemble the HINT: `/*! TDDL:node='TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005_RDS'*/`
2. Combine the assembled HINT and STATEMENT prefixed by EXPLAIN to form a new SQL statement and run it. The EXPLAIN command does not actually run. It only displays the execution plan of the SQL statement.

The following example describes how to query the execution plan of the located slow node.

```
mysql> /*! TDDL:node='TEST_123_1488766060743ACTJSANGUAN_TEST_123_WVVP_0005_RDS'*/ EXPLAIN select `t_item`.`detail_url`,SUM(distinct `t_item`.`price`) from `t_item` group by `t_item`.`detail_url` order by `t_item`.`detail_url` asc;
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len
| ref | rows      | Extra
+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE      | t_item | ALL | NULL          | NULL | NULL
| NULL | 1322263 | Using temporary; Using filesort |
+-----+-----+-----+-----+-----+-----+-----+
```

```
1 row in set (0.01 sec)
```

When the preceding SQL statement is run in ApsaraDB RDS for MySQL, the message `Using temporary; Using filesort` is returned. It indicates that low SQL execution is caused by improper use of the index. In this case, you can correct the index and run the SQL statement again.

7.14.10 Handle DDL exceptions

When you run any DDL commands of Distributed Relational Database Service (DRDS), DRDS performs the corresponding DDL operation on all table shards.

Failure can be divided into two types:

1. An error occurs when the DDL statement is run in a database shard. DDL execution failure in any database shard may result in inconsistent table shard structures.
2. The system does not respond for a long time after the DDL statement is run . When you perform a DDL operation on a large table, the system may make no response for a long time due to the long execution of a DDL statement in a database shard.

Execution failures in database shards may occur for various reasons. For example , the table you want to create already exists, the column you want to add already exists, or the disk space is insufficient.

No response for a long time is generally caused by the long execution of a DDL statement in a database shard. Taking ApsaraDB RDS for MySQL as an example, the DDL execution time depends mostly on whether the operation is an in-place (directly modifying the source table) or copy (copying data in the table) operation . An in-place operation only requires modifying metadata, while a copy operation reconstructs the whole table and also involves log and buffer operations.

To determine whether a DDL operation is an in-place or copy operation, you can view the return value of "rows affected" after the operation is completed.

Example:

- **Modify the default value of a column (this operation is very fast and does not affect the table data at all):**

```
Query OK, 0 rows affected (0.07 sec)
```

- **Add an index (this operation takes some time, but "0 rows affected" indicates that the table data is not replicated):**

```
Query OK, 0 rows affected (21.42 sec)
```

- **Modify the data type of a column (this operation takes a long time and reconstructs all data rows in the table):**

```
Query OK, 1671168 rows affected (1 min 35.54 sec)
```

Therefore, before executing a DDL operation on a large table, perform the following steps to determine whether the operation is a fast or slow operation:

1. Copy the table structure to generate a cloned table.
2. Insert some data.
3. Perform the DDL operation on the cloned table.
4. Check whether the value of "rows affected" is 0 after the operation is completed. A non-zero value means that this operation reconstructs the entire table. In this case, you need to perform this operation in off-peak hours.

Solution for failures

DRDS DDL operations distribute all SQL statements to all database shards for parallel execution. Execution failure on any database shard does not affect the execution on other database shards. In addition, DRDS provides the CHECK TABLE command to check the structure consistency of the table shards. Therefore, failed DDL operations can be performed again, and errors reported on database shards on which the operations have been successfully executed do not affect the execution on other database shards. Make sure that all table shards ultimately have the same structure.

Procedure for DDL failure handling

1. Run the CHECK TABLE command to check the table structure. If the returned result contains only one row and the status is normal, the table statuses are consistent. In this case, go to Step 2. Otherwise, go to Step 3.

2. Run the `SHOW CREATE TABLE` command to check the table structure. If the displayed table structure is the same as the expected structure after the DDL statement is run, the DDL statement is run successfully. Otherwise, go to Step 3.
3. Run the `SHOW PROCESSLIST` command to check the statuses of all SQL statements being executed. If any ongoing DDL operations are detected, wait until these operations are completed, and then perform Steps 1 and 2 to check the table structure. Otherwise, go to Step 4.
4. Perform the DDL operation again on DRDS. If the Lock conflict error is reported, go to Step 5; otherwise, go to Step 3.
5. Run the `RELEASE DBLOCK` command to release the DDL operation lock, and then go to Step 4.

The procedure is as follows:

1. Check the table structure consistency

Run the `CHECK TABLE` command to check the table structure. When the returned result contains only one row and the displayed status is OK, the table structures are consistent.



Notice:

If no result is returned after you run `CHECK TABLE` on , retry by using the CLI.

```
mysql> check table `xxxx`;
+-----+-----+-----+-----+
| TABLE                | OP    | MSG_TYPE | MSG_TEXT |
+-----+-----+-----+-----+
| TDDL5_APP.xxxx | check | status  | OK      |
+-----+-----+-----+-----+
1 row in set (0.05 sec)
```

2. Check the table structure

Run the `SHOW CREATE TABLE` command to check the table structure. If table structures are consistent and correct, the DDL statement has been run successfully.

```
mysql> show create table `xxxx`;
+-----+-----+
| Table      | Create Table
+-----+-----+
| xxxxx     | CREATE TABLE `xxxx` (
```



```
12 rows in set (0.03 sec)
```

The value in the **TIME** column indicates the number of seconds that the command has been executed. If a command execution is too slow, as shown in the figure, you can run the **KILL '0-0-402714795'** command to cancel the slow command.



Notice:

In DRDS, one logical SQL statement corresponds to multiple statements on database shards. Therefore, you may need to kill multiple commands to stop a logical DDL statement. You can determine the logical SQL statement to which the command belongs based on the **INFO** column in the **SHOW PROCESSLIST** result set.

4. Solution for the lock conflict error

DRDS adds a database lock before performing a DDL operation and releases the lock after the operation. The **KILL DDL** operation may not release the lock. If you perform the DDL operation again, the following error message is returned:

```
Lock conflict, the last DDL may still be running
```

Then, run **RELEASE DBLOCK** to release the lock. After the command is canceled and the lock is released, run the DDL statement again during off-peak hours or when the service is stopped.

Other problems

clients cannot display the modified table structure.

To enable some clients to obtain table structures from system tables (such as **COLUMNS** or **TABLES**), DRDS creates a shadow database in database shard 0 on your ApsaraDB RDS for MySQL instance. The shadow database name must be the same as the name of your DRDS logical database. It stores all table structures and other information in the user database.

The client obtains the DRDS table structure from the system table of the shadow database. During the processing of DDL exceptions, the table structure may be modified normally in the user database but not in the shadow database due to some reasons. In this case, you need to connect to the shadow database and perform the DDL operation on the table again in the database.

**Notice:**

CHECK TABLE does not check whether the table structure in the shadow database is consistent with that in the user database.

7.14.11 Efficiently scan DRDS data

Distributed Relational Database Service (DRDS) supports efficient data scanning and uses aggregate functions for statistical summary during full table scan.

The following describes common scanning scenarios:

- **Scan of table without database or table shards:** DRDS transmits the original SQL statement to the backend ApsaraDB RDS for MySQL database for execution. In this case, DRDS supports any aggregate functions.
- **Non-full table scan:** DRDS transmits the original SQL statement to each single ApsaraDB RDS for MySQL database for execution. For example, when the shard key in the WHERE clause is Equal, non-full table scan is performed. In this case, DRDS also supports any aggregate functions.
- **Full table scan:** Currently, the supported aggregate functions are COUNT, MAX, MIN, and SUM. In addition, LIKE, ORDER BY, LIMIT, and GROUP BY are also supported during full table scan.
- **Parallel scan of all table shards:** If you need to export data from all databases, you can run the SHOW command to view the table topology and scan all table shards in parallel. For more information, see the following section.

Traverse tables by using a hint

1. Run the SHOW TOPOLOGY FROM TABLE_NAME command to obtain the table topology.

```
mysql:> SHOW TOPOLOGY FROM DRDS_USERS;
+-----+-----+-----+
| ID   | GROUP_NAME       | TABLE_NAME |
+-----+-----+-----+
| 0    | DRDS_00_RDS     | drds_users  |
| 1    | DRDS_01_RDS     | drds_users  |
+-----+-----+-----+
2 rows in set (0.06 sec)
```

By default, the non-partition table is stored in database shard 0.

2. Traverse each table for TOPOLOGY.

a. Run the current SQL statement in database shard 0.

```
#!/ TDDL:node='DRDS_00_RDS'*/ SELECT * FROM DRDS_USERS;
```

b. Run the current SQL statement in database shard 1.

```
#!/ TDDL:node='DRDS_01_RDS'*/ SELECT * FROM DRDS_USERS;
```



Notice:

We recommend that you run `SHOW TOPOLOGY FROM TABLE_NAME` to obtain the latest table topology before each scan.

Parallel scans

DRDS allows you to run `mysqldump` to export data. However, if you want to scan data faster, you can enable multiple sessions for each table shard to scan tables in parallel.

```
mysql> SHOW TOPOLOGY FROM LJLTEST;
+-----+-----+-----+
| ID   | GROUP_NAME      | TABLE_NAME |
+-----+-----+-----+
| 0    | TDDL5_00_GROUP  | ljltest_00  |
| 1    | TDDL5_00_GROUP  | ljltest_01  |
| 2    | TDDL5_00_GROUP  | ljltest_02  |
| 3    | TDDL5_01_GROUP  | ljltest_03  |
| 4    | TDDL5_01_GROUP  | ljltest_04  |
| 5    | TDDL5_01_GROUP  | ljltest_05  |
| 6    | TDDL5_02_GROUP  | ljltest_06  |
| 7    | TDDL5_02_GROUP  | ljltest_07  |
| 8    | TDDL5_02_GROUP  | ljltest_08  |
| 9    | TDDL5_03_GROUP  | ljltest_09  |
| 10   | TDDL5_03_GROUP  | ljltest_10  |
| 11   | TDDL5_03_GROUP  | ljltest_11  |
+-----+-----+-----+
12 rows in set (0.06 sec)
```

As shown above, the table has four database shards, and each database shard has three table shards. Run the following SQL statement to operate on the table shards of the `TDDL5_00_GROUP` database:

```
#!/ TDDL:node='TDDL5_00_GROUP'*/ select * from ljltest_00;
```



Note:

`TDDL5_00_GROUP` in HINT corresponds to the `GROUP_NAME` column in the results of the `SHOW TOPOLOGY` command. In addition, the table name in the SQL statement is the table shard name.

At this time, you can enable up to 12 sessions (corresponding to 12 table shards respectively) to process data in parallel.

7.15 Appendix: DRDS terms

This topic lists common terms of Distributed Relational Database Service (DRDS) for your reference.

Term	Definition	Remarks
DRDS	DRDS is a distributed database service that was independently developed by Alibaba to solve the bottlenecks of single-instance database services. DRDS is compatible with MySQL protocols and syntax. DRDS supports automatic sharding, smooth scaling, auto scaling, and transparent read/write splitting. DRDS provides O&M capabilities for distributed databases throughout their entire lifecycle.	
TDDL	Taobao Distributed Data Layer (TDDL) was developed by Alibaba and has become a preferred component for nearly 1,000 core applications of Alibaba.	
DRDS console	The DRDS console is designed for database administrators (DBAs) to isolate resources as required and perform operations, such as instance management, database and table management, read/write splitting configuration, smooth scaling, monitoring data display, and IP address whitelist.	
DRDS manager	The DRDS manager is designed for global O &M personnel and DBAs to manage all DRDS resources and monitor the system.	
DRDS server	The DRDS servers are the service layer of DRDS. Multiple DRDS servers make up a cluster to provide distributed database services, including the read/write splitting, routed SQL execution, result merging, dynamic database configuration, and globally unique ID (GUID).	

Term	Definition	Remarks
Load balancer	DRDS servers are stateless, and thus requests can be randomly routed to any DRDS server. The load balancer is used to complete this task . Server Load Balancer (SLB) is used for overall output by Apsara Stack. VIPServer is typically used for Alibaba middleware output.	
Diamond	Diamond is a system responsible for DRDS configuration storage and management. It provides the configuration storage, query, and notification functions. Diamond stores the database source data, sharding rules, and DRDS switch configuration.	
Data Replication System	Data Replication System is responsible for data migration and synchronization of DRDS. Its core capabilities include full data migration and incremental data synchronization. Its derived features include smooth data import, smooth scale-out, and global secondary index (GSI). Data Replication System requires the support of ZooKeeper and DRDS Rtools.	
DRDS instance	A DRDS instance consists of multiple DRDS servers. A DRDS instance can contain multiple DRDS databases.	
DRDS instance ID	The unique identifier for each DRDS instance.	
Number of nodes in a DRDS instance	The number of DRDS servers in a DRDS instance.	
VIP	The virtual IP address (VIP) of the load balancer is divided into: <ul style="list-style-type: none"> · 1. Public VIP, which is accessible from the Internet. It is used for testing. · 2. Private VIP, which is accessible only from the Alibaba Cloud internal network. 	
VPC	Virtual Private Cloud (VPC) is generally used on Alibaba Cloud.	
Region	A region is a geographical location, such as East China. This concept is generally used for Alibaba Cloud.	

Term	Definition	Remarks
Zone	A physical area with independent power grids and networks within one region, such as Hangzhou Zone A. This concept is generally used for Alibaba Cloud.	
Logical SQL	A logical SQL is an SQL statement sent from an application to DRDS.	
Physical SQL	A physical SQL is an SQL statement obtained after DRDS parses a logical SQL statement. The obtained physical SQL statement is then sent to ApsaraDB for RDS for execution.	Logical SQLs and physical SQLs may be the same or different. Logical and physical SQL statements may be in a one-to-one or one-to-many mapping.
QPS	The queries per second (QPS) is the average number of logical SQL statements executed by DRDS per second in a statistical cycle,	instead of the number of transactions. Most control statements , such as COMMIT and SET, are not counted in QPS .
RT	The response time (RT) is the average response time (in milliseconds) of logical SQL statements executed by DRDS in a statistical cycle. The RT of an SQL statement is calculated as follows: (Time when DRDS writes the last packet of the result set) - (Time when DRDS receives the SQL statement)	

Term	Definition	Remarks
Physical QPS	The physical QPS is the average number of physical SQL statements that DRDS executes on ApsaraDB for RDS per second in a statistical cycle.	
Physical RT	<p>The physical RT is the average response time (in milliseconds) of physical SQL statements executed by DRDS on ApsaraDB for RDS in a statistical cycle.</p> <p>The RT of a physical SQL statement is calculated as follows:</p> <p>(Time when DRDS receives the result set returned by ApsaraDB for RDS) - (Time when DRDS starts to obtain the connection to ApsaraDB for RDS)</p>	This includes the time of establishing a connection to ApsaraDB for RDS or obtaining a connection from the connection pool, the network transmission time, and the time of executing the SQL statement by ApsaraDB for RDS.
Number of connections	The number of connections established between the application and DRDS,	instead of the number of connections established between DRDS and ApsaraDB for RDS.
Incoming traffic	The network traffic generated when the application sends SQL statements to DRDS.	This traffic is irrelevant to the traffic used for interaction between DRDS and ApsaraDB for RDS.

Term	Definition	Remarks
Outgoing traffic	The network traffic generated when DRDS sends the result set to the application.	This traffic is irrelevant to the traffic used for interaction between DRDS and ApsaraDB for RDS.
Number of active threads (ThreadRunning)	The number of threads running in a DRDS instance. This parameter can be used to indicate the load of the DRDS instance.	
Global monitoring data	The total monitoring data of all databases in a DRDS instance.	
Memory usage	The Java Virtual Machine (JVM) memory usage of a DRDS server process.	
Total memory usage	The memory usage of the machine where the DRDS server is located.	This metric is available only when the DRDS servers are deployed on Elastic Compute Service (ECS) instances. Generally, this metric is used for Alibaba Cloud.
CPU utilization	It indicates the CPU utilization of the machine where a DRDS server is located.	This metric is available only when DRDS servers are deployed on ECS instances. Generally, this metric is used for Alibaba Cloud.

Term	Definition	Remarks
System load	The load of the machine where a DRDS server is located.	This metric is available only when DRDS servers are deployed on ECS instances. Generally, this metric is used for Alibaba Cloud.
Service port	The port used by DRDS servers to provide MySQL-based services to external applications.	Generally, the port number is 3306. However, when multiple DRDS servers (mostly physical machines) are deployed on one machine, the port number will change accordingly.
Management port	The port used by DRDS servers to provide management application program interfaces (APIs).	Generally, the port number is the service port number + 100.
Start time	The time when DRDS servers start.	
Running time	The continuous running time of the DRDS servers since the last startup time.	
Total memory size	The maximum JVM memory size of a DRDS server.	
Memory usage	The JVM memory that is already used by the DRDS servers.	

Term	Definition	Remarks
Number of nodes	Mandatory. The number of machines. A DRDS instance is essentially a DRDS cluster, and the number of nodes refers to the number of machines in the cluster.	
Instance type	Mandatory. The type of the instance, including dedicated and shared instances. A dedicated instance works in the exclusive mode. A shared instance works in the multi-tenant mode, which is generally used in Alibaba Cloud.	
Machine type	Mandatory. The type of the machine where a DRDS server is deployed. Valid values are Auto-selected, PHY, and ECS. The DRDS inventory is divided into physical machine inventory and virtual machine inventory according to the type of machines where the DRDS servers are deployed. The two types cannot be mixed because their deployment and O&M methods are different.	
AliUid	Mandatory. The ID of the instance. In Apsara Stack, this ID is provided by the account system in the deployment environment.	
Backend port	The backend port of the VIP. For a DRDS server, this port is the service port of machine where the DRDS server is deployed.	
Frontend port	The frontend port of the VIP for user access. Each VIP has a set of frontend ports and backend ports. The VIP forwards data from the frontend port to the backend port.	
Public/private	The network type of the VIP. Valid values: <ul style="list-style-type: none"> · Public: the public VIP, which is accessible from public networks. · Private: the private VIP (including VPC VIP), which is accessible from private networks. 	
IbId	The ID of the SLB instance, which is the unique ID of VIP. The VIP is managed based on this ID.	

Term	Definition	Remarks
Forwarding mode	<p>The port forwarding mode of the VIP. The following modes are supported:</p> <ul style="list-style-type: none"> • FNAT: This mode is recommended when the backend machine is a virtual machine or VPC needs to be supported. • NAT: This mode can be selected when the backend machine is a physical machine. Currently, this mode is only used on Alibaba Cloud. • Open FNAT: This mode is applicable only to Alibaba Cloud. 	
VPC ID	The ID of the destination VPC, that is, the VPC to be accessed.	
VSwitch ID	The ID of the destination VSwitch, which determines the CIDR block where the VPC VIP of the instance is in.	
APPName	The app name of the destination DRDS database . Each DRDS database has a corresponding app name for loading configurations.	
UserName	The username used to log on to the destination DRDS database.	
DBName	The name of the destination DRDS database you want to log on to.	
IP address whitelist	Only the IP addresses specified in the IP address whitelist can access the DRDS instance.	

Term	Definition	Remarks
Read-only instance	<p>ApsaraDB for RDS instances where physical databases reside are divided into the following two types based on whether data is written to the instances:</p> <ul style="list-style-type: none"> • Primary instance: Both read and write requests are allowed on such an instance. In Apsara Stack, ApsaraDB RDS for MySQL is supported. In Alibaba Cloud, ApsaraDB for RDS is supported. • Read-only instance: Only read requests are allowed on such an instance. In Apsara Stack , ApsaraDB RDS for MySQL is supported. In Alibaba Cloud, ApsaraDB for RDS is supported • 	
Read SQL statement	<p>A type of SQL statement used to read data, such as the SELECT statement. DRDS determines whether an SQL statement is a read-only SQL statement when it is not in a transaction. If the SQL statement is in a transaction, DRDS treats it as a write SQL statement during read/write splitting.</p>	
Read/write splitting	<p>If read-only ApsaraDB RDS for MySQL instances exist, you can configure in the DRDS console to allocate read SQL statements to the primary and read-only instances proportionally. DRDS automatically identifies the type of SQL statements and allocates them proportionally.</p>	
Smooth scale-out	<p>On the basis of horizontal partitioning, the data distribution on ApsaraDB RDS for MySQL instances is dynamically adjusted for scale-out . Generally, scale-out is completed asynchronously without any modification to the business code.</p>	

Term	Definition	Remarks
Broadcast of small tables	You can synchronize the data in a single table in a database to all database shards in advance , to convert the cross-database JOIN query into JOIN query that can be completed on physical databases.	
Horizontal partitioning	Horizontal partitioning distributes the data rows originally stored in one table to multiple tables based on specified rules to achieve horizontal linear scaling.	
Partition mode	This mode allows you to create multiple database shards on an ApsaraDB RDS for MySQL instance. These database shards make up a DRDS database. In this mode, all DRDS functions can be used.	
Non-partition mode	In this mode, a database that has been built on an ApsaraDB RDS for MySQL instance is used as a DRDS database. In this mode, only DRDS read/write splitting is allowed, while other DRDS features such as database sharding and table sharding are not allowed.	
Imported database	An existing database on the ApsaraDB RDS for MySQL instance selected for creating a DRDS database. This is a unique concept for the creation of a DRDS database.	
Read policy	The ratio of read SQL statements assigned by DRDS to the primary and read-only ApsaraDB RDS for MySQL instances.	
Full table scan	If no shard field is specified in the SQL statement , DRDS runs the SQL statement on all table shards and summarizes the results. You can disable this function because of its high overheads.	
Shard key	A column in a logical table. DRDS routes data and SQL statements to a physical table based on this column.	
Data import	The act of importing data from an existing ApsaraDB RDS for MySQL instance to a DRDS database.	

Term	Definition	Remarks
Full migration	The act of migrating all existing records from a database to DRDS. An offset is recorded before full migration starts.	
Offset	In the MySQL Binlog file, each row represents a data change operation. The position of a line in the Binlog file is called an offset.	
Incremental migration	The act of reading all MySQL Binlog records from the recorded offset, converting them into SQL statements, and then running them in DRDS. Incremental migration continues before the switchover.	
Switchover	A step of data import and smooth scale-out, which writes all the remaining incremental records from MySQL Binlog to DRDS.	
Cleanup	The last step of smooth scale-out, which cleans redundant data and configurations generated during smooth scale-out.	
Heterogeneous indexing	For table shards of a DRDS database, the WHERE condition in the SQL statement for query must contain the shard key for sharding whenever possible. In this way, DRDS routes the query request to a specific database shard, improving the query efficiency. If the WHERE condition of the SQL statement does not contain the shard key, DRDS performs a full table scan. DRDS provides heterogeneous indexing to solve this problem. The data in a database shard or table shard of a DRDS instance is fully or partially synchronized to another table based on different shard keys. The destination table to which the data is synchronized is called a heterogeneous index table.	
DRDS sequence	The DRDS sequence (a 64-digit number of the signed BIGINT type in MySQL) aims to generate a globally unique number sequence (not necessarily in increments) to generate keys such as a primary key column and a unique key.	
DRDS hint	To facilitate DRDS usage, DRDS defines some hints to specify special actions.	

8 AnalyticDB for MySQL

8.1 What is AnalyticDB for MySQL?

AnalyticDB for MySQL (originally named ADS) is an Alibaba Cloud developed real-time online analytical processing (RT-OLAP) service that enables online analytics of large amounts of data at high concurrency. It can analyze hundreds of billions of data records from multiple dimensions at millisecond-level timing to provide you with data-driven insights into your business.



Note:

OLAP systems are often compared with online transaction processing (OLTP) systems. OLAP systems are good at performing multidimensional and complex query and analytics on large amounts of data, and the OLAP model is usually adopted in analytical databases. On the other hand, OLTP systems are good at performing transactional processing. In the OLTP model, data processing follows strong consistency and atomicity. The OLTP model supports frequent INSERT and UPDATE operations, and is usually adopted in relational database management systems such as MySQL and Microsoft SQL Server.

AnalyticDB for MySQL is an RT-OLAP system that has the following benefits:

- Compatible with MySQL, business intelligence (BI) tools, and extract, transform, and load (ETL) tools. You can use AnalyticDB for MySQL to analyze and integrate your data in a cost-effective, efficient, and simple manner.
- Uses relational models to store data and provides SQL statements to flexibly compute and analyze data. No advanced data modeling is required.
- Uses distributed computing technologies to provide excellent real-time computing capabilities.

When processing tens of billions of data records or more, the performance of AnalyticDB for MySQL can achieve or even surpass that of multidimensional online analytical processing (MOLAP) systems. AnalyticDB for MySQL can compute tens of billions of data records within several hundred milliseconds. You can then explore large amounts of data without constraints, instead of viewing data reports based on a predefined logic.

- Computes hundreds of billions of data records in real time.

In the AnalyticDB for MySQL system, all data generated in your business system is used for data analysis instead of a sample. This maximizes the effectiveness of analysis results.

- Supports a large number of concurrent queries and ensures high system availability through dynamic multi-copy storage and computing technology. Therefore, AnalyticDB for MySQL can serve as a backend system for end user products (including Internet products and internal enterprise analysis products).

AnalyticDB for MySQL has been used in Internet business systems that have hundreds of thousands to tens of millions of users, such as Data Cube, Taobao Index, Kuaidi Dache, Alimama DMP, and Taobao Groceries.

AnalyticDB for MySQL is a real-time computing system that provides rapid and flexible online data analysis and computation.

8.2 Limits

Note the following limits before you use AnalyticDB for MySQL.

Object	Naming convention	Limit
Database name	The database name can be up to 64 characters in length and can contain letters, digits, and underscores (_). It must start with a lowercase letter and cannot contain two or more consecutive underscores (_).	The database name cannot be analyticdb. This name is reserved for built-in databases.
Table name	The table name must be 1 to 127 characters in length and can contain letters, digits, and underscores (_). It must start with a letter or underscore (_).	<ul style="list-style-type: none"> • It cannot contain quotation marks ('), exclamation marks (!), or spaces. • The table name cannot be SQL reserved keywords.

Object	Naming convention	Limit
Column name	The column name must be 1 to 127 characters in length and can contain letters, digits, and underscores (_). It must start with a letter or underscore (_).	<ul style="list-style-type: none"> It cannot contain quotation marks ('), exclamation marks (!), or spaces. The column name cannot be SQL reserved keywords.
Account name	The account name must be 2 to 16 characters in length and can contain lowercase letters, digits, and underscores (_). It must start with a lowercase letter and end with a lowercase letter or digit.	N/A
Password	The password must be 8 to 32 characters in length and contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. Special characters include !@#\$%^&*()_+ -=	N/A
Comment in a table	N/A	The comment can be up to 1,024 characters in length.
Comment in a column	N/A	The comment can be up to 1,024 characters in length.
Index name	N/A	The index name can be up to 64 characters in length.
Default value of a column	N/A	The default value can be up to 127 characters in length.

8.3 Quick start

8.3.1 Log on to the AnalyticDB for MySQL console

This topic describes how to log on to the AnalyticDB for MySQL console.

Prerequisites

- Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel.
- The URL used to access the ASCM console is in the following format: `http://IP address or domain name of the ASCM console/manage`.
- We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to access the ASCM console. Press Enter.
2. Enter your username and password.

The system has a default super administrator, whose username is super. The super administrator can create system administrators. A system administrator can create system users and notify the users of the default passwords by SMS or email.



Note:

When you log on to the ASCM console for the first time, you must modify the password of your username as instructed. For security concerns, your password must meet the minimum complexity requirements: The password must be 8 to 20 characters in length and must contain at least two types of the following characters: letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs (\$), and percent signs (%).

3. Click Login to go to the ASCM console homepage.
4. In the top navigation bar, click Products and choose Database Services > AnalyticDB for MySQL.

8.3.2 Create a database cluster

This topic describes how to create an AnalyticDB for MySQL cluster.

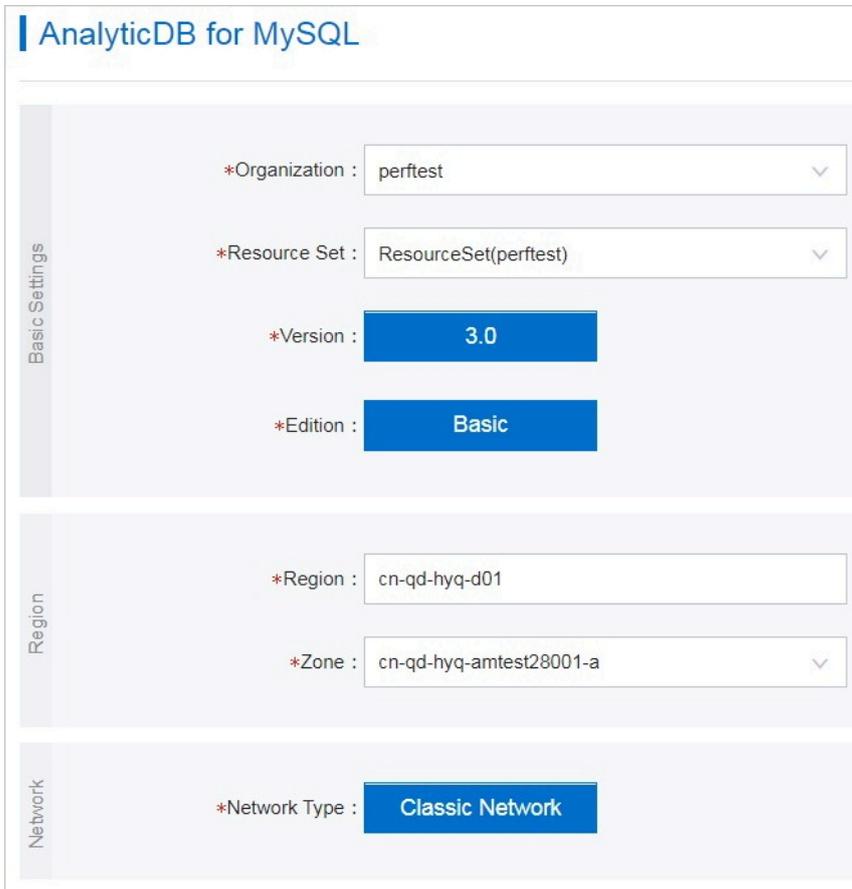
Procedure

1. [Log on to the AnalyticDB for MySQL console.](#)

2. Click **Create Cluster** in the upper-right corner of the **Clusters** page and configure parameters as prompted.

Parameter	Description
Region	The region where the cluster resides. You cannot change the region after the cluster is created. We recommend that you select a region that is closest to the geographic area of your business to improve access speed and stability.
Zone	The zone of the cluster. A zone is an independent physical area located within a region. There are no substantive differences between zones.
Organization	The organization to which the cluster belongs.
Resource Set	The resource set of the cluster.
Version	Only version 3.0 is supported.
Edition	Only Basic is supported.
Network Type	AnalyticDB for MySQL only supports classic networks. Cloud services in a classic network are not isolated. Access control to cloud services in a classic network is implemented by the security groups or whitelist policies of the services.
Specifications	The ECU specifications.
Node Groups	The number of node groups. By default, each node group consists of three replicas.

Parameter	Description
Storage	The storage space of a node group.



AnalyticDB for MySQL

Basic Settings

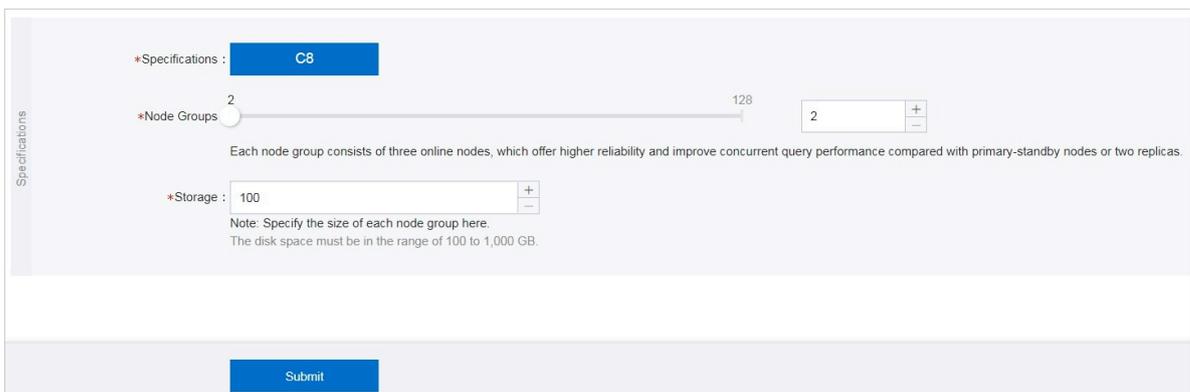
- *Organization : perftest
- *Resource Set : ResourceSet(perftest)
- *Version : 3.0
- *Edition : Basic

Region

- *Region : cn-qd-hyq-d01
- *Zone : cn-qd-hyq-amtest28001-a

Network

- *Network Type : Classic Network



Specifications

- *Specifications : C8
- *Node Groups : 2 (range 2 to 128)
- *Storage : 100

Each node group consists of three online nodes, which offer higher reliability and improve concurrent query performance compared with primary-standby nodes or two replicas.

Note: Specify the size of each node group here.
The disk space must be in the range of 100 to 1,000 GB.

Submit

3. After you have configured the preceding parameters, click **Submit**.

8.3.3 Configure a whitelist

This topic describes how to configure a whitelist for a cluster in the AnalyticDB for MySQL console.

Context

After creating an AnalyticDB for MySQL cluster, you must configure a whitelist for the cluster to allow external devices to access the cluster. The default whitelist contains only the default IP address 127.0.0.1, which indicates that no devices are allowed to access the cluster. The whitelist can enhance access security of AnalyticDB for MySQL clusters. We recommend that you maintain the whitelist on a regular basis. The whitelist does not affect the normal operation of AnalyticDB for MySQL clusters.

Procedure

1. [Log on to the AnalyticDB for MySQL console.](#)
2. In the upper-left corner of the page, select the region where the cluster resides.
3. On the Clusters page, click Cluster ID corresponding to the cluster for which you want to configure a whitelist.
4. In the left-side navigation pane, click Data Security.
5. On the Whitelist Settings tab, click Edit to the right of the default whitelist.



Note:

You can also click Create Whitelist to create a new whitelist.

6. In the Edit Whitelist dialog box that appears, delete the default IP address 127.0.0.1, enter the IP addresses or CIDR blocks you want to allow, and then click OK.
 - The CIDR block 0.0.0.0/0 indicates that all IP addresses are allowed to access the cluster. Exercise caution when you add this CIDR block.
 - If you enter a CIDR block such as 10.10.10.0/24, any IP addresses in the 10.10.10.X format can access the cluster.
 - If you want to add multiple IP addresses or CIDR blocks, separate multiple entries with commas (,). Do not add spaces before or after the commas.
Example: 192.168.0.1,172.16.213.9.
 - The whitelist modification takes effect within one minute.

8.3.4 Create a database account

This topic describes the database account types in AnalyticDB for MySQL and how to create database accounts.

Account types

AnalyticDB for MySQL supports two types of database account: **privileged and standard.**

Table 8-1: Account types

Account type	Description
Privileged account	<ul style="list-style-type: none"> • You can create and manage privileged accounts only in the console. • You can create only one privileged account for a cluster. You can use the privileged account to manage all standard accounts and databases of the cluster. • You can use the privileged account of a cluster to disconnect any standard account of the cluster from AnalyticDB for MySQL. A privileged account has more permissions, which allows you to perform fine-grained management operations. For example , you can grant query permissions on different tables to different users. • The privileged account in AnalyticDB for MySQL is equivalent to the root account in MySQL.
Standard account	<ul style="list-style-type: none"> • You can use only SQL statements to create and manage standard accounts • You can create up to 256 standard accounts for a cluster. • You need to manually grant a standard account the permissions to access a specific database. • You cannot use a standard account to disconnect other accounts from AnalyticDB for MySQL.

Create a privileged account

1. [Log on to the AnalyticDB for MySQL console](#)
2. In the upper-left corner of the page, select the region where the cluster resides.
3. On the Clusters page, click the ID of the cluster for which you want to create a privileged account. In the left-side navigation pane, click Accounts. On the Accounts page, click Create Account.
4. In the Create Account dialog box, configure the parameters as prompted.

Parameter	Description
Account	The name of the privileged account. The account name must be 2 to 16 characters in length and can contain lowercase letters, digits, and underscores (_). It must start with a lowercase letter and end with a lowercase letter or digit.
Account Type	The type of the account. The value is Privileged Account. This value cannot be changed.
Password	The password of the privileged account. The password must be 8 to 32 characters in length and contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. Special characters include ! @ # \$ % ^ & * () _ + - =
Confirm Password	Enter the password of the privileged account again.
Description	Optional. The description of the privileged account.

5. Click OK to create the privileged account.

Create a standard account

For more information about how to create a standard account and grant permissions to the standard account, see [CREATE USER](#).

What to do next

[Create a database.](#)

8.3.5 Connect to a database cluster

This topic describes how to establish connections to an AnalyticDB for MySQL cluster.

Prerequisites

After you [Configure a whitelist](#)[Create a database account](#), you can use a MySQL client (such as Navicat for MySQL, DBeaver, DbVisualizer, or SQL Workbench/J), or the MySQL command-line tool to connect to the AnalyticDB for MySQL cluster. You can also configure the connection information such as the endpoint, port, and account of an AnalyticDB for MySQL cluster in an application to connect to the cluster.

Use the code in an application to connect to AnalyticDB for MySQL

- [C#](#)
- [PHP](#)
- [Python](#)
- [Configure the Druid connection pool](#)
- [Java](#)

Use the MySQL command-line tool to connect to AnalyticDB for MySQL

[Use the MySQL command-line tool to connect to AnalyticDB for MySQL](#)

Use a client to connect to AnalyticDB for MySQL

- [SQL Workbench/J](#)
- [DbVisualizer](#)
- [DBeaver](#)
- [Navicat](#)

8.3.6 Create a database

You can use a MySQL client (such as Navicat for MySQL, DBeaver, DbVisualizer, or SQL Workbench/J), application code, or the MySQL command-line tool to connect to an AnalyticDB for MySQL cluster, and execute the `CREATE DATABASE` statement to create a database.

Prerequisites

You have connected to a database cluster. For more information, see [Connect to a database cluster](#).

Context

- **Syntax:** `CREATE DATABASE [IF NOT EXISTS] db_name`
- **Examples:**

```
create database adb_demo;
```

```
create database if not exists adb_demo2;
```

8.3.7 Apply for a public endpoint

This topic describes how to use the AnalyticDB for MySQL console to apply for a public endpoint for a cluster.

Context

- You must manually apply for a public endpoint. You can release the public endpoint if you do not need it.
- The public endpoint applies to the following scenarios:
 - You need to access an AnalyticDB for MySQL cluster on devices that are not deployed in the Alibaba Cloud environment.
 - You need to access an AnalyticDB for MySQL cluster from a region or of a network type that is different from the cluster.

Apply for a public endpoint

1. [Log on to the AnalyticDB for MySQL console](#).
2. In the upper-left corner of the page, select the region where the cluster resides.
3. On the Clusters page, click Cluster ID corresponding to the cluster.
4. On the Cluster Information page, click Apply for Public Endpoint.

5. In the Apply for Public Endpoint dialog box that appears, click OK to obtain a public endpoint.



Note:

Before you use the public endpoint to access the AnalyticDB for MySQL cluster, you must add the IP address of the device that you use to access the cluster to the whitelist. For more information, see [Configure a whitelist](#).

Release a public endpoint

1. [Log on to the AnalyticDB for MySQL console](#).
2. In the upper-left corner of the page, select the region where the cluster resides.
3. On the Clusters page, click Cluster ID corresponding to the cluster.
4. On the Cluster Information page, click Release Public Endpoint.
5. In the Release Public Endpoint dialog box that appears, click OK to release the public endpoint.

8.3.8 Synchronize data

To synchronize data in different scenarios, you can use the following data loading solutions provided by AnalyticDB for MySQL:

- Use Kettle to synchronize data from relational databases, NoSQL databases such as HBase, or Microsoft Office Excel or Access to AnalyticDB for MySQL.
- Use external tables to import data from OSS to AnalyticDB for MySQL or export data from AnalyticDB for MySQL to OSS.
- Use the Load Data statement to write local data to AnalyticDB for MySQL.
- If the data already exists in AnalyticDB for MySQL tables, use the `INSERT INTO... SELECT FROM` statement to synchronize data.

8.4 Connect to a database cluster

8.4.1 Use the MySQL command-line tool to connect to AnalyticDB for MySQL

This topic describes how to use the MySQL command-line tool to establish connections to an AnalyticDB for MySQL cluster.

Syntax

```
mysql -hadb_url -P3306 -uadb_user -padb_password
```

Parameters

- **adb_url**: the endpoint of the AnalyticDB for MySQL cluster to which you want to connect. You can obtain the endpoint of the cluster in the Network Information section of the Cluster Information page in the AnalyticDB for MySQL console.
- **3306**: the port number of the AnalyticDB for MySQL cluster endpoint. The default port number is 3306.
- **adb_user**: the account used to connect to the AnalyticDB for MySQL cluster. The account can be a privileged account or a standard account that has the required permissions.
- **adb_password**: the password of the account used to connect to the AnalyticDB for MySQL cluster.

Example

```
mysql -ham-bp****.ads.aliyuncs.com -P3306 -utest -pTest123
```

8.4.2 Use the code in a business system to connect to AnalyticDB for MySQL

8.4.2.1 C#

This topic describes how to use the MySQL Connector/NET connector to establish connections to an AnalyticDB for MySQL cluster.

```
using System;
using System.Data;
using MySql.Data;
using MySql.Data.MySqlClient;
namespace adbdemo
{
    public class Tutorial2
    {
        public static void Main()
```

```

    {
        // server: the endpoint of the AnalyticDB for MySQL cluster to
        // which you want to connect. You can obtain the endpoint on the Cluster
        // Information page of the AnalyticDB for MySQL console.
        // UID: the account used to connect to the AnalyticDB for MySQL
        // cluster. There are two types of accounts: privileged and standard.
        // database: the name of the database in the AnalyticDB for MySQL
        // cluster.
        // port: the port number of the AnalyticDB for MySQL cluster
        // endpoint.
        // password: the password of the account used to connect to the
        // AnalyticDB for MySQL cluster.
        string connStr = "server=... ;UID=...;database=...;port
        =...;password=... ;SslMode=none;";
        MySqlConnection conn = new MySqlConnection(connStr);
        try
        {
            Console.WriteLine("Connecting to MySQL...");
            conn.Open();
            string sql = "select c_custkey, c_name from customer
limit 1";
            MySqlCommand cmd = new MySqlCommand(sql, conn);
            MySqlDataReader rdr = cmd.ExecuteReader();
            while (rdr.Read())
            {
                Console.WriteLine(rdr[0] + " --- " + rdr[]);
            }
            rdr.Close();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.ToString());
        }
        conn.Close();
        Console.WriteLine("Done." ) ;
    }
}
}

```

8.4.2.2 PHP

This topic describes how to connect to an AnalyticDB for MySQL cluster in PHP.

Precautions

- **If your operating system is Linux, you must install the `php-mysql 5.1.x` module.**
- **If your operating system is Windows, you must install the `php_MySQL.dll` library**
-
- **For more information about how to enable `PrepareStatement` if you use [PDO](#) to connect to AnalyticDB for MySQL, see [Enable PrepareStatement for a client in different programming languages](#).**

Use MySQLi to connect to AnalyticDB for MySQL

```

// am-bp***.ads.aliyuncs.com: the endpoint of the AnalyticDB for MySQL
// cluster to which you want to connect. You can obtain the endpoint on
// the Cluster Information page of the AnalyticDB for MySQL console.

```

```

$sads_server_name="am-bp***.ads.aliyuncs.com";
// account_name: the account used to connect to the AnalyticDB for
MySQL cluster. There are two types of accounts: privileged and
standard.
$sads_username="account_name";
// account_password: the password of the account used to connect to
the AnalyticDB for MySQL cluster.
$sads_password="account_password";
// db_name: the name of the database in the AnalyticDB for MySQL
cluster.
$sads_database="db_name";
// 3306: the port number of the AnalyticDB for MySQL cluster endpoint.
$sads_port=3306;
// Connect to the AnalyticDB for MySQL cluster.
$sads_conn=mysqli_connect($sads_server_name,$sads_username,$sads_password,
$sads_database, $sads_port);

$strsql="SELECT user_id FROM my_ads_db.my_first_table limit 20;";
$result=mysqli_query($sads_conn, $strsql);
while($row = mysqli_fetch_array($result)) {
    // user_id: the name of the column to query.
    echo $row["user_id"] ;
}

```

Use PDO to connect to AnalyticDB for MySQL

```

// am-bp***.ads.aliyuncs.com: the endpoint of the AnalyticDB for MySQL
cluster to which you want to connect. You can obtain the endpoint on
the Cluster Information page of the AnalyticDB for MySQL console.
$sads_server_name = "am-bp***.ads.aliyuncs.com";
// account_name: the account used to connect to the AnalyticDB for
MySQL cluster. There are two types of accounts: privileged and
standard.
$sads_username = "account_name";
// account_password: the password of the account used to connect to
the AnalyticDB for MySQL cluster.
$sads_password = "account_password";
// db_name: the name of the database in the AnalyticDB for MySQL
cluster.
$sads_database = 'db_name';
// 3306: the port number of the AnalyticDB for MySQL cluster endpoint.
$sads_port = 3306;
$dsn = "mysql:host={$sads_server_name};dbname={$sads_database};port={$
ads_port}";
try {
    $dbh = new PDO($dsn, $sads_username, $sads_password);
    echo 'PDO Success !' ;
} catch (PDOException $e) {
    echo 'PDO Connection failed: ' . $e->getCode() ."\n" . $e->
getMessage() ."\n". $e->getTraceAsString();
}

```

8.4.2.3 Python

This topic describes how to use Python MySQLdb to connect to an AnalyticDB for MySQL cluster.

```

#!/usr/bin/python
# -*- coding: UTF-8 -*-
import MySQLdb

```

```
# Create a database connection.
# host: the endpoint or IP address of the AnalyticDB for MySQL cluster
to which you want to connect.
# port: the port number of the AnalyticDB for MySQL cluster endpoint.
# user: the account used to connect to the AnalyticDB for MySQL
cluster. There are two types of accounts: privileged and standard.
# passwd: the password of the account used to connect to the
AnalyticDB for MySQL cluster.
# db: the name of the database in the AnalyticDB for MySQL cluster.
db = MySQLdb.connect(host='am-bp***.ads.aliyuncs.com', port=3306, user
='account_name', passwd='account_password', db='db_name')
# Use the cursor() method to obtain an operation cursor.
cursor = db.cursor()
# Use the execute() method to execute SQL statements.
cursor.execute("SELECT VERSION()")
# Use the fetchone() method to obtain a data entry.
data = cursor.fetchone()
print "Database version : %s " % data
# Close the database connection.
db.close()
```

8.4.2.4 Configure the Druid connection pool

This topic describes how to use the Java Database Connectivity (JDBC) Druid connection pool to connect to an AnalyticDB for MySQL cluster.

Precautions

- **When you use the Druid connection pool to connect to an AnalyticDB for MySQL cluster, we recommend that you set `keepAlive=true`. In this way, you can reuse connections and avoid short-lived connections.**
- **Use Druid 1.1.12 or later versions.**

Configure the Druid connection pool

```
<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource"
init-method="init" destroy-method="close">
  <!-- jdbc_url: the endpoint of the AnalyticDB for MySQL cluster
to which you want to connect. You can obtain the endpoint on the
Cluster Information page of the AnalyticDB for MySQL console.-->
  <property name="url" value="${jdbc_url}" />
  <!-- jdbc_user: the account used to connect to the AnalyticDB
for MySQL cluster. There are two types of accounts: privileged and
standard. -->
  <property name="username" value="${jdbc_user}" />
  <!-- jdbc_password: the password of the account used to connect
to the AnalyticDB for MySQL cluster. -->
  <property name="password" value="${jdbc_password}" />
  <!-- Set the initial size of the connection pool, and the
minimum and maximum number of connections. -->
  <property name="initialSize" value="5" />
  <property name="minIdle" value="10" />
  <property name="maxActive" value="20" />
  <!-- Set the timeout period for obtaining a connection from the
connection pool. -->
  <property name="maxWait" value="60000" />
  <!-- Set the interval for detecting idle connections to be
closed. Unit: milliseconds. -->
```

```

    <property name="timeBetweenEvictionRunsMillis" value="2000" />
    <!-- Set the minimum validity period of a connection in the
connection pool. Unit: milliseconds. -->
    <property name="minEvictableIdleTimeMillis" value="600000" />
    <property name="maxEvictableIdleTimeMillis" value="900000" />
    <property name="validationQuery" value="select 1" />
    <property name="testWhileIdle" value="true" />
    <!-- Specify whether to check the validity of the connection
each time you obtain a connection from the connection pool. A value
of true indicates that the validity of the connection is checked. A
value of false indicates that the validity of the connection is not
checked. -->
    <property name="testOnBorrow" value="false" />
    <!-- Specify whether to check the validity of the connection
each time you return a connection to the connection pool. A value of
true indicates that the validity of the connection is checked. A value
of false indicates that the validity of the connection is not checked
. -->
    <property name="testOnReturn" value="false" />
    <property name="keepAlive" value="true" />
    <property name="phyMaxUseCount" value="100000" />
    <!-- Set the filters to be used for monitoring statistics. -->
    <property name="filters" value="stat" />
</bean>

```

8.4.2.5 Java

This topic describes how to use MySQL Java Database Connectivity (JDBC) to connect to an AnalyticDB for MySQL cluster.

Supported MySQL JDBC driver versions

AnalyticDB for MySQL supports the following MySQL JDBC driver versions:

- **5.0 series:** 5.0.2, 5.0.3, 5.0.4, 5.0.5, 5.0.7, and 5.0.8.
- **5.1 series:** 5.1.1, 5.1.2, 5.1.3, 5.1.4, 5.1.5, 5.1.6, 5.1.7, 5.1.8, 5.1.11, 5.1.12, 5.1.13, 5.1.14, 5.1.15, 5.1.16, 5.1.17, 5.1.18, 5.1.19, 5.1.20, 5.1.21, 5.1.22, 5.1.23, 5.1.24, 5.1.25, 5.1.26, 5.1.27, 5.1.28, 5.1.29, 5.1.31, 5.1.32, 5.1.33, and 5.1.34.

Precautions

To create MySQL JDBC connections in Java, you must add the MySQL JDBC driver package to your project. You must add the path of the `mysql-connector-java-x.x.x.jar` file to the value of the `CLASSPATH` variable in your project. Otherwise, you cannot create a MySQL JDBC connection.

Sample code for creating a MySQL JDBC connection without retries

To connect to AnalyticDB for MySQL databases through MySQL JDBC, you can add the following Java code to your business system:

```

Connection connection = null;
Statement statement = null;
ResultSet rs = null;

```

```

try {
    Class.forName("com.mysql.jdbc.Driver");
    // adb_url: the endpoint of the AnalyticDB for MySQL cluster to
    // which you want to connect. You can obtain the endpoint on the Cluster
    // Information page of the AnalyticDB for MySQL console. The default port
    // number is 3306.
    // db_name: the name of the database in the AnalyticDB for MySQL
    // cluster.
    String url = "jdbc:mysql://adb_url:3306/db_name? useUnicode=true&
characterEncoding=UTF-8";
    Properties connectionProps = new Properties();
    // account_name: the account used to connect to the AnalyticDB
    // for MySQL cluster. There are two types of accounts: privileged and
    // standard.
    connectionProps.put("user", "account_name");
    // account_password: the password of the account used to connect
    // to the AnalyticDB for MySQL cluster.
    connectionProps.put("password", "account_password");
    connection = DriverManager.getConnection(url, connectionProps);
    statement = connection.createStatement();
    String query = "select count(*) from information_schema.tables";
    rs = statement.executeQuery(query);
    while (rs.next()) {
        System.out.println(rs.getObject(1));
    }
} catch (ClassNotFoundException e) {
    e.printStackTrace();
} catch (SQLException e) {
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
} finally {
    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    if (statement != null) {
        try {
            statement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    if (connection != null) {
        try {
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
}

```

Sample code for creating a MySQL JDBC connection with retries

When you create a MySQL JDBC connection, you can configure the following parameters to implement a retry mechanism:

```

public static final int MAX_QUERY_RETRY_TIMES = 3;
public static Connection conn = null;

```

```

public static Statement statement = null;
public static ResultSet rs = null;
public static void main(String[] args) throws ClassNotFoundException {
    // db_name: the name of the database in the AnalyticDB for MySQL
    cluster.
    String yourDB = "db_name";
    // account_name: the account used to connect to the AnalyticDB
    for MySQL cluster. There are two types of accounts: privileged and
    standard.
    String username = "account_name";
    // account_password: the password of the account used to connect
    to the AnalyticDB for MySQL cluster.
    String password = "account_password";
    Class.forName("com.mysql.jdbc.Driver");
    // adb_url: the endpoint of the AnalyticDB for MySQL cluster to
    which you want to connect. You can obtain the endpoint on the Cluster
    Information page of the AnalyticDB for MySQL console. The default port
    number is 3306.
    String url = "jdbc:mysql://adb_url:3306/" + yourDB + "? useUnicode
    =true&characterEncoding=UTF-8";
    Properties connectionProps = new Properties();
    connectionProps.put("user", username);
    connectionProps.put("password", password);
    String query = "select id from test4dmp.test limit 10";
    int retryTimes = 0;
    // Run automatic retries through loops.
    while (retryTimes < MAX_QUERY_RETRY_TIMES) {
        try {
            getConn(url, connectionProps);
            execQuery(query); // Run a query.
            break; // If the query is run, exit the loop.
        } catch (SQLException e) {
            System.out.println("Met SQL exception: " + e.getMessage()
            + ", then go to retry task ...");
            try {
                if (conn == null || conn.isClosed()) {
                    retryTimes++;
                }
            } catch (SQLException e1) {
                if (conn != null) {
                    try {
                        conn.close();
                    } catch (SQLException e2) {
                        e.printStackTrace();
                    }
                }
            }
        }
    }
    // Clear connection resource.
    closeResource();
}
/**
 * Get connection.
 *
 * @param url
 * @param connectionProps
 * @throws SQLException
 */
public static void getConn(String url, Properties connectionProps)
throws SQLException {
    conn = DriverManager.getConnection(url, connectionProps);
}
/**

```

```
* Query task execution logic.
*
* @param sql
* @throws SQLException
*/
public static void execQuery(String sql) throws SQLException {
    Statement statement = null;
    ResultSet rs = null;
    statement = conn.createStatement();
    for (int i = 0; i < 10; i++) {
        long startTs = System.currentTimeMillis();
        rs = statement.executeQuery(sql);
        int cnt = 0;
        while (rs.next()) {
            cnt++;
            System.out.println(rs.getObject(1) + " ");
        }
        long endTs = System.currentTimeMillis();
        System.out.println("Elapse Time: " + (endTs - startTs));
        System.out.println("Row count: " + cnt);
        try {
            Thread.sleep(160000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
/**
 * Close connection resource.
 */
public static void closeResource() {
    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    if (statement != null) {
        try {
            statement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    if (conn != null) {
        try {
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

```
}
```

8.4.3 Enable PreparedStatement for a client in different programming languages

Background information

In most databases, SQL statements can be preprocessed on the server to improve database performance. AnalyticDB for MySQL databases deliver high performance without preprocessing SQL statements on the server side because these databases have the strong query capability, computing power, and plan cache feature.

AnalyticDB for MySQL databases do not support the protocol for preprocessing SQL statements on the server. In most programming languages, you can enable PreparedStatement for a client to prepare statements or interpolate parameters on the client.

This topic describes how to enable PreparedStatement for a client in different programming languages.

MySQL Connector/J (JDBC) driver

You can enable PreparedStatement for the MySQL Connector/J (JDBC) driver when the following condition is met: `useServerPrepStmts=false`. For more information, visit [Configuration Properties for Connector/J](#).



Note:

If the `useCursorFetch=true` condition is met, the `useServerPrepStmts=false` condition is overwritten and PreparedStatement cannot be enabled.

MariaDB Connector/J

You can enable PreparedStatement for MariaDB Connector/J if the following condition is met: `useServerPrepStmts=false`. For more information, visit [About MariaDB Connector/J](#).

Go MySQL driver

You can enable PreparedStatement for the Go MySQL driver if the following condition is met: `interpolateParams=true`. For more information, visit [Go-MySQL-Driver](#).

PDO

You can enable PrepareStatement for PDO if the following condition is met: `PDO::ATTR_EMULATE_PREPARES=TRUE`. For more information, visit [setAttribute](#).

8.4.4 Use a client to connect to AnalyticDB for MySQL

8.4.4.1 SQL Workbench/J

SQL Workbench/J is a DBMS-independent, cross-platform SQL query tool. This topic describes how to use SQL Workbench/J to connect to an AnalyticDB for MySQL cluster.

Preparations

Before you use SQL Workbench/J, complete the following steps:

- Install the MySQL JDBC driver.
- Install SQL Workbench/J.
- Add the IP address of the device on which SQL Workbench/J is installed to the whitelist of the AnalyticDB for MySQL cluster. For more information, see [Configure a whitelist](#).

Precautions

For more information about how to enable PrepareStatement in SQL Workbench/J, see [Enable PrepareStatement for a client in different programming languages](#).

Procedure

1. Start SQL Workbench/J and choose File > Manage Drivers....



Note:

If you use SQL Workbench/J for the first time, you must add the JDBC driver and its JAR file. When you use SQL Workbench/J later, you can skip step 2.

2. In the Manage drivers dialog box that appears, select MySQL as the driver, add the JAR file of the driver, and then click OK.

3. Choose File > Connect window. In the Select Connection Profile dialog box that appears, configure the connection parameters as required.

Parameter	Description
New profile	The name of the connection, which facilitates subsequent management.
Driver	The type of the driver. Select MySQL from the drop-down list.
URL	<p>The endpoint of the AnalyticDB for MySQL cluster to which you want to connect, including the endpoint, port number, and database name. The format is <code>jdbc:mysql://hostname:port/name_of_database</code>, where:</p> <ul style="list-style-type: none"> <code>hostname</code>: the public endpoint or VPC endpoint of the cluster. <code>port</code>: the port number of the AnalyticDB for MySQL cluster endpoint. The default port number is 3306. <code>name_of_database</code>: the name of the database in the cluster. This parameter is optional.
Username	<p>The account used to connect to the AnalyticDB for MySQL cluster. You can use either of the following accounts:</p> <ul style="list-style-type: none"> Privileged accounts. Standard accounts that have the permissions to connect to the cluster.
Password	The password of the account used to connect to the AnalyticDB for MySQL cluster.

4. After configuring the preceding parameters, click Test to test connectivity. After the connection passes the test, click OK to connect to the AnalyticDB for MySQL database.

8.4.4.2 DbVisualizer

This topic describes how to use DbVisualizer to connect to an AnalyticDB for MySQL cluster.

Preparations

Before you use DbVisualizer, complete the following steps:

- Install the MySQL JDBC driver.
- Install DbVisualizer.
- Add the IP address of the device on which DbVisualizer is installed to the whitelist of the AnalyticDB for MySQL cluster. For more information, see [Configure a whitelist](#).

Procedure

1. Start DbVisualizer and choose Tools > Connection Wizard. In the New Connection Wizard dialog box that appears, enter a name for the connection to facilitate subsequent management.
2. Click Next and select MySQL as the database driver from the drop-down list.
3. Click Next and configure the connection parameters as required.

Parameter	Description
Notes	The description of the connection.
Database Server	The endpoint of the AnalyticDB for MySQL cluster to which you want to connect. You can view the connection information about the cluster on the Cluster Information page of the AnalyticDB for MySQL console.
Database Port	The port number of the AnalyticDB for MySQL cluster endpoint. The default port number is 3306.
Database	The name of the database in the AnalyticDB for MySQL cluster.

Parameter	Description
Database Userid	The account used to connect to the AnalyticDB for MySQL cluster. You can use either of the following accounts: <ul style="list-style-type: none"> Privileged accounts. Standard accounts that have the permissions to connect to the cluster.
Database Password	The password of the account used to connect to the AnalyticDB for MySQL cluster.

- After configuring the preceding parameters, click Ping Server to test connectivity. After the connection passes the test, click Finish.

After the AnalyticDB for MySQL cluster is connected, you can manage data through DbVisualizer.

8.4.4.3 DBeaver

This topic describes how to use DBeaver to connect to an AnalyticDB for MySQL cluster.

Context

DBeaver is a free and open-source database management tool distributed under General Public License (GPL). It is designed for developers and database administrators. DBeaver supports databases that are compatible with Java Database Connectivity (JDBC), such as MySQL, PostgreSQL, Oracle, DB2, Microsoft SQL Server, and Sybase. DBeaver provides a graphical user interface (GUI), on which you can view database schemas, execute SQL statements and scripts, view and export data, and process binary large object (BLOB) or character large object (CLOB) data.

Preparations

Before you use DBeaver, complete the following steps:

- Install DBeaver.
- Install the MySQL JDBC driver.
- Add the IP address of the device on which DBeaver is installed to the whitelist of the AnalyticDB for MySQL cluster. For more information, see [Configure a whitelist](#).

Procedure

1. Start DBeaver and choose Database > New Connection.
2. In the Create new connection dialog box that appears, select MySQL as the connection type and click Next.
3. On the General tab of the Create new connection dialog box, configure the connection parameters as required.

Parameter	Description
Server Host	The endpoint of the AnalyticDB for MySQL cluster to which you want to connect. You can view the connection information about the cluster on the Cluster Information page of the AnalyticDB for MySQL console.
Port	The port number of the AnalyticDB for MySQL cluster endpoint. The default port number is 3306.
Database	The name of the database in the AnalyticDB for MySQL cluster.
User Name	The account used to connect to the AnalyticDB for MySQL cluster. You can use either of the following accounts: <ul style="list-style-type: none"> • Privileged accounts. • Standard accounts that have the permissions to connect to the cluster.
Password	The password of the account used to connect to the AnalyticDB for MySQL cluster.

4. After configuring the preceding parameters, click Test Connection.... After the connection passes the test, click Finish to connect to the cluster.

8.4.4.4 Navicat

This topic describes how to use Navicat to connect to an AnalyticDB for MySQL cluster.

Context

Navicat is a fast, reliable, and cost-effective database management tool designed to simplify database management and reduce costs. Navicat provides a graphical user interface (GUI) for you to create remote connections from the local computer to AnalyticDB for MySQL clusters and manage data.

Preparations

Before you use Navicat for MySQL, complete the following steps:

- **Install Navicat for MySQL.**
- **Add the IP address of the device on which Navicat for MySQL is installed to the whitelist of the target AnalyticDB for MySQL cluster. For more information, see [Configure a whitelist](#).**

Procedure

1. **Start Navicat for MySQL, and choose File > New Connection > MySQL. In the New Connection dialog box that appears, configure the connection parameters as required.**

Parameter	Description
Connection Name	The name of the connection. We recommend that you choose an identifiable name, which facilitates subsequent management.
Host	The endpoint of the AnalyticDB for MySQL cluster to which you want to connect. You can view the connection information about the cluster on the Cluster Information page of the AnalyticDB for MySQL console.
Port	The port number of the AnalyticDB for MySQL cluster endpoint. The default port number is 3306.
User Name	The account used to connect to the AnalyticDB for MySQL cluster. You can use either of the following accounts: <ul style="list-style-type: none"> • Privileged accounts. • Standard accounts that have the permissions to connect to the cluster.

Parameter	Description
Password	The password of the account used to connect to the AnalyticDB for MySQL cluster.

If your operating system is macOS, add the database name after configuring the connection information.

2. Click **Test Connection**. After the connection passes the test, click **OK**.

The connection to the AnalyticDB for MySQL cluster is created but is not activated. You need to activate the connection manually.

3. **Right-click** a connection name and select **Open Connection**. **Right-click** a database name to activate the database connection. Then, you can use Navicat for MySQL to manage data.

8.5 Manage database clusters

8.5.1 View monitoring information

You can view cluster monitoring information in real time in the AnalyticDB for MySQL console.

Procedure

1. [Log on to the AnalyticDB for MySQL console](#).
2. In the upper-left corner of the page, select the region where the cluster resides.
3. On the Clusters page, click **Cluster ID** corresponding to the cluster.
4. In the left-side navigation pane, click **Monitoring Information** to view the cluster monitoring information.

The monitoring information includes CPU utilization, cluster connections, QPS, and query response time.

8.5.2 Change specifications

This topic describes how to change the specifications of AnalyticDB for MySQL clusters.

Context



Note:

You can modify only the number of node groups.

Procedure

1. [Log on to the AnalyticDB for MySQL console.](#)
2. In the upper-left corner of the page, select the region where the cluster resides.
3. Click **Change Specifications** in the **Actions** column corresponding to the cluster.
4. On the **Change Specifications** page, set **Node Groups** and click **Submit**.

8.5.3 Delete a cluster

You can delete AnalyticDB for MySQL clusters based on business needs.

Procedure

1. [Log on to the AnalyticDB for MySQL console.](#)
2. In the upper-left corner of the page, select the region where the cluster resides.
3. Click **Delete** in the **Actions** column corresponding to the cluster.
4. In the **Delete Cluster** dialog box that appears, click **OK**.

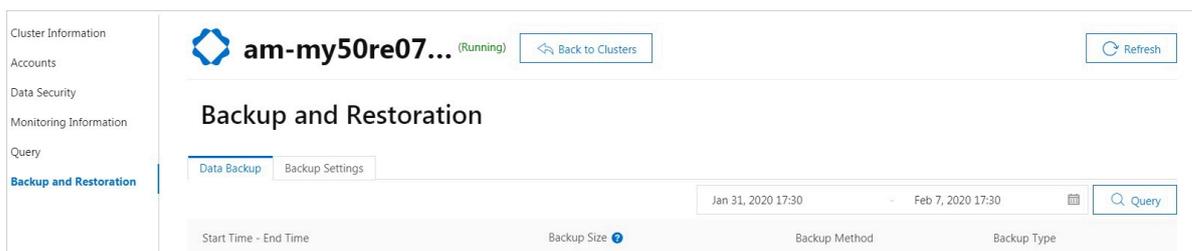
8.6 Backup and restoration

8.6.1 Back up data

AnalyticDB for MySQL uses physical backups (snapshots) to automatically back up data every Tuesday and Friday from 02:00 to 03:00. The backup files are retained for seven days.

Procedure

1. [Log on to the AnalyticDB for MySQL console.](#)
2. In the upper-left corner of the page, select the region where the cluster resides.
3. On the **Clusters** page, click **Cluster ID** corresponding to the cluster.
4. In the left-side navigation pane, click **Backup and Restoration**.
5. Click the **Backup Settings** tab to view the automatic settings.



8.6.2 Restore data

In AnalyticDB for MySQL, the cluster administrator can restore data from backup sets.

8.7 Diagnostics and optimization

8.7.1 Use functions related to slow SQL statements

AnalyticDB for MySQL provides the slow SQL analysis feature. You can view the trends and statistics of slow SQL statements and obtain diagnostic analysis and suggestions.

Procedure

1. [Log on to the AnalyticDB for MySQL console](#).
2. In the upper-left corner of the page, select the region where the cluster resides.
3. On the Clusters page, click Cluster ID corresponding to the cluster.
4. In the left-side navigation pane, click Query.

You can use one of the following methods to view the slow query logs:

- **Slow SQL Trend:** views the trend line chart of the slow query logs.
- **Slow SQL Details:** views the detailed data of the slow query logs.

8.8 Account and permission management

8.8.1 Permission model

Permission levels

An AnalyticDB for MySQL cluster supports the following four levels of permission control:

- **GLOBAL:** cluster-level permissions
- **DB:** database-level permissions
- **TABLE:** table-level permissions
- **COLUMN:** column-level (field) permissions

If you want a user to query the data of one specific column in a table, you can grant the **SELECT** permission on the column to the user. Example: `GRANT select (customer_id) ON customer TO 'test321'.`

Operations and corresponding permissions

Operation	Required permission	Supported permission level
SELECT	SELECT	<ul style="list-style-type: none"> • DB • TABLE • COLUMN
INSERT	INSERT	<ul style="list-style-type: none"> • DB • TABLE • COLUMN
INSERT...SELECT...FROM ...	<ul style="list-style-type: none"> • INSERT • SELECT 	<ul style="list-style-type: none"> • DB • TABLE • COLUMN
UPDATE	UPDATE	<ul style="list-style-type: none"> • DB • TABLE • COLUMN
DELETE	DELETE	<ul style="list-style-type: none"> • DB • TABLE
TRUNCATE TABLE	DROP	<ul style="list-style-type: none"> • DB • TABLE
ALTER TABLE	<ul style="list-style-type: none"> • ALTER • INSERT • CREATE 	<ul style="list-style-type: none"> • DB • TABLE
CREATE DATABASE	CREATE	N/A
CREATE TABLE	CREATE	<ul style="list-style-type: none"> • DB • TABLE
SHOW CREATE TABLE	SELECT	<ul style="list-style-type: none"> • DB • TABLE
DROP DATABASE	DROP	DB
DROP TABLE	DROP	<ul style="list-style-type: none"> • DB • TABLE

Operation	Required permission	Supported permission level
CREATE VIEW	<ul style="list-style-type: none"> • CREATE_VIEW • SELECT 	<ul style="list-style-type: none"> • DB • TABLE <p>To execute the <code>CREATE VIEW REPLACE</code> statement, the DROP permission is also required in addition to the preceding permissions.</p>
DROP VIEW	DROP	<ul style="list-style-type: none"> • DB • TABLE
SHOW CREATE VIEW	<ul style="list-style-type: none"> • SHOW_VIEW • SELECT 	<ul style="list-style-type: none"> • DB • TABLE
CREATE_PROCEDURE	CREATE_ROUTINE	N/A
DROP_PROCEDURE	ALTER_ROUTINE	N/A
CREATE_EVENT	EVENT	N/A
DROP_EVENT	EVENT	N/A
CREATE USER/DROP USER/RENAME USER	CREATE_USER	N/A
SET PASSWORD	SUPER	N/A
GRANT/REVOKE	GRANT	N/A

8.8.2 Manage database accounts and permissions

- For more information about how to create a RAM user, see `CREATE USER`.
- For more information about how to grant permissions to a RAM user, see `GRANT`.
- For more information about how to revoke the permissions of a RAM user, see `REVOKE`.
- For more information about how to modify the name of a database account, see `RENAME USER`.
- For more information about how to delete a database account, see `DROP USER`.

8.9 Data visualization

8.9.1 Tableau

This topic describes how to use Tableau to connect to an AnalyticDB for MySQL cluster and perform data analytics and visualization.

Preparations

Before you use Tableau, complete the following steps:

- Install the MySQL Open Database Connectivity (ODBC) driver. We recommend that you use MySQL Connector/ODBC 3.5.1 or 5.3.
- Install Tableau 9.0 or later.

Procedure

1. Start Tableau and click MySQL in the left-side navigation pane to create a MySQL connection. In the MySQL dialog box that appears, configure the connection parameters as prompted.

Parameter	Description
Server	The endpoint of the AnalyticDB for MySQL cluster to which you want to connect.
Port	The port number of the AnalyticDB for MySQL cluster endpoint.
User Name	The account used to connect to the AnalyticDB for MySQL cluster. You can use either of the following accounts: <ul style="list-style-type: none"> • Privileged accounts. • Standard accounts that have the permissions to connect to the cluster.
Password	The password of the account used to connect to the AnalyticDB for MySQL cluster.

2. After you configure the preceding parameters, click Log On to connect to the AnalyticDB for MySQL cluster.

Use Tableau

In Tableau, you can view the databases on which you have permissions. After selecting a database, you can read tables, preview data, and generate visual reports. For more information about how to use Tableau, visit [Tableau](#).

8.9.2 QlikView

This topic describes how to use QlikView to connect to an AnalyticDB for MySQL cluster and build a business intelligence (BI) system.

Preparations

Before you use QlikView, complete the following steps:

- Install the MySQL Open Database Connectivity (ODBC) driver. We recommend that you use MySQL Connector/ODBC 3.5.1 or 5.3.
- Install QlikView 11.20.x.

Procedure

1. On the host where QlikView is installed, click Control Panel, and choose System and Security > Administrative Tools > ODBC Data Sources. This path may vary with different operating systems. In the ODBC Data Source Administrator dialog box that appears, add a system data source name (DSN) and select MySQL ODBC 5.xx Driver as the data source.

Parameter	Description
Data Source Name	The name of the database in the AnalyticDB for MySQL cluster.
TCP/IP Server	The endpoint of the AnalyticDB for MySQL cluster to which you want to connect.
Port	The port number of the AnalyticDB for MySQL cluster endpoint.
User	The account used to connect to the AnalyticDB for MySQL cluster. You can use either of the following accounts: <ul style="list-style-type: none"> • Privileged accounts. • Standard accounts that have the permissions to connect to the cluster.

Parameter	Description
Password	The password of the account used to connect to the AnalyticDB for MySQL cluster.

2. After you configure the preceding parameters, click **Test** to test connectivity. After the connection passes the test, click **OK** to create the ODBC connection.
3. Start QlikView and choose **File > Edit Script**. Select the database name specified in step 1 to test connectivity.
4. After the database passes the connectivity test, you can use the following **SELECT** statement to obtain the data in the AnalyticDB for MySQL database:

```
SELECT * FROM DATABASE_NAME.TABLE_NAME;
```

For example, use the following statement to obtain the data in the `user_info` table:

```
SELECT * FROM adb_database.user_info;
```

Use QlikView

After obtaining data from AnalyticDB for MySQL databases, you can use QlikView to perform more operations. For more information about how to use QlikView, visit [QlikView](#).

8.9.3 FineReport

This topic describes how to use FineReport to connect to an AnalyticDB for MySQL cluster and manage reports.

Preparations

Before you use FineReport, complete the following steps:

- Install the MySQL JDBC driver.
- Install FineReport.

Procedure

1. Start FineReport and choose **Server > Define Data Connection**.

2. In the Define Data Connection dialog box that appears, configure the parameters as required.

Parameter	Description
Database	The type of the database. Select MySQL from the drop-down list.
Driver	The type of the driver. Select MySQL JDBC driver from the drop-down list.
URL	The endpoint of the AnalyticDB for MySQL cluster to which you want to connect, including the endpoint and port number. The format is <code>jdbc:mysql://hostname:port</code> , where: <ul style="list-style-type: none"> • <code>hostname</code>: the public endpoint or VPC endpoint of the cluster. • <code>port</code>: the port number of the AnalyticDB for MySQL cluster endpoint. The default port number is 3306.
User Name	The account used to connect to the AnalyticDB for MySQL cluster. You can use either of the following accounts: <ul style="list-style-type: none"> • Privileged accounts. • Standard accounts that have the permissions to connect to the cluster.
Password	The password of the account used to connect to the AnalyticDB for MySQL cluster.

3. After you configure the preceding parameters, click Test Connection. After the connection passes the test, click Confirm to connect to the AnalyticDB for MySQL cluster.

Use FineReport

After FineReport is connected to the AnalyticDB for MySQL cluster, you can obtain the data in AnalyticDB for MySQL databases and use FineReport to generate reports. For more information about how to use FineReport, visit [FineReport](#).

8.10 Data migration and synchronization

8.10.1 Use Kettle to synchronize the local data to AnalyticDB for MySQL

This topic uses Excel as an example to describe how to use Kettle to synchronize the local Excel data to AnalyticDB for MySQL.

Background information

Kettle is a popular open-source extract-transform-load (ETL) tool used for data collection, conversion, and migration. Kettle supports not only various relational databases and NoSQL databases such as HBase and MongoDB, but also niche data sources such as Microsoft Office Excel and Access. With extensions and plug-ins, Kettle can support more data sources.

For more information, visit [Kettle](#).

Preparations

Before you use Kettle, complete the following steps:

- **Install Kettle.**
- **Create a database and a table in an AnalyticDB for MySQL cluster.**

For more information about how to create databases and tables, see [CREATE DATABASE](#) and [CREATE TABLE](#).

- **Add the IP address of the device on which Kettle is installed to the whitelist of the AnalyticDB for MySQL cluster. For more information, see [Configure a whitelist](#).**

Procedure

1. **Start Kettle, and choose File > New > Conversion to create a conversion task.**
2. **Choose File > New > Database Connection to create a database connection for the conversion task.**

Parameter	Description
Connection Name	The name of the connection. We recommend that you choose an identifiable name, which facilitates subsequent management.
Connection Type	The type of the connection. Select MySQL from the drop-down list.

Parameter	Description
Access	The access mode of the connection. Select Native (JDBC).
Host Name	The endpoint of the AnalyticDB for MySQL cluster to which you want to connect. You can view the connection information about the cluster on the Cluster Information page of the AnalyticDB for MySQL console.
Database Name	The name of the database in the AnalyticDB for MySQL cluster.
Port Number	The port number of the AnalyticDB for MySQL cluster endpoint. The default port number is 3306.
User Name	The account used to connect to the AnalyticDB for MySQL cluster. You can use either of the following accounts: <ul style="list-style-type: none"> Privileged accounts. Standard accounts that have the permissions to connect to the cluster.
Password	The password of the account used to connect to the AnalyticDB for MySQL cluster.

**Note:**

Do not select Use Result Streaming Cursor when you configure the parameters.

- After you configure the preceding parameters, click Test. In the Database Connection Test dialog box that appears, follow the prompts to verify if the connection to the database is successful. After the connection passes the test, click OK.
- In the left-side navigation pane of Kettle, click the Core objects tab, and choose Input > Excel Input. Drag and drop Excel Input to the workspace.

5. Double-click Excel Input in the workspace. In the Excel Input dialog box that appears, click Browse and Add to add an Excel file to Selected Files.

Configure parameters on tabs such as Worksheet, Content, and Field as required, and click Preview to check whether the entered data meets your requirements.

6. In the left-side navigation pane of Kettle, click the Core objects tab, and choose Output > Table Output. Drag and drop Table Output to the workspace.

7. Add a connection line from Excel Input to Table Output.

8. Double-click Table Output in the workspace. In the Table Output dialog box that appears, configure the parameters as required.

- Target Schema: Enter the name of the AnalyticDB for MySQL database.
- Target Table: Enter the name of the table in the AnalyticDB for MySQL database.
- Select Specify database fields.
- Select Use batch update for inserts.

On the Database fields tab of the Table Output dialog box, click Get fields and Enter field mapping to map columns in the Excel file to those in the AnalyticDB for MySQL table.

9. Click the white arrow to perform the conversion. During this period, you can check the operations log and operating status.

After the data in the Excel file is synchronized to the AnalyticDB for MySQL database, you can use AnalyticDB for MySQL to analyze the data.

8.11 SQL manual

8.11.1 Data types

Data types supported by AnalyticDB for MySQL

Keyword	Data type	Valid value
BOOLEAN	The Boolean type	Valid values: 0 and 1. A value of 0 indicates false. A value of 1 indicates true. A BOOLEAN value is 1 bit in size.
TINYINT	The tiny integer type	Valid values: -128 to 127. A TINYINT value is 1 byte in size.
SMALLINT	The small integer type	Valid values: -32768 to 32767. A SMALLINT value is 2 bytes in size.
INT or INTEGER	The integer type	Valid values: -2147483648 to 2147483647. An INT value is 4 bytes in size.
BIGINT	The big integer type	Valid values: -9223372036854775808 to 9223372036854775807. A BIGINT value is 8 bytes in size.
FLOAT	The single-precision floating-point type	Valid values: -3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38. The FLOAT type follows the IEEE standard. A FLOAT value is 4 bytes in size.

Keyword	Data type	Valid value
DOUBLE	The double-precision floating-point type	Valid values: -1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+308. The DOUBLE type follows the IEEE standard. A DOUBLE value is 8 bytes in size.
DECIMAL(M,D)	N/A	M is the maximum precision, and its value range is 1 to 1000. D is the decimal scale. The value of D must be less than or equal to that of M, that is, $D \leq M$.
VARCHAR	The variable-length string type	A VARCHAR value can be up to 16 MB in size. You do not need to specify the size when using VARCHAR.
DATE	The date type	Valid values: '0001-01-01' to '9999-12-31'. A DATE value is in the 'YYYY-MM-DD' format and is 4 bytes in size.
TIME	The time type	Valid values: '00:00:00' to '23:59:59'. A TIME value is in the 'HH:MM:SS' format and is 8 bytes in size.

Keyword	Data type	Valid value
DATETIME	The date and time type	<p>Valid values: '0001-01-01 00:00:00.000' to '9999-12-31 23:59:59.999'. A DATETIME value is in the 'YYYY-MM-DD HH:MM:SS' format. It is 8 bytes in size and in the UTC format.</p> <p> Note: DATETIME uses UTC time. You cannot change the time zone for DATETIME values.</p>
TIMESTAMP	The timestamp type	<p>Valid values: '0001-01-01 00:00:00.000' to '9999-12-31 23:59:59.999'. A TIMESTAMP value is in the 'YYYY-MM-DD HH:MM:SS' format. It is 4 bytes in size and in the UTC format.</p> <p> Note: TIMESTAMP uses the time zone of the database system by default. You can specify the time zone for each session.</p>

Comparison with MySQL data types

AnalyticDB for MySQL	MySQL	Difference
BOOLEAN	BOOL and BOOLEAN	No difference.
TINYINT	TINYINT	No difference.
SMALLINT	SMALLINT	No difference.
INT and INTEGER	INT and INTEGER	No difference.
BIGINT	BIGINT	No difference.

AnalyticDB for MySQL	MySQL	Difference
FLOAT	FLOAT[(M,D)]	No difference.
DOUBLE	DOUBLE[(M,D)]	No difference.
DECIMAL	DECIMAL	AnalyticDB for MySQL supports a maximum precision of 1000 digits, while MySQL supports a maximum precision of only 65 digits.
VARCHAR	VARCHAR	The VARCHAR type in AnalyticDB for MySQL corresponds to the CHAR, VARCHAR, TEXT, MEDIUMTEXT, and LONGTEXT types in MySQL.
DATE	DATE	MySQL supports the value 0000-00-00, while AnalyticDB for MySQL automatically converts the value 0000-00-00 to NULL.
TIME	TIME	AnalyticDB for MySQL is precise to the millisecond. MySQL supports custom precision levels.
DATETIME	DATETIME	MySQL supports the value 0000-00-00 00:00:00, while AnalyticDB for MySQL automatically converts the value 0000-00-00 00:00:00 to NULL. AnalyticDB for MySQL is precise to the millisecond. MySQL supports custom precision levels.
TIMESTAMP	TIMESTAMP	AnalyticDB for MySQL is precise to the millisecond. MySQL supports custom precision levels.

8.11.2 Data definition statements

8.11.2.1 CREATE DATABASE

Create a database



Note:

You can create up to 256 databases in each AnalyticDB for MySQL cluster.

Syntax

```
CREATE DATABASE [IF NOT EXISTS] db_name
```

Parameter

db_name: the name of the database. The database name can be up to 64 characters in length and can contain letters, digits, and underscores (_). It must start with a lowercase letter and cannot contain two or more consecutive underscores (_).



Note:

The database name cannot be `analyticdb`. This name is reserved for built-in databases.

Example

```
CREATE DATABASE adb_demo;
```

Use a database

You can execute the `USE db_name` statement to use a database after it is created.

Syntax

```
USE db_name
```

Example

```
use adb_demo;
show tables;
+-----+
|Tables_in_adb_demo|
+-----+
|customer          |
```

|test_table |

8.11.2.2 CREATE TABLE

You can execute the **CREATE TABLE** statement to create a table in AnalyticDB for MySQL.

Syntax

```
CREATE TABLE [IF NOT EXISTS] table_name
({column_name column_type [column_attributes] [ column_constraints ] [
COMMENT 'string']
| table_constraints}
[, ... ] )
table_attribute
[partition_options]
[AS] query_expression
COMMENT 'string'

column_attributes:
    [DEFAULT default_expr]
    [AUTO_INCREMENT]

column_constraints:
    [{NOT NULL|NULL} ]
    [PRIMARY KEY]

table_constraints:
    [{INDEX|KEY} [index_name] (column_name,...)]
    [PRIMARY KEY [index_name] (column_name,...)]
    [CLUSTERED KEY [index_name] (column_name,...)]

table_attribute:
    DISTRIBUTED BY HASH(column_name,...) | DISTRIBUTED BY BROADCAST

partition_options:
    PARTITION BY
        {VALUE(column_name) | VALUE(date_format(column_name, ?))}
```

[LIFECYCLE N]

Parameters

Parameter	Description
table_name	<p>The name of the table.</p> <p>The table name must be 1 to 127 characters in length and can contain letters, digits, and underscores (_). The table name must start with a letter or underscore (_).</p> <p>Specify the table name in the db_name .table_name format to distinguish tables that have the same name across different databases.</p>
column_name	<p>The name of the column.</p> <p>The column name must be 1 to 127 characters in length and can contain letters, digits, and underscores (_). The table name must start with a letter or underscore (_).</p>
column_type	<p>The data type of the column to be added.</p> <p>For more information about the data types supported by AnalyticDB for MySQL, see Data types.</p>

Parameter	Description
column_attributes	<ul style="list-style-type: none"> • DEFAULT default_expr: the default value of the column. Enter an expression without variables, such as current_timestamp. If this parameter is not specified, the default value is NULL. • AUTO_INCREMENT: optional. Specifies whether the column is an auto-increment column. The data type of an auto-increment column must be BIGINT. This is because AnalyticDB for MySQL provides unique values for an auto-increment column, but these values are not incremented in sequence.
column_constraints	<ul style="list-style-type: none"> • NOT NULL NULL: specifies whether the column accepts the NULL value. A value of NOT NULL indicates that the column does not accept the NULL value. A value of NULL indicates that the column accepts the NULL value. Default value: NULL. • PRIMARY KEY: the primary key of the column. You can define multiple primary keys in the PRIMARY KEY(column_name [, ...]) format.
table_constraints	<p>INDEX KEY: the inverted index.</p> <p>AnalyticDB for MySQL automatically creates indexes for whole tables. You do not need to manually create an index.</p>

Parameter	Description
PRIMARY KEY	<p>The index for the primary keys.</p> <ul style="list-style-type: none"> • Only tables with primary keys support the DELETE and UPDATE operations. • The primary keys must include the partition key. We recommend that you put the partition key before the primary key combination.
CLUSTERED KEY	<p>The clustered index. It defines the columns used for sorting data in the table. The logical order of the key values in the clustered index determines the physical order of the corresponding rows in the table. You can add only one clustered index for each table.</p> <p>For example, clustered key <code>col5_col6_cls_index(col5,col6)</code> specifies the <code>col5 col6</code> clustered index. <code>col5 col6</code> and <code>col6 col5</code> are different clustered indexes.</p>
DISTRIBUTED BY HASH(<code>column_name</code> ,...)	<p>The distributed key of the fact table. The data of the table is distributed based on the hash value of the columns specified by <code>column_name</code>.</p> <p>AnalyticDB for MySQL allows you to select multiple fields as the distributed key.</p>
DISTRIBUTED BY BROADCAST	<p>The dimension table. The dimension table is stored on each node of a cluster. For performance reasons, we recommend that you do not store large amounts of data in the dimension table.</p>

Parameter	Description
partition_options	<p>The options for fact table partitions.</p> <p>AnalyticDB for MySQL manages the lifecycle of tables based on the <code>LIFECYCLE N</code> parameter. That is, AnalyticDB for MySQL only stores <code>N</code> number of partitions. All other partitions are deleted.</p> <p>For example, <code>PARTITION BY VALUE (column_name)</code> specifies that the table is partitioned based on the column specified by <code>column_name</code>. <code>PARTITION BY VALUE (DATE_FORMAT (column_name , '%Y%m%d'))</code> specifies that the table is partitioned based on the column specified by <code>column_name</code> after the column is formatted to a date format such as <code>20190101</code>. <code>LIFECYCLE 365</code> specifies that a maximum of 365 partitions can be retained on each node. That is, only data of the last 365 days is stored. When you write data on the 366th day, the data from the first day is deleted.</p>

Precautions

- AnalyticDB for MySQL clusters use UTF-8 encoding, which is equivalent to utf8mb4 in MySQL. AnalyticDB for MySQL does not support changing encoding formats.
- You can create up to `256 × Number of node groups` tables in an AnalyticDB for MySQL cluster.

Examples

- **Create a test table.**

```
create table test (
  id bigint auto_increment,
  name varchar,
  value int,
  ts timestamp
)
  DISTRIBUTED BY HASH(id)
```

The test table is a fact table. The `id` column is an auto-increment column. The distributed key is `id`. The data of the table is distributed based on the hash value of the `id` column.

- **Create a customer table.**

```
CREATE TABLE customer (
  customer_id bigint NOT NULL COMMENT 'Customer ID',
  customer_name varchar NOT NULL COMMENT 'Customer name',
  phone_num bigint NOT NULL COMMENT 'Phone number',
  city_name varchar NOT NULL COMMENT 'City',
  sex int NOT NULL COMMENT 'Gender',
  id_number varchar NOT NULL COMMENT 'ID card number',
  home_address varchar NOT NULL COMMENT 'Home address',
  office_address varchar NOT NULL COMMENT 'Office address',
  age int NOT NULL COMMENT 'Age',
  login_time timestamp NOT NULL COMMENT 'Logon time',
  PRIMARY KEY (login_time,customer_id,phone_num)
) DISTRIBUTED BY HASH(customer_id)
PARTITION BY VALUE (DATE_FORMAT(login_time, '%Y%m%d')) LIFECYCLE 30
COMMENT 'Customer information table';
```

The customer table is a fact table. In the table, the distributed key is `customer_id`. The partition key is `login_time`. `login_time`, `customer_id`, and `phone_num` form the primary key combination.

8.11.2.3 ALTER TABLE

You can execute the ALTER TABLE statement to modify a table.



Note:

You cannot modify the data type of a column in AnalyticDB for MySQL.

Syntax

```
ALTER TABLE table_name
  ADD COLUMN (column_name column_definition,...)
  | ADD {INDEX|KEY} [index_name] (column_name,...)
  | ADD CLUSTERED [INDEX|KEY] [index_name] (column_name,...)
  | DROP COLUMN column_name
  | DROP {INDEX|KEY} index_name
  | DROP CLUSTERED [INDEX|KEY] index_name
```

```

| MODIFY COLUMN column_name column_definition
| RENAME new_table_name
| TRUNCATE PARTITION {partition_names | ALL}

```

Add a column

Syntax

```
ALTER TABLE db_name.table_name ADD column_name data_type;
```

Example

Add the province column of the VARCHAR type to the customer table.

```
ALTER TABLE adb_demo.customer ADD COLUMN province varchar comment 'Province';
```

Delete a column

Syntax

```
ALTER TABLE db_name.table_name DROP column_name data_type;
```

Example

Delete the province column of the VARCHAR type from the customer table.

Modify the comment of a column

Syntax

```
ALTER TABLE db_name.table_name MODIFY COLUMN column_name data_type
comment 'new_comment';
```

Example

Modify the COMMENT of the province column to the province where the customer locates in the customer table.

```
ALTER TABLE adb_demo.customer MODIFY COLUMN province varchar comment 'Province where the customer locates';
```

Modify the value constraint for a column from NOT NULL to NULL



Note:

You can modify only the value constraint from NOT NULL to NULL, but not from NULL to NOT NULL.

Syntax

```
ALTER TABLE db_name.table_name MODIFY COLUMN column_name data_type {
NULL}
```

Example

Modify the value of the province column to NULL in the customer table.

```
ALTER TABLE adb_demo.customer MODIFY COLUMN province varchar NULL;
```

Modify the default value of a column

Syntax

```
ALTER TABLE db_name.table_name MODIFY COLUMN column_name data_type
DEFAULT 'default'
```

Example

Modify the default value of the sex column to 0 (indicating male) in the customer table.

```
ALTER TABLE adb_demo.customer MODIFY COLUMN sex int(11) NOT NULL
DEFAULT 0;
```

Add a clustered index



Note:

You can add only one clustered index for each table.

Syntax

```
ALTER TABLE db_name.table_name ADD CLUSTERED KEY index_name(column_name1, column_name2);
```

```
ALTER TABLE db_name.table_name ADD CLUSTERED KEY index_name(column_name1);
```

Example

Create a clustered index for the `customer_id` and `id_number` columns in the `customer` table.

```
ALTER TABLE adb_demo.customer ADD CLUSTERED KEY c_k(customer_id, id_number);
```

Delete a clustered index

Syntax

```
ALTER TABLE db_name.table_name DROP CLUSTERED KEY index_name;
```

Example

Delete the clustered index of the `customer` table.

```
ALTER TABLE adb_demo.customer DROP CLUSTERED KEY c_k;
```

8.11.2.4 CREATE VIEW

You can execute the `CREATE VIEW` statement to create a view.

Syntax

```
CREATE VIEW view_name AS select_stmt
```

Parameters

- `view_name`: **the name of the view to create. You can prefix the view name with the database name to distinguish views with the same name in different databases.**
- `select_stmt`: **the data source of the view.**

Example

Create a view named `v` and set its data source to the `customer` table.

```
CREATE VIEW adb_demo.v AS SELECT * FROM customer;
```

8.11.2.5 DROP DATABASE

You can execute the `DROP DATABASE` statement to delete a database.

Syntax

```
DROP DATABASE db_name;
```



Note:

Before you delete a database, you must first delete the tables in the database.

Example

```

use adb_demo2;
+-----+
show tables;
+-----+
| Tables_in_adb_demo2 |
+-----+
| test2                |
+-----+
drop table test2;
drop database adb_demo2;

```

8.11.2.6 DROP TABLE

You can execute the DROP TABLE statement to delete a table.

Syntax

```
DROP TABLE db_name.table_name;
```

**Note:**

When you execute this statement, both the table data and schema are deleted.

Example

```
DROP TABLE adb_demo.customer;
```

8.11.2.7 DROP VIEW

You can execute the DROP VIEW statement to delete a view.

Syntax

```
DROP VIEW [IF EXISTS] view_name, [, view_name] ...
```

Parameters

view_name: the name of the view to delete. You can prefix the view name with the database name to distinguish views with the same name in different databases.

Example

Delete the view named v.

```
DROP VIEW v;
```

8.11.3 Data manipulation statements

8.11.3.1 INSERT INTO

You can execute the `INSERT INTO` statement to insert data to a table. If a primary key is the same as an existing one in the records, the new record is not inserted. This statement is equivalent to the `INSERT IGNORE INTO` statement.

Syntax

```
INSERT [IGNORE]
      INTO table_name
      [( column_name [, ...] )]
      [VALUES]
      [(value_list[, ...])]
      [query];
```

Parameters

- **`IGNORE`:** optional. Specifies that a newly inserted record is ignored if the primary key of the record is the same as that of an existing record.
- **`column_name`:** optional. The name of the column.
- **`query`:** inserts one or more records queried from another table to this table.

Precautions

If the column names are not specified, the sequence of the columns in the data to be inserted must be the same as that specified in the `CREATE TABLE` statement.

Example

Create the customer and courses tables.

```
CREATE TABLE customer (
  customer_id bigint NOT NULL COMMENT 'Customer ID',
  customer_name varchar NOT NULL COMMENT 'Customer name',
  phone_num bigint NOT NULL COMMENT 'Phone number',
  city_name varchar NOT NULL COMMENT 'City',
  sex int NOT NULL COMMENT 'Gender',
  id_number varchar NOT NULL COMMENT 'ID card number',
  home_address varchar NOT NULL COMMENT 'Home address',
  office_address varchar NOT NULL COMMENT 'Office address',
  age int NOT NULL COMMENT 'Age',
  login_time timestamp NOT NULL COMMENT 'Logon time',
  PRIMARY KEY (login_time,customer_id,phone_num)
)DISTRIBUTED BY HASH(customer_id)
```

```
PARTITION BY VALUE(DATE_FORMAT(login_time, '%Y%m%d')) LIFECYCLE 30
COMMENT 'Customer information table';
```

```
CREATE TABLE courses(
id bigint AUTO_INCREMENT PRIMARY KEY,
name varchar(20) NOT NULL,
grade varchar(20) default 'Grade 3',
submission_date timestamp)DISTRIBUTED BY HASH(id)
```

- **Insert a record to the customer table.**

```
INSERT INTO customer(customer_id,customer_name,phone_num,city_name,
sex,id_number,home_address,office_address,age,login_time)
values
(002367,'Alan','13678973421','Hangzhou',0,'987300','West Lake','
Cloud Town',23,'2018-03-02 10:00:00');
```

- **Insert multiple records to the customer table.**

```
INSERT INTO customer(customer_id,customer_name,phone_num,city_name,
sex,id_number,home_address,office_address,age,login_time)
values
(002367,'Tom','13678973421','Hangzhou',0,'987300','West Lake','
Cloud Town',23,'2018-03-02 10:00:00'),(002368,'Alex','13878971234','
Hangzhou',0,'987300','West Lake','Cloud Town',28,'2018-08-01 11:00
:00'),(002369,'Eric','13968075284','Hangzhou',1,'987300','West Lake
','Cloud Town',35,'2018-09-12 08:11:00');
```

- **Insert multiple records to the customer table and do not specify the column names.**

```
INSERT INTO customer
values(002367,'Tom','13678973421','Hangzhou',0,'987300','West Lake
','Cloud Town',23,'2018-03-02 10:00:00'),(002368,'Alex','1387897123
4','Hangzhou',0,'987300','West Lake','Cloud Town',28,'2018-08-01 11
:00:00'),(002369,'Eric','13968075284','Hangzhou',1,'987300','West
Lake','Cloud Town',35,'2018-09-12 08:11:00');
```

- **Insert a record to the courses table.**

```
insert into courses (name,submission_date) values("Jams",NOW());
```

8.11.3.2 REPLACE INTO

You can execute the `REPLACE INTO` statement to insert data to a table by overwriting the existing data in real time. The system first checks whether the primary key of a record to be inserted is the same as that of an existing record. If they are the same, the system deletes the existing record and inserts the new record. If they are not the same, the system inserts the new record.

Syntax

```
REPLACE INTO table_name [(column_name,...)] VALUES ({Constant|NULL|
DEFAULT},...),(...),...
```

Example

- **Insert a record to the customer table by specifying the column names in the REPLACE INTO statement.**

```
REPLACE INTO customer(customer_id,customer_name,phone_num,city_name
,sex,id_number,home_address,office_address,age,login_time) values (
002367,'Alan','13678973421','Hangzhou',0,'987300','West Lake','Cloud
Town',23,'2018-03-02 10:00:00');
```

- **Insert multiple records to the customer table and do not specify the column names.**

```
REPLACE INTO customer values(002367,'Tom','13678973421','Hangzhou',0
,'987300','West Lake','Cloud Town',23,'2018-03-02 10:00:00'),(002368
,'Alex','13878971234','Hangzhou',0,'987300','West Lake','Cloud Town
',28,'2018-08-01 11:00:00'),(002369,'Eric','13968075284','Hangzhou',
1,'987300','West Lake','Cloud Town',35,'2018-09-12 08:11:00');
```

8.11.3.3 INSERT SELECT FROM

You can execute the `INSERT SELECT FROM` statement to copy records from one table to another.

Syntax

```
INSERT INTO table_name [( column_name [, ...] )]query;
```

Parameters

- `column_name`: **the name of the column. If you want to copy data in only some columns of the source table to the target table, the columns specified in the SELECT clause must have the same sequence and data types as those specified in the INSERT clause.**
- `query`: **the SELECT FROM TABLE or SELECT FROM VIEW statement.**

Example

- **Copy data only in the specified columns of the customer table to the new_customer table by specifying the column names.**

```
INSERT INTO new_customer (customer_id, customer_name, phone_num)
SELECT customer_id, customer_name, phone_num FROM customer
```

```
WHERE customer.customer_name = 'Alan';
```

- **Copy data in all columns of the customer table to the new_customer table and do not specify the column names.**

```
INSERT INTO new_customer
  SELECT customer_id, customer_name, phone_num, city_name, sex, id_number
  , home_address, office_address, age, login_time)
  FROM customer
  WHERE customer.customer_name = 'Alan';
```

8.11.3.4 REPLACE SELECT FROM

You can execute the `REPLACE SELECT FROM` statement to copy records from one table to another by overwriting existing data in real time. The system first checks whether the primary key of a record to be inserted is the same as that of an existing record. If they are the same, the system deletes the existing record and inserts the new record. If they are not the same, the system inserts the new record.

Syntax

```
REPLACE INTO table_name [(column_name,...)]query;
```

Parameter

- **query:** the `SELECT FROM TABLE` or `SELECT FROM VIEW` statement.
- **column_name:** the name of the column. If you want to copy data in only some columns of the source table to the target table, the columns specified in the `SELECT` clause must have the same sequence and data types as those specified in the `REPLACE` clause.

Precautions

The target table to which the records are inserted must exist before you execute the `REPLACE SELECT FROM` statement.

Example

Copy data only in the specified columns of the customer table to the new_customer table by specifying the column names.

```
REPLACE INTO new_customer (customer_id, customer_name, phone_num)
  SELECT customer_id, customer_name, phone_num
  FROM customer
```

```
WHERE customer.customer_name = 'Alan';
```

8.11.3.5 INSERT OVERWRITE INTO SELECT

You can execute the `INSERT OVERWRITE INTO SELECT` statement to insert multiple records to a table at a time.

Syntax

```
INSERT OVERWRITE INTO table_name [(column_name,...)]  
SELECT select_statement FROM from_statement
```

Precautions

- The target table to which the records are inserted must exist before you execute the `INSERT OVERWRITE INTO SELECT` statement.
- The existing data in the target table will not change before the `INSERT OVERWRITE INTO SELECT` statement is completed. The system writes data to the target table after the `INSERT OVERWRITE INTO SELECT` statement is completed.
- If you execute the `INSERT OVERWRITE INTO SELECT` statement to insert a primary key that is the same as an existing one in the records, the new record is not inserted.

8.11.3.6 UPDATE

You can execute the `UPDATE` statement to update data in a table.

Syntax

```
UPDATE table_reference  
SET assignment_list  
[WHERE where_condition]  
[ORDER BY ...]
```

Precautions

The table on which you execute the `UPDATE` statement must have a primary key.

Example

Change the name of the customer who has the `customer_id = '2369'` attribute to Claire in the customer table.

```
update customer set customer_name = 'Claire' where customer_id = '2369';
```

8.11.3.7 DELETE

You can execute the DELETE statement to delete records from a table.

Syntax

```
DELETE FROM table_name[ WHERE condition ]
```

Precautions

- **The table on which you execute the DELETE statement must have a primary key.**
- **You cannot use the alias of a table to execute the DELETE statement.**
- **If you need to delete all records of a table or all partitions, we recommend that you use the TRUNCATE TABLE or TRUNCATE PARTITION statement, instead of the DELETE statement.**

Example

- **Delete the record where name is Alex from the customer table.**

```
DELETE FROM customer WHERE customer_name='Alex';
```

- **Delete the records where age is less than 18 from the customer table.**

```
DELETE FROM customer WHERE age<18;
```

8.11.3.8 TRUNCATE TABLE

You can execute the `TRUNCATE TABLE` statement to clear the data of a table or specified partitions in a table.

Syntax

- **Clear the data of a table.**

```
TRUNCATE TABLE db_name.table_name;
```

- **Clear the data of specified partitions in a table.**

```
TRUNCATE TABLE db_name.table_name PARTITION partition_name;
```

- **The data type of partition names is BIGINT. You can execute the following SQL statement to obtain the names of all partitions in a table:**

```
select partition_name from information_schema.partitions where  
table_name = 'your_table_name' order by partition_name desc limit  
100;
```

Precautions

When you execute the `TRUNCATE TABLE` statement to clear the data of a table, the table schema is not deleted.

Example

- **Clear the data of the customer table.**

```
TRUNCATE TABLE adb_demo.customer;
```

- **Clear the data of specified partitions in the customer table.**

```
TRUNCATE TABLE adb_demo.customer partition 20170103,20170104,
20170108
```

8.11.3.9 KILL PROCESS

You can execute the `KILL PROCESS` statement to terminate a running process.

Syntax

```
KILL PROCESS process_id
```

Parameters

`process_id`: the ID of the process to terminate. You can query the process ID by executing the `SHOW PROCESSLIST` statement and checking the `ProcessId` field.

Permission

- You can execute the `KILL PROCESS` statement to terminate the running processes under your account.
- The privileged account can grant the `PROCESS` permission to a standard account by using the `GRANT` statement. The standard account can then terminate the running processes of all accounts in the cluster.

```
GRANT process on *.* to account_name;
```

8.11.3.10 SHOW PROCESSLIST

You can execute the `SHOW PROCESSLIST` statement to view the running processes.



Note:

You can also execute the `INFORMATION_SCHEMA PROCESSLIST` statement to view the running processes.

Syntax

```
SHOW [FULL] PROCESSLIST
```

Response parameters

The following response parameters are returned after you execute the `SHOW FULL PROCESSLIST` or `SHOW PROCESSLIST` statement:

- **Id:** the ID of the process.
- **ProcessId:** the unique ID of the task. This parameter is required when you execute the `KILL PROCESS` statement.
- **User:** the current account.
- **Host:** the hostname of the client that sends the `SHOW PROCESSLIST` statement, which consists of the IP address and port number.
- **DB:** the database to which the process is connected.
- **Command:** the command that is executed in the current connection. The command types include `sleep`, `query`, and `connect`.
- **Time:** the time when the `Command` is executed. **Unit:** seconds.
- **State:** the execution status of the SQL statement in the current connection.
- **Info:** the SQL statement.



Note:

If you do not specify the `FULL` keyword, you can view only the first 100 characters of the `Info` field in each record.

Permission

- You can execute the `SHOW PROCESSLIST` statement to view the processes running under your account.

- A privileged account can grant the PROCESS permission to a standard account by using the GRANT statement. The standard account can then view the running processes of all accounts in the cluster.

```
GRANT process on *.* to account_name;
```

8.11.4 SELECT

8.11.4.1 Syntax

You can use the SELECT statement to query data from one or more tables. The syntax is as follows:

```
[ WITH with_query [, ...] ]
SELECT
[ ALL | DISTINCT ] select_expr [, ...]
[ FROM table_reference [, ...] ]
[ WHERE condition ]
[ GROUP BY [ ALL | DISTINCT ] grouping_element [, ...] ]
[ HAVING condition]
[ WINDOW window_name AS (window_spec) [, window_name AS (window_spec
)] ...]
[ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] select ]
[ ORDER BY {column_name | expr | position} [ASC | DESC], ... [WITH
ROLLUP]]
[ LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

- **table_reference**: the data source from which data is queried. The source can be a table, view, associative table, or subquery.
- Table names and column names are not case-sensitive.
- If a table name or column name contains keywords or space characters, you can quote the table name or column name with backticks (``).

WHERE

You can enter a BOOLEAN expression in the WHERE keyword to query data records that meet the specified condition. For example, you can execute the following statement to query the information of the customer whose customer_id is 2368.

```
SELECT * FROM CUSTOMER where customer_id=2368;
```

ALL and DISTINCT

You can use the ALL and DISTINCT keywords to specify whether duplicate rows are retained in the query result. The default value is ALL, indicating that all rows are

returned. The **DISTINCT** keyword indicates that duplicate rows are deleted in the query result.

```
SELECT col1, col2 FROM t1;SELECT DISTINCT col1, col2 FROM t1;
```

The following methods show you how to use other keywords.

8.11.4.2 WITH

This topic describes how to use the **WITH** clause in a **SELECT** statement.

You can use the **WITH** clause to define subqueries by using common table expressions (CTEs) in a **SELECT** statement. The subqueries defined in the **WITH** clause can be referenced in the **SELECT** statement. The **WITH** clause can be used to flatten nested queries or simplify subqueries. Each subquery is performed only once in the **SELECT** statement. This improves the query performance.



Note:

- A CTE is a named temporary result set, and is valid only in a single SQL statement, such as a **SELECT**, **INSERT**, and **DELETE** statement.
- A CTE is valid only during the period when the SQL statement is executing.

Precautions

- A CTE can be followed by a SQL statement, such as a **SELECT**, **INSERT**, or **UPDATE** statement, or other CTEs in the same **WITH** clause. Separate multiple CTEs with commas (,). Otherwise, the CTEs will not take effect.
- The paging feature is not supported in a CTE.

Use the WITH clause

- The following queries are equivalent:

```
SELECT a, b FROM (SELECT a, MAX(b) AS b FROM t GROUP BY a) AS x;
```

```
WITH x AS (SELECT a, MAX(b) AS b FROM t GROUP BY a) SELECT a, b FROM x;
```

- The **WITH** clause can be used to define multiple subqueries.

```
WITH
  t1 AS (SELECT a, MAX(b) AS b FROM x GROUP BY a),
  t2 AS (SELECT a, AVG(d) AS d FROM y GROUP BY a)
SELECT t1.*, t2.*
```

```
FROM t1 JOIN t2 ON t1.a = t2.a;
```

- **After a subquery is defined, other subqueries following this subquery in the same WITH clause can reference this subquery.**

```
WITH
  x AS (SELECT a FROM t),
  y AS (SELECT a AS b FROM x),
  z AS (SELECT b AS c FROM y)
SELECT c FROM z;
```

8.11.4.3 GROUP BY

You can use the `GROUP BY` clause to group the query result. You can use the `GROUPING SETS`, `CUBE`, and `ROLLUP` options in the `GROUP BY` clause to display the grouping result in different forms.

Syntax

```
GROUP BY expression [, ...]
```

- **GROUPING SETS**

The `GROUPING SETS` option is used to specify multiple `GROUP BY` conditions for one query, which is equal to the `UNION` of multiple `GROUP BY` clauses.

```
SELECT origin_state, origin_zip, destination_state, sum(package_weight)
FROM shipping
GROUP BY GROUPING SETS (
  (origin_state),
  (origin_state, origin_zip),
  (destination_state));
```

The preceding statement is equal to the following statement:

```
SELECT origin_state, NULL, NULL, sum(package_weight)
FROM shipping GROUP BY origin_state
UNION ALL
SELECT origin_state, origin_zip, NULL, sum(package_weight)
FROM shipping GROUP BY origin_state, origin_zip
UNION ALL
SELECT NULL, NULL, destination_state, sum(package_weight)
FROM shipping GROUP BY destination_state;
```

- **CUBE**

The `CUBE` option is used to list all possible grouping sets.

```
SELECT origin_state, destination_state, sum(package_weight)
FROM shipping
```

```
GROUP BY origin_state, destination_state WITH CUBE
```

The preceding statement is equal to the following statement:

```
SELECT origin_state, destination_state, sum(package_weight)
FROM shipping
GROUP BY GROUPING SETS (
    (origin_state, destination_state),
    (origin_state),
    (destination_state),
    ())
```

- **ROLLUP**

The **ROLLUP** option is used to list the grouping sets in a hierarchical manner.

```
SELECT origin_state, origin_zip, sum(package_weight)
FROM shipping
GROUP BY ROLLUP (origin_state, origin_zip)
```

The preceding statement is equal to the following statement:

```
SELECT origin_state, origin_zip, sum(package_weight)
FROM shipping
GROUP BY GROUPING SETS ((origin_state, origin_zip), (origin_state),
    ())
```

Precautions

- You must use standard aggregate functions such as **SUM**, **AVG**, and **COUNT** to specify the columns that are not used for grouping. Otherwise, the **GROUP BY** clause does not take effect.
- The **GROUP BY** clause must contain all the columns and non-aggregate expressions specified for the **SELECT** statement.

Example

The following statement contains two aggregate expressions. The first aggregate expression uses the **SUM** function, and the second uses the **COUNT** function. The **LISTID** and **EVENTID** columns are the columns that are used for grouping.

```
select listid, eventid, sum(pricepaid) as revenue,
count(qtysold) as numtix
from sales
group by listid, eventid
order by 3, 4, 2, 1
limit 5;
```

listid	eventid	revenue	numtix
89397	47	20.00	1
106590	76	20.00	1
124683	393	20.00	1
103037	403	20.00	1

```
47685 |      429 |    20.00 |      1
(5 rows)
```

You can also use the sequence numbers to reference columns in the **GROUP BY** clause.

For example, you can modify the preceding statement as follows:

```
select listid, eventid, sum(pricepaid) as revenue,
count(qtysold) as numtix
from sales
group by 1,2
order by 3, 4, 2, 1
limit 5;
listid | eventid | revenue | numtix
-----+-----+-----+-----
89397  |      47 |    20.00 |      1
106590 |      76 |    20.00 |      1
124683 |     393 |    20.00 |      1
103037 |     403 |    20.00 |      1
147685 |     429 |    20.00 |      1
```

8.11.4.4 HAVING

You can use the **HAVING** clause with the **GROUP BY** clause and aggregate functions to display groups that only meet certain conditions after grouping and aggregation.

Syntax

```
[ HAVING condition ]
```

Precautions

- The columns that you reference in the **HAVING** clause must be used for grouping or contain an aggregate function.
- The **HAVING** clause must be used together with aggregate functions and the **GROUP BY** clause to display groups that only meet certain conditions after grouping based on **GROUP BY**.

Example

Group records in the CUSTOMER table and query the records whose account balance is greater than the specified value.

```
SELECT count(*), mktsegment, nationkey,
CAST(sum(acctbal) AS bigint) AS totalbal
FROM customer
GROUP BY mktsegment, nationkey
HAVING sum(acctbal) > 5700000
ORDER BY totalbal DESC;
_col0 | mktsegment | nationkey | totalbal
-----+-----+-----+-----
1272  | AUTOMOBILE |      19  | 5856939
```

1253	FURNITURE	14	5794887
1248	FURNITURE	9	5784628
1243	FURNITURE	12	5757371
1231	HOUSEHOLD	3	5753216
1251	MACHINERY	2	5719140
1247	FURNITURE	8	5701952

8.11.4.5 JOIN

Syntax

```

join_table:
    table_reference [INNER] JOIN table_factor [join_condition]
    | table_reference {LEFT|RIGHT|FULL} [OUTER] JOIN table_reference
join_condition
    | table_reference CROSS JOIN table_reference [join_condition])
table_reference:
    table_factor
    | join_table

table_factor:
    tbl_name [alias]
    | table_subquery alias
    | ( table_references )

join_condition:
    ON expression

```

Example

```

select catgroup1, sold, unsold
from
(select catgroup, sum(qtysold) as sold
from category c, event e, sales s
where c.catid = e.catid and e.eventid = s.eventid
group by catgroup) as a(catgroup1, sold)
join
(select catgroup, sum(numtickets)-sum(qtysold) as unsold
from category c, event e, sales s, listing l
where c.catid = e.catid and e.eventid = s.eventid
and s.listid = l.listid
group by catgroup) as b(catgroup2, unsold)
on a.catgroup1 = b.catgroup2
order by 1;

```

8.11.4.6 LIMIT

You can use the LIMIT clause to specify the maximum number of rows to return in a query. Typically, you need to specify one or two numbers in the LIMIT clause.

The first number specifies the number of rows to skip from the beginning, and the second number specifies the maximum number of rows to return.

Examples

Query the orders table and limit the number of rows to return to five by using the **LIMIT** clause.

```
SELECT orderdate FROM orders LIMIT 5;
-----
o_orderdate
-----
1996-04-14
1992-01-15
1995-02-01
1995-11-12
1992-04-26
```

Query the customer table and return the information of the third customer to the seventh customer sorted by the creation date.

```
SELECT * FROM customer ORDER BY create_date LIMIT 2,5
```

8.11.4.7 ORDER BY

You can use the **ORDER BY** clause to sort the query result. Each expression in the **ORDER BY** clause consists of column names or the sequence numbers (starting from 1) of columns.

Syntax

```
[ ORDER BY expression
[ ASC | DESC ]
[ NULLS FIRST | NULLS LAST ]
[ LIMIT { count | ALL } ]
```

8.11.4.8 Subqueries

The following example shows how to query the top ten sellers in ticket sales. The **WHERE** clause defines a table subquery, and the result of the subquery consists of multiple rows and one column.



Note:

The result of a table subquery can contain multiple rows and columns.

```
select firstname, lastname, cityname, max(qtysold) as maxsold
from users join sales on users.userid=sales.sellerid
where users.city not in(select venuecity from venue)
group by firstname, lastname, city
order by maxsold desc, city desc
limit 10;
```

firstname	lastname	cityname	maxsold
Noah	Guerrero	Worcester	8
Isadora	Moss	Winooski	8
Kieran	Harrison	Westminster	8
Heidi	Davis	Warwick	8
Sara	Anthony	Waco	8
Bree	Buck	Valdez	8
Evangeline	Sampson	Trenton	8
Kendall	Keith	Stillwater	8
Bertha	Bishop	Stevens Point	8
Patricia	Anderson	South Portland	8

8.11.4.9 UNION, INTERSECT, and EXCEPT

You can use the **UNION**, **INTERSECT**, and **EXCEPT** operators to combine multiple query result sets to a single result set.

Syntax

```
query
{ UNION [ ALL ] | INTERSECT | EXCEPT | MINUS }
query
```

Parameters

- **UNION**: returns the union of two result sets.
- **UNION ALL**: returns the union of two result sets with duplicate rows retained. The **ALL** keyword specifies that the duplicate rows are retained after the **UNION** calculation.
- **INTERSECT**: returns the intersection of two result sets.
- **EXCEPT|MINUS**: returns rows that appear in the first result set, but not in the other result set. The **MINUS** operator is equal to the **EXCEPT** operator.

Calculation sequence

- The **UNION** and **EXCEPT** operators are left-associative. That is, if you do not use parentheses () to change the calculation sequence, the calculation starts from left to right.

For example, in the following statement, the **UNION** operator returns the union of T1 and T2. Then the **EXCEPT** operator returns the rows that appear only in the union returned by the **UNION** operator, but not T3.

```
select * from t1
union
select * from t2
except
select * from t3
```

```
order by c1;
```

- **In the same statement, the `INTERSECT` operator is prioritized before the `UNION` and `EXCEPT` operators.**

For example, the following statement finds the intersection of T2 and T3, and then the union of the intersection and T1.

```
select * from t1
union
select * from t2
intersect
select * from t3
order by c1;
```

- **You can use parentheses () to change the calculation sequence of these operators.**

For example, the following statement finds the union of T1 and T2, and then the intersection of the union and T3.

```
(select * from t1
union
select * from t2)
intersect
(select * from t3)
order by c1;
```

8.11.5 CREATE USER

You can execute the `CREATE USER` statement to create an account.

Syntax

```
CREATE USER
    [if not exists] user [auth_option] [, [if not exists] user [
auth_option]] ...
```

Precautions

To create an account by executing the `CREATE USER` statement, you must have the `CREATE_USER` permission.

Examples

Create an account named account2 and set the password to Account2.

```
CREATE USER if not exists 'account2' IDENTIFIED BY 'Account2';
```

8.11.6 GRANT

You can execute the GRANT statement to grant permissions to an account.

Syntax

```
GRANT
  priv_type [(column_list)]
  [, priv_type [(column_list)]] ...
ON priv_level
TO user [auth_option]
[WITH {GRANT OPTION}]
```

Parameters

- **priv_type**: the type of permission to grant.
- **column_list**: optional. If the **priv_type** parameter is set to **SELECT**, you can enter the names of columns to grant the **SELECT** permission on these columns.
- **priv_level**: the level of the permission to grant.
 - ***. ***: the cluster level.
 - **db_name. ***: the database level.
 - **db_name.table_name** or **table_name**: the table level.

Precautions

To grant permissions to the other accounts by executing the GRANT statement, you must have the GRANT OPTION permission.

Examples

- **Grant the cluster-level `all` permission to account2.**

```
GRANT all ON *.* TO 'account2';
```

- **Grant the database-level `all` permission to account3.**

```
GRANT all ON adb_demo.* TO 'account3';
```

- **Execute the GRANT statement to create and grant permissions to an account. For example, create an account with the cluster-level data manipulation permissions.**

```
GRANT insert,select,update,delete on *.* to 'test'@'%' identified by 'Testpassword1';
```

Create an account with the database-level data manipulation permissions.

```
GRANT insert,select,update,delete on adb_demo.* to 'test123' identified by 'Testpassword123';
```

- **Create an account that has the `SELECT` permission on the specified columns.**

```
GRANT select (customer_id, sex) ON customer TO 'test321' identified by 'Testpassword321';
```

8.11.7 REVOKE

You can execute the REVOKE statement to revoke permissions from an account.

Syntax

```
REVOKE
  priv_type [(column_list)]
  [, priv_type [(column_list)]] ...
  ON [object_type] priv_level
  FROM user
```

Parameters

- `priv_type`: the type of permission to revoke.
- `column_list`: optional. If the `priv_type` parameter is set to `SELECT`, you can enter the names of columns to revoke the `SELECT` permission on these columns.
- `priv_level`: the level of the permission to revoke.
 - `*.*`: the cluster level.
 - `db_name.*`: the database level.
 - `db_name.table_name` or `table_name`: the table level.

Precautions

To revoke permissions from other accounts by executing the REVOKE statement, you must have the `GRANT OPTION` permission.

Examples

Revoke the database-level `all` permission of account3.

```
REVOKE all ON adb_demo. * FROM 'account3';
```

8.11.8 Query users

AnalyticDB for MySQL is compatible with MySQL databases. AnalyticDB for MySQL provides a built-in database named `MySQL`, which stores user information, permission information, and stored procedures of AnalyticDB for MySQL. You can execute a `SELECT` statement to query user information of AnalyticDB for MySQL.

Precautions

- In AnalyticDB for MySQL, only the privileged account can query user information.
- The privileged account in AnalyticDB for MySQL is equivalent to the root account in MySQL.

Examples

```
USE MYSQL;
SELECT User, Host, Password FROM mysql.user;
+-----+-----+-----+
| User      | Host | Password                |
+-----+-----+-----+
| account1  | %    | *61f3777f02386598cd***** |
| account2  | %    | *0fe79c07e168cabc99***** |
```

8.11.9 RENAME USER

You can execute the `RENAME USER` statement to modify the name of an account.

Syntax

```
RENAME USER old_user TO new_user [, old_user TO new_user] ...
```

Examples

```
RENAME USER account2 TO account_2;
SELECT User, Host, Password FROM mysql.user;
+-----+-----+-----+
| User      | Host | Password                |
+-----+-----+-----+
| account2  | %    | *61f3777f02386598cda***** |
```

```
| account_2| % | *0fe79c07e168cab99b****|
```

8.11.10 DROP USER

You can execute the **DROP USER** statement to delete an account.

Syntax

```
DROP USER [if exists] user [, [if exists] user] ...
```

Precautions

To delete an account by executing the **DROP USER** statement, you must have the **CREATE_USER** permission.

Examples

```
DROP USER account_2;
```

8.11.11 SHOW

SHOW DATABASES

You can execute the **SHOW DATABASES** statement to view the databases.

Syntax

```
SHOW DATABASES [EXTRA];
```

You can specify the **EXTRA** parameter to view extra information about the databases, such as the IDs of the creators and the connection information.

Examples

```
SHOW DATABASES EXTRA;
+-----+
| Database |
+-----+
| adb_demo |
| MYSQL    |
| INFORMATION_SCHEMA |
| adb_demo2 |
```

SHOW TABLES

Syntax

You can execute the SHOW TABLES statement to view the tables in the current database.

```
SHOW TABLES [IN db_name];
```

Examples

```
SHOW TABLES IN adb_demo;
+-----+
| Tables_in_adb_demo |
+-----+
| customer            |
| customer2          |
| new_customer       |
| test_table         |
| v                  |
```

SHOW COLUMNS

You can execute the SHOW COLUMNS statement to view the columns in a table.

Syntax

```
SHOW COLUMNS IN db_name.table_name;
```

Examples

```
SHOW COLUMNS IN adb_demo.test_table;
```

SHOW CREATE TABLE

You can execute the SHOW CREATE TABLE statement to view the statement that is used to create a table.

Syntax

```
SHOW CREATE TABLE db_name.table_name;
```

Examples

```
SHOW CREATE TABLE adb_demo.customer;
```

SHOW GRANTS

You can execute the SHOW GRANTS statement to view the permissions that are granted to the current account.

Syntax

```
SHOW GRANTS;
```

8.12 System functions

This topic describes the system functions supported by AnalyticDB for MySQL.

8.12.1 Aggregate functions

The aggregate functions described in this topic use the `testtable` table as test data.

```
create table testtable(a int) distributed by hash(a);
```

```
insert into testtable values (1),(2),(3);
```

8.12.1.1 AVG

This function calculates the average value of all input values.

```
avg(bigint x) avg(double x) avg(float x)
```

- **Return value type:** DOUBLE.
- **Example:**

```
select avg(a) from testtable;
+-----+
| avg(a) |
+-----+
|    2.0 |
```

8.12.1.2 BIT_AND

This function returns the result of bitwise AND of all bits in the parameter.

```
bit_and(float x)
bit_and(bigint x)
bit_and(double x)
```

- **Return value type:** BIGINT.
- **Example:**

```
select bit_and(a) from testtable;
+-----+
| bit_and(a) |
+-----+
```

```
|          0          |
```

8.12.1.3 BIT_OR

This function returns the result of bitwise OR of all bits in the parameter.

```
bit_or(float x)
bit_or(bigint x)
bit_or(double x)
```

- **Return value type: BIGINT.**
- **Example:**

```
select bit_or(a) from testtable;
+-----+
|          bit_or(a)          |
+-----+
|              3              |
+-----+
```

8.12.1.4 BIT_XOR

This function returns the result of bitwise XOR of all bits in the parameter.

```
bit_xor(double x)
bit_xor(bigint x)
bit_xor(float x)
```

- **Return value type: BIGINT.**
- **Example:**

```
select bit_xor(a) from testtable;
+-----+
|          bit_xor(a)          |
+-----+
|              0              |
+-----+
```

8.12.1.5 COUNT

This function counts the number of records.

```
count([distinct|all] value x)
```

- **Description: distinct and all specify whether to exclude duplicate records in the counting process. The default value is all, indicating that all records are counted . If distinct is specified, only records with distinct values are counted.**
- **Return value type: BIGINT.**
- **Example:**

```
select count(distinct a) from testtable;
+-----+
| count(DISTINCT a) |
+-----+
```

8.12.1.6 MAX

This function returns the maximum value of all input values.

```
max(value x)
```

- **Description:** value can be of any data type. However, data of the BOOLEAN type is not supported in the calculation. If the value for a row in the specified column is null, this row is ignored.
- **Return value type:** LONG.
- **Example:**

```
select max(a) from testtable;
+-----+
| max(a) |
+-----+
|      3 |
```

8.12.1.7 MIN

This function returns the minimum value of all input values.

```
min(value x)
```

- **Description:** value can be of any data type. However, data of the BOOLEAN type is not supported in the calculation. If the value for a row in the specified column is null, this row is ignored.
- **Return value type:** LONG.
- **Example:**

```
select min(a) from testtable;
+-----+
| min(a) |
+-----+
|      1 |
```

8.12.1.8 STD/STDDEV

This function returns the sample standard deviation of all input values.

```
std(double x)
std(bigint x)
stddev(double x)
stddev(bigint x)
```

- **Return value type:** DOUBLE.

- **Example:**

```
select std(a) from testtable;
+-----+
| std(a) |
+-----+
| 0.816496580927726 |
```

8.12.1.9 STDDEV_POP

This function returns the population standard deviation of all input values.

```
stddev_pop(double x)
stddev_pop(bigint x)
```

- **Return value type: DOUBLE.**

- **Example:**

```
select stddev_pop(a) from testtable;
+-----+
| stddev_pop(a) |
+-----+
| 0.816496580927726 |
```

8.12.1.10 STDDEV_SAMP

This function returns the population standard deviation for a group of integers, decimals, or floating-point numbers.

```
stddev_samp(double x)
stddev_samp(bigint x)
```

- **Return value type: DOUBLE.**

- **Example:**

```
select stddev_samp(a) from testtable;
+-----+
| stddev_samp(a) |
+-----+
| 1.0 |
```

8.12.1.11 SUM

This function calculates the sum of all input values.

```
sum(double x)
sum(float x)
sum(bigint x)
```

- **Return value type: BIGINT.**

- **Example:**

```
select sum(a) from testtable;
```

```

+-----+
|          sum(a)          |
+-----+
|              6          |
+-----+

```

8.12.1.12 VAR_POP

This function returns the population variance for a group of integers, decimals, or floating-point numbers.

```

var_pop(double x)
var_pop(bigint x)

```

- **Description:** You can also use the `VARIANCE()` function, which has the same meaning as the `VAR_POP` function. However, the `VARIANCE()` function is not a standard SQL function. If no matches are found, `VAR_POP()` returns the value `NULL`.
- **Return value type:** `DOUBLE`.
- **Example:**

```

select var_pop(a) from testtable;
+-----+
|          var_pop(a)          |
+-----+
|  0.6666666666666666        |
+-----+

```

8.12.1.13 VAR_SAMP

This function returns the sample variance for a group of integers, decimals, or floating-point numbers.

```

var_samp(double x)
var_samp(bigint x)

```

- **Return value type:** `DOUBLE`.
- **Example:**

```

select var_samp(a) from testtable;
+-----+
|          var_samp(a)          |
+-----+
|              1.0            |
+-----+

```

8.12.1.14 VARIANCE

This function returns the population variance for a group of integers, decimals, or floating-point numbers.

```

variance(double x)

```



```
| 2001-01-25 |
```

```
select adddate(timestamp '2001-1-22',interval '3' day);
+-----+
| adddate(TIMESTAMP '2001-1-22', INTERVAL '3' DAY) |
+-----+
| 2001-01-25 00:00:00 |
```

```
select adddate(datetime '2001-1-22',interval '3' day);
+-----+
| adddate(DATETIME '2001-1-22', INTERVAL '3' DAY) |
+-----+
| 2001-01-25 00:00:00 |
```

```
select adddate('2001-1-22',interval '3' day);
+-----+
| adddate('2001-1-22', INTERVAL '3' DAY) |
+-----+
| 2001-01-25 |
```

```
select adddate(datetime '2001-1-22',interval '3' second);
+-----+
| adddate(DATETIME '2001-1-22', INTERVAL '3' SECOND) |
+-----+
| 2001-01-22 00:00:03 |
```

8.12.2.2 ADDTIME

This function adds time to a specified time. That is, this function returns the result of expr1 plus expr2.

```
ADDTIME(expr1,expr2)
```

- **Parameter types:**

```
addtime(date,varchar)
addtime(time,varchar)
addtime(datetime,varchar)
addtime(timestamp,varchar)
addtime(varchar,varchar)
```

- **Return value type: VARCHAR.**

- **Example:**

```
select addtime(date '1998-01-01','01:01:01');
+-----+
| addtime(DATE '1998-01-01', '01:01:01') |
+-----+
```

```
| 1998-01-01 01:01:01 |
```

8.12.2.3 CONVERT_TZ

This function converts the value of `dt` from `from_tz` to `to_tz` and returns the result.

```
CONVERT_TZ(dt, from_tz, to_tz)
```

- **Parameter types:**

```
convert_tz(varchar, varchar, varchar)
```

- **Return value type: DATETIME.**

- **Example:**

```
select convert_tz('2004-01-01 12:00:00', '+00:00', '+10:00');
+-----+
| convert_tz('2004-01-01 12:00:00', '+00:00', '+10:00') |
+-----+
|                                     2004-01-01 22:00:00 |
```

8.12.2.4 CURDATE

This function returns the current date.

```
CURDATE()
```

- **Return value type: DATE.**

- **Example:**

```
select curdate;
+-----+
| curdate() |
+-----+
| 2019-05-25 |
```

8.12.2.5 CURTIME

This function returns the current time.

```
CURTIME()
```

- **Return value type: TIME.**

- **Example:**

```
select curtime();
+-----+
| curtime() |
+-----+
```

| 14:39:22.109 |

8.12.2.6 DATE

This function returns the date part of a date or datetime expression.

DATE(expr)

- **Parameter types:**

```
date(timestamp)
date(datetime)
date(varchar)
```

- **Return value type: DATE.**

- **Example:**

```
select date(timestamp '2003-12-31 01:02:03');
+-----+
| date(TIMESTAMP '2003-12-31 01:02:03') |
+-----+
| 2003-12-31                             |
+-----+
```

8.12.2.7 DATE_FORMAT

This function returns a date as a string in the specified format.

DATE_FORMAT(date, format)

- **Description:** The following table describes the format specifiers.

Specifier	Description
%a	The abbreviation of a day of a week. Valid values: Sun to Sat.
%b	The abbreviation of a month name. Valid values: Jan to Dec.
%c	The month in numeric format. Valid values: 0 to 12.
%d	The day of the month in numeric format. Valid values: 00 to 31.
%e	The day of the month in numeric format. Valid values: 0 to 31.
%f	The microseconds. Valid values: 000000 to 999999.
%H	The hour. Valid values: 00 to 23.
%h	The hour. Valid values: 01 to 12.

Specifier	Description
%I	The hour. Valid values: 01 to 12.
%i	The minutes in numeric format. Valid values: 00 to 59.
%j	The day of the year. Valid values: 001 to 366.
%k	The hour. Valid values: 0 to 23.
%l	The hour. Valid values: 1 to 12.
%M	The name of the month. Valid values: January to December.
%m	The month in numeric format. Valid values: 00 to 12.
%p	The abbreviation of a time period in 12-hour format. Valid values: AM and PM.
%r	The time in 12-hour format (hh:mm:ss AM or hh:mm:ss PM).
%S	The seconds. Valid values: 00 to 59.
%s	The seconds. Valid values: 00 to 59.
%T	The time in 24-hour format (hh:mm:ss).
%v	The week number. Monday is the first day of a week. This specifier applies to WEEK() mode 3 and is used with %x.
%W	The name of the weekday. Valid values : Sunday to Saturday.
%x	The year of the week in numeric format. Monday is the first day of a week. This specifier is used with %v, and the value contains four digits.
%Y	The year of the week in numeric format. The value contains four digits.
%y	The year of the week in numeric format. The value contains two digits.
%%	The percent sign (%).
%x	x, for any "x" not listed above.

- **Parameter types:**

```
date_format(timestamp, varchar)
date_format(varchar, varchar)
date_format(datetime, varchar)
date_format(date, varchar)
```

- **Return value type: VARCHAR.**

- **Example:**

```
select date_format(timestamp '2019-05-27 13:23:00', '%W %M %Y') as
result;
+-----+
| result          |
+-----+
| Monday May 2019 |
```

8.12.2.8 SUBDATE/DATE_SUB

This function returns the value of date minus the specified INTERVAL.

```
DATE_SUB(date, INTERVAL expr unit)
```

- **Description: Valid values of unit: second, minute, hour, day, month, year, minute_second, hour_second, hour_minute, day_second, day_minute, day_hour, and year_month. Default value of unit: day.**
- **Parameter types:**

```
subdate(date, INTERVAL expr unit)
subdate(timestamp, INTERVAL expr unit)
subdate(datetime, INTERVAL expr unit)
subdate(varchar, INTERVAL expr unit)
subdate(date, bigint)
subdate(date, varchar)
subdate(datetime, bigint)
subdate(datetime, varchar)
subdate(timestamp, bigint)
subdate(timestamp, varchar)
subdate(varchar, bigint)
subdate(varchar, varchar)
```

- **Return value type: DATE.**
- **Example:**

```
select date_sub(date '2001-1-22', interval '3' day);
+-----+
| date_sub(DATE '2001-1-22', INTERVAL '3' DAY) |
+-----+
```

```
| 2001-01-19 |
```

8.12.2.9 DATEDIFF

This function returns the number of days between expr1 and expr2.

```
DATEDIFF(expr1,expr2)
```

- **Parameter types:**

```
datediff(varchar, varchar)
datediff(datetime, varchar)
datediff(varchar, datetime)
datediff(datetime, datetime)
datediff(varchar, timestamp)
datediff(timestamp, timestamp)
datediff(timestamp, varchar)
datediff(date, date)
datediff(date, varchar)
datediff(varchar, date)
```

- **Return value type: BIGINT.**

- **Example:**

```
select datediff('2007-12-31 23:59:59','2007-12-30');
+-----+
| datediff('2007-12-31 23:59:59', '2007-12-30') |
+-----+
|                                             1 |
```

8.12.2.10 DAY/DAYOFMONTH

This function returns the day in date. Valid values: [1,31].

```
DAY(date)
DAYOFMONTH(date)
```

- **Parameter types:**

```
dayofmonth(timestamp)
dayofmonth(datetime)
dayofmonth(date)
dayofmonth(time)
dayofmonth(varchar)
```

- **Return value type: BIGINT.**

- **Example:**

```
select dayofmonth(timestamp '2007-02-03 12:23:09');
+-----+
| dayofmonth(TIMESTAMP '2007-02-03 12:23:09') |
+-----+
```

8.12.2.11 DAYNAME

This function returns the day of the week for a date, such as Monday.

DAYNAME(date)

- **Parameter types:**

```
dayname(timestamp)
dayname(datetime)
dayname(date)
dayname(varchar)
```

- **Return value type: VARCHAR.**

- **Example:**

```
select dayname(timestamp '2007-02-03 00:00:00');
+-----+
| dayname(TIMESTAMP '2007-02-03 00:00:00') |
+-----+
| Saturday                                |
```

8.12.2.12 DAYOFWEEK

This function returns the weekday index for a date. For example, the weekday indexes for Sunday, Monday, and Saturday are 1, 2, and 7, respectively.

DAYOFWEEK(date)

- **Parameter types:**

```
dayofweek(timestamp)
dayofweek(datetime)
dayofweek(date)
dayofweek(varchar)
```

- **Return value type: BIGINT.**

- **Example:**

```
select dayofweek(timestamp '2007-02-03 00:00:00');
+-----+
| dayofweek(TIMESTAMP '2007-02-03 00:00:00') |
+-----+
```

8.12.2.13 DAYOFYEAR

This function returns the day of the year for a date. Valid values: [1,366].

```
DAYOFYEAR(date)
```

- **Parameter types:**

```
dayofyear(timestamp)
dayofyear(datetime)
dayofyear(date)
dayofyear(varchar)
```

- **Return value type: BIGINT.**

- **Example:**

```
select dayofyear(timestamp '2007-02-03 00:12:12');
+-----+
| dayofyear(TIMESTAMP '2007-02-03 00:12:12') |
+-----+
|                                     34 |
```

8.12.2.14 EXTRACT

This function returns a part of a date or time, which is specified by unit. For example, this function can return the year, month, day, hour, or minute of a date or time.

```
EXTRACT(unit FROM date)
```

- **Valid values of unit: second, minute, hour, day, month, year, minute_second, hour_second, hour_minute, day_second, day_minute, day_hour, and year_month.**
- **Supported parameter types: VARCHAR, TIMESTAMP, DATETIME, and TIME.**
- **Return value type: BIGINT.**
- **Example:**

```
select extract(second from '2019-07-02 00:12:34');
+-----+
| _col0 |
+-----+
```

8.12.2.15 FROM_DAYS

This function returns a DATE value based on the parameter N indicating the number of days.

```
FROM_DAYS(N)
```

- **Parameter types:**

```
from_days(varchar)
from_days(bigint)
```

- **Return value type: DATE.**

- **Example:**

```
select from_days(730669);
+-----+
| from_days(730669) |
+-----+
| 2000-07-03       |
+-----+
```

8.12.2.16 FROM_UNIXTIME

This function returns a Unix timestamp in the specified format.

```
FROM_UNIXTIME(unix_timestamp[,format])
```

- **Description: The format parameter conforms to the format in the DATE_FORMAT function.**

- **Parameter types:**

```
from_unixtime(varchar, varchar)
from_unixtime(varchar)
from_unixtime(double, varchar)
from_unixtime(double)
```

- **Return value type: DATETIME.**

- **Example:**

```
select from_unixtime('1447430881','%Y %M %h:%i:%s %x');
+-----+
| from_unixtime('1447430881', '%Y %M %h:%i:%s %x') |
+-----+
```

```
| 2015 November 12:08:01 2015 |
```

8.12.2.17 HOUR

This function returns the hour part of a specified time.

```
HOUR(time)
```

- **Parameter types:**

```
hour(timestamp)
hour(datetime)
hour(date)
hour(time)
hour(varchar)
```

- **Return value type: BIGINT.**

- **Example:**

```
select hour(timestamp '2019-12-07 10:05:03');
+-----+
| hour(TIMESTAMP '2019-12-07 10:05:03') |
+-----+
|                                     10 |
```

8.12.2.18 LAST_DAY

This function returns the last day of the month for a date or datetime.

```
LAST_DAY(date)
```

- **Parameter types:**

```
last_day(varchar)
last_day(timestamp)
last_day(datetime)
last_day(date)
```

- **Return value type: DATE.**

- **Example:**

```
select last_day('2003-02-05');
+-----+
| last_day('2003-02-05') |
+-----+
| 2003-02-28             |
```

8.12.2.19 LOCALTIME/LOCALTIMESTAMP/NOW

This function returns the current timestamp.

```
localtime
localtime()
localtimestamp
localtimestamp()
```

```
now()
```

- **Return value type: DATETIME.**
- **Example:**

```
select now();
+-----+
| now() |
+-----+
| 2019-05-25 00:28:37 |
```

8.12.2.20 MAKEDATE

This function returns a date based on the year and dayofyear parameters.

```
MAKEDATE(year,dayofyear)
```

- **Parameter types:**

```
makedate(bigint, bigint)
makedate(varchar, varchar)
```

- **Return value type: DATE.**
- **Example:**

```
select makedate(2011,31), makedate(2011,32);
+-----+-----+
| makedate(2011, 31) | makedate(2011, 32) |
+-----+-----+
| 2011-01-31         | 2011-02-01         |
```

8.12.2.21 MAKETIME

This function returns a time based on the hour, minute, and second parameters.

```
MAKETIME(hour,minute,second)
```

- **Parameter types:**

```
maketime(bigint, bigint, bigint)
maketime(varchar, varchar, varchar)
```

- **Return value type: TIME.**
- **Example:**

```
select maketime(12,15,30);
+-----+
| maketime(12, 15, 30) |
+-----+
```

12:15:30

8.12.2.22 MINUTE

This function returns the minute part of a specified time.

MINUTE(time)

- **Parameter types:**

```
minute(timestamp)
minute(datetime)
minute(date)
minute(time)
minute(varchar)
```

- **Return value type: BIGINT.**

- **Example:**

```
select minute(timestamp '2008-02-03 10:05:03');
+-----+
| minute(TIMESTAMP '2008-02-03 10:05:03') |
+-----+
|                                     5 |
```

8.12.2.23 MONTH

This function returns the month for a date.

MONTH(date)

- **Parameter types:**

```
month(timestamp)
month(datetime)
month(date)
month(time)
month(varchar)
```

- **Return value type: BIGINT.**

- **Example:**

```
select month(timestamp '2008-02-03 00:00:00');
+-----+
| month(TIMESTAMP '2008-02-03 00:00:00') |
+-----+
```

8.12.2.24 MONTHNAME

This function returns the full name of the month for a date.

```
MONTHNAME(date)
```

- **Parameter types:**

```
monthname(timestamp)
monthname(datetime)
monthname(date)
monthname(varchar)
```

- **Return value type: VARCHAR.**

- **Example:**

```
select monthname(datetime '2008-02-03 00:00:00');
+-----+
| monthname(DATETIME '2008-02-03 00:00:00') |
+-----+
| February |
```

8.12.2.25 PERIOD_ADD

This function adds N months to period P.

```
PERIOD_ADD(P,N)
```

- **Parameter types:**

```
period_add(bigint, bigint)
period_add(varchar, varchar)
period_add(varchar, bigint)
```

- **Return value type: BIGINT.**

- **Example:**

```
select period_add(200801,2);
+-----+
| period_add(200801, 2) |
+-----+
| 200803 |
```

8.12.2.26 PERIOD_DIFF

This function returns the number of months between periods P1 and P2.

```
PERIOD_DIFF(P1,P2)
```

- **Parameter types:**

```
period_diff(bigint, bigint)
```

```
period_diff(vvarchar, varchar)
```

- **Return value type:** BIGINT.
- **Example:**

```
select period_diff(200802,200703);
+-----+
| period_diff(200802, 200703) |
+-----+
|                               11 |
```

8.12.2.27 QUARTER

This function returns the quarter of the year for a date. Valid values: [1,4].

```
QUARTER(date)
```

- **Parameter types:**

```
quarter(datetime)
quarter(vvarchar)
quarter(timestamp)
quarter(date)
```

- **Return value type:** BIGINT.
- **Example:**

```
select quarter(datetime '2008-04-01 12:12:12');
+-----+
| quarter(DATETIME '2008-04-01 12:12:12') |
+-----+
|                                           2 |
```

8.12.2.28 SEC_TO_TIME

This function converts seconds to time.

```
SEC_TO_TIME(seconds)
```

- **Parameter types:**

```
sec_to_time(bigint)
sec_to_time(vvarchar)
```

- **Return value type:** TIME.
- **Example:**

```
select sec_to_time(2378);
+-----+
| sec_to_time(2378) |
+-----+
```

```
| 00:39:38 |
```

8.12.2.29 SECOND

This function returns the second part of a specified time. Valid values: [0,59].

```
SECOND(time)
```

- **Parameter types:**

```
second(timestamp)
second(datetime)
second(date)
second(time)
second(varchar)
```

- **Return value type: BIGINT.**

- **Example:**

```
select second(timestamp '2019-03-12 12:13:14');
+-----+
| second(TIMESTAMP '2019-03-12 12:13:14') |
+-----+
|                                     14 |
```

8.12.2.30 STR_TO_DATE

This function converts a string to a date or datetime in the specified format.

```
STR_TO_DATE(str, format)
```

- **Parameter types:**

```
str_to_date(varchar, varchar)
```

- **Return value type: DATETIME.**

- **Example:**

```
select str_to_date('2017-01-06 10:20:30', '%Y-%m-%d %H:%i:%s') as
result;
+-----+
| result                |
+-----+
| 2017-01-06 10:20:30 |
```

8.12.2.31 SUBTIME

This function subtracts expr2 from expr1.

```
SUBTIME(expr1, expr2)
```

- **Parameter types:**

```
subtime(date, varchar)
```

```
subtime(datetime, varchar)
subtime(timestamp, varchar)
subtime(time, varchar)
subtime(varchar, varchar)
```

- **Return value type: DATETIME.**
- **Example:**

```
select subtime(date '2018-10-31','0:1:1');
+-----+
| subtime(DATE '2018-10-31', '0:1:1') |
+-----+
|                2018-10-30 23:58:59 |
```

8.12.2.32 SYSDATE

This function returns the system time.

```
SYSDATE()
```

- **Return value type: DATETIME.**
- **Example:**

```
select sysdate();
+-----+
| sysdate()          |
+-----+
| 2019-05-26 00:47:21 |
```

8.12.2.33 TIME

This function returns the time in expr as a string.

```
TIME(expr)
```

- **Parameter types:**

```
time(varchar)
time(datetime)
time(timestamp)
```

- **Return value type: VARCHAR.**
- **Example:**

```
select time('2003-12-31 01:02:03');
+-----+
| time('2003-12-31 01:02:03') |
+-----+
```

| 01:02:03 |

8.12.2.34 TIME_FORMAT

This function returns time as a string in the specified format.

```
TIME_FORMAT(time, format)
```

- **Description:** The format parameter conforms to the format in the [DATE_FORMAT](#) function.
- **Parameter types:**

```
time_format(varchar, varchar)
time_format(timestamp, varchar)
time_format(datetime, varchar)
time_format(time, varchar)
time_format(date, varchar)
```

- **Return value type:** VARCHAR.
- **Example:**

```
select time_format('12:00:00', '%H %k %h %I %l');
+-----+
| time_format('12:00:00', '%H %k %h %I %l') |
+-----+
| 12 12 12 12 12 |
```

8.12.2.35 TIME_TO_SEC

This function converts time to seconds.

```
TIME_TO_SEC(time)
```

- **Parameter types:**

```
time_to_sec(varchar)
time_to_sec(datetime)
time_to_sec(timestamp)
time_to_sec(date)
time_to_sec(time)
```

- **Return value type:** BIGINT.
- **Example:**

```
select time_to_sec(datetime '2009-12-12 22:23:00');
+-----+
| time_to_sec(DATETIME '2009-12-12 22:23:00') |
+-----+
```

8.12.2.36 TIMEDIFF

This function subtracts `expr2` from `expr1`.

```
TIMEDIFF(expr1,expr2)
```

- **Parameter types:**

```
timediff(time, varchar)
timediff(time, time)
timediff(varchar, varchar)
```

- **Return value type: DATETIME.**

- **Example:**

```
select timediff(time '12:00:00','10:00:00');
+-----+
| timediff(TIME '12:00:00', '10:00:00') |
+-----+
| 02:00:00                               |
```

8.12.2.37 TIMESTAMP

This function returns `expr` as a datetime value.

```
TIMESTAMP(expr)
```

- **Parameter types:**

```
timestamp(date)
timestamp(varchar)
```

- **Return value type: DATETIME.**

- **Example:**

```
select timestamp(date '2019-05-27');
+-----+
| timestamp(DATE '2019-05-27') |
+-----+
| 2019-05-27 00:00:00          |
```

8.12.2.38 TIMESTAMPADD

This function adds `interval` to `datetime_expr`.

```
TIMESTAMPADD(unit,interval,datetime_expr)
```

- **Description: unit specifies the unit of interval. Valid values of unit: second, minute, hour, day, week, month, quarter, and year.**

- **Parameter types:**

```
timestampadd(varchar, varchar, timestamp)
timestampadd(varchar, bigint, timestamp)
timestampadd(varchar, varchar, date)
timestampadd(varchar, bigint, date)
timestampadd(varchar, varchar, datetime)
timestampadd(varchar, bigint, datetime)
timestampadd(varchar, varchar, varchar)
timestampadd(varchar, bigint, varchar)
```

- **Return value type: DATETIME.**

- **Example:**

```
select timestampadd(second,'1',timestamp '2003-01-02 12:12:12')as
result;
+-----+
| result |
+-----+
| 2003-01-02 12:12:13 |
```

8.12.2.39 TIMESTAMPDIFF

This function subtracts `datetime_expr2` from `datetime_expr1`. `unit` specifies the unit of the result.

```
TIMESTAMPDIFF(unit,datetime_expr1,datetime_expr2)
```

- **Description: Valid values of unit: second, minute, hour, day, week, month, quarter, and year. Use this function in the same way as the TIMESTAMPADD function.**

- **Parameter types:**

```
timestampdiff(varchar, timestamp, timestamp)
timestampdiff(varchar, date, date)
timestampdiff(varchar, datetime, datetime)
timestampdiff(varchar, varchar, varchar)
```

- **Return value type: BIGINT.**

- **Example:**

```
select timestampdiff(second,datetime '2003-02-01 10:12:13',
datetime '2003-05-01 10:12:13')as result;
+-----+
| result |
+-----+
```

| 7689600 |

8.12.2.40 TO_DAYS

This function returns the number of days since year 0 based on date.

TO_DAYS(date)

- **Parameter types:**

```
to_days(date)
to_days(time)
to_days(varchar)
to_days(timestamp)
to_days(datetime)
```

- **Return value type: BIGINT.**

- **Example:**

```
select to_days(date '2018-12-12');
+-----+
| to_days(DATE '2018-12-12') |
+-----+
|                          737405 |
```

8.12.2.41 TO_SECONDS

This function returns the number of seconds since year 0 based on expr.

TO_SECONDS(expr)

- **Parameter types:**

```
to_seconds(date)
to_seconds(datetime)
to_seconds(timestamp)
to_seconds(varchar)
to_seconds(time)
```

- **Return value type: BIGINT.**

- **Example:**

```
select to_seconds(date '2019-09-08');
+-----+
| to_seconds(DATE '2019-09-08') |
+-----+
```

```
| 63735120000 |
```

8.12.2.42 UNIX_TIMESTAMP

UNIX_TIMESTAMP returns the Unix timestamp for the current time in seconds since '1970-01-01 00:00:00' UTC.

```
UNIX_TIMESTAMP([date])
```

- **Description:** UNIX_TIMESTAMP(date) returns the Unix timestamp for the date parameter in seconds since '1970-01-01 00:00:00' UTC.

- **Parameter types:**

```
unix_timestamp()
unix_timestamp(varchar)
unix_timestamp(timestamp)
unix_timestamp(date)
unix_timestamp(datetime)
```

- **Return value type:** BIGINT.

- **Example:**

```
select unix_timestamp();
+-----+
| unix_timestamp() |
+-----+
|          1558935850 |
```

8.12.2.43 UTC_DATE

This function returns the UTC date.

```
UTC_DATE()
```

- **Return value type:** VARCHAR.

- **Example:**

```
select utc_date();
+-----+
| utc_date() |
+-----+
| 2019-05-27 |
```

8.12.2.44 UTC_TIME

This function returns the UTC time.

```
UTC_TIME()
```

- **Return value type:** VARCHAR.

- **Example:**

```
select utc_time();
+-----+
| utc_time() |
+-----+
| 05:53:19   |
```

8.12.2.45 UTC_TIMESTAMP

This function returns the UTC timestamp.

```
utc_timestamp()
```

- **Return value type:** VARCHAR.

- **Example:**

```
select utc_timestamp();
+-----+
| utc_timestamp()      |
+-----+
| 2019-05-27 05:55:15 |
```

8.12.2.46 WEEK

This function returns the week number for date, which is the week to which date belongs in the year.

```
WEEK(date[,mode])
```

- **Description:**

- **date** specifies the date for which you want to obtain the week number.
- **mode** is an optional parameter that specifies the logic for calculating the week number. It allows you to specify whether the week starts from Monday or Sunday. The returned value ranges from 0 to 52 or from 1 to 53. The following table describes the formats that mode supports.

Mode	First day of the week	Range
0	Sunday	0 to 53
1	Monday	0 to 53
2	Sunday	1 to 53
3	Monday	1 to 53
4	Sunday	0 to 53
5	Monday	0 to 53

Mode	First day of the week	Range
6	Sunday	1 to 53
7	Monday	1 to 53

- **Parameter types:**

```
week(varchar)
week(varchar, bigint)
week(date)
week(date, bigint)
week(datetime)
week(datetime, bigint)
week(timestamp)
week(timestamp, bigint)
```

- **Return value type: BIGINT.**

- **Example:**

```
select week('2019-05-27');
+-----+
| week('2019-05-27') |
+-----+
|                    21 |
```

8.12.2.47 WEEKDAY

This function returns the weekday for date. The result is an integer indicating the weekday. The mapping is as follows: 0 = Monday, 1 = Tuesday, and 6 = Sunday.

```
WEEKDAY(date)
```

- **Parameter types:**

```
weekday(timestamp)
weekday(datetime)
weekday(date)
weekday(varchar)
```

- **Return value type: BIGINT.**

- **Example:**

```
select weekday(timestamp '2019-05-27 00:09:00');
+-----+
| weekday(TIMESTAMP '2019-05-27 00:09:00') |
+-----+
```

	0
--	---

8.12.2.48 WEEKOFYEAR

This function returns the calendar week for date. Valid values: [1, 53].

WEEKOFYEAR(date)

- **Parameter types:**

```
weekofyear(timestamp)
weekofyear(datetime)
weekofyear(date)
weekofyear(varchar)
```

- **Return value type: BIGINT.**

- **Example:**

```
select weekofyear(timestamp '2019-05-27 09:00:00');
+-----+
| weekofyear(TIMESTAMP '2019-05-27 09:00:00') |
+-----+
|                                             22 |
```

8.12.2.49 YEAR

This function returns the year for date.

YEAR(date)

- **Parameter types:**

```
year(timestamp)
year(datetime)
year(date)
year(time)
year(varchar)
```

- **Return value type: BIGINT.**

- **Example:**

```
select year(timestamp '2019-05-27 00:00:00');
+-----+
| year(TIMESTAMP '2019-05-27 00:00:00') |
+-----+
|                                     2019 |
```

8.12.2.50 YEARWEEK

This function returns the year and week for a date.

YEARWEEK(date)

```
YEARWEEK(date,mode)
```

- **Description:** The year in the result may be different from the year in the date parameter for the first and the last week of the year.

`mode` works in the same way as `mode` in the [WEEK](#) function. For the single-parameter syntax, the value of `mode` is 0.

- **Parameter types:**

```
yearweek(timestamp)
yearweek(timestamp, bigint)
yearweek(datetime)
yearweek(datetime, bigint)
yearweek(date, bigint)
yearweek(date)
yearweek(varchar)
yearweek(varchar, bigint)
```

- **Return value type:** BIGINT.

- **Example:**

```
select yearweek(timestamp '2019-05-27 00:00:00');
+-----+
| yearweek(TIMESTAMP '2019-05-27 00:00:00') |
+-----+
|                                     201921 |
```

8.12.3 String functions

8.12.3.1 ASCII

This function returns the decimal ASCII code of the `str` character or the leftmost character in the string `str`.

```
ASCII(varchar str)
```

- **Return value type:** BIGINT.

- **Example:**

```
select ascii('2');
+-----+
| ascii('2') |
+-----+
|          50 |
```

8.12.3.2 BIN

This function returns the binary string of the integer `N`.

```
BIN(bigint N)
```

- **Description:** If `N` is null, null is returned.

- **Return value type:** VARCHAR.
- **Example:**

```
select bin(12);
+-----+
| bin(12) |
+-----+
| 1100    |
```

8.12.3.3 BIT_LENGTH

This function returns the length of the string *str*, measured in bits.

```
BIT_LENGTH(varchar str)
```

- **Return value type:** BIGINT.
- **Example:**

```
select bit_length('text');
+-----+
| bit_length('text') |
+-----+
|                    | 32 |
```

8.12.3.4 CHAR

This function returns a string of decimal ASCII codes of the integers *N1*, *N2*, and so on.

```
CHAR(bigint N1, bigint N2...)
```

- **Return value type:** VARBINARY.
- **Example:**

```
select char(97,110,97,108,121,116,105,99,100,98);
+-----+
| char(97, 110, 97, 108, 121, 116, 105, 99, 100, 98) |
+-----+
| analyticdb                                         |
```

8.12.3.5 CHAR_LENGTH/CHARACTER_LENGTH

This function returns the length of the string *str*, measured in characters.

```
CHAR_LENGTH(varchar str)
```

- **Return value type:** BIGINT.
- **Example:**

```
select char_length('China');
+-----+
| char_length('China') |
```

```
+-----+
|                2 |
```

8.12.3.6 CONCAT

This function concatenates strings. If any of the parameters is null, null is returned.

```
concat(varchar str1, ..., varchar strn)
```

- **Return value type:** VARCHAR.
- **Example:**

```
select concat('aliyun', ', ', 'analyticdb');
+-----+
| concat('aliyun', ', ', 'analyticdb') |
+-----+
| aliyun, analyticdb                  |
```

8.12.3.7 CONCAT_WS

This function concatenates strings and separates them with delimiters. The separator parameter specifies the delimiter. Parameters whose value is null are not included in the final string.

```
concat_ws(varchar separator, varchar str1, ..., varchar strn)
```

- **Return value type:** VARCHAR.
- **Example:**

```
select concat_ws(',', 'First name', 'Second name', 'Last Name')as
result;
+-----+
| result                |
+-----+
| First name,Second name,Last Name |
```

8.12.3.8 ELT

This function returns the string specified by the integer N.

```
ELT(bigint N, varchar str1, varchar str2, varchar str3,...)
```

- If N is less than 1 or greater than the number of the specified strings, null is returned.
- **Return value type:** VARCHAR.
- **Example:**

```
select elt(4, 'Aa', 'Bb', 'Cc', 'Dd');
+-----+
| elt(4, 'Aa', 'Bb', 'Cc', 'Dd') |
+-----+
```

| Dd |

8.12.3.9 EXPORT_SET

This function returns a string in which the strings on and off are placed based on the bits 0 or 1 from right to left (from low-order to high-order) in the binary value of the bits integer.

```
EXPORT_SET(bigint bits, varchar on, varchar off[, varchar separator[,
bigint number_of_bits]])EXPORT_SET(bigint bits, varchar on, varchar
off[, varchar separator[, bigint number_of_bits]])
```

- **Description:** The string on is placed for bit 1, and the string off is placed for bit 0. These strings are separated by delimiters. The default delimiter is a comma (,). The number_of_bits parameter specifies the number of bits that are checked. Default value: 64.

If the value of the number_of_bits parameter is greater than 64, the parameter is silently clipped to 64.

The same value is returned when the number_of_bits parameter is set to -1 or 64.

- **Return value type:** VARCHAR.
- **Example:**

```
select export_set(5, '1', '0', ',', 2);
+-----+
| export_set(5, '1', '0', ',', 2) |
+-----+
| 1,0 |
```

8.12.3.10 FIELD

This function returns the position of the string str in the strings such as str1, str2, and str3. If the string str is not found, 0 is returned.

```
field(varchar str, varchar str1, varchar str2, varchar str3,...)
```

- **Return value type:** BIGINT.
- **Example:**

```
select field('Bb', 'Aa', 'Bb', 'Cc', 'Dd', 'Ff');
+-----+
| field('Bb', 'Aa', 'Bb', 'Cc', 'Dd', 'Ff') |
+-----+
```

8.12.3.11 FIND_IN_SET

This function returns the position of the string `str` in the string list `strlist`.

```
FIND_IN_SET(varchar str, varchar strlist)
```

- **Description:** If `str` is not found in `strlist` or `strlist` is empty, 0 is returned.

If either `str` or `strlist` is null, null is returned.

- **Return value type:** BIGINT.

- **Example:**

```
select find_in_set('b','a,b,c,d');
+-----+
| find_in_set('b','a,b,c,d') |
+-----+
|                               2 |
```

8.12.3.12 FORMAT

This function formats the integer `X` to the `#,###,###.##` format rounded to `D` decimal places, and returns the result as a string.

```
format(double X, bigint D)
```

- **Description:** If `D` is 0, the result has no decimal point or fractional part.
- **Return value type:** BIGINT.
- **Example:**

```
select format(12332.123456, 4)as result1, format(12332.1,4)as
result2, format(12332.2,0)as result3;
+-----+-----+-----+
| result1 | result2 | result3 |
+-----+-----+-----+
| 12,332.1235 | 12,332.1000 | 12,332 |
```

8.12.3.13 HEX

This function returns the hexadecimal string of the integer `N`, or returns a string consisting of the hexadecimal values of all characters in the string `str`.

```
HEX(bigint N)
HEX(varchar str)
```

- **Return value type:** VARCHAR.
- **Example:**

```
select hex(16);
+-----+
```

```
| hex(16) |
+-----+
| 10      |
```

8.12.3.14 INSTR

This function returns the position of the first occurrence of the substring `substr` in the string `str`.

```
INSTR(varchar str, varchar substr)
```

- **Return value type:** BIGINT.
- **Example:**

```
select instr('foobarbar', 'bar');
+-----+
| instr('foobarbar', 'bar') |
+-----+
|                          4 |
```

8.12.3.15 LEFT

This function returns the leftmost `len` characters of the string `str`.

```
LEFT(varchar str, bigint len)
```

- **Description:** If `str` or `len` is null, null is returned.
- **Return value type:** VARCHAR.
- **Example:**

```
select left('foobarbar', 5);
+-----+
| left('foobarbar', 5) |
+-----+
| fooba                |
```

8.12.3.16 LENGTH/OCTET_LENGTH

This function returns the length of the string `str`.

```
length(varchar str)
```

- **Return value type:** BIGINT.
- **Example:**

```
select length('aliyun');
+-----+
| length('aliyun') |
+-----+
```

8.12.3.17 LIKE

The **LIKE** operator is used to match the string expression with the string pattern. If the two strings match, 1 is returned. Otherwise, 0 is returned.

```
expression [ NOT ] LIKE pattern [ESCAPE 'escape_char']
```

- **Description:** The string pattern can contain the following wildcard characters:

- **%:** matches strings in any length.
- **_:** matches a single character.

escape_char: escapes the percent signs (%) and underscores (_) in the string pattern. The percent signs (%) and underscores (_) following the escape character are not used as wildcard characters.

- **Return value type:** BIGINT.
- **Example:**

```
select 'David!' like 'David_' as result1, 'David!' not like '
David_' as result2, 'David!' like '%D%v%' as result3;
+-----+-----+-----+
| result1 | result2 | result3 |
+-----+-----+-----+
|          1 |          0 |          1 |
```

8.12.3.18 LOCATE

This function returns the position of the first occurrence of substring **substr** in string **str**, or returns the position of the first occurrence of the substring **substr** in the string **str**, starting at position **pos**.

```
LOCATE(varchar substr, varchar str)
LOCATE(varchar substr, varchar str, bigint pos)
```

- **Description:** If **substr** is not found in **str**, 0 is returned.

If **substr** or **str** is null, null is returned.

- **Return value type:** BIGINT.
- **Example:**

```
select locate('bar', 'foobarbar');
+-----+
| locate('bar', 'foobarbar') |
+-----+
```

8.12.3.19 LOWER/LCASE

This function converts the string `str` to lowercase.

```
lower(varchar str)
```

- **Return value type:** VARCHAR.
- **Example:**

```
select lower('Aliyun');
+-----+
| lower('Aliyun') |
+-----+
| aliyun          |
```

8.12.3.20 LPAD

This function returns the string `str`, left-padded with the string `padstr` to a length of `len` characters.

```
lpad(varchar str, bigint len, varchar padstr)
```

- **Description:** If `str` is longer than `len`, the return value is shortened to `len` characters.
- **Return value type:** VARCHAR.
- **Example:**

```
select lpad('Aliyun',9,'#');
+-----+
| lpad('Aliyun', 9, '#') |
+-----+
| ###Aliyun              |
```

8.12.3.21 LTRIM

This function removes the leading space characters of the string `str`.

```
LTRIM(varchar str)
```

- **Return value type:** VARCHAR.
- **Example:**

```
select ltrim(' abc');
+-----+
| ltrim(' abc') |
+-----+
```

```
| abc |
```

8.12.3.22 MAKE_SET

This function returns a set value (a string containing substrings separated by delimiters) consisting of the string that have the corresponding bit in the bits set.

```
MAKE_SET(bits, str1, str2,...)
```

- **Description:** The string `str1` corresponds to bit 0 and the string `str2` corresponds to bit 1. The same rule applies to the rest numbers. The strings whose value is null are not included in the final string.
- **Return value type:** VARCHAR.
- **Example:**

```
select make_set(5,'hello','nice','world');
+-----+
| make_set(5, 'hello', 'nice', 'world') |
+-----+
| hello,world |
```

8.12.3.23 MID

This function returns a substring that contains `len` characters in length from the string `str`, starting from the `pos` position.

```
MID(varchar str, bigint pos, bigint len)
```

- **Return value type:** VARCHAR.
- **Example:**

```
select mid('Quadratically',5,6);
+-----+
| mid('Quadratically', 5, 6) |
+-----+
| ratica |
```

8.12.3.24 OCT

This function returns the octal string of the integer `N`.

```
OCT(bigint N)
```

- **Description:** If `N` is null, null is returned.
- **Return value type:** VARCHAR.
- **Example:**

```
select oct(12);
+-----+
| oct(12) |
```

```
+-----+
| 14    |
```

8.12.3.25 POSITION

This function returns the position of the first occurrence of the substring `substr` in the string `str`, starting from position 1. If `substr` is not found in `str`, 0 is returned.

```
POSITION(varchar substr IN varchar str)
```

- **Return value type:** BIGINT.
- **Example:**

```
select position('bar' in 'foobarbar');
+-----+
| locate('bar', 'foobarbar') |
+-----+
|                               4 |
```

8.12.3.26 REPEAT

This function repeats the string `str` for `count` times.

```
REPEAT(varchar str, bigint count)
```

- If `count < 1`, an empty string is returned.
- If `str` or `count` is null, null is returned.
- **Return value type:** VARCHAR.
- **Example:**

```
select repeat('a', 3);
+-----+
| repeat('a', 3) |
+-----+
| aaa            |
```

8.12.3.27 REPLACE

This function replaces all occurrences of the string `from_str` in the string `str` with the string `to_str`.

```
REPLACE(varchar str, varchar from_str, varchar to_str)
```

- **Return value type:** VARCHAR.
- **Example:**

```
select replace('WWW.aliyun.com', 'W', 'w');
+-----+
| replace('WWW.aliyun.com', 'W', 'w') |
+-----+
```

| www.aliyun.com |

8.12.3.28 REVERSE

This function returns the string `str` with the order of the characters reversed.

```
REVERSE(varchar str)
```

- **Return value type:** VARCHAR.
- **Example:**

```
select reverse('123456');
+-----+
| reverse('123456') |
+-----+
| 654321           |
```

8.12.3.29 RIGHT

This function returns the rightmost `len` characters from the string `str`.

```
RIGHT(varchar str, bigint len)
```

- **Description:** If `str` or `len` is null, null is returned.
- **Return value type:** VARCHAR.
- **Example:**

```
select right('abc',3);
+-----+
| presto_right('abc', 3) |
+-----+
| abc                   |
```

8.12.3.30 RLIKE/REGEXP

This function matches the expression string with the pattern string, which is a regular expression. If the two strings match, 1 is returned. Otherwise, 0 is returned.

```
expression RLIKE pattern
expression REGEXP pattern
```

- **If expression or pattern is null, null is returned.**
- **Return value type:** BIGINT.
- **Example:**

```
select 'Michael!' regexp '.*';
+-----+
| regexp_like('Michael!', '.*') |
+-----+
```

8.12.3.31 RPAD

This function returns the string `str`, right-padded with the string `padstr` to a length of `len` characters.

```
rpad(varchar str, bigint len, varchar padstr)
```

- If `str` is longer than `len`, the return value is shortened to `len` characters.
- Return value type: **VARCHAR**.
- Example:

```
select rpad('Aliyun',9,'#');
+-----+
| rpad('Aliyun', 9, '#') |
+-----+
| Aliyun###              |
```

8.12.3.32 RTRIM

This function removes the trailing space characters of the string `str`.

```
RTRIM(varchar str)
```

- Return value type: **VARCHAR**.
- Example:

```
select rtrim('barbar ');
+-----+
| rtrim('barbar ') |
+-----+
| barbar          |
```

8.12.3.33 SPACE

This function returns a string consisting of a specified number of space characters.

```
SPACE(bigint N)
```

- Return value type: **VARCHAR**.
- Example:

```
select concat("#", space(6), "#");
+-----+
| concat('#', space(6), '#') |
+-----+
```

```
| # # |
```

8.12.3.34 STRCMP

If the string `str1` is the same as the string `str2`, 0 is returned. If the string `str1` is smaller than the string `str2` based on the current sort order, -1 is returned. Otherwise, 1 is returned.

```
STRCMP(varchar str1, varchar str2)
```

- **Return value type:** BIGINT.
- **Example:**

```
select strcmp('text', 'text2');
+-----+
| strcmp('text', 'text2') |
+-----+
|                          -1 |
```

8.12.3.35 SUBSTR/SUBSTRING

```
SUBSTRING(varchar str, bigint pos)
SUBSTRING(varchar str FROM pos)
SUBSTRING(varchar str, bigint pos, bigint len)
SUBSTRING(varchar str FROM pos FOR len)
```

- **Description:**
 - **SUBSTRING(varchar str, bigint pos) or SUBSTRING(varchar str FROM pos):** returns the substring starting from the `pos` position to the end of the string. If `pos < 0`, the beginning of the substring is `pos` characters from the end of the string.
 - **SUBSTRING(varchar str, bigint pos, bigint len) or SUBSTRING(varchar str FROM pos FOR len):** returns a substring that contains `len` characters in length from the string, starting from the `pos` position. If `pos < 0`, the beginning of the substring is `pos` characters from the end of the string.
- **Return value type:** VARCHAR.
- **Example:**

```
select substr('helloworld', 6);
+-----+
| substr('helloworld', 6) |
+-----+
```

```
| world |
```

8.12.3.36 SUBSTRING_INDEX

This function returns the substring before the last occurrence of the **delim** delimiter in the **str** string.

```
SUBSTRING_INDEX(varchar str, varchar delim, bigint count)
```

- **Description:**
 - If **count** > 0, this function returns all the characters to the left of the last **delim** delimiter.
 - If **count** < 0, this function returns all the characters to the right of the last **delim** delimiter.
 - The **SUBSTRING_INDEX** function performs a case-sensitive match when searching for the **delim** delimiter.
- **Return value type:** VARCHAR.
- **Example:**

```
select substring_index('www.aliyun.com', '.', 2);
+-----+
| substring_index('www.aliyun.com', '.', 2) |
+-----+
| www.aliyun                               |
+-----+
```

8.12.3.37 TRIM

This function removes the leading and trailing space characters or other specified characters from a string.

```
TRIM([remstr FROM] str)
TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM] str)
```

- **Return value type:** VARCHAR.
- **Example:**

```
select trim(' bar ');
+-----+
| trim(' bar ') |
+-----+
```

| bar |

8.12.3.38 UPPER/UCASE

This function converts the string `str` to uppercase.

```
upper(varchar str)
```

- **Return value type:** VARCHAR.
- **Example:**

```
select upper('Aliyun');
+-----+
| upper('Aliyun') |
+-----+
| ALIYUN          |
```

8.12.4 Numeric functions

8.12.4.1 ABS

This function returns the absolute value of `x`.

```
abs(tinyint x)
abs(smallint x)
abs(int x)
abs(bigint x)
abs(float x)
abs(double x)
abs(decimal x)
```

- **Return value types:** LONG, DECIMAL, and DOUBLE.
- **Example:**

```
select abs(4.5);
+-----+
| abs(4.5) |
+-----+
|         4.5 |
```

8.12.4.2 ROUND

This function rounds the parameter `x`. `d` specifies the number of decimal places.

The default value of `d` is 0. The rounding algorithm changes based on the data type of `x`.

```
round(tinyint x)
round(smallint x)
round(int x)
round(bigint x)
round(float x)
round(double x)
```

```
round(x, d)
```

- **Description:**

- If **x** is null, null is returned.
- If **d** > 0, the parameter is rounded to the specified decimal place.
- If **d** = 0, the parameter is rounded to the nearest integer.
- If **d** < 0, digits to the left of the decimal point of the parameter are rounded.

- **Return value types:** LONG, DECIMAL, and DOUBLE.

- **Example:**

```
select round(4);
+-----+
| round(4) |
+-----+
|         4 |
```

8.12.4.3 SQRT

This function returns the square root of **x**.

```
sqrt(double x)
```

- **Return value type:** DOUBLE.

- **Example:**

```
select sqrt(4.222);
+-----+
| sqrt(4.222) |
+-----+
| 2.054750593137766 |
```

8.12.4.4 LN

This function returns the natural logarithm of **x**.

```
ln(double x)
```

- **Return value type:** DOUBLE.

- **Example:**

```
select ln(2.718281828459045);
+-----+
| ln(2.718281828459045) |
+-----+
```

8.12.4.5 LOG

If called with one parameter, this function returns the natural logarithm of x . If called with two parameters, this function returns the logarithm of y to the base x .

- **Return value type:** DOUBLE.
- **Example:**

```
select log(16);
+-----+
| log(16) |
+-----+
| 2.772588722239781 |
```

8.12.4.6 LOG2

This function returns the base-2 logarithm of the parameter.

- **Return value type:** DOUBLE.
- **Example:**

```
select log2(8.7654);
+-----+
| log2(8.7654) |
+-----+
| 3.131819928389146 |
```

8.12.4.7 PI

This function returns the value of Pi.

```
pi()
```

- **Return value type:** DOUBLE.
- **Example:**

```
select pi();
+-----+
| pi() |
+-----+
| 3.141592653589793 |
```

8.12.4.8 LOG10

This function returns the base-10 logarithm of the parameter.

```
log10(double x)
```

- **Return value type:** DOUBLE.

- **Example:**

```
select log10(100.876);
+-----+
| log10(100.876) |
+-----+
| 2.0037878529824615 |
+-----+
```

8.12.4.9 POWER/POW

This function returns the value of x raised to the power of y.

```
power(double x, double y)
pow(double x, double y)
```

- **Return value type: DOUBLE.**

- **Example:**

```
select power(1.2,3.4);
+-----+
| power(1.2, 3.4) |
+-----+
| 1.858729691979481 |
+-----+
```

8.12.4.10 RADIANS

This function converts degrees to radians.

```
radians(double x)
```

- **Return value type: DOUBLE.**

- **Example:**

```
select radians(60.0);
+-----+
| radians(60.0) |
+-----+
| 1.0471975511965976 |
+-----+
```

8.12.4.11 DEGREES

This function converts radians to degrees.

```
degrees(double x)
```

- **Return value type: DOUBLE.**

- **Example:**

```
select degrees(1.3);
+-----+
| degrees(1.3) |
+-----+
```

```
| 74.48451336700703 |
```

8.12.4.12 SIGN

This function returns the sign of x.

```
sign(smallint x)
sign(tinyint x)
sign(int x)
sign(bigint x)
sign(float x)
sign(double x)
sign(decimal x)
```

- **Return value type:** LONG.
- **Example:**

```
select sign(12);
+-----+
| sign(12) |
+-----+
|         1 |
```

8.12.4.13 CEILING/CEIL

This function returns the minimum integer value that is greater than the value of x.

```
ceiling(tinyint x)
ceiling(smallint x)
ceiling(int x)
ceiling(bigint x)
ceiling(float x)
ceiling(double x)
```

- **Return value types:** LONG, DECIMAL, and DOUBLE.
- **Example:**

```
select ceiling(4);
+-----+
| ceiling(4) |
+-----+
|         4 |
```

8.12.4.14 FLOOR

This function returns the maximum integer value that is less than the value of x.

```
floor(tinyint x)
floor(smallint x)
floor(int x)
floor(bigint x)
floor(float x)
floor(double x)
```

- **Return value types:** LONG, DECIMAL, and DOUBLE.

- **Example:**

```
select floor(4.5);
+-----+
| floor(4.5) |
+-----+
|          4.0 |
+-----+
```

8.12.4.15 EXP

This function returns the value of e (the base of natural logarithms) raised to the power of x.

```
exp(double x)
```

- **Return value type: DOUBLE.**

- **Example:**

```
select exp(4.5);
+-----+
| exp(4.5) |
+-----+
| 90.01713130052181 |
+-----+
```

8.12.4.16 COS

This function returns the cosine of x.

```
cos(double x)
```

- **Return value type: DOUBLE.**

- **Example:**

```
select cos(1.3);
+-----+
| cos(1.3) |
+-----+
| 0.26749882862458735 |
+-----+
```

8.12.4.17 ACOS

This function returns the arc cosine of x.

```
acos(double x)
```

- **Description: If $x > 1$ or $x < -1$, null is returned.**

- **Return value type: DOUBLE.**

- **Example:**

```
select acos(0.5);
+-----+
| acos(0.5) |
+-----+
```

```
+-----+
| 1.0471975511965979 |
```

8.12.4.18 TAN

This function returns the tangent of x.

```
tan(double x)
```

- **Return value type: DOUBLE.**
- **Example:**

```
select tan(8);
+-----+
| tan(8) |
+-----+
| -6.799711455220379 |
```

8.12.4.19 ATAN

This function returns the arc tangent of x.

```
atan(double x)
```

- **Return value type: DOUBLE.**
- **Example:**

```
select atan(0.5);
+-----+
| atan(0.5) |
+-----+
| 0.4636476090008061 |
```

8.12.4.20 ATAN2

This function returns the arc tangent of the result of x divided by y.

```
atan2(double x, double y)
atan(double x, double y)
```

- **Return value type: DOUBLE.**
- **Example:**

```
select atan2(0.5,0.3);
+-----+
| atan2(0.5, 0.3) |
+-----+
```

```
| 1.0303768265243125 |
```

8.12.4.21 COT

This function returns the cotangent of x.

```
cot(double x)
```

- **Return value type: DOUBLE.**
- **Example:**

```
select cot(1.234);
+-----+
| cot(1.234) |
+-----+
| 0.35013639786701445 |
```

8.12.4.22 ASIN

This function returns the arc sine of x.

```
asin(double x)
```

- **Return value type: DOUBLE.**
- **Example:**

```
select asin(0.5);
+-----+
| asin(0.5) |
+-----+
| 0.5235987755982989 |
```

8.12.4.23 SIN

This function returns the sine of x.

```
sin(double x)
```

- **Return value type: DOUBLE.**
- **Example:**

```
select sin(1.234);
+-----+
| sin(1.234) |
+-----+
| 0.9438182093746337 |
```

8.12.5 Arithmetic operators

+

- **Description: This operator is used for addition.**

- **Example:**

```
select 3+5;
+-----+
| _col0 |
+-----+
|      8 |
```

-

- **Description: This operator is used for subtraction.**

- **Example:**

```
select 3-5;
+-----+
| _col0 |
+-----+
|     -2 |
```

*

- **Description: This operator is used for multiplication.**

- **Example:**

```
select 3*pi();
+-----+
| _col0 |
+-----+
| 9.42477796076938 |
```

/

- **Description: This operator is used for division.**

- **Example:**

```
select 3/pi();
+-----+
| _col0 |
+-----+
| 0.954929658551372 |
```

DIV

- **Description: This operator is used for division. The decimal part of the result is discarded.**

- **Example:**

```
select 3 div pi();
+-----+
| _col0 |
+-----+
```

```
| 0 |
```

% or MOD

- **Description:** This operator returns the remainder of one parameter divided by the other parameter.
- **Example:**

```
select 3 mod pi();
+-----+
| _col0 |
+-----+
| 3.0 |
```

-

- **Description:** This operator converts a positive number to a negative number or a negative number to a positive number.
- **Example:**

```
select - 2;
+-----+
| _col0 |
+-----+
| -2 |
```

8.12.6 Bit functions and operators

BIT_COUNT

- **Description:** This function converts the parameter to a binary value, and then returns the number of bits that are set to 1 in the value.
- **Return value type:** BIGINT.
- **Example:**

```
select bit_count(2);
+-----+
| bit_count(2) |
+-----+
| 1 |
```

&

- **Description:** This function is used for bitwise AND.
- **Return value type:** BIGINT.
- **Example:**

```
select 12 & 15;
+-----+
| bitwise_and(12, 15) |
```



```

+-----+
| bitwise_right_shift(3, 2) |
+-----+
|                               0 |

```

<<(BITWISE_LEFT_SHIFT)

```

bitwise_left_shift(double x, double y)
bitwise_left_shift(varchar x, varchar y)
bitwise_left_shift(bigint x, bigint y)

```

- **Description:** This function shifts a value to the left.
- **Return value type:** BIGINT.
- **Example:**

```

SELECT 3 << 2;
+-----+
| bitwise_left_shift(3, 2) |
+-----+
|                               12 |

```

8.12.7 Control flow functions

The control flow functions described in this topic use the `conditiontest` table as test data.

```
create table conditiontest(a int) distributed by hash(a);
```

```
insert into conditiontest values (1),(2),(3);
```

```

SELECT * FROM conditiontest;
+----+
| a |
+----+
| 2 |
| 1 |
| 3 |

```

CASE

```

CASE expression
  WHEN value THEN result
  [ WHEN ... ]
  [ ELSE result ]
END

```

- **Description:** A simple CASE expression checks each value from left to right, and returns the result for the first value that is equal to the expression. If no match is found, result is returned.

- **Example:**

```
SELECT a,
       CASE a
         WHEN 1 THEN 'one'
         WHEN 2 THEN 'two'
         ELSE 'three'
       END as caseresult
FROM conditiontest;
```

a	caseresult
2	two
1	one
3	three

```
CASE
  WHEN condition THEN result
  [ WHEN ... ]
  [ ELSE result ]
END
```

- **Description:** An advanced CASE expression calculates conditions from left to right, and returns the result for the first condition that is TRUE. If no condition is TRUE, result is returned.

- **Example:**

```
SELECT a,
       CASE a
         WHEN a=1 THEN 'one1'
         WHEN a=2 THEN 'two2'
         ELSE 'three3'
       END as caseresult
FROM conditiontest;
```

a	caseresult
1	one1
3	three3
2	three3

IF

```
if(condition, true_value)
```

- **Description:** If the condition is true, true_value is returned. Otherwise, null is returned.
- **Example:**

```
SELECT IF((2+3)>4,5);
```

_col0
5

```
| 5 |
```

```
if(condition, true_value, false_value)
```

- **Description:** If the condition is true, true_value is returned. Otherwise, false_value is returned.
- **Example:**

```
SELECT IF((2+3)<5,5,6);
+-----+
| _col0 |
+-----+
| 6 |
```

IFNULL

```
IFNULL(expr1,expr2)
```

- **Description:** If expr1 is not null, expr1 is returned. Otherwise, expr2 is returned.
- **Example:**

```
SELECT IFNULL(NULL,2);
+-----+
| _col0 |
+-----+
| 2 |
SELECT IFNULL(1,0);
+-----+
| _col0 |
+-----+
| 1 |
```

NULLIF

```
NULLIF(expr1,expr2)
```

- **Description:** If expr1 is equal to expr2, null is returned. Otherwise, expr1 is returned.
- **Example:**

```
SELECT NULLIF (2,1);
+-----+
| _col0 |
+-----+
| 2 |
SELECT NULLIF (2,2);
+-----+
| _col0 |
+-----+
| NULL |
```