

Data Replication Service

Troubleshooting

Issue 01
Date 2026-01-21



Copyright © Huawei Technologies Co., Ltd. 2026. All rights reserved.

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

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

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

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

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: <https://e.huawei.com>

Security Declaration

Product Lifecycle

Huawei's regulations on product lifecycle are subject to the *Product End of Life Policy*. For details about this policy, visit the following web page:

<https://support.huawei.com/ecolumnsweb/en/warranty-policy>

Vulnerability

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

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

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

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

Initial Digital Certificate

The Initial digital certificates on Huawei devices are subject to the *Rights and Responsibilities of Initial Digital Certificates on Huawei Devices*. For details about this document, visit the following web page:

<https://support.huawei.com/enterprise/en/bulletins-service/ENEWS2000015789>

Huawei Enterprise End User License Agreement

This agreement is the end user license agreement between you (an individual, company, or any other entity) and Huawei for the use of the Huawei Software. Your use of the Huawei Software will be deemed as your acceptance of the terms mentioned in this agreement. For details about this agreement, visit the following web page:

<https://e.huawei.com/en/about/eula>

Lifecycle of Product Documentation

Huawei after-sales user documentation is subject to the *Product Documentation Lifecycle Policy*. For details about this policy, visit the following web page:

<https://support.huawei.com/enterprise/en/bulletins-website/ENEWS2000017761>

Contents

1 Solutions to Failed Check Items.....	1
1.1 Disk Space.....	1
1.1.1 Checking Whether the Destination Database Has Sufficient Storage Space.....	1
1.1.2 Checking Whether the Destination Server Has Sufficient Storage Space.....	2
1.2 Database Parameters.....	2
1.2.1 Checking Whether the Source Database Binlog Is Enabled.....	2
1.2.2 Checking Whether the Source Database Binlog Is Row-Based.....	3
1.2.3 Checking Whether the expire_logs_days Value in the Source Database Is Correct.....	3
1.2.4 Checking Whether the Source and Destination Database Character Sets Are Consistent.....	4
1.2.5 Checking Whether the Source Database server_id Meets the Incremental Migration Requirements	5
1.2.6 Checking Whether the Source and Destination Database Table Names Are Consistent in Case Sensitivity.....	6
1.2.7 Checking Whether the Source Database Contains Object Names with Non-ASCII Characters.....	6
1.2.8 Checking Whether the time_zone Values of the Source and Destination Databases Are the Same.....	7
1.2.9 Checking Whether the collation_server Values of the Source and Destination Databases Are the Same.....	8
1.2.10 Checking Whether the SERVER_UUID Values of the Source and Destination Databases Are the Same.....	8
1.2.11 Checking Whether the SERVER_ID Values of the Source and Destination Databases Are Different	8
1.2.12 Checking Whether the Source Database Contains Invalid sql_mode Values.....	9
1.2.13 Checking Whether the sql_mode Values of the Source and Destination Databases Are the Same....	9
1.2.14 Checking Whether the sql_mode Value in the Destination Database Is Not no_engine.....	10
1.2.15 Checking Whether the innodb_strict_mode Values of the Source and Destination Databases Are the Same.....	10
1.2.16 Checking Whether the max_wal_senders Value of the Source Database Is Correctly Configured.....	11
1.2.17 Checking Whether the WAL_LEVEL Value in the Source Database Is Correct.....	11
1.2.18 Checking Whether the MAX_REPLICATION_SLOTS Value in the Source Database Is Correct.....	12
1.2.19 Checking Whether the Source Database Is on Standby.....	12
1.2.20 Checking Whether the log_slave_updates Value in the Source Database Is Correct.....	13
1.2.21 Checking Whether the BLOCK_SIZE Value of the Source Database Is the Same as That of the Destination Database.....	13
1.2.22 Checking Whether the binlog_row_image Value is FULL.....	13
1.2.23 Checking Whether the Transaction Isolation Levels are Consistent.....	14

1.2.24 Checking Whether the lc_monetary Values of the Source and Destination Databases Are the Same	14
1.2.25 Checking Whether the Source Database Contains Trigger Names with Non-ASCII Characters	15
1.2.26 Checking Whether the Source Database Collections Contain More Than 10 Indexes	15
1.2.27 Checking Whether the Source Database Collections Contain TTL Indexes	15
1.2.28 Checking Whether log_bin_trust_function_creators Is Set to On in Both the Source and Destination Databases	16
1.2.29 Checking Whether GTID Is Enabled for the Source Database	16
1.2.30 Checking Whether GTID Is Enabled for the Destination Database	17
1.2.31 Checking Whether log_bin_trust_function_creators Is Set to On in the Destination Database	17
1.2.32 Checking Whether the Values in the Source Oracle Database Are Out of the Ranges of the MySQL Database	18
1.2.33 Checking Whether the max_allowed_packet Value of the Destination Database Is too Small	18
1.2.34 Checking Whether the Source Database User Has the Permission to Parse Logs	19
1.2.35 Checking Whether the ExpireLogsDays Value Is 0	19
1.2.36 Checking Whether the Source Database Character Set Is Supported	19
1.2.37 Checking Whether the Length Table and Field Names Is Not Supported	20
1.2.38 Checking Whether the Databases and Tables Exist	20
1.2.39 Checking Whether the Supplemental Log Level of the Source Database Meets Requirements	20
1.2.40 Checking Whether the Length of the Source Database Object Names Exceeds the Limit	21
1.2.41 Checking Whether session_replication_role of the Destination Database Is correctly Set	21
1.2.42 Checking the Database Compatibility Type	22
1.2.43 Checking Whether the Collation of the Destination Database Is Correct	22
1.2.44 Checking Whether the Index Name Is Supported	22
1.2.45 Checking Whether Tables Structures Are Consistent	23
1.2.46 Checking Whether Existing Data Meets the Constraints	23
1.2.47 Checking the Additional Column of the Destination Database	24
1.2.48 Checking Whether Implicit Primary Key Check Is Enabled for the Primary and Standby Databases	24
1.2.49 Checking Whether the Source Table Replication Attribute Is Correct	24
1.2.50 Checking Whether the Specified Replication Slot Exists in the Source Database	25
1.2.51 Checking Whether the MongoDB Instance Type Matches the Migration Mode	25
1.2.52 Checking the Physical Standby Database	25
1.2.53 Checking Whether the Case Sensitivity of the Destination Database Is Configured	26
1.2.54 Checking Whether CDC is Enabled for Tables in the Source Database	26
1.2.55 Checking Whether the CDC Retention Period in the Source Database Is Long Enough	27
1.2.56 Checking Whether the Source and Destination Databases Have Different Computer Names	27
1.2.57 Checking Whether the Length of the Source Database Name Exceeds 64	27
1.2.58 Synchronization Object Name Check	28
1.2.59 Checking Whether the Source Database Contains Disabled Clustered Indexes	28
1.2.60 Checking Whether the Source Database Is Empty	28
1.2.61 Checking Whether the Source Database Uses the Full Recovery Model	28
1.2.62 Checking the Synchronization Objects	29

1.2.63 Checking Whether the Source Database Tables Contain Data Types Not Supported for Migration	30
1.2.64 Checking Whether the SQL Server Agent of the Source Database Is Enabled	30
1.2.65 Checking Whether the Values of group_concat_max_len Are Consistent	30
1.2.66 Checking Whether the Table Structures of the Source Database Are Consistent	31
1.2.67 Checking Whether the Character Sets Are Compatible	31
1.2.68 Whether There Are XA Transactions That Have Not Been Submitted for a Long Time in the Source Database	31
1.2.69 Whether the Selected Objects Exist in the Destination Database	32
1.2.70 Destination Database Same Name Check	32
1.2.71 Whether the Destination Database User (Schema) and Table Exist	33
1.2.72 Whether the Maximum Number of Indexed Columns Has Been Reached	34
1.2.73 Checking the Length of the Index Column in the Source Database	34
1.2.74 Whether the Table Structures (Including Primary Key Indexes and the Number of Columns) of the Source Oracle Database and Destination Database Middleware Are Aligned	35
1.2.75 Whether Synchronization Objects Exist in the Destination Database	36
1.2.76 Whether the Source Database Contains Encrypted Objects	36
1.2.77 Checking Whether the Source Database Contains Unsupported Table Field Types	37
1.2.78 Checking Replication Attribute of Primary Key Columns	37
1.2.79 Whether There Are Tables Containing Fields of the longtext or longblob Type in the Synchronization Object	38
1.2.80 Checking Database Mapping Objects	39
1.2.81 Whether the Source Database Is the Standby Database of a TaurusDB Instance	40
1.2.82 Checking Whether the Source and Destination Database Character Sets Are Consistent	40
1.2.83 Checking Whether Data Replication Is Enabled for the Source Database	41
1.2.84 Checking Whether the Maximum Sequence Number of the Source Database is Less Than That of the Destination Database	42
1.2.85 Checking Whether Interval Partitioned Tables Are Included in the Source Database	42
1.2.86 Oracle Account Check in the Source Database	43
1.2.87 Checking the Number of DNs in the Source Database	43
1.2.88 Whether the Selected Objects Exist in the Destination Database	44
1.2.89 Whether There Are Foreign Keys Containing Unsupported Reference Operations in the Source Database	44
1.2.90 Whether the Selected Table Contains Delay Constraints	45
1.2.91 Whether the Source Database Tables Contain Primary Keys	46
1.2.92 Whether Foreign Keys Are Disabled or Tables to Be Synchronized Have Foreign Keys in the Destination Database	46
1.2.93 Whether There Are Composite Hash Indexes in the Source Collection	47
1.2.94 Whether There Are Composite Hash Shard Keys in the Source Collection	47
1.2.95 Whether the Source Table Structure Contains Newline Characters	48
1.2.96 Whether There Are Tables Containing Fields of the bytea or text Type in the Synchronization Object	48
1.2.97 Checking Whether the Source Table Structure Contains Virtual Columns	48
1.2.98 Whether the max_allowed_packet Value of the Source Database Is Too Small	49

1.2.99 Whether the Supplemental Log Level in the Source Database Is Correct.....	49
1.2.100 Whether Kafka Topics Have Been Created.....	50
1.2.101 Whether the Source Database Kernel Encoding Supports Data Replication.....	50
1.2.102 block_encryption_mode Consistency Check.....	51
1.2.103 Character Type and Sorting Rule Check in the Destination Database.....	52
1.2.104 Column Name Check in the Source Database.....	52
1.2.105 Source Encrypted Table Check.....	53
1.2.106 Checking Whether the Source Table Replication Attribute Is Correct.....	53
1.2.107 Partition Table Check on the Source Database.....	54
1.2.108 Checking Whether the Source Database Contains Unsupported Generated Columns.....	54
1.2.109 Checking Whether the ENABLE_SLOT_LOG Value of the Source Database Is Correct.....	55
1.3 Source DB Instance Statuses.....	55
1.3.1 Checking Whether the Source DB Instance Is Available.....	55
1.3.2 Checking Whether the Source and Destination Databases Are of the Same Type.....	55
1.3.3 Checking Whether the ChangeStream API of the source DB instance is available.....	56
1.4 Destination DB Instance Statuses.....	56
1.4.1 Checking Whether the Destination DB Instance Is Available.....	57
1.4.2 Checking Whether the Destination Database Is Involved in Another Migration Task.....	57
1.4.3 Checking Whether the Destination Database Has a Read Replica.....	58
1.4.4 Checking Whether the Destination Database Is Read-Only.....	58
1.4.5 Checking Whether the Extensions Are Supported.....	59
1.4.6 Checking Whether Destination Contains the Configured Database.....	60
1.4.7 Checking Whether the Destination DB Instance Is Available.....	61
1.4.8 Checking Whether the Destination Database Is Empty.....	61
1.4.9 Checking Whether There Are Foreign Keys that Are Not Disabled in the Destination Database Table	62
1.4.10 Checking Whether There Are Enabled Triggers in an Existing Destination Database Table.....	63
1.5 Database User Permissions.....	63
1.5.1 Whether the Source Database User Has Sufficient Permissions.....	63
1.5.2 Checking Whether the Destination Database User Has Sufficient Permissions.....	64
1.5.3 Checking Whether the Destination Database Account Has Required Permissions to Migrate Definer	65
1.6 Database Versions.....	65
1.6.1 Checking Whether the Source Database Version Is Supported.....	65
1.6.2 Checking Whether the Destination Database Version Is Supported.....	66
1.6.3 Checking Whether the Migration Is from an Earlier Database Version to the Same or a Later Version	66
1.7 Networks.....	66
1.7.1 Checking Whether the Source Database Is Connected.....	66
1.7.2 Checking Whether the Destination Database Is Connected.....	68
1.8 Database Objects.....	69
1.8.1 Checking Whether the Source Database Contains a MyISAM Table.....	70

1.8.2 Checking Whether the Source Database Contains the Functions or Stored Procedures that the Source Database User Is Not Authorized to Migrate.....	70
1.8.3 Checking Whether Objects with the Same Names Exist in the Source Database.....	71
1.8.4 Whether the Source Database Contains Unlogged Tables.....	71
1.8.5 Checking Whether the Names of Views to Be Migrated Are the Same.....	72
1.8.6 Checking Whether the _Id Fields in the Collection of the Source Database Have Indexes.....	72
1.8.7 Checking Whether the Index Length of the Source Database Exceeds the Limit.....	73
1.8.8 Checking Whether the Source Database Tables Use Storage Engines Not Supported by the Destination Database.....	74
1.8.9 Checking Whether the Database Names Mapped to the Destination DB Instance Contain Unsupported Characters.....	74
1.8.10 Checking Whether the Source Database Tables Contain Primary Keys.....	75
1.8.11 Checking Whether the Source Database Contains Triggers or Events.....	77
1.8.12 Checking Whether the Source Database Referenced Roles Pass the Check.....	77
1.8.13 Checking Whether the Source Database Referenced Accounts Pass the Check.....	78
1.8.14 Checking Whether the Source Database Contains Schemas or Users Named cdc.....	78
1.8.15 Checking Whether Associated Objects Are Selected.....	79
1.8.16 Checking Whether the Specified Objects Exist In the Destination Database.....	79
1.9 Database Configuration Items.....	80
1.9.1 Checking Whether the Source Database Name Is Valid.....	80
1.9.2 Checking Whether the Source Database Table Name Is Valid.....	81
1.9.3 Checking Whether the Source Database View Name Is Valid.....	82
1.9.4 Checking Whether the Source Database Collection Name Is Valid.....	83
1.9.5 Checking Whether the Shard Key Can Be Obtained from the Source Database.....	83
1.9.6 Checking Whether the Source Database Schema Name Is Valid.....	84
1.9.7 Checking Whether the Maximum Number of Chunks in the Destination Database Is Sufficient.....	85
1.9.8 Checking Whether Archive Logs Are Enabled on the Source Oracle Database.....	85
1.9.9 Checking Whether Supplemental Logging Is Correctly Enabled on the Source Database.....	86
1.10 Conflicts.....	86
1.10.1 Checking Whether the Names of the Source and Destination Databases Are the Same.....	86
1.10.2 Checking Whether the Same View Names Exist in Both the Source and Destination Databases.....	90
1.10.3 Checking Whether the Destination Database Contains a Non-Empty Collection with the Same Name As the Source Database.....	91
1.10.4 Checking Whether Destination Database Contains the Same Table Names As the Synchronization Objects.....	91
1.10.5 Checking Whether the Destination Database Contains Objects with the Same Name As Those in the Source Database.....	93
1.10.6 Checking Whether Collections in Both the Source and Destination Databases Are Not Capped.....	96
1.11 SSL Connections.....	96
1.11.1 Checking Whether the SSL Connection Is Correctly Configured.....	96
1.11.2 Checking Whether the SSL Connection Is Enabled for the Source Database.....	97
1.11.3 Checking Whether the SSL Certificate of the Source Database Exists.....	97
1.11.4 Checking Whether the SSL Certificate of the Destination Database Exists.....	98
1.11.5 Checking Whether Both the Source and Destination Databases Use SSL.....	99

1.12 Object Dependencies.....	99
1.12.1 Checking Whether the Objects Referenced by Views Are Selected for Migration.....	99
1.12.2 Checking Whether Referenced Tables Are Selected for Migration.....	100
1.13 Source Database Information.....	101
1.13.1 Checking Whether the Shards and Mongos Are in the Same Cluster.....	101
1.13.2 Checking Whether the Balancers of the Source Database Is Enabled.....	101
1.13.3 Checking Whether the Source and Destination Database Types Match.....	102
1.13.4 Long Transaction Check in the Source Database.....	102
1.14 Pre-Check Timeout.....	103
2 Failure Cases.....	105
2.1 Case Overview.....	105
2.2 Real-Time Migration from MongoDB to DDS.....	112
2.2.1 Full Migration Error: Prematurely reached end of stream.....	112
2.2.2 Full Migration Error: not authorized on *** to execute command {***}.....	112
2.2.3 Full Migration Error: GC overhead limit exceeded.....	113
2.2.4 Full Migration Error: The background process is unavailable. Maybe it has been killed manually or by the operating system. Please restart the task if possible or wait for restarting by itself within 5 minutes.".....	113
2.2.5 Full Migration Error: Timed out after 60000 ms while waiting to connect.....	114
2.2.6 Full or Incremental Migration Error: Timed out after 60000 ms while waiting to connect.....	114
2.2.7 Full or Incremental Migration Error: Invalid BSON field name ***.....	115
2.2.8 Incremental Migration Error: Timed out after 60000 ms while waiting for a server that matches com.mongodb.client.internal.....	115
2.2.9 Incremental Migration Error: Command failed with error *** (***):***. The full response is {***}".....	116
2.3 Real-Time Migration and Synchronization from MySQL to MySQL.....	116
2.3.1 Full Phase Error: Table *** doesn't exist.....	116
2.3.2 Full Phase Error: The background process is unavailable.....	117
2.3.3 Full Phase Error: Communications link failure The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.....	117
2.3.4 Full Phase Error: Error writing file *** (errno: 28 - No space left on device).....	118
2.3.5 Full Phase Error: The MySQL server is running with the --super-read-only option so it cannot execute this statement.....	118
2.3.6 Full Phase Error: The table *** is full.....	119
2.3.7 Full Phase Error: Unknown column *** in 'field list'.....	120
2.3.8 Full Phase Error: Lock wait timeout exceeded; try restarting transaction.....	120
2.3.9 Full Phase Error: Java heap space.....	121
2.3.10 Full Phase Error: Table *** already exists.....	121
2.3.11 Full Phase Error: temp table: *** not exist.....	121
2.3.12 Full Phase Error: failed to create new session.....	122
2.3.13 Full Phase Error: load table: *** failed.....	122
2.3.14 Full Phase Error: extract table structure failed!.....	123
2.3.15 Full Phase Error: read table=*** failed.....	123
2.3.16 Full Phase Error: CANNOT UPDATE USER WITH NULL PASSWORD.....	124

2.3.17 Full Phase Error: Access denied for user *** to database ***.....	124
2.3.18 Full Phase Error: The MySQL server is running with the --super-read-only option so it cannot execute this statement.....	125
2.3.19 Full Phase Error: Temporary file write failure.....	125
2.3.20 Full Phase Error: Incorrect prefix key; the used key part isn't a string, the used length is longer than the key part, or the storage engine doesn't support unique prefix keys.....	126
2.3.21 Full Phase Error: Unknown database ***.....	126
2.3.22 Full Phase Error: Access denied; you need (at least one of) the SUPER privilege(s) for this operation.....	127
2.3.23 Full Phase Error: retry structures failed events and Table *** doesn't exist.....	128
2.3.24 Full Phase Error: shard table=*** failed.....	128
2.3.25 Full Phase Error: error when split table shard occur!.....	128
2.3.26 Full Phase Error: Column name 'AUTO_PK_ROW_ID' is reserved.....	129
2.3.27 Full Phase Error: transfer account failed, can not find password from src DB.....	129
2.3.28 Full Phase Error: Failed to add the foreign key constraint '****' to system tables.....	130
2.3.29 Full Phase Error: Too many keys specified; max 64 keys allowed.....	130
2.3.30 Full Phase Error: Unknown collation: 'utf8mb4_0900_ai_ci'.....	131
2.3.31 Full Phase Error: exist some xa transactions for long times, may lack some data for this Job!.....	132
2.3.32 Full Phase Error: Invalid GIS data provided to function st_geometryfromtext'.....	132
2.3.33 Full or Incremental Phase Error: Access denied for user ***.....	133
2.3.34 Full or Incremental Phase Error: binlog is not existed.....	133
2.3.35 Full or Incremental Phase Error: database log download failed.....	134
2.3.36 Full or Incremental Phase Error: Can not read response from server.....	134
2.3.37 Full or Incremental Phase Error: Communications link failure.....	135
2.3.38 Full or Incremental Phase Error: EOF Packet received, master disconnected.....	135
2.3.39 Full or Incremental Phase Error: Extract db create sql failed.....	136
2.3.40 Full or Incremental Phase Error: load database structure failed in source database.....	136
2.3.41 Full or Incremental Phase Error: load table: *** failed.....	137
2.3.42 Full or Incremental Phase Error: Reached end of input stream.....	137
2.3.43 Full or Incremental Phase Error: Read timed out.....	138
2.3.44 Full or Incremental Phase Error: The background process is unavailable.....	138
2.3.45 Full or Incremental Phase Error: Duplicate entry *** for key 'PRIMARY'.....	138
2.3.46 Full or Incremental Phase Error: cause by: Index: ***, Size: ***.....	140
2.3.47 Full or Incremental Phase Error: The offset and file name between src and parser is inconsistency.....	141
2.3.48 Full or Incremental Phase Error: core process is not healthy or crashed.....	141
2.3.49 Full or Incremental Phase Error: table info of table `****` from metadata miss.....	142
2.3.50 Full or Incremental Phase Error: binlog parse fail, data dictionary may be not complete!.....	142
2.3.51 Full or Incremental Phase Error: table *** record field size for insert/delete dml.....	142
2.3.52 Full or Incremental Phase Error: service *** failed, cause by: Unable to connect to DBMS: ***.....	143
2.3.53 Full or Incremental Phase Error: The binlog fetch connection may be interrupted.....	143
2.3.54 Full or Incremental Phase Error: Received error packet: errno - 1047, solstate - HY000 errmsg = Unknown command.....	144
2.3.55 Incremental Phase Error: not equals to target db column count.....	144

2.3.56 Incremental Phase Error: The MySQL server is running with the --super-read-only option.....	145
2.3.57 Incremental Phase Error: you need (at least one of) the SUPER privilege(s) for this operation.....	145
2.3.58 Incremental Phase Error: Can't DROP ***; check that column/key exists.....	146
2.3.59 Incremental Phase Error: Can't find file: *** (errno: 2 - No such file or directory).....	146
2.3.60 Incremental Phase Error: Data truncation: Data too long for column.....	147
2.3.61 Incremental Phase Error: Failed to read file header from.....	147
2.3.62 Incremental Phase Error: Lock wait timeout exceeded.....	148
2.3.63 Incremental Phase Error: Must seek before attempting to read next event.....	148
2.3.64 Incremental Phase Error: Table *** already exists.....	148
2.3.65 Incremental Phase Error: Table *** doesn't exist.....	149
2.3.66 Incremental Phase Error: Table *** not found in database.....	149
2.3.67 Incremental Phase Error: source has more columns than target.....	150
2.3.68 Incremental Phase Error: Unknown storage engine.....	150
2.3.69 Incremental Phase Error: Unknown table.....	151
2.3.70 Incremental Phase Error: You have an error in your SQL syntax.....	151
2.3.71 Incremental Phase Error: not illegal for mariaDb gtid position.....	152
2.3.72 Incremental Phase Error: without PK execute failed.....	152
2.3.73 Incremental Phase Error: Deadlock found when trying to get lock.....	152
2.3.74 Incremental Phase Error: current serverUUID not equals to this session.....	153
2.3.75 Incremental Phase Error: Slave has more GTIDs than the master has, using the master's SERVER_UUID.....	153
2.3.76 Incremental Phase Error: Operation not allowed when innodb_force_recovery > 0.....	154
2.3.77 Incremental Phase Error: filter data in config condition filter error.....	154
2.4 Real-Time Migration and Synchronization from MySQL to TaurusDB.....	155
2.4.1 Full or Incremental Phase Error: Illegal mix of collations (utf8mb4_0900_ai_ci,IMPLICIT) and (utf8mb4_general_ci,IMPLICIT) for operation.....	155
2.5 Real-Time Synchronization from PostgreSQL to PostgreSQL.....	156
2.5.1 Task Startup Error: Initialize logical replication stream failed, the source database may have a long transaction: ****	156
2.5.2 Full Synchronization Error: function *** does not exist.....	156
2.5.3 Full Synchronization Error: relation *** does not exist.....	157
2.5.4 Full Synchronization Error: GC overhead limit exceeded.....	158
2.5.5 Full Synchronization Error: Java heap space.....	158
2.5.6 Full Synchronization Error: column *** of relation *** does not exist.....	159
2.5.7 Full Synchronization Error: column *** does not exist.....	159
2.5.8 Full Synchronization Error: type 'hstore' does not exist.....	160
2.5.9 Full Synchronization Error: type 'geometry' does not exist.....	160
2.5.10 Full Synchronization Error: Check that the hostname and port are correct and that the postmaster is accepting TCP/IP connections.....	161
2.5.11 Full Synchronization Error: invalid locale name.....	162
2.5.12 Full Synchronization Error: password must not equal user name.....	162
2.5.13 Full Synchronization Error: permission denied for schema ***	162
2.5.14 Full or Incremental Phase Error: service *** failed, cause by: Unable to connect to DBMS: ***	163

2.5.15 Full or Incremental Phase Error: Initialize logical replication stream failed, the source database may have a long transaction.....	163
2.5.16 Full or Incremental Phase Error: memory required is *** MB, maintenance_work_mem is *** MB..	164
2.5.17 Full or Incremental Phase Error: temporary file size exceeds temp_file_limit.....	165
2.5.18 Incremental Synchronization Error: Table *** not found in target database.....	165
2.5.19 Incremental Synchronization Error: remaining connection slots are reserved.....	166
2.5.20 Incremental Synchronization Error: PL/pgSQL function *** line *** at SQL statement.....	167
2.5.21 Incremental Synchronization Error: The replication slot does not exist and the task is not started for the first time.....	167
2.6 Real-Time Synchronization with Oracle Serving as the Source.....	168
2.6.1 Full Synchronization Error: has date/datetime: *** which is outside of dest allowed range.....	168
2.6.2 Full or Incremental Phase Error: Got minus one from a read call.....	168
2.6.3 Incremental Synchronization Error: Source supplemental log level is PK/UI. Missing column data at delete+insert on ***".....	169
2.6.4 Incremental Synchronization Error: timeout when get next file log, maybe has been deleted, please check it.....	170
2.6.5 Incremental Synchronization Error: Failed to construct kafka producer.....	170
2.6.6 Incremental Synchronization Error: Topic *** not present in metadata after 300000 ms.....	171
2.7 Real-Time Synchronization with GaussDB Serving as the Source.....	171
2.7.1 Task Startup Error: Initialize logical replication stream failed, the source database may have a long transaction: ****.....	171
2.7.2 Incremental Synchronization Error: The replication slot does not exist and the task is not started for the first time.....	172
2.8 Real-Time DR with MySQL Serving as the Source.....	172
2.8.1 DR Error: A dml without pk write target db fail.....	172
2.9 Backup Migration.....	173
2.9.1 Backup Migration Failed Because Backup Files Cannot Be Found.....	174
2.9.2 Backup Migration Failed Because a Backup Database Cannot Be Found in the Backup Files.....	175
2.9.3 Backup Migration Failed Because the Database with the Same Name Already Exists.....	175
2.9.4 Backup Migration Failed Because an Incremental Backup File Is Used.....	176
2.9.5 Backup Migration Failed Because a Log Backup File Is Used.....	176
2.9.6 Backup Migration Failed Because the Backup File Verification Failed.....	177
2.9.7 Backup Migration Failed Because of Insufficient Space.....	177
2.9.8 Backup Migration Failed Because Database Names Are Not Specified.....	177
2.9.9 Backup Migration Failed Because a Full Backup File Is Used.....	178
2.9.10 Backup Migration Failed Because the LSNs of Incremental Backup Files Are Inconsecutive.....	178
2.9.11 Backup Migration Failed Because the Number of Databases to Be Restored Exceeds the Destination Database Threshold.....	179
2.10 Workload Replay.....	179
2.10.1 Parsing Failed, and a Message Is Displayed Indicating That the OBS Connection Failed.....	179
2.11 Data-Level Comparison.....	180
2.11.1 Data-Level Comparison Error: service SDV failed! cause by: the size of records in one shard[***] of target database, exceeds the max size 200000.....	180
2.11.2 Row Comparison Is Inconsistent, and SQL Execution in the Destination Database Times Out.....	181

1 Solutions to Failed Check Items

1.1 Disk Space

1.1.1 Checking Whether the Destination Database Has Sufficient Storage Space

In the migration phase, DRS uses the row-level parallel migration mode to ensure migration performance and transmission stability. If the source database data is compact, the destination database will use more storage space than the source database after DRS migration. DRS checks the available storage space of the destination database during the pre-check. If the storage space is insufficient, the migration may fail.

Failure Cause

If the storage space of the destination database is insufficient, it is recommended that the size of the destination database disk be set to the smaller value of the following two values:

1. 1.5 or 2.5 times the size of the data to be migrated in the source database.
2. The size of the data to be migrated in the source database plus 200 GB.

The available storage space of the destination database is subject to the information displayed on the page.

Handling Suggestion

- If the pre-check fails, scale up the storage of the destination database or clear data in the destination database and perform the pre-check again.

For details about how to scale up an RDS for MySQL DB instance, see [Scaling Up Storage Space](#).

If you choose to clear data in the destination database, the storage usage decreases within 2 to 3 minutes.

- If the task fails, scale up the storage of the destination database or clear data in the destination database and click **Resume** in the **Operation** column to continue the task.

1.1.2 Checking Whether the Destination Server Has Sufficient Storage Space

Table 1-1 Checking whether the destination server has sufficient storage space

Check Item	Whether the destination server has sufficient storage space
Description	If the destination server's storage space is insufficient, the migration will fail.
Failure Cause and Handling Suggestion	Failure cause: The amount of data in the source database is greater than the remaining storage space of the destination server. Handling suggestion: Modify the synchronization object.

1.2 Database Parameters

1.2.1 Checking Whether the Source Database Binlog Is Enabled

Check whether Binlog is enabled for the source database. During an incremental MySQL migration, Binlog of the source database must be enabled.

Failure Cause

Binlog is not enabled for the source database.

Handling Suggestion

- If the source is an on-premises MySQL database, perform the following operations to enable Binlog.

- a. Run the following command to check whether Binlog is enabled:
`show variables like "log_bin";`

```
mysql> show variables like "log_bin";
***** 1. row *****
Variable_name: log_bin
Value: OFF
1 row in set (0.01 sec)
```

- b. If Binlog is disabled, add `log-bin = mysql-bin` under `[mysqld]` in the MySQL configuration file `my.cnf` or `my.ini`.

```
[mysqld]  
log-bin = mysql-bin
```

- c. Restart the database.

```
mysql> show variables like "log_bin"\G;  
***** 1. row *****  
Variable_name: log_bin  
Value: ON  
1 row in set (0.00 sec)
```

- If the source is an RDS for MySQL DB instance, Binlog is enabled by default and no additional configuration is required. **binlog_format** is set to **row**.
- If the source is a TiDB database, add **enable=true** under **[binlog]** in the database configuration file and restart the database to apply the settings.

1.2.2 Checking Whether the Source Database Binlog Is Row-Based

Check whether the source database binlog format is correct. The binlog of the source database must be enabled and the row-based format must be used during incremental MySQL migration.

Failure Cause

The source database binlog is not row-based.

Handling Suggestion

- If the source database is an on-premises MySQL database, perform the following operations to change the binlog format of the source database:
 - Method 1: You can modify the **my.cnf** or **my.ini** configuration file and restart the database.
binlog_format=row
 - Method 2: Stop all service connections.
set global binlog_format='ROW'
Modify the **my.cnf** or **my.ini** configuration file.
binlog_format=row

In the **ROW** format, the log growth rate increases, which may occupy more disk space.

- If the source database is an RDS for MySQL instance, binlog is enabled by default and binlog_format is set to row. No additional configuration is required.

1.2.3 Checking Whether the expire_logs_days Value in the Source Database Is Correct

During MySQL migration, you can set **expire_logs_days** to change the binlog retention period. Set **expire_logs_day** to a proper value to ensure that the binlog does not expire before data transfer resumes. This ensures that services can be recovered after interruption.

Failure Cause

The `expire_logs_days` parameter of the source database is set to 0.

Handling Suggestion

- If the source database is an RDS for MySQL DB instance, set `expire_logs_days` to a proper value by following the instructions provided in [Setting a Local Retention Period for RDS for MySQL Binlogs](#).
- If the source database is an on-premises MySQL database, perform the following steps:
 - a. Log in to the server where the MySQL source database is located.
 - b. Run the following command to check the configured binlog retention period:

```
show variables like 'expire_logs_days';
```

Or

```
show variables like 'binlog_expire_logs_seconds';
```
 - c. Manually modify the `my.cnf` configuration file and set the binlog retention period. The following uses three days as an example.

```
expire_logs_days=3
```

Or

```
binlog_expire_logs_seconds=259200;
```
 - d. After the modification, restart the source database during a non-service period.

1.2.4 Checking Whether the Source and Destination Database Character Sets Are Consistent

Check whether the source and destination database character sets are consistent. If the character set of the source database is different from that of the destination database, some data may be garbled or inconsistent.

Failure Cause

The character sets of the source and destination databases are inconsistent.

Handling Suggestion

Change the character set of the source or destination database.

- If a MySQL database is used, perform the following operations:
 - If the database is a self-managed database, run commands to change the character set.
 - i. Run the following command to check the character set of the database:

```
SHOW VARIABLES LIKE "character_set_server"\G;
```

```
mysql> show variables like "character_set_server"\G;
***** 1. row *****
Variable_name: character_set_server
Value: utf8
1 row in set (0.00 sec)
```

- ii. Modify the character set of the source database server.

```
SET character_set_server='utf8';
```

```
mysql> set character_set_server='utf8';  
Query OK, 0 rows affected (0.00 sec)
```

- If the database is an RDS for MySQL DB instance, modify the **character_set_server** parameter.
- If a PostgreSQL database is used, perform the following operations:
 - If the database is a self-managed database, run commands to change the character set.
 - i. Run the following command to check the character set of the database:

```
show server_encoding;
```
 - ii. Modify the character set of the source database server.

```
set server_encoding='utf8';
```
 - If the database is an RDS for PostgreSQL DB instance, modify the **server_encoding** parameter.

For details about how to modify character set parameters for other types of databases, see the usage guide of the corresponding database.

1.2.5 Checking Whether the Source Database **server_id** Meets the Incremental Migration Requirements

During an incremental MySQL migration, the source database **server_id** must meet the following requirements:

- If the source database version is MySQL 5.6 or earlier, the value of **server_id** ranges from 2 to 4294967296.
- If the source database version is MySQL 5.7 or later, the value of **server_id** ranges from 1 to 4294967296.

Failure Cause

The **server_id** value of the source database does not meet requirements.

Handling Suggestion

Step 1 Log in to the server where the MySQL source database is located.

Step 2 Run the following SQL statement to check the value of **server_id**:

```
show variables like '%server_id%';
```

Step 3 If the value of **server_id** does not meet requirements, run the following command to change the value of **server_id**:

```
set global server_id=n
```

The value **n** indicates the source database **server_id**. If the source database version is MySQL 5.6, the value **n** ranges from 2 to 4294967296. Otherwise, the value **n** ranges from 1 to 4294967296.

Step 4 After the modification, perform the pre-check again.

----End

1.2.6 Checking Whether the Source and Destination Database Table Names Are Consistent in Case Sensitivity

During MySQL migration, the value of **lower_case_table_names** of the source database is inconsistent with that of destination database. If the value of **lower_case_table_names** is **0**, the database is case sensitive. If the value of **lower_case_table_names** is **1**, the database is case insensitive and all uppercase letters are converted to lowercase letters for storage. If the **lower_case_table_names** values of the source and destination databases are different, and a database or table is named using uppercase letters, the task may fail.

Failure Cause

The **lower_case_table_names** values of the source and destination databases must be the same.

Handling Suggestion

- If the database is an RDS for MySQL instance, change the value of **lower_case_table_names** by referring to [Tables Failed to Be Found After Case-Sensitivity Setting Changes for RDS for MySQL](#).
- If the database is an on-premises MySQL database, perform the following steps:
 - a. Log in to the server where the MySQL source database is located.
 - b. Add **lower_case_table_names=n** under [mysqld] in the **my.cnf** file.
 - c. After the modification, restart the source database during a non-service period.

1.2.7 Checking Whether the Source Database Contains Object Names with Non-ASCII Characters

During MySQL migration, if the source database contains object names with non-ASCII characters, the migration may fail.

Failure Cause

The source database contains object names with non-ASCII characters.

Handling Suggestion

Change the object names containing non-ASCII characters in the source database and perform the pre-check again.

1.2.8 Checking Whether the `time_zone` Values of the Source and Destination Databases Are the Same

If the `time_zone` values of the source and destination databases are different, the migration may fail.

Failure Cause

The `time_zone` or `system_time_zone` values of the source and destination databases must be the same.

Handling Suggestion

Change the value of `time_zone (timezone)` or `system_time_zone` of the source database to be the same as that of the destination database.

- To change the value of `time_zone` in the MySQL database, perform the following steps:
 - If the database is a self-managed database, run commands to change the time zone.
 - i. Run the following command to check the time zone of the database:

```
SHOW VARIABLES LIKE "%time_zone%";
```
 - ii. Run the following command to change the time zone:

```
SET time_zone = 'Timezone';
```
 - If the database is an RDS for MySQL DB instance, change the time zone by following the instructions provided in [How Can I Change the Time Zone?](#)
- To change the value of `time_zone` in the Oracle database, perform the following steps:
 - a. Run the following statement to query the value of `time_zone` in the database:

```
SELECT DBTIMEZONE FROM DUAL;
```
 - b. Run the following statement to change the value of `time_zone` in the database:

```
ALTER DATABASE SET TIME_ZONE='Time zone';
```

Example of changing the time zone to GMT+8:

```
ALTER DATABASE SET TIME_ZONE='+08:00';
```
 - c. Restart the database after changing the value of `time_zone`.

```
SQL> shutdown immediate
SQL> startup
```
- To change the value of `time_zone` in the DDM database, perform the following steps:

Log in to the DDM console and change the time zone.

For details about how to change the time zone for other types of databases, see the usage guide of the corresponding database.

1.2.9 Checking Whether the `collation_server` Values of the Source and Destination Databases Are the Same

If the `collation_server` values of the source and destination databases are different, the migration may fail.

Failure Cause

The `collation_server` values of the source and destination databases must be the same.

Handling Suggestion

Change the value of `collation_server` of the source database to be the same as that of the destination database.

- If the database is a self-managed MySQL database, run commands to change the value.
 - a. Run the following command to check the value:

```
SHOW VARIABLES LIKE "collation_server";
```
 - b. Run the following command to change the value:

```
SET collation_server='utf8_unicode_ci';
```
- If the database is an RDS for MySQL DB instance, change the value of the `collation_server` parameter by referring to [Modifying Parameters of an RDS for MySQL Instance](#).

For details about how to change the value of `collation_server` for other types of databases, see the usage guide of the corresponding database.

1.2.10 Checking Whether the `SERVER_UUID` Values of the Source and Destination Databases Are the Same

During MySQL migration, if the `SERVER_UUID` values of the source and destination databases are the same, the migration may fail.

Failure Cause

The `SERVER_UUID` values of the source and destination databases are the same.

Handling Suggestion

Check whether the source and destination databases are the same MySQL database. If yes, change the source or destination database.

1.2.11 Checking Whether the `SERVER_ID` Values of the Source and Destination Databases Are Different

During MySQL migration, if the `SERVER_ID` values of the source and destination databases are the same, the migration may fail.

Failure Cause

The **SERVER_ID** values of the source and destination databases are the same.

Handling Suggestion

Change **SERVER_ID** of the source and destination databases to different values.

1.2.12 Checking Whether the Source Database Contains Invalid `sql_mode` Values

During MySQL migration, the source database cannot have invalid values of **sql_mode**. Otherwise, the migration may fail.

Failure Cause

The **sql_mode** value of the source database cannot be **no_engine_substitution**.

Handling Suggestion

Change **sql_mode** of the source database to a proper value.

- If the database is a self-managed MySQL database, run the following command to change the value.
`SET sql_mode = 'New value';`
- If the database is an RDS for MySQL DB instance, change the value of the **sql_mode** parameter by referring to [Modifying Parameters of an RDS for MySQL Instance](#).

1.2.13 Checking Whether the `sql_mode` Values of the Source and Destination Databases Are the Same

Check whether the **sql_mode** values of the source and destination databases are the same. If they are different, the migration may fail. You are advised to change them to the same value.

Failure Cause

The **sql_mode** values of the source and destination databases must be the same.

Handling Suggestion

Change the value of **sql_mode** of the source database to be the same as that of the destination database.

- If the database is a self-managed MySQL database, run commands to change the value.
`SET sql_mode = 'New value';`
- If the database is an RDS for MySQL DB instance, change the value of the **sql_mode** parameter by referring to [Modifying Parameters of an RDS for MySQL Instance](#).

If MyISAM tables are to be migrated, the **sql_mode** values in the destination database cannot contain **no_engine_substitution**.

For details about how to change the value of **sql_mode** for other types of databases, see the usage guide of the corresponding database.

1.2.14 Checking Whether the **sql_mode** Value in the Destination Database Is Not **no_engine**

During MySQL migration, if the MyISAM tables are included in the migration objects, the **sql_mode** value in the destination database cannot be **no_engine_substitution**. Otherwise, the migration fails.

Failure Cause

In the destination database, the **sql_mode** value is **no_engine_substitution**.

Handling Suggestion

In the destination database, set **sql_mode** to a value other than **no_engine_substitution**.

- If the database is a self-managed MySQL database, run the following command to change the value.
`SET sql_mode = 'New value';`
- If the database is an RDS for MySQL DB instance, change the value of the **sql_mode** parameter by referring to [Modifying Parameters of an RDS for MySQL Instance](#).

1.2.15 Checking Whether the **innodb_strict_mode** Values of the Source and Destination Databases Are the Same

Check whether the **innodb_strict_mode** values of the source and destination databases are the same. If they are different, the task may fail. You are advised to change them to the same value.

The database parameter **innodb_strict_mode** specifies whether to enable the strict check mode. If **innodb_strict_mode** is set to **ON**, InnoDB returns errors rather than warnings when checking for invalid CREATE TABLE, ALTER TABLE, or CREATE INDEX statement. If **innodb_strict_mode** is set to **OFF**, InnoDB uses the default syntax when checking for invalid syntax. For example, in MySQL 5.7.34, if **innodb_strict_mode** is set to **ON** and the table creation statement **CREATE TABLE t1(c1 int, c2 varchar(32)) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=FIXED** is executed, the error **Table storage engine for 't1' doesn't have this option** is reported.

In a DRS task, if **innodb_strict_mode** of the source database is set to **OFF** and **innodb_strict_mode** of the destination database is set to **ON**, no error will be reported when DDL statements are executed in the source database due to syntax errors, and an error will be reported when data is synchronized to the destination database. As a result, the task fails.

Failure Cause

The **innodb_strict_mode** values of the source and destination databases must be the same.

Handling Suggestion

Change the value of **innodb_strict_mode** of the source database to be the same as that of the destination database.

- If the database is a self-managed MySQL database, run commands to change the value.
 - a. Run the following command to check the value:

```
SHOW VARIABLES LIKE "innodb_strict_mode";
```
 - b. Run the following command to change the value:

```
SET GLOBAL innodb_strict_mode = <value>;
```

To disable this option, set **<value>** to **0**. To enable this option, set **<value>** to **1**.
- If the database is an RDS for MySQL DB instance, change the value of the **innodb_strict_mode** parameter by referring to [Modifying Parameters of an RDS for MySQL Instance](#).

For details about how to change the value of **innodb_strict_mode** for other types of databases, see the usage guide of the corresponding database.

1.2.16 Checking Whether the max_wal_senders Value of the Source Database Is Correctly Configured

During PostgreSQL synchronization, the **max_wal_senders** value of the source database must be greater than the number of used replication slots. Otherwise, the synchronization may fail.

Failure Cause

The **max_wal_senders** value of the source database is less than or equal to the number of used replication slots.

Handling Suggestion

Set **max_wal_senders** to a value greater than the number of used replication slots and restart the database to apply the changes. Run the following command to query the number of used replication slots in the current database:

```
select count(1) from pg_replication_slots;
```

1.2.17 Checking Whether the WAL_LEVEL Value in the Source Database Is Correct

During synchronization from PostgreSQL or GaussDB, the **wal_level** parameter of the source database must be set to **logical**. If the value is not **logical**, the incremental logs of the source database cannot be logically decoded. As a result, incremental synchronization cannot be performed.

Failure Cause

- The **wal_level** value of the source database is not **logical**.
- The source or destination database version is not supported.

Handling Suggestion

- Change the value of **wal_level** in the source database to **logical**. For details, see the following:
 - Run **alter system set wal_level = logical** in the source database as a super user and restart the database to apply the changes.
 - Alternatively, modify the **postgresql.conf** configuration file, set **wal_level** to **logical**, and restart the database to apply the changes.
 - You can set GaussDB parameters on the database management page and restart the database to apply the settings.
- Change the source or destination database version by referring to [Supported Databases](#).

1.2.18 Checking Whether the MAX_REPLICATION_SLOTS Value in the Source Database Is Correct

During synchronization from PostgreSQL or GaussDB, the **max_replication_slots** value of the source database must be greater than the number of used replication slots. Otherwise, the synchronization may fail.

Failure Cause

The **max_replication_slots** value of the source database is less than or equal to the number of used replication slots.

Handling Suggestion

Set **max_replication_slots** to a value greater than the number of used replication slots and restart the database to apply the changes. Run the following command to query the number of used replication slots in the current database:

```
select count(1) from pg_replication_slots;
```

1.2.19 Checking Whether the Source Database Is on Standby

For a full+incremental synchronization task where the source database is a PostgreSQL database, the source database cannot be a standby database. Otherwise, the incremental synchronization cannot be performed. For a full synchronization task, the source database can be a standby database, but **hot_standby_feedback** must be set to **on**. Otherwise, the synchronization may fail.

Failure Cause

- In a full+incremental synchronization task, the source database is a standby database, and incremental synchronization cannot be performed.
- For a full synchronization task, the source database is a standby database, and **hot_standby_feedback** is set to **off**.

Handling Suggestion

- For a full+incremental synchronization task, change the source database to a primary database.

- For a full synchronization task, change the source database to a primary database. Alternatively, change the **hot_standby_feedback** value of the source database to **on** before starting full synchronization. After the full synchronization is complete, change the value of this parameter to **off**.

1.2.20 Checking Whether the `log_slave_updates` Value in the Source Database Is Correct

To ensure that DRS can obtain all binlogs during MySQL migration, enable the `log_slave_updates` parameter.

Failure Cause

The `log_slave_updates` parameter of the source database is set to **OFF**.

Handling Suggestion

Step 1 Log in to the server where the MySQL source database is located.

Step 2 Run the following SQL statement to check whether the value of `log_slave_updates` is **ON**:

```
show variables like '%log_slave_updates%';
```

Step 3 Add the following content followed by `[mysqld]` in the MySQL configuration file `my.cnf`:

```
log_slave_updates=1
```

Step 4 After the modification, restart the source database during a non-service period.

----End

1.2.21 Checking Whether the `BLOCK_SIZE` Value of the Source Database Is the Same as That of the Destination Database

During PostgreSQL synchronization, the `BLOCK_SIZE` value of the destination database must be greater than or equal to that of the source database. Otherwise, the synchronization may fail.

Failure Cause

The `BLOCK_SIZE` value of the destination database is less than that of the source database.

Handling Suggestion

Change the `BLOCK_SIZE` value of the source or destination database to ensure that value of the destination database must be greater than or equal to that of the source database.

1.2.22 Checking Whether the `binlog_row_image` Value is **FULL**

During MySQL migration and synchronization, the `binlog_row_image` parameter of the source database must be set to **FULL**. Otherwise, the migration fails.

Failure Cause

The **binlog_row_image** value of the source database is not **FULL**.

Handling Suggestion

- If the source database is an RDS DB instance on the cloud, change the value of **binlog_row_image** to **FULL** on the RDS console, and then restart the source database.
- If the source database is an on-premises database, perform the following steps:
 - a. Log in to the server where the MySQL source database is located.
 - b. Manually change the value of **binlog_row_image** in the **my.cnf** configuration file to **FULL** and save the file.

```
binlog_row_image=full
```
 - c. To ensure a successful task, restart the source database during off-peak hours.

1.2.23 Checking Whether the Transaction Isolation Levels are Consistent

During MySQL migration, the transaction isolation levels of the source and destination databases must be the same.

Failure Cause

The transaction isolation levels of the source and destination databases are inconsistent.

Handling Suggestion

Change the value of the database isolation level parameter (**tx_isolation** or **transaction_isolation**) to ensure that the transaction isolation levels of the source and destination databases are the same.

- MySQL 5.7 and 5.6: **tx_isolation**
- MySQL 8.0: **transaction_isolation**

1.2.24 Checking Whether the lc_monetary Values of the Source and Destination Databases Are the Same

During PostgreSQL synchronization, check whether the **lc_monetary** values of the source and destination databases are the same. If they are inconsistent, the synchronization fails.

Failure Cause

The **lc_monetary** values of the source and destination databases are different.

Handling Suggestion

Change the value of **lc_monetary**.

- For a self-managed database, run the following command to change the value.

```
SET lc_monetary TO ='Currency format'
```
- For an RDS for PostgreSQL instance, change the parameter by referring to [Modifying Parameters of an RDS for PostgreSQL Instance](#).

1.2.25 Checking Whether the Source Database Contains Trigger Names with Non-ASCII Characters

During MySQL migration, if the source database contains trigger names with non-ASCII characters, the migration will fail.

Failure Cause

The source database cannot contain trigger names with non-ASCII characters.

Handling Suggestion

- Method 1: Click **Previous** to return to the **Select Migration Type** page. Do not select triggers whose names contain non-ASCII characters.
- Method 2: Change the trigger name in the source database and perform the pre-check again.

1.2.26 Checking Whether the Source Database Collections Contain More Than 10 Indexes

During MongoDB migration, the number of indexes affects the migration duration. DRS checks whether the source database contains collections with more than 10 indexes. If yes, the migration speed will be affected. Check whether all indexes need to be migrated.

Item to Be Confirmed

The source database has collections containing more than 10 indexes, which are migrated slowly.

Handling Suggestion

The number of indexes affects the migration duration. Check whether all indexes need to be migrated. If an index does not need to be migrated, delete the index before starting the migration.

Run the following command to delete an index:

```
run the db.<Collection name>.dropIndex(<Index name>)
```

1.2.27 Checking Whether the Source Database Collections Contain TTL Indexes

During MongoDB migration, due to inconsistency of time zones and clocks between source and destination databases, migrating TTL indexes will cause data inconsistency. DRS checks whether the source database contains collections with

TTL indexes. If yes, delete the TTL indexes or do not migrate the collections with TTL indexes.

Item to Be Confirmed

If the objects to be migrated contain TTL indexes, due to inconsistency of time zones and clocks between source and destination databases, migrating TTL indexes will cause data inconsistency.

Handling Suggestion

If data consistency is required, delete TTL indexes. Alternatively, do not migrate the collections containing TTL indexes.

Run the following command to delete an index:

```
run the db.Collection name.dropIndex(Index name)
```

1.2.28 Checking Whether `log_bin_trust_function_creators` Is Set to On in Both the Source and Destination Databases

During the out-of-cloud migration from MySQL to MySQL, the `log_bin_trust_function_creators` value of the source database must be the same as that of the destination database. If the source database supports user-defined functions (UDFs) but the destination database does not, change the `log_bin_trust_function_creators=off` parameter of the destination database to `log_bin_trust_function_creators=on`. If the parameters of the source and destination are different, the migration may fail.

Item to Be Confirmed

The destination database does not support user-defined functions.

Handling Suggestion

In the `my.cnf` file of the destination database, check whether `log_bin_trust_function_creators=on` exists. If it does not exist, add `log_bin_trust_function_creators=on` and restart the database for the modification to take effect.

1.2.29 Checking Whether GTID Is Enabled for the Source Database

During data migration, GTID must be enabled for the source database. If GTID is disabled for the source database and a primary/standby switchover is performed on the source database, the task may fail.

Failure Cause

GTID is disabled on the source database.

Handling Suggestion

- If the source database is an RDS for MySQL DB instance, GTID is enabled by default and cannot be disabled. If GTID is disabled, contact RDS for MySQL O&M personnel.
- If the source database version is MySQL 5.5, GTID cannot be set, and DRS cannot be used for synchronization and DR tasks. Create a migration task or contact O&M personnel.
- If the source database version is MySQL 5.6 and later, set the following parameters to enable GTID in the database configuration file. Then, restart the database for the modifications to take effect.

Parameters to be configured:

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

1.2.30 Checking Whether GTID Is Enabled for the Destination Database

During real-time DR, GTID must be enabled for the destination database. If GTID is disabled for the destination database and a primary/standby switchover is performed on the destination database, the task may fail.

Failure Cause

GTID is disabled on the destination database.

Handling Suggestion

- If the destination database is an RDS for MySQL DB instance, GTID is enabled by default and cannot be disabled. If GTID is disabled, contact RDS for MySQL O&M personnel.
- If the destination database version is MySQL 5.5, GTID cannot be set, and DRS cannot be used for disaster recovery. Create a migration task or contact O&M personnel.
- If the destination database version is MySQL 5.6 or later, set the following parameters to enable GTID in the database configuration file. Then, restart the database for the modifications to take effect.

Parameters to be configured:

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

1.2.31 Checking Whether log_bin_trust_function_creators Is Set to On in the Destination Database

During the migration from RDS for MySQL to MySQL out of the cloud, the destination database does not support user-defined functions.

Item to Be Confirmed

The destination database does not support user-defined functions.

Handling Suggestion

In the **my.cnf** file of the destination database, check whether **log_bin_trust_function_creators=on** exists. If it does not exist, add **log_bin_trust_function_creators=on** and restart the database for the modification to take effect.

1.2.32 Checking Whether the Values in the Source Oracle Database Are Out of the Ranges of the MySQL Database

During heterogeneous data synchronization, DRS checks the compatibility. You need to confirm the detailed item.

Item to Be Confirmed

- Ensure that the columns that use character strings as primary keys or unique keys in the table to be migrated do not contain spaces. If the source database uses character strings as primary keys or unique keys, it will support spaces, but the destination database will not. As a result, there may be data inconsistencies or the migration may fail.
- Due to differences between the source and destination databases, the migration will fail if the values of the following data types in the source database are out of the ranges of the destination database: number, int, float, double, date, and timestamp.
- When a non-accurate data type is used as a primary key, there may be data inconsistencies due to database compatibility.

Handling Suggestion

Check whether the preceding problems occur. You need to determine whether to solve the problems as required. You can also confirm the problems and go to the next step.

1.2.33 Checking Whether the **max_allowed_packet** Value of the Destination Database Is too Small

In MySQL or MariaDB migration and synchronization scenarios, if the value of **max_allowed_packet** of the destination database is less than 100 MB, data cannot be written to the destination database. As a result, the full migration fails.

Failure Cause

The **max_allowed_packet** value of the destination database is too small, which may cause data fails to be written during the migration.

Handling Suggestion

Set the **max_allowed_packet** value greater than 100 MB in the destination database.

- For a self-managed database, run the following command to change the value.

```
SET GLOBAL max_allowed_packet=Size;
```

- For an RDS for MySQL instance, change the parameter by referring to [Modifying Parameters of an RDS for MySQL Instance](#).

1.2.34 Checking Whether the Source Database User Has the Permission to Parse Logs

During an incremental synchronization with Oracle serving as the source database, DRS checks the permissions of the source database user. If the source database user does not have the log parsing permission, the incremental migration will fail.

Failure Cause

- The source database user does not have the EXECUTE_CATALOG_ROLE role.
- The source database user does not have the permission to parse logs.

Handling Suggestion

Assign the required role or permission to the user and perform the check again.

- Run the **GRANT EXECUTE_CATALOG_ROLE TO *UserName*** command to assign the EXECUTE_CATALOG_ROLE role to a user.
- Run the **GRANT LOGMINING TO *UserName*** command to grant the log parsing permission to a user.

1.2.35 Checking Whether the ExpireLogsDays Value Is 0

In MySQL synchronization and DR scenarios, if the value of **expire_logs_days** is 0, the task may fail.

Failure Cause

If **expire_logs_days** is set to **0** in the source database, operations such as startup and flush logs will trigger binlog clearance and result in a task failure.

Handling Suggestion

Run the following command on the source database to set the binlog retention period: In the following command, **value** indicates the binlog retention period. The value is an integer greater than 0.

```
set global expire_logs_days=value
```

1.2.36 Checking Whether the Source Database Character Set Is Supported

During a data synchronization with Oracle serving as the source database, DRS checks the character set of the source database. If the character set of the source database is not supported, data synchronization may fail.

Failure Cause

The source database character set is not supported. A synchronization task from Oracle only supports the following character sets: ZHS16GBK, AL32UTF8, UTF8,

US7ASCII, WE8MSWIN1252, WE8ISO8859P1, WE8ISO8859P2, WE8ISO8859P4, WE8ISO8859P5, WE8ISO8859P7, WE8ISO8859P9, WE8ISO8859P13, WE8ISO8859P15.

Handling Suggestion

Go back to the connection test page and select a source database with supported character sets or change the character set of the source database.

1.2.37 Checking Whether the Length Table and Field Names Is Not Supported

For synchronization from DDM to Oracle or MySQL to Oracle, the length of the source database table name or field name cannot exceed 30 characters. Otherwise, the synchronization fails.

Failure Cause

There are tables or fields whose name lengths exceed 30 characters in the source database.

Handling Suggestion

- Method 1: Click **Previous** to return to the object selection page and deselect the tables.
- Change the lengths of the table or field names fewer than 30 characters.

1.2.38 Checking Whether the Databases and Tables Exist

Check whether the task objects meet the migration or synchronization requirements.

Failure Cause

- During object file import, there are databases and tables that do not exist in the source database. The synchronization fails.
- The selected tables contain identifier columns, but the destination database does not support identifier columns. As a result, data synchronization fails.

Handling Suggestion

- Remove these objects that do not exist in the source database and import the file again.
- Click **Previous** to return to the object selection page and deselect the tables containing identifier columns.

1.2.39 Checking Whether the Supplemental Log Level of the Source Database Meets Requirements

During an incremental synchronization with Oracle serving as the source database, DRS checks whether the source database has the supplemental logging level permission. The supplemental logging level of the source Oracle database does not meet requirements. The synchronization fails.

Failure Cause

The supplemental logging level of the source Oracle database does not meet requirements.

Handling Suggestion

In the source database, perform either of the following operations:

- Enable all-level (database-level) supplemental logging: **alter database add supplemental log data (all) columns**
- Enable minimal-level supplemental logging in the source Oracle database: **alter database add supplemental log data**. Then run the following command to enable all-level (table-level) supplemental logging for each to-be-synchronized table: **alter table TABLE_NAME add supplemental log data(all) columns**

1.2.40 Checking Whether the Length of the Source Database Object Names Exceeds the Limit

The length of the destination object names is limited. If the length of the object name in the source database exceeds the limit, the synchronization may fail.

Failure Cause

The length of source object names mapped to the destination ones exceeds the upper limit.

Handling Suggestion

- Method 1: Click **Previous** to return to the object selection page and delete the objects from the synchronization objects.
- Method 2: If the objects are tables, change the table names to meet the mapping conditions.

1.2.41 Checking Whether session_replication_role of the Destination Database Is correctly Set

During PostgreSQL synchronization, the **session_replication_role** parameter of the destination database is not set to **replica**. Data synchronization may fail when the synchronized table has associated foreign key constraints or triggers.

Failure Cause

The **session_replication_role** parameter of the destination database is not set to **replica**.

Handling Suggestion

Before starting the synchronization task, set **session_replication_role** of the destination database to **replica**. After the synchronization is complete, change the value of this parameter to **origin**.

- For a self-managed PostgreSQL database, refer to the following SQL command:
`SET session_replication_role TO 'replica';`
- For an RDS for PostgreSQL instance, change the parameter by referring to [Modifying Parameters of an RDS for PostgreSQL Instance](#).

1.2.42 Checking the Database Compatibility Type

Check the compatibility type of the destination database. If the database type is incompatible, data may contain garbled characters or the synchronization may fail.

Failure Cause

- During synchronization from MySQL to GaussDB, the destination database is incompatible with MySQL.

Handling Suggestion

- Use a destination GaussDB instance that is compatible with MySQL.
If the destination instance is a distributed instance, run the following statement to create a compatible database: `CREATE DATABASE mysql_compatible_db DBCOMPATIBILITY 'MYSQL';`
If the destination instance is a primary/standby instance, run the following statement to create a compatible database: `CREATE DATABASE mysql_compatible_db DBCOMPATIBILITY 'B';`

1.2.43 Checking Whether the Collation of the Destination Database Is Correct

The primary key or unique key in the destination database contains a collation ended in `_ci`. The collation ending in `_ci` is case insensitive, so an error indicating duplicate keys may be reported during synchronization and cause the synchronization to fail.

Failure Cause

The destination database collation is not supported.

Handling Suggestion

Change the destination database collation to a case-sensitive collation (not ending with `_ci`).

1.2.44 Checking Whether the Index Name Is Supported

Check whether an index name of the synchronization object complies with the specifications. The index name rules are as follows:

- Cannot contain uppercase letters.
- Cannot contain `/*? "<>|#` and spaces.
- Can contain colons (`:`).

- Cannot start with a hyphen (-), underscore (_), or plus sign (+).
- Cannot contain periods (.) or ellipsis (...).
- Can contain up to 255 characters.

Failure Cause

There are index names that do not meet requirements.

Handling Suggestion

Return to the **Set Synchronization Task** page, select the tables that meet the requirements. Alternatively, create indexes required in the destination table.

1.2.45 Checking Whether Tables Structures Are Consistent

If you select only data for synchronization, DRS checks the consistency of table structures between the source and destination databases.

Failure Cause

- The number of columns in the destination table is less than that in the source table.
- If the source and destination databases are different, the column names in the source table are converted to lowercase letters for comparison with those in the destination table.
- Column names contain spaces or special characters.

Handling Suggestion

Return to the **Set Synchronization Task** page, select the tables that meet the requirements. Alternatively, modify columns required in the destination table. Ensure that the source table columns are a subset of the destination columns.

1.2.46 Checking Whether Existing Data Meets the Constraints

The constraint type of the source database table does not check whether existing data meets constraints. Some data in the source table may not meet the constraints, and the destination database may not support the constraint types. As a result, data transmission fails or data is lost because the data written to the destination database fails the constraint check.

Item to Be Confirmed

Some check item constraints are not met.

Handling Suggestion

Ensure that all data in the corresponding table meets the current constraints. If any data does not meet the constraints, consider not configuring constraints in the destination (contact DRS online support to skip the table structure consistency check), or process the source data to ensure that the data meets the constraints.

1.2.47 Checking the Additional Column of the Destination Database

Check whether the additional column is added to the additional column list of the destination database. If no additional column is added, the incremental synchronization task fails.

Failure Cause

Some columns are missing in the additional column processing table in the destination database.

Handling Suggestion

Add the missing additional columns to the destination database tables. Reference statement:

```
ALTER TABLE `database`.`table` ADD COLUMN `column_name`  
column_definition;
```

1.2.48 Checking Whether Implicit Primary Key Check Is Enabled for the Primary and Standby Databases

Check whether **create_default_primary_key** is enabled for the source or destination database. The **create_default_primary_key** parameter of the source or destination database is enabled. As a result, the primary/standby switchover may fail or data may be inconsistent. You are advised to disable **create_default_primary_key**. If this parameter is disabled, the task may still fail after a primary/standby switchover. The possible cause is that a table without a primary key is created when this parameter is enabled.

Item to Be Confirmed

The **create_default_primary_key** parameter is enabled for the source or destination database.

Handling Suggestion

This parameter was deprecated, but in some DB instances of earlier versions, this parameter may be enabled. As a result, an error may occur during the primary/standby switchover, which may cause the task to fail or data inconsistency, you are advised to disable this parameter or ensure that the source database does not contain tables that do not have primary keys.

1.2.49 Checking Whether the Source Table Replication Attribute Is Correct

The replica identity attribute of the source database table must be **FULL**. If the source database contains tables whose replication attribute is not **Full**, incremental synchronization may fail.

Failure Cause

- The selected source database table contains the primary key column, but the replication attribute is not FULL.
- The source database contains tables whose replication attribute is not Full. As a result, incremental synchronization may fail.

Handling Suggestion

Change the replication attribute of the table to FULL. Run the following statement in the source database:

```
ALTER TABLE table_name REPLICA IDENTITY FULL;
```

1.2.50 Checking Whether the Specified Replication Slot Exists in the Source Database

The replication slot with the specified name is automatically created after the task is started and cannot be the same as an existing replication slot in the source database.

Failure Cause

The replication slot with a specified name already exists in the source database.

Handling Suggestion

Delete the replication slot with the same name from the source database, or specify a replication slot name different from the existing one in the source database.

1.2.51 Checking Whether the MongoDB Instance Type Matches the Migration Mode

Check whether the MongoDB instance type matches the migration mode. If not, the migration fails.

Failure Cause

During DRS task creation, the source DB instance type is set to **Cluster**, but the source database is not a cluster.

Handling Suggestion

If the source DB instance type is set to **Cluster**, ensure that the source database is a cluster database.

1.2.52 Checking the Physical Standby Database

During an incremental synchronization with Oracle serving as the source database, DRS checks whether the source database is a physical standby database.

Item to Be Confirmed

- The source database is a physical standby database, where data of the LOB type cannot be parsed.
- The physical standby database does not generate logs. It replicates logs from the primary database. Check whether supplemental logging of the primary database meets the incremental synchronization requirements.

Handling Suggestion

- Change the Oracle startup mode and restart the Oracle database.
- Check whether supplemental logging of the primary database meets the incremental synchronization requirements. For details, see [How Do I Check Supplemental Logging of the Source Oracle Database?](#)

1.2.53 Checking Whether the Case Sensitivity of the Destination Database Is Configured

Check whether the value of `lower_case_table_names` in the destination database meets requirements. If the value of `lower_case_table_names` is `1`, the database and table names are case insensitive. The databases or tables whose names contain uppercase letters cannot be migrated.

Failure Cause

The value of destination database parameter `lower_case_table_names` is `1`, and the names of the selected databases or tables contain uppercase letters.

Handling Suggestion

- If the destination database is a self-built database, add `lower_case_table_names=0` under `[mysqld]` in the MySQL configuration file `my.cnf` and then restart the destination database.
- If the destination database is an RDS instance, modify the parameter on the RDS management console. If the modification fails, contact customer service.

1.2.54 Checking Whether CDC is Enabled for Tables in the Source Database

Check whether CDC is enabled for the source Microsoft SQL Server database. Incremental synchronization of the source Microsoft SQL Server database is based on the CDC capability provided by SQL Server. If CDC of the source SQL Server database is disabled, incremental synchronization will be affected.

Failure Cause

CDC is not enabled for the tables to be synchronized in the source database.

Handling Suggestion

Enable CDC for the preceding table in the source database by following the instructions provided in the [official Microsoft SQL Server documentation](#).

1.2.55 Checking Whether the CDC Retention Period in the Source Database Is Long Enough

Check whether the CDC retention period in the source Microsoft SQL Server database is long enough. If the CDC data is retained less than one day in the source database, incremental synchronization will be abnormal. Change the retention period to 1440 minutes (one day) or longer. The recommended value is 4320 minutes (three days).

Failure Cause

If the CDC data is retained less than one day in the source database, incremental synchronization will be abnormal.

Handling Suggestion

Change the CDC retention period to 1440 minutes (one day) or longer. The recommended value is 4320 minutes (three days).

For details, see the following statements:

```
EXECUTE sys.sp_cdc_change_job  
    @job_type = N'cleanup',  
    @retention = 4320;
```

1.2.56 Checking Whether the Source and Destination Databases Have Different Computer Names

If the source and destination databases have the same computer name, it does not meet the migration requirements.

Failure Cause

The source and destination databases cannot have the same computer name.

Handling Suggestion

Change the computer name of the source database and restart the computer for the modification to take effect.

1.2.57 Checking Whether the Length of the Source Database Name Exceeds 64

Check whether source database name contains more than 64 characters. If yes, the migration fails.

Failure Cause

The source database name cannot contain more than 64 characters.

Handling Suggestion

Change the length of the source database names.

1.2.58 Synchronization Object Name Check

If the source database contains database names, schema names, or table names that do not meet requirements, the migration will fail.

Failure Cause

The source database contains database names, schema names, or table names that do not meet requirements. The database names, schema names, and table names in the source database can contain only letters, underscores (_), hyphens (-), and digits.

Handling Suggestion

Change the object names that do not meet requirements.

1.2.59 Checking Whether the Source Database Contains Disabled Clustered Indexes

If the source database contains disabled clustered indexes, the migration may fail.

Failure Cause

The source database contains disabled clustered indexes.

Handling Suggestion

Run the following command to enable the clustered indexes:

```
ALTER INDEX [Index name] ON [Table name] REBUILD
```

1.2.60 Checking Whether the Source Database Is Empty

If the source instance does not contain any database, the migration cannot be performed.

Failure Cause

- The source database fails to be connected.
- The source instance is empty.

Handling Suggestion

- Connect to the source database and perform the pre-check again.
- Change the source database or create objects in the source database. Ensure that the source database is not empty before the migration.

1.2.61 Checking Whether the Source Database Uses the Full Recovery Model

Check whether the source database uses the full recovery model.

Failure Cause

A database does not use the full recovery model in the source database.

Handling Suggestion

Run the following SQL statements on each database that does not use the full recovery model:

```
USE [master]
GO
ALTER DATABASE [Database name] SET RECOVERY FULL WITH NO_WAIT
GO
```

1.2.62 Checking the Synchronization Objects

Check whether the source database objects meet the migration or synchronization requirements.

Failure Cause

- The source database does not contain the objects to be synchronized.
- The source database is temporarily unavailable.
- For a task with Oracle serving as the source database, a schema name, table name, or column name of the selected table exceed 30 characters.
- For a task with Microsoft SQL Server serving as the source database, the number of tables to be synchronized in a single task exceeds 1000.
- An internal error occurred.

Handling Suggestion

- Return to the object selection page, select the objects to be synchronized again, and then perform the pre-check.
- In the pre-check phase, if the source database is disconnected and the pre-check fails, ensure that the source database is properly connected and perform the pre-check again.
- Due to the restrictions of Oracle Logminer, a schema name, table name, or column name of the selected table in the source Oracle database in the incremental synchronization phase cannot exceed 30 characters. Change the schema name, table name, and column name that do not meet requirements to ensure that the length does not exceed 30 characters. Then, perform the precheck again.
- For a task with Microsoft SQL Server serving as the source database, the number of tables to be synchronized in a single task cannot exceed 1000. Reduce the number of tables to be synchronized in a single task, deselect the tables that do not need to be synchronized, or split the task into multiple synchronization tasks for data synchronization, and perform the pre-check again.
- Check whether the source or destination DB engine is of a special version. The special version may not support some syntax. As a result, the pre-check fails. Change the source or destination database and perform the pre-check again.

1.2.63 Checking Whether the Source Database Tables Contain Data Types Not Supported for Migration

The source Microsoft SQL Server database tables cannot contain the SQL_VARIANT, GEOMETRY, and GEOGRAPHY data types. Otherwise, the synchronization task fails.

Failure Cause

The source database tables contain data types that are not supported.

Handling Suggestion

Click **Previous** to return to the object selection page and delete the objects from the synchronization objects.

1.2.64 Checking Whether the SQL Server Agent of the Source Database Is Enabled

Check whether the SQL Server Agent of the source database is enabled.

Failure Cause

If the SQL Server Agent of the source database is not enabled, the migration will fail.

Handling Suggestion

Enable the Microsoft SQL Server Agent and perform the pre-check again.

1.2.65 Checking Whether the Values of `group_concat_max_len` Are Consistent

If the values of `group_concat_max_len` in the source and destination databases are different, the queried fields may be truncated. Change the parameter values to the same.

Failure Cause

The `group_concat_max_len` values of the source and destination databases must be the same.

Handling Suggestion

Change the value of `group_concat_max_len` of the source database to be the same as that of the destination database.

- If the database is a self-managed MySQL database, run commands to change the value.
 - a. Run the following command to check the value:

```
SHOW VARIABLES LIKE "group_concat_max_len";
```

- b. Run the following command to change the value:

```
SET SESSION group_concat_max_len = <New maximum length>
```
- If the database is an RDS for MySQL DB instance, change the value of the **group_concat_max_len** parameter by referring to [Modifying Parameters of an RDS for MySQL Instance](#).

For details about how to change the value of **group_concat_max_len** for other types of databases, see the usage guide of the corresponding database.

1.2.66 Checking Whether the Table Structures of the Source Database Are Consistent

During many-to-one synchronization, the structures of the tables in the source database must be inconsistent.

Failure Cause

Many-to-one synchronization failed because the table structures in the source database are inconsistent.

Handling Suggestion

Modify the source table structures to ensure that they are consistent.

1.2.67 Checking Whether the Character Sets Are Compatible

During a data synchronization with Oracle serving as the source database, DRS checks the compatibility between the character sets of the source and destination databases. If the character set of the source database is incompatible with that of the destination database, data synchronization may fail.

Failure Cause

The character set of the destination database is incompatible with that of the source database.

Handling Suggestion

Change the character set of the destination database to be the same as that of the source database.

1.2.68 Whether There Are XA Transactions That Have Not Been Submitted for a Long Time in the Source Database

There are XA transactions that have been there for a long time without being submitted in the source database. As a result, the data of these XA transactions may be missing.

Failure Cause

There are XA transactions that have been there for a long time without being submitted in the source database.

Handling Suggestion

Ensure that XA transaction has been correctly committed and perform the pre-check again.

1.2.69 Whether the Selected Objects Exist in the Destination Database

During data synchronization, DRS checks destination database object consistency. If the destination database objects do not meet requirements, data synchronization may fail.

Failure Cause

- The selected schema does not exist in the destination database.
- Source database names are the same except for letter cases.
- The selected table does not exist in the destination database or the table structure is inconsistent with that in the source database.

Handling Suggestion

- Create the schema in the destination database. Statement for creating a schema:

```
CREATE SCHEMA schema_name;
```
- Change the table name or return to the object selection page and deselect the tables with the same name.
- If you choose not to synchronize the table structure, create a table in the destination database and ensure that the table structure is the same as that in the source database

1.2.70 Destination Database Same Name Check

Check whether there is a database table with the same name as the source table in the destination database.

- If **Table structure** is selected, the destination database cannot contain tables whose names are the same as the source tables to be synchronized.
- If **Table structure** is not selected, the destination database must have tables that match the source tables, and the table structure must be the same as the selected source table structures.

Failure Cause

- The table structure synchronization is selected, and there are tables to be synchronized in the destination database.
- The table structure synchronization is not selected, and there are no tables to be synchronized in the destination database.
- The source table columns are not a subset of the destination columns.
- Some databases cannot be synchronized because the databases with the same names do not exist in the destination databases. For synchronization tasks, you need to create the corresponding database (user) in the destination database in advance.

Handling Suggestion

- To synchronize table structures, delete existing tables from the destination database.
- Select **Table structure** in the previous step or create the corresponding tables in the destination database.
- Return to the **Set Synchronization Task** page, select the tables that meet the requirements. Alternatively, add columns required in the destination table.
- Create these databases or users in the destination database or do not synchronize these databases. Statement for creating a user:

```
CREATE USER user_name IDENTIFIED BY password;
```

1.2.71 Whether the Destination Database User (Schema) and Table Exist

GaussDB -> Oracle Synchronization

Table 1-2 Whether the destination database user (schema) and table exist

Check Item	Whether the destination database user (schema) and table exist
Description	To synchronize data from GaussDB to the Oracle database, you need to create the corresponding user (schema) and table in the destination database in advance.
Failure Cause and Handling Suggestion	<p>Failure cause: The destination database user corresponding to the selected schema does not exist.</p> <p>Handling suggestion: Create a user with the same source database schema name in the destination database. Statement for creating a user: <pre>CREATE USER user_name IDENTIFIED BY password;</pre></p>
	<p>Failure cause: Source database names are the same except for letter cases.</p> <p>Handling suggestion: Change the table name or return to the object selection page and deselect the tables with the same name. Statement for changing the table name: <pre>ALTER TABLE old_table_name RENAME TO new_table_name;</pre></p>

1.2.72 Whether the Maximum Number of Indexed Columns Has Been Reached

DB2 for LUW -> GaussDB Synchronization

Table 1-3 Whether the maximum number of indexed columns has been reached

Check Item	Whether the maximum number of indexed columns has been reached
Description	The number of indexed columns to be migrated in the source database cannot exceed 32.
Failure Cause and Handling Suggestion	<p>Failure cause: The number of indexed columns to be migrated in the source database cannot exceed 32.</p> <p>Handling suggestion: Check the number of indexed columns in the source table and modify the indexes.</p> <p>Statement for viewing indexed columns: SELECT T1.INDNAME ,T1.COLNAMES FROM SYSCAT.INDEXES AS T1 JOIN SYSCAT.INDEXCOLUSE AS T2 ON T1.INDNAME= T2.INDNAME WHERE T1.TABNAME='table_name' AND T1. INDNAME= 'index_name';</p> <p>Statement for deleting an index: DROP INDEX index_name;</p> <p>Statement for creating an index: CREATE INDEX index_name ON table_name(col1,col2);</p>

1.2.73 Checking the Length of the Index Column in the Source Database

Oracle to MySQL Synchronization

Table 1-4 Checking the length of the index column in the source database

Check Item	Checking the length of the index column in the source database
Description	Check whether the length of index column in the source database exceeds the limit.
Failure Cause and Handling Suggestion	<p>Failure cause: The source database contains a data table with more than 64 indexes.</p> <p>Handling suggestion: Do not synchronize indexes or delete some indexes so that the number of indexes in a single table in the source database does not exceed 64.</p> <p>Statement for deleting an index: DROP INDEX index_name;</p>

	<p>Failure cause: There are indexes in the source database exceed the column length limit of the destination database.</p> <p>Handling suggestion: Deselect the table or change the index length.</p> <p>Statement for deleting an index: DROP INDEX index_name;</p> <p>Statement for creating an index: CREATE INDEX index_name ON table_name(col1,col2);</p>
--	--

1.2.74 Whether the Table Structures (Including Primary Key Indexes and the Number of Columns) of the Source Oracle Database and Destination Database Middleware Are Aligned

Oracle to DDM Synchronization

Table 1-5 Whether the table structures (including primary key indexes and the number of columns) of the source Oracle database and destination database middleware are aligned

Check Item	Whether the table structures (including primary key indexes and the number of columns) of the source Oracle database and destination database middleware are aligned
Description	Check whether the table structures (including primary key indexes and the number of columns) of the source Oracle database and destination DDM database are aligned.
Failure Cause and Handling Suggestion	<p>Failure cause: The redundant columns (columns that do not exist in the source database) of the destination database cannot contain not-null constraints. The not-null constraints will cause the migration to fail.</p> <p>Handling suggestion:</p> <ol style="list-style-type: none"> 1. Check the not-null constraints on the redundant columns in the destination database. DESC [table_name]; 2. Modify the non-null constraint on the redundant columns. Alter Table table_name Modify column_name NULL; <p>Failure cause: Source database names are the same except for letter cases.</p> <p>Handling suggestion: Change the table name or return to the object selection page and deselect the tables with the same name. Statement for changing the table name: ALTER TABLE old_table_name RENAME TO new_table_name;</p>

1.2.75 Whether Synchronization Objects Exist in the Destination Database

GaussDB -> MySQL Synchronization

Table 1-6 Whether synchronization objects exist in the destination database

Check Item	Whether synchronization objects exist in the destination database
Description	Check whether synchronization objects exist in the destination database.
Failure Cause and Handling Suggestion	<p>Failure cause: Source database names are the same except for the letter case.</p> <p>Handling suggestion: Change the table name or deselect the tables with the same name on the object selection page. Statement for changing the table name: <code>ALTER TABLE old_table_name RENAME TO new_table_name;</code></p>

1.2.76 Whether the Source Database Contains Encrypted Objects

Microsoft SQL Server as the Source

Table 1-7 Whether the source database contains encrypted objects

Check Item	Whether the source database contains encrypted objects
Description	Check whether the source database contains encrypted objects.
Failure Cause and Handling Suggestion	<p>Failure cause: The source database contains encrypted objects.</p> <p>Handling suggestion: Go back to the object selection page and select database objects that are not encrypted.</p>

1.2.77 Checking Whether the Source Database Contains Unsupported Table Field Types

Oracle Synchronization

Table 1-8 Checking whether the source database contains unsupported table field types

Check Item	Whether the source database contains unsupported table field types
Description	Check whether the source database contains unsupported table field types.
Failure Cause and Handling Suggestion	Failure cause: The source database contains unsupported table field types. The following table field types are supported: VARCHAR, VARCHAR2, NVARCHAR2, NUMBER, FLOAT, LONG, DATE, BINARY_FLOAT, BINARY_DOUBLE, RAW, LONG RAW, CHAR, NCHAR, CLOB, NCLOB, BLOB, ROWID, TIMESTAMP, TIMESTAMP WITH TIME ZONE and TIMESTAMP WITH LOCAL TIME ZONE. Handling suggestion: Select other tables that can be synchronized.

Synchronization from MySQL to PostgreSQL

Table 1-9 Checking whether the source database contains unsupported table field types

Check Item	Whether the source database contains unsupported table field types
Description	The following table field types are not supported: geometry, point, lineString, polygon, geometrycollection, multipoint, multilinestring, and multipolygon. The source database contains unsupported table field types, resulting in migration failures.
Failure Cause and Handling Suggestion	Failure cause: The source database contains unsupported table field types. Handling suggestion: Delete the columns containing the unsupported field types. Alternatively, do not migrate the table containing the unsupported table field types.

1.2.78 Checking Replication Attribute of Primary Key Columns

During a full+incremental synchronization or an incremental synchronization task with PostgreSQL serving as the source database, the replication attribute of

primary key columns in the source database table is checked. If the toast attribute of the primary key column in the source database is main, external, or extended, the replica identity attribute must be full.

Failure Cause

- All primary key columns of the tables to be synchronized are columns whose storage attribute is plain, and the replication attribute of the tables is neither full nor default. Incremental synchronization may fail.
- The primary key columns of the tables to be synchronized contain columns whose storage attribute is not plain, and the replication attribute of the tables is neither full nor default. There is a high probability that incremental synchronization will fail.
- The primary key columns of the tables to be synchronized contain columns whose storage attribute is not plain, and the replication attribute of the tables is not full. Incremental synchronization may fail.

Handling Suggestion

Run the following SQL statement to change the replication attribute of the tables to full: (If the replication attribute is changed to default, incremental synchronization may still fail.)

```
alter table schema.table replica identity full;
```

1.2.79 Whether There Are Tables Containing Fields of the longtext or longblob Type in the Synchronization Object

MySQL as the Source

Table 1-10 Whether there are tables containing fields of the longtext or longblob type in the synchronization object

Check Item	Whether there are tables containing fields of the longtext or longblob type in the synchronization object
Description	If there are tables containing fields of the longtext or longblob type in the synchronization object, DRS tasks with small specifications may fail.
Failure Cause and Handling Suggestion	Failure cause: There are tables containing fields of the longtext or longblob type in the synchronization object. Handling suggestion: If tables containing fields of the longtext or longblob type exist in the synchronization object, create a DRS task with large specifications.

MariaDB Synchronization

Table 1-11 Checking whether there are tables containing fields of the longtext or longblob type in the synchronization object

Check Item	Whether there are tables containing fields of the longtext or longblob type in the synchronization object
Description	If there are tables containing fields of the longtext or longblob type in the synchronization object, DRS tasks with small specifications may fail.
Failure Cause and Handling Suggestion	Failure cause: There are tables containing fields of the longtext or longblob type in the synchronization object. Handling suggestion: If tables containing fields of the longtext or longblob type exist in the synchronization object, create a DRS task with large specifications.

1.2.80 Checking Database Mapping Objects

MySQL to MySQL, MySQL to TaurusDB, and TaurusDB to MySQL Synchronization

Table 1-12 Checking database mapping objects

Check Item	Whether the database mapping objects are supported
Description	Whether the database mapping objects are supported
Item to Be Confirmed and Handling Suggestion	Item to be confirmed: The source database contains objects that cannot be synchronized. Handling suggestion: After database mapping, the source database contains data types that cannot be synchronized, including functions, stored procedures, and views. All the preceding objects cannot be synchronized during database mapping. Check whether the objects to be synchronized meet service requirements.

1.2.81 Whether the Source Database Is the Standby Database of a TaurusDB Instance

If the source database is a standby TaurusDB database and there is no binlogs, the incremental migration will fail.

Failure Cause

The source database is a standby database and does not contain binlogs.

Handling Suggestion

Use a primary TaurusDB database as the source database, and perform the pre-check again.

1.2.82 Checking Whether the Source and Destination Database Character Sets Are Consistent

MySQL->MySQL/DDM Migration

Table 1-13 Checking whether the source and destination database character sets are consistent

Check Item	Whether the source and destination database character sets are consistent
Description	Checking whether the source and destination database character sets are consistent
Item to Be Confirmed and Handling Suggestion	Item to be confirmed: The source database supports character sets of a later version. Handling suggestion: The source database supports character sets of a later version. Check whether the source database uses a character set of a later version and whether the destination database supports the character set.

DDM -> MySQL/DDM Synchronization

Table 1-14 Checking whether the source and destination database character sets are consistent

Check Item	Whether the source and destination database character sets are consistent
Description	Checking whether the source and destination database character sets are consistent

Item to Be Confirmed and Handling Suggestion	<p>Item to be confirmed: The source database supports character sets of a later version.</p> <p>Handling suggestion: The source database supports character sets of a later version. Check whether the source database uses a character set of a later version and whether the destination database supports the character set.</p>
---	---

MariaDB Synchronization

Table 1-15 Checking whether the source and destination database character sets are consistent

Check Item	Whether the source and destination database character sets are consistent
Description	Checking whether the source and destination database character sets are consistent
Item to Be Confirmed and Handling Suggestion	<p>Item to be confirmed: The source database supports character sets of a later version.</p> <p>Handling suggestion: The source database supports character sets of a later version. Check whether the source database uses a character set of a later version and whether the destination database supports the character set.</p>

1.2.83 Checking Whether Data Replication Is Enabled for the Source Database

Redis Migration

Table 1-16 Checking whether data replication is enabled for the source database

Check Item	Whether data replication is enabled for the source database
Description	Checking whether data replication is enabled for the source database

Failure Cause and Handling Suggestion	Failure cause: Data replication is not enabled on the source database. Handling suggestion: Connect to the source database and set enable-replication to 1 to enable data replication.
--	---

1.2.84 Checking Whether the Maximum Sequence Number of the Source Database is Less Than That of the Destination Database

Redis Migration

Table 1-17 Checking whether the maximum sequence number of the source database is less than that of the destination database

Check Item	Whether the maximum sequence number of the source database is less than that of the destination database
Description	Checking whether the maximum sequence number of the source database is less than that of the destination database
Item to Be Confirmed and Handling Suggestion	Item to Be Confirmed: The maximum sequence number of the source database to be migrated is greater than that of the destination database. If incremental data is generated in the database with the maximum sequence number, the migration task will fail. Handling suggestion: Increase the maximum sequence number of the destination database.

1.2.85 Checking Whether Interval Partitioned Tables Are Included in the Source Database

GaussDB Serving as the Source in Synchronization

Table 1-18 Checking whether interval partitioned tables are included in the source database

Check Item	Whether interval partitioned tables are included in the source database
-------------------	---

Description	Checking whether interval partitioned tables are included in the source database
Failure Cause and Handling Suggestion	Failure cause: Interval partitioned tables cannot be synchronized. Handling suggestion: Go back to the Synchronization Object area and deselect the interval partitioned tables.

1.2.86 Oracle Account Check in the Source Database

During an incremental synchronization with Oracle serving as the source database, DRS checks the source database account.

Failure Cause

The source database account is an Oracle account, but not a user account. Incremental synchronization cannot be performed.

Handling Suggestion

Use a user account instead of the Oracle account.

1.2.87 Checking the Number of DNs in the Source Database

Self-built Distributed GaussDB Serving as the Source in Synchronization

Table 1-19 Checking the number of DNs in the source database

Check Item	Checking the Number of DNs in the Source Database
Description	Check whether the number of DNs entered by the user is the same as that in the source database.
Item to Be Confirmed and Handling Suggestion	Item to be confirmed: The number of DNs entered by the user is different from that in the source database. Handling suggestion: If the number of DNs entered by the user is different from that in the source database, there may be data inconsistencies. You must confirm the risky item before you continue.

1.2.88 Whether the Selected Objects Exist in the Destination Database

During a data synchronization with Oracle serving as the source database, DRS checks the consistency of destination database objects.

Failure Cause

- Object names will be converted to lowercase letters after being synchronized to the destination database. To avoid synchronization failures, ensure that the selected source database tables do not contain columns with the same name but different letter cases.
- The selected table does not exist in the destination database or the table structure is inconsistent with that in the source database.
- The constraints on the destination database are inconsistent with those on the source database. The synchronization may fail due to inconsistent constraints.

Handling Suggestion

- Delete columns with the same name but different letter cases from the source database tables, or change the names of columns with the same name but different letter cases in the source database tables.
- If the selected table does not exist in the destination database, create the table in the destination database and ensure that the table structure is the same as that in the source database. Statement for creating a table:

```
CREATE TABLE table_name (column_name data_type);
```

If the table structure is inconsistent with that in the source database, create missing columns in the destination database table, or convert the names of columns with the same name but different letter cases in the destination database table to lowercase letters, or delete redundant columns from the source database table.
- Ensure that the constraints on the destination database are consistent with those on the source database, or confirm that inconsistent constraints do not adversely affect data migration.

1.2.89 Whether There Are Foreign Keys Containing Unsupported Reference Operations in the Source Database

MySQL, TaurusDB, and DDM Serving as the Source in Full+Incremental or Incremental Migration and Synchronization, MySQL, TaurusDB, and DDM Serving as the Source in DR

Table 1-20 Whether there are foreign keys containing unsupported reference operations in the source database

Check Item	Whether there are foreign keys containing unsupported reference operations in the source database
-------------------	---

Description	In a synchronization object, there are foreign keys that contain reference operations such as CASCADE, SET NULL, and SET DEFAULT. These operations will cause the update or deletion of rows in parent tables and affect records in child tables. Also, operations related to child tables are not recorded in binlogs. The DRS cannot synchronize data, and data in child tables is inconsistent.
Failure Cause and Handling Suggestion	<p>Failure cause: In a synchronization object, there are foreign keys that contain reference operations such as CASCADE, SET NULL, and SET DEFAULT. These operations will cause the update or deletion of rows in parent tables and affect records in child tables. Also, operations related to child tables are not recorded in binlogs. The DRS cannot synchronize data, and data in child tables is inconsistent.</p> <p>Handling suggestion: Delete foreign keys that contain reference operations such as CASCADE, SET NULL, and SET DEFAULT from child tables, or do not synchronize these child tables.</p> <p>Reference statement for deleting a foreign key: <code>ALTER TABLE table_name DROP FOREIGN KEY foreign_key_name</code></p>

1.2.90 Whether the Selected Table Contains Delay Constraints

PostgreSQL or GaussDB Serving as the Source in Synchronization

Table 1-21 Whether the selected table contains delay constraints

Check Item	Whether the selected table contains delay constraints
Description	Tables that contain delay constraints may fail to be synchronized.
Failure Cause and Handling Suggestion	<p>Failure cause: Tables that contain delay constraints may fail to be synchronized.</p> <p>Handling suggestion: Remove the delay constraints.</p> <ul style="list-style-type: none"> SQL statement for deleting a constraint: <code>alter table Schema_name.Table_name drop CONSTRAINT Constraint_name</code> SQL statement for adding a constraint: <code>alter table Schema_name.Table_name add CONSTRAINT Constraint_name Constraint_type (Field list) NOT DEFERRABLE</code>

1.2.91 Whether the Source Database Tables Contain Primary Keys

MySQL as the Source

Table 1-22 Checking whether the source database tables contain primary keys

Check Item	Whether the source database tables contain primary keys
Description	The tables to be synchronized in the source database do not contain primary keys.
Item to Be Confirmed and Handling Suggestion	<p>Item to be confirmed: The tables to be synchronized in the source database do not contain primary keys.</p> <p>Handling suggestion: Create primary keys for the tables as the performance of a table without a primary key is lower than that of a table with a primary key.</p>

1.2.92 Whether Foreign Keys Are Disabled or Tables to Be Synchronized Have Foreign Keys in the Destination Database

Oracle -> GaussDB Synchronization

Table 1-23 Checking whether foreign keys are disabled or tables to be synchronized have foreign keys in the destination database

Check Item	Whether foreign keys are disabled or tables to be synchronized have foreign keys in the destination database
Description	Check whether foreign keys are disabled or tables to be synchronized have foreign keys in the destination database.
Failure Cause and Handling Suggestion	<p>Failure cause: Tables to be synchronized have foreign keys in the destination database and cannot be synchronized.</p> <p>Handling suggestion: Delete the foreign key, disable the trigger, or change the value of <code>session_replication_role</code> to <code>replica</code>.</p>

1.2.93 Whether There Are Composite Hash Indexes in the Source Collection

Migration and Synchronization from MongoDB to DDS

Table 1-24 Checking whether there are composite hash indexes in the source collection

Check Item	Whether there are composite hash indexes in the source collection
Description	There are collections containing composite hash indexes in the source database.
Failure Cause and Handling Suggestion	Failure cause: The selected collections contain composite hash indexes. Handling suggestion: Deselect the preceding collections or create non-composite hash indexes in the source database.

1.2.94 Whether There Are Composite Hash Shard Keys in the Source Collection

Migration and Synchronization from MongoDB to DDS

Table 1-25 Checking whether there are composite hash shard keys in the source collection

Check Item	Whether there are composite hash shard keys in the source collection
Description	There are collections containing composite hash shard keys in the source database.
Failure Cause and Handling Suggestion	Failure cause: The selected collections contain composite hash shard keys. Handling suggestion: Deselect the preceding collections.

1.2.95 Whether the Source Table Structure Contains Newline Characters

The source databases, tables, columns, indexes, or constraints contain newline characters, which may cause service problems.

Failure Cause

The source databases, tables, columns, indexes, or constraints contain newline characters.

Handling Suggestion

The source databases, tables, columns, indexes, or constraints contain newline characters, which may cause service problems. Modify the objects in the source database and perform the pre-check again.

1.2.96 Whether There Are Tables Containing Fields of the bytea or text Type in the Synchronization Object

PostgreSQL Serving as the Source in Synchronization

Table 1-26 Checking whether there are tables containing fields of the bytea or text type in the synchronization object

Check Item	Whether there are tables containing fields of the bytea or text type in the synchronization object
Description	Fields of the bytea or text type may result in out of memory (OOM) during synchronization.
Item to Be Confirmed and Handling Suggestion	<p>Potential problem: If there are tables containing fields of the bytea or text type in the synchronization object, fields of the bytea or text type may result in task OOM during synchronization.</p> <p>Handling suggestion: If there are tables containing fields of the bytea or text type in the synchronization object, create a DRS task with large specifications for synchronization.</p>

1.2.97 Checking Whether the Source Table Structure Contains Virtual Columns

During a data synchronization with Oracle serving as the source database, DRS checks whether the source table structure contains virtual columns.

Failure Cause

The source database contains virtual columns, but data in virtual columns cannot be migrated. As a result, the migrated data is incomplete.

Handling Suggestion

After the pre-check, create a table structure that contains virtual columns in the destination database.

1.2.98 Whether the max_allowed_packet Value of the Source Database Is Too Small

MySQLor TaurusDB as the Source

Table 1-27 Checking whether the max_allowed_packet value of the source database is too small

Check Item	Whether the max_allowed_packet value of the source database is too small
Description	If the max_allowed_packet value of the source database is too small, data migration may fail.
Item to Be Confirmed and Handling Suggestion	Potential problem: If there is a lot of data to be migrated or there are too many fields to be migrated, and the max_allowed_packet value of the source database is too small, then the migration task may fail. Handling suggestion: Change the max_allowed_packet value of the source database to a value greater than 16777216.

1.2.99 Whether the Supplemental Log Level in the Source Database Is Correct

During a data synchronization with Oracle serving as the source database, if all data is required for the synchronization object, all-level supplemental logging must be enabled.

Failure Cause

All data is required for the synchronization object, but all-level supplemental logging is not enabled.

Handling Suggestion

Enable all-level supplemental logging.

Commands for enabling all-level supplemental logging:

Database level: alter database add supplemental log data (all) columns

Table level: alter table *Schema_name.Table_name* add supplemental log data(all) columns

1.2.100 Whether Kafka Topics Have Been Created

GaussDB to Kafka Synchronization

Table 1-28 Checking whether Kafka topics have been created

Check Item	Whether Kafka topics have been created
Description	If you do not create topics in Kafka, DRS automatically creates them during incremental synchronization.
Item to Be Confirmed and Handling Suggestion	Item to be confirmed: If topics were not created in Kafka, DRS automatically creates them during incremental synchronization. Handling suggestion: Create the required topics in Kafka.

1.2.101 Whether the Source Database Kernel Encoding Supports Data Replication

Out-of-Cloud GeminiDB Redis Migration

Table 1-29 Checking whether the source database kernel encoding supports data replication

Check Item	Whether the source database kernel encoding supports data replication
Description	Check whether the source database kernel encoding supports data replication.
Failure Cause and Handling Suggestion	Failure cause: The source database kernel encoding does not support data replication. Handling suggestion: Contact GeminiDB Redis personnel.

1.2.102 block_encryption_mode Consistency Check

Check whether the **block_encryption_mode** values of the source and destination databases are the same. If they are different, the destination database may be unavailable after data migration. You are advised to set **block_encryption_mode** to the same value.

block_encryption_mode is used to set the encryption mode when the encryption and decryption functions are used. If the **block_encryption_mode** parameter of the source database is different from that of the destination database, the query results of the encryption and decryption functions are different.

For example, if you set **block_encryption_mode** to **aes-128-ecb** and run **SELECT HEX(AES_ENCRYPT('test','k'))**, the command output is **1521CE1E7B33581DF75BA0DF53F8F6D3**. If you set **block_encryption_mode** to **aes-256-ecb** and run **SELECT HEX(AES_ENCRYPT('test','k'))**, the command output is **0201581D9FC84F7BBF136A80E7FC9572**. The two command outputs are different. In this case, after the encrypted data in the source database is synchronized to the destination database, the original data cannot be restored after decryption in the destination database.

Failure Cause

The **block_encryption_mode** values of the source and destination databases must be the same.

Handling Suggestion

- If related encryption and decryption functions are not used to perform operations on services, skip this check item.
- To change the **block_encryption_mode** parameter of a database, perform the following operations:
 - The source database is an on-premises MySQL database.
 - Method 1: Run the following command and restart the database to apply the change:

```
SET GLOBAL block_encryption_mode = 'the encryption mode'
```
 - Method 2: Add the following line to the **my.cnf** or **my.ini** configuration file and restart the database to apply the change.

```
block_encryption_mode = 'the encryption mode'
```
 - If the source database is an RDS for MySQL instance on the cloud, change the **block_encryption_mode** parameter of a database by referring to [Modifying Parameters of an RDS for MySQL Instance](#).

1.2.103 Character Type and Sorting Rule Check in the Destination Database

PostgreSQL -> PostgreSQL Synchronization

Table 1-30 Checking the character type and sorting rule in the destination database

Check Item	Character type and sorting rule check in the destination database
Description	Check whether the destination database supports the value of lc_ctype or lc_collate in the database to be synchronized.
Item to Be Confirmed and Handling Suggestion	Item to be confirmed: The destination database does not support the value of lc_ctype or lc_collate in the database to be synchronized. Handling suggestion: Check whether the parameter lc_ctype or lc_collate can be set to the default value when a database is created in the destination database during full synchronization. The value of lc_collate affects the sorting rule of character strings, and the value of lc_ctype affects character type and conversion.

1.2.104 Column Name Check in the Source Database

Microsoft SQL Server as the Source in Synchronization

Table 1-31 Column name check in the source database

Check Item	Column name check in the source database
Description	Check whether the column names in the source database contain special characters.
Failure Cause and Handling Suggestion	Failure cause: The source database contains column names that do not meet requirements. The column names cannot contain the following special characters: []? Handling suggestion: Ensure that column names meet requirements.

1.2.105 Source Encrypted Table Check

MySQLor TaurusDB as the Source

Table 1-32 Source encrypted table check

Check Item	Source encrypted table check
Description	Check whether there are encrypted tables in the source database.
Failure Cause and Handling Suggestion	<p>Item to be confirmed: The source database contains encrypted tables. Check whether the destination database supports encrypted tables. If the destination database does not support these tables, the task may fail.</p> <p>Handling suggestion: Check whether the destination database supports these tables. If not, deselect them from the synchronization objects.</p>

1.2.106 Checking Whether the Source Table Replication Attribute Is Correct

PostgreSQL or GaussDB Serving as the Source in Incremental Synchronization

Table 1-33 Checking whether the source table replication attribute is correct

Check Item	Whether the source table replication attribute is correct
Description	Check the replication attribute of the table in the source database.
Failure Cause and Handling Suggestion	<p>Item to be confirmed: The source database table contains the primary key column, but the replication attribute is not FULL. When the source table data is updated, if the replication attribute of the table is not FULL, the source database logs may not record the old values of all columns, causing data loss.</p> <p>Handling suggestion: Run the following statements to change the replication attribute of the preceding tables to FULL:</p> <pre>alter table sch1.t varchar replica identity full; alter table sch1.t char replica identity full;</pre>

1.2.107 Partition Table Check on the Source Database

Synchronization from PostgreSQL to GaussDB

Table 1-34 Partition table check on the source database

Check Item	Partition table check on the source database
Description	Check whether the source database has partition tables.
Failure Cause and Handling Suggestion	<p>Item to be confirmed: If the source database contains partition tables and source partitions are modified or are deleted during data synchronization, partition data may fail to be synchronized or the synchronization may fail.</p> <p>Handling suggestion: The source partition must be a newly-created partition, and the partition name must be unique. This partition can be deleted from the source database only after all data is synchronized to the destination database.</p>

1.2.108 Checking Whether the Source Database Contains Unsupported Generated Columns

If there are tables where a generated column is a primary key or unique key in the source database, DRS cannot synchronize the structure of generated columns. As a result, data may be inconsistent after being synchronized to the destination database.

Failure Cause

There are tables where a generated column is a primary key or unique key in the source database.

Handling Suggestion

- Create a table structure in the destination database to ensure that logic and constraints of generated columns in the destination database are the same as those in the source database. Return to the object selection page, deselect **Table structure** for **Synchronization Object Type** and perform the pre-check again.
- Return to the object selection page, delete the tables where a generated column is a primary key or unique key from the synchronization object, and perform the pre-check again.

1.2.109 Checking Whether the ENABLE_SLOT_LOG Value of the Source Database Is Correct

GaussDB Serving as the Source in Synchronization

Check the **ENABLE_SLOT_LOG** parameter in the source database. If the source database is a DR cluster, this parameter is not involved.

Check Item	Whether the ENABLE_SLOT_LOG value of the source database is correct
Description	During incremental synchronization, this parameter must be set to on , indicating whether to enable primary/standby synchronization for replication slots.
Handling Suggestion	Set ENABLE_SLOT_LOG of the source database to on .

1.3 Source DB Instance Statuses

1.3.1 Checking Whether the Source DB Instance Is Available

In the pre-check phase, DRS checks the status of the source DB instance.

Failure Cause

The source DB instance is unavailable.

Handling Suggestion

If the source DB instance is abnormal and cannot be accessed by DRS, wait until the DB instance becomes available and perform the pre-check again.

1.3.2 Checking Whether the Source and Destination Databases Are of the Same Type

MongoDB Migration

Table 1-35 Checking whether the source and destination databases are of the same type

Check Item	Whether the source and destination databases are of the same type
-------------------	---

Description	Check whether the source and destination databases are of the same type. If they are of different types, the migration fails.
Failure Cause and Handling Suggestion	Failure cause: The destination database is a cluster but the source database is a replica set. Handling suggestion: Change the type of the source or destination database.
	Failure cause: The destination database is a replica set but the source database is a cluster. Handling suggestion: Change the type of the source or destination database.

1.3.3 Checking Whether the ChangeStream API of the source DB instance is available

MongoDB-to-DDS Migration

Table 1-36 Checking whether the ChangeStream API of the source DB instance is available

Check Item	Whether the ChangeStream API of the source DB instance is available
Description	Check whether the ChangeStream API of the source database is available.
Failure Cause and Handling Suggestion	Failure cause: The source database cannot use the ChangeStream API. Handling suggestion: <ol style="list-style-type: none"> 1. Check whether the source database version is MongoDB 4.0 or later. 2. Check whether the WiredTiger storage engine is enabled for the source database. If not, you are advised to create a DRS task and select the oplog mode. Run the following command (on a shard node): <code>db.serverStatus().storageEngine.name;</code>
	Failure cause: The source database cannot use the ChangeStream API. Handling suggestion: Check whether the source database version is DDS 4.0 or later. If not, upgrade the source database to DDS 4.0 or later.

1.4 Destination DB Instance Statuses

1.4.1 Checking Whether the Destination DB Instance Is Available

In the pre-check phase, DRS checks the status of the destination DB instance.

Failure Cause

- The destination DB instance is unavailable.
- The destination DB instance is read-only.

Handling Suggestion

- If the destination DB instance is abnormal and cannot be accessed by DRS, wait until the DB instance becomes available and perform the pre-check again.
- If the destination DB instance is read-only and cannot be written, replace the destination database and perform the pre-check again.

1.4.2 Checking Whether the Destination Database Is Involved in Another Migration Task

MySQL

Table 1-37 Checking whether the destination database is involved in another migration task

Check Item	Whether the destination database is involved in another migration task
Description	Check whether the destination database is being used in another migration task. If more than one migration task uses the same destination database, the migration may fail.
Failure Cause and Handling Suggestion	Failure cause: The destination RDS DB instance is being used in another migration task. Handling suggestion: Wait for the migration task to complete. You can also stop or delete an unused migration task.

1.4.3 Checking Whether the Destination Database Has a Read Replica

MySQL

Table 1-38 Checking whether the destination database has a read replica

Check Item	Whether the destination database has a read replica
Description	Check whether the destination database has read replicas. If the destination database has read replicas, the incremental migration may fail.
Failure Cause and Handling Suggestion	Failure cause: In an incremental migration, the destination database cannot have read replicas. Handling suggestion: Delete the read replicas from the destination database. After the migration is complete, create read replicas.

1.4.4 Checking Whether the Destination Database Is Read-Only

MySQL Migration, Synchronization, and Backward DR

Table 1-39 Checking whether the destination database is read-only

Check Item	Whether the destination database is read-only
Description	The destination database is read-only, and data cannot be written to the destination database.
Failure Cause and Handling Suggestion	Failure cause: The destination database is read-only. Handling suggestion: Run the following commands to change the destination database to read/write and then, restart the database. Sample commands: set global read_only=0; set global super_read_only=0;

1.4.5 Checking Whether the Extensions Are Supported

PostgreSQL Synchronization

Table 1-40 Checking whether the extensions are supported

Check Item	Whether the extensions are supported
Description	Check whether the source database has plug-ins that are not installed on the destination database.
Failure Cause and Handling Suggestion	<p>Failure cause: Extensions installed in the source database are not supported in the destination database.</p> <p>Handling suggestion:</p> <ul style="list-style-type: none"> • If the source database services do not depend on those extensions, run the following statement to delete the extensions. Replace <i>plugin_name</i> with the name of the extension to be deleted. <code>drop extension plugin_name;</code> • Alternatively, use a destination database that supports these extensions.
	<p>Failure cause: The source database has extensions that contain tables as members.</p> <p>Handling suggestion: Check whether the source database extensions contain metadata generated after the extensions are created. If yes, use the dedicated syntax of the extension to rebuild the metadata after the migration is complete.</p>
	<p>Failure cause: The destination database user does not have the permission to create extensions.</p> <p>Handling suggestion: Grant the permission to the user in the destination database as user root. Run the following SQL statements (replace <i>username</i> with the destination database username): <code>alter user username inherit;</code> <code>grant root to username;</code></p>
	<p>Failure cause: The extension version supported by the destination database is earlier than that installed in the source database.</p> <p>Handling suggestion: Use the destination database that supports extensions of a later version (not earlier than the source database extension version) and create a synchronization task again.</p>

1.4.6 Checking Whether Destination Contains the Configured Database

MySQL > PostgreSQL

Table 1-41 Checking whether destination contains the configured database

Check Item	Whether the destination contains the configured database.
Description	Databases and schemas cannot be migrated. You need to manually create databases and schemas on the destination database. Otherwise, the migration will fail.
Failure Cause and Handling Suggestion	Failure cause: Databases cannot be migrated from MySQL to PostgreSQL. Handling suggestion: In the destination database, manually create databases and schemas with the same names as those of the source database.
	Failure cause: The objects to be synchronized already exist in the destination database. Handling suggestions: Delete the tables to be synchronized from the destination database or select the tables that do not exist in the destination database for synchronization.

MySQL -> GaussDB Synchronization

Table 1-42 Checking whether destination contains the configured database

Check Item	Whether the destination contains the configured database.
Description	In the MySQL to GaussDB synchronization scenario, the mapped database must exist in the destination database. Otherwise, the synchronization fails.
Failure Cause and Handling Suggestion	Failure cause: The destination database does not contain the configured database. Handling suggestion: Before the synchronization, manually create a mapped database in the destination database.

GaussDB->GaussDB Synchronization

Table 1-43 Checking whether destination contains the configured database

Check Item	Whether the destination contains the configured database.
Description	In the GaussDB to GaussDB synchronization scenario, the mapped database must exist in the destination database. Otherwise, the synchronization fails.
Failure Cause and Handling Suggestion	Failure cause: The configured database does not exist in the destination. Handling suggestion: Before the synchronization, manually create a configured database in the destination.

1.4.7 Checking Whether the Destination DB Instance Is Available

Table 1-44 Checking whether the destination DB instance is available

Check Item	Whether the destination DB instance is available
Description	Check whether the primary instance and read replicas are available in the destination database. If not, the migration fails.
Failure Cause and Handling Suggestion	Failure cause: The destination DB instance is not available. Handling suggestion: Repair the destination DB instance.
	Failure cause: Read replicas in the destination database are abnormal. Handling suggestion: Repair the abnormal read replicas in the destination database.
	Failure cause: The RDS service is abnormal. Try again later. Handling suggestion: Try again later.

1.4.8 Checking Whether the Destination Database Is Empty

MySQL

Table 1-45 Checking whether the destination database is empty

Check Item	Whether the destination database is empty

Description	Check whether the destination database is empty. If the destination database is not empty, disaster recovery will fail.
Failure Cause and Handling Suggestion	Failure cause: The destination database is not empty. Handling suggestion: Delete all the user-created databases from the destination database.

1.4.9 Checking Whether There Are Foreign Keys that Are Not Disabled in the Destination Database Table

GaussDB Serving as the Destination in Synchronization

Table 1-46 Whether foreign keys are disabled or tables to be synchronized have foreign keys in the destination database

Check Item	Whether foreign keys are disabled or tables to be synchronized have foreign keys in the destination database
Description	The destination database has foreign keys that are not disabled. As a result, data synchronization may fail.
Failure Cause and Handling Suggestion	<p>Failure Cause: The destination database has foreign keys that are not disabled. As a result, data synchronization may fail.</p> <p>Handling Suggestion: Use any of the following methods:</p> <ol style="list-style-type: none"> 1. Change the value of session_replication_role of the destination database user to replica. set session_replication_role to replica; 2. Deselect these tables from the synchronization objects. 3. Disable or delete foreign keys. <p>Disabling foreign keys: alter table table_name disable constraint constraint_name;</p> <p>Deleting foreign keys: alter table table_name drop constraint constraint_name;</p>

1.4.10 Checking Whether There Are Enabled Triggers in an Existing Destination Database Table

GaussDB Serving as the Destination in Synchronization

Table 1-47 Checking whether there are enabled triggers in an existing destination database table

Check Item	Checking whether there are enabled triggers in an existing destination database table
Description	An existing table in the destination database contains valid triggers, which may cause data synchronization task failures.
Failure Cause and Handling Suggestion	Failure Cause: An existing table in the destination database contains valid triggers, which may cause data synchronization task failures. Handling suggestion: 1. Disable triggers. alter table table_name disable trigger all; 2. After the synchronization is complete, enable the triggers. alter table table_name enable trigger all;

1.5 Database User Permissions

1.5.1 Whether the Source Database User Has Sufficient Permissions

Check whether the source database user has sufficient permissions. The source database user permissions required in the full and incremental phases vary depending on the DB engine.

Failure Cause

The source database user does not have sufficient permissions.

Handling Suggestion

When you use DRS to migrate or synchronize data, the source database user must have required permissions. Tasks of different DB engines and modes require different account permissions. DRS automatically checks the database account permissions in the pre-check phase and provides handling suggestions.

Take the MySQL migration as an example. The source database user permissions are as follows:

- Full migration:
SELECT, SHOW VIEW, and EVENT

Reference statement: **GRANT** SELECT, SHOW VIEW, EVENT **ON** *.* **TO** 'user1';

- Full+incremental migration:
SELECT, SHOW VIEW, EVENT, LOCK TABLES, REPLICATION SLAVE, and REPLICATION CLIENT

REPLICATION SLAVE and REPLICATION CLIENT are global permissions and must be enabled separately. The reference statement is as follows:

GRANT REPLICATION SLAVE, REPLICATION CLIENT **ON** *.* **TO** 'user1';

SELECT, SHOW VIEW, EVENT, and LOCK TABLES are non-global permissions. The reference statement is as follows:

GRANT SELECT, SHOW VIEW, EVENT, LOCK TABLES, **ON** [Database to be migrated].* **TO** 'user1';

Related Documents

- [Which MySQL Permissions Are Required for DRS?](#)
- [How Do I Set an Independent Oracle Account That Has the Least Privilege and Uses DRS?](#)

1.5.2 Checking Whether the Destination Database User Has Sufficient Permissions

Check whether the destination database user has sufficient permissions. The destination database user permissions required in the full and incremental phases vary depending on the DB engine.

Failure Cause

The destination database user does not have sufficient permissions.

Handling Suggestion

When you use DRS to migrate or synchronize data, the destination database user must have required permissions. Tasks of different DB engines require different account permissions. DRS automatically checks the database account permissions in the pre-check phase and provides handling suggestions.

Take the MySQL migration as an example. The destination database user permissions are as follows:

SELECT, CREATE, ALTER, DROP, DELETE, INSERT, UPDATE, INDEX, EVENT, CREATE VIEW, CREATE ROUTINE, TRIGGER, REFERENCES, and WITH GRANT OPTION. If the destination database version is in the range 8.0.14 to 8.0.18, the SESSION_VARIABLES_ADMIN permission is required.

Reference statement: **GRANT**SELECT, CREATE, ALTER, DROP, DELETE, INSERT, UPDATE, INDEX, EVENT, CREATE VIEW, CREATE ROUTINE, TRIGGER, REFERENCES **ON** [Databases to be migrated].* **TO** 'user1' **WITH GRANT OPTION**;

Related Documents

- [Which MySQL Permissions Are Required for DRS?](#)
- [How Do I Set an Independent Oracle Account That Has the Least Privilege and Uses DRS?](#)

1.5.3 Checking Whether the Destination Database Account Has Required Permissions to Migrate Definer

MySQL Migration

Table 1-48 Checking Whether the Destination Database Account Has Required Permissions to Migrate Definer

Check Item	Checking whether the destination database account has required permissions to migrate Definer
Description	To migrate Definers to the cloud, the source database user must have the all privileges permission.
Failure Cause and Handling Suggestion	<p>Failure cause: The permission of the specified destination database account is insufficient.</p> <p>Handling suggestions: Migrate all Definers to the specified destination database account. Alternatively, do not migrate Definers to the specified destination database account and run the following command to grant the destination database account the all privileges permission.</p> <p>Example command:</p> <pre>grant all privileges on *.* to 'user' '@' host'</pre>
	<p>Failure cause: The specified source database user does not have required permissions.</p> <p>Handling suggestion:</p> <ol style="list-style-type: none"> 1. When configuring the destination database, select Migrate Definer to User to ensure that Definers of all the objects are under the specified user. 2. Retain the Definer settings and run the following command to grant the all privileges permission to the source database user. <p>Example command:</p> <pre>grant all privileges on *.* to 'user' '@' host'</pre>

1.6 Database Versions

1.6.1 Checking Whether the Source Database Version Is Supported

Check whether the source database version is supported. Different DB engines have different supported versions. For details, see [Supported Databases](#).

Failure Cause

The source database version does not meet the migration requirements.

Handling Suggestion

Select a source database version that meets requirements.

1.6.2 Checking Whether the Destination Database Version Is Supported

Check whether the destination database version is supported. Different DB engines have different supported versions. For details, see [Supported Databases](#).

Failure Cause

The destination database version does not meet the migration requirements.

Handling Suggestion

Select a destination database version that meets requirements.

1.6.3 Checking Whether the Migration Is from an Earlier Database Version to the Same or a Later Version

For homogeneous migration, DRS checks whether data is migrated from an earlier version to a later version or the same version. The database of a later version has new features. If the destination database does not have such features, data migration may fail.

Failure Cause

Data cannot be migrated or synchronized from a newer version database to an older version database.

Handling Suggestion

Select a source or destination database version that meets requirements, or ensure that the source database does not use new features provided by a later version. Otherwise, the migration may fail.

1.7 Networks

1.7.1 Checking Whether the Source Database Is Connected

In the pre-check phase, DRS checks the connectivity and accuracy of the source database IP address, port, username, and password.

Failure Cause

- The username or password is incorrect.
- The port cannot be accessed.
- The database account does not allow remote connection.
- The connection fails.

Handling Suggestion

- Check whether the username and password entered during the DRS connection test are correct. Enter the correct database username and password and perform the pre-check again.
- If the port entered during the connection test cannot be accessed, check whether the port exists. If the port is correct, check whether the firewall is enabled.
- If the source database is PostgreSQL, and the database configuration file **pg_hba.conf** does not contain the database account configuration, grant the remote connection permission for the account.

Add the following to **pg_hba.conf**, and restart the database for the modification to take effect:

```
host all xxx(dbuser) 0.0.0.0/0 method
```

After the task is complete, delete this record and restart the database again.

- Before data migration, ensure that the network has been prepared well and security rules have been configured. If the connection fails, perform the following operations to check whether the network configuration is correct:
 - Public network
 - i. Ensure that public accessibility is enabled for the database.
 - ii. Ensure that the security rules of the database are correctly configured.

You need to add the EIP of the DRS instance to the whitelist of the database to ensure that the DRS instance can access the database. The IP address displayed on the **Configure Source and Destination Databases** page is the EIP of the DRS instance.
 - iii. Ensure that the firewall settings are correct.

The firewall of the data center must allow access from the EIP of the DRS instance so that the DRS instance can access the database.

Inbound access: Allows access from the EIP of the DRS instance to the database listening port.

Outbound access: Allows data transmission from the database listening port to the EIP of the DRS instance.
 - VPC
 - i. Ensure that the database security group is correctly configured.

View inbound rules to allow traffic from the private IP address of the DRS instance to the database listening port. The IP address displayed on the **Configure Source and Destination Databases** page is the private IP address of the DRS instance.
 - ii. Ensure that the database network ACL is correctly configured.

By default, a VPC does not have a network ACL. If you have configured a network ACL, add an inbound rule.

- VPN or Direct Connect
 - i. Ensure that the database VPN or Direct Connect is correctly configured.
 - ii. Ensure that the security rules of the database are correctly configured.

You need to add the private IP address of the DRS instance to the whitelist of the database to ensure that the DRS instance can access the database. The IP address displayed on the **Configure Source and Destination Databases** page is the private IP address of the DRS instance.

For more details about network settings, see [Overview of Preparations](#).

1.7.2 Checking Whether the Destination Database Is Connected

In the pre-check phase, DRS checks the connectivity and accuracy of the destination database IP address, port, username, and password.

Failure Cause

- The username or password is incorrect.
- The port cannot be accessed.
- The database account does not allow remote connection.
- Kafka brokers are unavailable.
- The connection fails.

Handling Suggestion

- Check whether the username and password entered during the DRS connection test are correct. Enter the correct database username and password and perform the pre-check again.
- If the port entered during the connection test cannot be accessed, check whether the port exists. If the port is correct, check whether the firewall is enabled.
- If the destination database is PostgreSQL, and the database configuration file **pg_hba.conf** does not contain the database account configuration, grant the remote connection permission for the account.

Add the following to **pg_hba.conf**, and restart the database for the modification to take effect:

```
host all xxx(dbuser) 0.0.0.0/0 method
```

After the task is complete, delete this record and restart the database again.

- If the destination database is Kafka, the possible causes are as follows:
 - Check whether Kafka brokers are normal.
 - Check whether security authentication is enabled on Kafka. If security authentication is enabled, select the corresponding security connection mode. For details, see [Kafka Authentication](#).

- Before data migration, ensure that the network has been prepared well and security rules have been configured. If the connection fails, perform the following operations to check whether the network configuration is correct:
 - Public network
 - i. Ensure that public accessibility is enabled for the database.
 - ii. Ensure that the security rules of the database are correctly configured.

You need to add the EIP of the DRS instance to the whitelist of the database to ensure that the DRS instance can access the database. The IP address displayed on the **Configure Source and Destination Databases** page is the EIP of the DRS instance.
 - iii. Ensure that the firewall settings are correct.

The firewall of the data center must allow access from the EIP of the DRS instance so that the DRS instance can access the database.

Inbound access: Allows access from the EIP of the DRS instance to the database listening port.

Outbound access: Allows data transmission from the database listening port to the EIP of the DRS instance.
 - VPC
 - i. Ensure that the database security group is correctly configured.

View inbound rules to allow traffic from the private IP address of the DRS instance to the database listening port. The IP address displayed on the **Configure Source and Destination Databases** page is the private IP address of the DRS instance.
 - ii. Ensure that the database network ACL is correctly configured.

By default, a VPC does not have a network ACL. If you have configured a network ACL, add an inbound rule.
 - VPN or Direct Connect
 - i. Ensure that the database VPN or Direct Connect is correctly configured.
 - ii. Ensure that the security rules of the database are correctly configured.

You need to add the private IP address of the DRS instance to the whitelist of the database to ensure that the DRS instance can access the database. The IP address displayed on the **Configure Source and Destination Databases** page is the private IP address of the DRS instance.

For more details about network settings, see [Overview of Preparations](#).

1.8 Database Objects

1.8.1 Checking Whether the Source Database Contains a MyISAM Table

MySQL

Table 1-49 Checking whether the source database contains a MyISAM table

Check Item	Whether the source database contains a MyISAM table
Description	If the source database contains a MyISAM table, the migration will fail.
Item to Be Confirmed and Handling Suggestion	Item to be confirmed: The source database contains MyISAM tables that are not supported by the destination database, which may cause the migration to fail. Handling suggestion: Convert the tables in the source database to InnoDB tables and try again. Alternatively,

1.8.2 Checking Whether the Source Database Contains the Functions or Stored Procedures that the Source Database User Is Not Authorized to Migrate

MySQL

Table 1-50 Checking whether the source database contains the functions or stored procedures that the source database user is not authorized to migrate

Check Item	Whether the source database contains the functions or stored procedures that the source database user is not authorized to migrate.
Description	The source database contains the functions or stored procedures that the source database user is not authorized to migrate.
Failure Cause and Handling Suggestion	Failure cause: The source database user does not have the permission to migrate functions and stored procedures. Handling suggestion: Ensure that the source database user has the highest-level right.

1.8.3 Checking Whether Objects with the Same Names Exist in the Source Database

MySQL -> PostgreSQL Synchronization

Table 1-51 Checking whether objects with the same names exist in the source database

Check Item	Whether objects with the same names exist in the source database
Description	Failure cause: The source databases selected are not mapped to the same database, or tables with same names exist in the selected databases.
Failure Cause and Handling Suggestion	Failure cause: PostgreSQL does not support synchronization of multiple databases, or objects have the same names in the source databases. Handling suggestion: Select one database for migration, or map the selected databases to the same database and ensure that the object in the databases have unique names.

1.8.4 Whether the Source Database Contains Unlogged Tables

PostgreSQL as the Source

Table 1-52 Whether the source database contains unlogged tables

Check Item	Whether the source database contains unlogged tables
Description	Check whether the source database contains unlogged tables. If yes, the synchronization fails.
Item to Be Confirmed and Handling Suggestion	Item to be confirmed: The source database contains unlogged tables and modifications to these tables are not recorded in logs. As a result, incremental data of unlogged tables cannot be synchronized. Handling suggestions: Check whether incremental data of the unlogged tables needs to be synchronized. If yes, run the following command to change the unlogged tables to logged: ALTER TABLE TABLE_NAME SET LOGGED."

1.8.5 Checking Whether the Names of Views to Be Migrated Are the Same

Oracle Migration and Synchronization

Table 1-53 Checking whether the source database meets integrity constraints

Check Item	Whether the source database meets constraint integrity
Description	Check the constraint integrity of the source database. If the check result does not meet the migration requirements, the migration fails.
Item to Be Confirmed and Handling Suggestion	Failure cause: Tables to be migrated are dependent but the referenced tables are not to be migrated. Handling suggestion: Select the referenced tables in the migration object list.
	Failure cause: Views that cannot be migrated are selected in the migration object list. Handling suggestions: Check all objects which the views reference and select them for migration.
	Failure cause: The names of the views to be migrated are the same except for letter cases. Handling suggestions: Change the view names or do not migrate these views.
	Failure cause: Source database constraint names are the same except for letter cases. Handling suggestions: Change the constraint names or do not migrate the constraints whose names are the same except for letter cases.

1.8.6 Checking Whether the _id Fields in the Collection of the Source Database Have Indexes

MongoDB Migration

Table 1-54 Checking whether the _id fields in the collections of the source database have indexes

Check Item	Whether the _id fields in the collections of the source database have indexes
Description	Check whether the _id fields in the collections of the source database have indexes. If not, the migration fails.

Item to Be Confirmed and Handling Suggestion	<p>Failure cause: The <code>_id</code> fields in the collections of the source database must have indexes.</p> <p>Handling suggestion: Run the <code>db.CollectionName.ensureIndex({_id: 1})</code> command to add an index. If the index fails to be added and the system displays a message indicating that the <code>_id</code> value already exists, the collection cannot be migrated.</p>
---	---

1.8.7 Checking Whether the Index Length of the Source Database Exceeds the Limit

Oracle -> MySQL and Oracle -> TaurusDB Migration and Synchronization

Table 1-55 Checking whether the index length of the source database exceeds the limit

Check Item	Whether the index length of the source database exceeds the limit
Description	The migration fails because the index length of the source database exceeds the column length limit of the destination database.
Failure Cause and Handling Suggestion	<p>Failure cause: There are indexes in the source database exceed the column length limit of the destination database.</p> <p>Handling suggestion: 1. Delete the tables from the migration object. 2. Modify the index length.</p>

MySQL Migration, Synchronization, and Disaster Recovery

Table 1-56 Checking whether the index length of the source database exceeds the column length limit

Check Item	Whether the index length of the source database exceeds the limit
Description	The migration fails because the index length of the source database exceeds the column length limit of the destination database.

Failure Cause and Handling Suggestion	<p>Failure cause: There are indexes in the source database exceed the column length limit of the destination database.</p> <p>Handling suggestion: Set innodb_large_prefix of the destination database to ON, or return to the previous step and select the table to be migrated again.</p>
--	---

1.8.8 Checking Whether the Source Database Tables Use Storage Engines Not Supported by the Destination Database

Check whether the source database tables use storage engines not supported by the destination database. If yes, the migration fails.

Failure Cause

The source database tables use the storage engines that are not supported by the destination database.

Handling Suggestion

Step 1 Go back to the object selection page.

Step 2 Deselect the tables that use the storage engines not supported by the destination database.

Step 3 Click **Next** to perform the pre-check again.

----End

1.8.9 Checking Whether the Database Names Mapped to the Destination DB Instance Contain Unsupported Characters

MySQL

Table 1-57 Checking whether the database names mapped to the destination database contain unsupported characters.

Check Item	Whether the database names mapped to the destination DB instance contain unsupported characters.
Description	The following characters are not supported in the database names mapped to the destination DB instance: .<>\'

Item to Be Confirmed and Handling Suggestion	Failure cause: The database names mapped to the destination DB instance contain unsupported characters. Handling suggestion: Go back to the object selection page and change the source database names to be mapped to the destination DB instance.
---	--

1.8.10 Checking Whether the Source Database Tables Contain Primary Keys

MySQL Migration and Disaster Recovery

Table 1-58 Checking whether the source database tables contain primary keys

Check Item	Whether the source database tables contain primary keys
Description	If tables to be migrated in the source database do not contain primary keys, the migration may fail.
Item to Be Confirmed and Handling Suggestion	Item to be confirmed: The tables to be migrated in the source database do not contain primary keys. Handling suggestion: Create a primary key for the table. If the table does not have a primary key to uniquely identify every row, value comparison cannot be performed. Also, if the network connection is unstable, the data in the destination database may be inconsistent with that in the source database.

MySQL Synchronization

Table 1-59 Checking whether the source database tables contain primary keys

Check Item	Whether the source database tables contain primary keys
Description	If tables to be synchronized in the source database do not contain primary keys, the synchronization may fail.

Failure Cause and Handling Suggestion	<p>Failure cause: The tables to be synchronized in the source database do not contain primary keys.</p> <p>Handling suggestion: Create primary keys for the tables to improve performance.</p>
	<p>Failure cause: In a many-to-one task, tables with no primary key that have the same name as those in the destination database are not allowed in the source database.</p> <p>Handling suggestion: Modify tables without primary keys, delete tables with no primary key from the destination database, or do not migrate tables without primary keys.</p>
Item to Be Confirmed and Handling Suggestion	<p>Item to be confirmed: The tables to be synchronized in the source database do not contain primary keys.</p> <p>Handling suggestion: Create a primary key for the table. If the table does not have a primary key to uniquely identify every row, value comparison cannot be performed. Also, if the network connection is unstable, the data in the destination database may be inconsistent with that in the source database.</p>

Oracle Synchronization

Table 1-60 Checking whether the source database tables contain primary keys

Check Item	Whether the source database tables contain primary keys
Description	If the source database contains tables that do not have primary keys, a small amount of data may be inconsistent during synchronization.
Failure Cause and Handling Suggestion	<p>Failure cause: The table without a primary key lacks a unique identifier for rows. When the network is unstable, you may need to retry the task several times, or data inconsistency may occur.</p> <p>Handling suggestion: Add a primary key to a table, or do not synchronize the table that does not have a primary key.</p> <p>Statement for adding a primary key: <code>ALTER TABLE table_name ADD CONSTRAINT constraint-name PRIMARY KEY (column_name);</code></p> <p>To synchronize tables with primary keys, do not perform the ALTER TABLE MOVE, SPLIT/MERGE, FLASHBACK, or ALTER TABLE SHRINK SPACE operation during full synchronization. Otherwise, data duplication may increase.</p>

1.8.11 Checking Whether the Source Database Contains Triggers or Events

MySQL Migration

Table 1-61 Checking whether the source database contains triggers or events

Check Item	Whether the source database contains triggers or events
Description	To prevent unexpected operations on the destination database automatically triggered by triggers or events, this task starts the trigger or event migration only after you stop the task. If you close or disconnect the source database connection during the task running, triggers or events are not migrated.
Item to Be Confirmed and Handling Suggestion	Item to be confirmed: The source database contains triggers or events. Handling suggestion: Stop the task first and then disconnect the network to ensure the completeness of the migration.

1.8.12 Checking Whether the Source Database Referenced Roles Pass the Check

MongoDB Migration

Table 1-62 Checking whether the source database referenced roles pass the check

Check Item	Whether the source database referenced roles pass the check
Description	If the roles referenced by accounts to be migrated are not migrated to the destination database, the migration may fail.
Failure Cause and Handling Suggestion	Failure cause: The databases referenced by source database roles do not exist in the destination database, and are not displayed in the object selection list. Handling suggestion: Select the databases referenced by roles for migration or do not migrate the roles.
Suggestion	Failure cause: The roles referenced by source database roles do not exist in the destination database, and are not displayed in the role selection list. Handling suggestion: Select the referenced roles or do not migrate the roles that fail the check.

1.8.13 Checking Whether the Source Database Referenced Accounts Pass the Check

MongoDB Migration

Table 1-63 Checking whether the source database referenced accounts pass the check

Check Item	Whether the source database referenced accounts pass the check
Description	If the roles referenced by accounts to be migrated are not migrated to the destination database, the migration may fail.
Failure Cause and Handling Suggestion	Failure cause: The databases referenced by source database account do not exist in the destination database, and are not displayed in the object selection list. Handling suggestion: Select the roles referenced by accounts for migration or do not migrate the accounts.
	Failure cause: The databases referenced by source database account do not exist in the destination database, and are not displayed in the object selection list. Handling suggestion: Select the databases referenced by accounts for migration or do not migrate the accounts.

1.8.14 Checking Whether the Source Database Contains Schemas or Users Named cdc

Microsoft SQL Server as the Source

Table 1-64 Checking whether the source database contains schemas or users named cdc

Check Item	Whether the source database contains schemas or users named cdc
Description	In full+incremental mode, the source database contains a schema or user named cdc.
Failure Cause and Handling Suggestion	Failure cause: The source database contains a schema or user named cdc. Handling suggestion: Go back to the object selection page and deselect the schema and user named cdc.

1.8.15 Checking Whether Associated Objects Are Selected

PostgreSQL Serving as the Source in Synchronization

Table 1-65 Checking whether associated objects are selected

Check Item	Whether the associated objects are selected
Description	The associated objects must be selected for migration. Otherwise, the migration may fail.
Failure Cause and Handling Suggestion	Failure cause: Tables referenced by the foreign key in the table to be migrated are not selected for migration. Handling suggestion: Select the associated objects.
	Failure cause: The selected objects contain views associated with some tables or views that are not selected for migration. Handling suggestion: Select the associated objects.
	Failure cause: The tables associated with the child tables to be migrated are not selected for migration. Handling suggestion: Select the associated objects.

1.8.16 Checking Whether the Specified Objects Exist In the Destination Database

PostgreSQL to RDS for PostgreSQL Migration and Synchronization

Table 1-66 Checking whether the specified migration objects exist in the destination database.

Check Item	Whether the specified migration objects exist in the destination database.
Description	The objects with the same names as the source objects cannot exist in the destination database. Otherwise, the migration may fail.
Failure Cause and Handling Suggestion	Failure cause: The objects to be synchronized exist in the destination database. Handling suggestion: Delete the objects with the same names as the source objects and perform the verification again.

1.9 Database Configuration Items

1.9.1 Checking Whether the Source Database Name Is Valid

MySQL Migration

Table 1-67 Checking whether the source database name is valid

Check Item	Whether the source database name is valid
Description	The source database name cannot contain invalid characters. It must contain 1 to 64 characters, including only lowercase letters, digits, hyphens (-), and underscores (_). If the source database name contains any invalid character, the migration fails.
Failure Cause and Handling Suggestion	Failure cause: This item cannot be checked because the source database fails to be connected. Handling suggestion: Check whether the source database is connected.
	Failure cause: The source database name cannot contain invalid characters. It must contain 1 to 64 characters, including only lowercase letters, digits, hyphens (-), and underscores (_). Handling suggestion: Change the source database names that contain unsupported characters or go back to the previous page and select the databases that do not contain unsupported characters.

MongoDB Migration

Table 1-68 Checking whether the source database name is valid

Check Item	Whether the source database name is valid
Description	If the source database name contains invalid characters, the migration fails.
	Failure cause: The source database names cannot contain the following special characters: '<>'. Handling suggestion: Change the source database names that contain unsupported characters or go back to the previous page and select the databases that do not contain unsupported characters.

Oracle Migration

Table 1-69 Checking whether the source database name is valid

Check Item	Whether the source database name is valid
Description	The source database names cannot contain non-ASCII characters or special characters .><`\,?!" If the source database name contains invalid characters, the migration fails.
	Failure cause: The source database names contain unsupported characters. Handling suggestion: Change the source database names that contain unsupported characters or go back to the previous page and select the databases that do not contain unsupported characters.

1.9.2 Checking Whether the Source Database Table Name Is Valid

MySQL Migration

Table 1-70 Checking whether the source database table name is valid

Check Item	Whether the source database table name is valid
Description	If the source database table name contains invalid character, the synchronization task fails.
Failure Cause and Handling Suggestion	Failure cause: The source database table names contain unsupported characters, non-ASCII characters, or the following characters: ><\ Handling suggestion: To solve this problem, perform the following steps: Click Previous to return to the Select Migration Type page. Select a customized object and do not select the table that contains unsupported characters. Method 2: Change the table name.

PostgreSQL Migration

Table 1-71 Checking whether the source database table name is valid

Check Item	Whether the source database table name is valid
-------------------	---

Description	The source database table name cannot contain single quotation marks ('), double quotation marks ("), and periods (.). Ensure that the source database table name does not contain invalid characters. Otherwise, the synchronization task fails.
Failure Cause and Handling Suggestion	Failure cause: The source database names contain unsupported characters. Handling suggestion: Run the following command to change the source database names that contain unsupported characters: alter table old_name rename to new_name;

Oracle Migration

Table 1-72 Checking whether the source database table name is valid

Check Item	Whether the source database table name is valid
Description	The source database names cannot contain non-ASCII characters or special characters .><`\,?!" If the table name or view name of the source database contains any invalid character, the migration fails.
Failure Cause and Handling Suggestion	Failure cause: The table name or view name of the source database contains unsupported characters. Handling suggestion: <ul style="list-style-type: none"> Run the following command to change the table name that contains unsupported characters: alter table old_name rename to new_name; Run the following command to change the view name that contains unsupported characters: rename old_view_name to new_view_name;

1.9.3 Checking Whether the Source Database View Name Is Valid

MySQL

Table 1-73 Checking whether the source database contains view names with non-ASCII characters

Check Item	Whether the source database contains view names with non-ASCII characters
-------------------	---

Description	If the source database contains non-ASCII characters, the migration will fail.
Item to Be Confirmed and Handling Suggestion	<p>Failure cause: The source database view names contain unsupported characters, non-ASCII characters, or the following characters: ></\</p> <p>Handling suggestion: To solve this problem, perform the following steps:</p> <p>Method 1:</p> <p>Click Previous to return to the Select Migration Type page. Select a customized object and do not select the view name that contains unsupported characters.</p> <p>Method 2: Change the view name.</p>

1.9.4 Checking Whether the Source Database Collection Name Is Valid

MongoDB Migration

Table 1-74 Checking whether the source database collection name is valid

Check Item	Whether the source database collection name is valid
Description	<p>The source database collection names cannot contain slashes (/) or backslashes (\).</p> <p>Failure cause: The source database collection names cannot contain the following characters: ,<></p> <p>Handling suggestion: Modify the source database collection names that contain invalid characters or go back to the Select Migration Type page and select the collections that do not contain invalid characters.</p>

1.9.5 Checking Whether the Shard Key Can Be Obtained from the Source Database

MongoDB Migration

Table 1-75 Checking whether the shard keys can be obtained from the source database

Check Item	Whether the shard keys can be obtained from the source database
-------------------	---

Description	Check whether the destination database user permissions meet the migration requirements. If the permissions are insufficient, the migration will fail.
Failure Cause and Handling Suggestion	Item to be confirmed: The source database is a replica set but the shard keys have not been configured in the destination database. Handling suggestion: If the destination database cannot obtain the shard keys of the source database, the data in the source database will be migrated to the primary shard node of the sharded cluster in the destination database. To fully utilize the read/write performance, storage capability, and high availability of the cluster, see FAQs .
	Item to be confirmed: The source database type is unknown, and the shard keys have not been configured in the destination database. Handling suggestion: If the destination database cannot obtain the shard keys of the source database, the data in the source database will be migrated to the primary shard node of the sharded cluster in the destination database. To fully utilize the read/write performance, storage capability, and high availability of the cluster, see FAQs .
	Item to be confirmed: The source database contains collections that do not have shard keys configured. Handling suggestion: If the destination database cannot obtain the shard keys of the source database, the data in the source database will be migrated to the primary shard node of the sharded cluster in the destination database. To fully utilize the read/write performance, storage capability, and high availability of the cluster, see FAQs .

1.9.6 Checking Whether the Source Database Schema Name Is Valid

PostgreSQL

Table 1-76 Checking whether the source database schema name is valid

Check Item	Whether the source database schema name is valid
Description	The source database schema name cannot contain single quotation marks ('), double quotation marks ("), and periods (.). Ensure that the source database schema name does not contain invalid characters. Otherwise, the synchronization task fails.
Failure Cause and Handling Suggestion	Failure cause: The source database schema names contain unsupported characters. Handling suggestion: Run the following command to change the source database schema names that contain unsupported characters: alter schema old_name rename to new_name;

1.9.7 Checking Whether the Maximum Number of Chunks in the Destination Database Is Sufficient

MongoDB Migration

Table 1-77 Checking whether the maximum number of chunks in the destination database is sufficient

Check Item	Whether the maximum number of chunks in the destination database is sufficient
Description	The maximum number of chunks in the destination database is insufficient to support sharding and splitting of the source database. If the maximum number of chunks is reached, chunks are not split and the write performance is negatively affected.
Failure Cause and Handling Suggestion	Item to be confirmed: The maximum number of chunks in the destination database is insufficient to support sharding and splitting of the source database. If the maximum number of chunks is reached, chunks are not split and the write performance is negatively affected. Handling suggestion: Select a destination DB instance of higher specifications.

1.9.8 Checking Whether Archive Logs Are Enabled on the Source Oracle Database

Oracle -> MySQL Migration and Synchronization

Table 1-78 Checking whether archive logs are enabled on the source Oracle database

Check Item	Whether archive logs are enabled on the source Oracle database
Description	During incremental migration from Oracle to MySQL, archive logs must be enabled on the source Oracle database.
Failure Cause and Handling Suggestion	Failure cause: Archive logs are not enabled on the source Oracle database. Handling suggestion: Run the alter database archivelog command to enable archive logs on the source Oracle database.

Failure Cause and Handling Suggestion	<p>Failure cause: This item cannot be checked because the source database fails to be connected.</p> <p>Handling suggestion: Check whether the source database is connected.</p>
	<p>Failure cause: This item cannot be checked because the destination database fails to be connected.</p> <p>Handling suggestion: Check whether the destination database is connected.</p>
	<p>Failure cause: Insufficient user permissions</p> <p>Handling suggestion: Check whether the database user permissions meet the migration requirements.</p>
	<p>Handling suggestion:</p> <ul style="list-style-type: none"> • If you are migrating data to the cloud, determine whether to delete the databases with the same names as the source databases or specify a new destination DB instance based on site requirements. • If you are migrating data out of the cloud, determine whether to use the original destination database or specify a new destination DB instance based on site requirements.
	<p>Failure cause: During an incremental migration, the source and destination databases cannot have the same names.</p> <p>Handling suggestion: Determine whether to retain these databases in the destination RDS DB instance or specify another destination RDS DB instance.</p>

Migration from Redis to GeminiDB Redis

Table 1-81 Checking whether the names of the source and destination databases are the same

Check Item	Whether the names of the source and destination databases are the same
Description	Check whether the names of the source and destination databases are the same.
Failure Cause and Handling Suggestion	<p>Failure cause: The destination instance cannot contain databases with the same names as those in the source.</p> <p>Handling suggestion: Determine whether to retain these databases in the destination instance or specify another destination instance.</p>

Oracle -> MySQL/Oracle -> TaurusDB Synchronization

Table 1-82 Checking Whether the Names of the Source and Destination Databases Are the Same

Check Item	Whether the names of the source and destination databases are the same
Description	The mapping names of the tables to be synchronized in the source databases are the same as the names of the destination database tables.
Failure Cause and Handling Suggestion	<p>Failure cause: Source database names are the same except for letter cases.</p> <p>Handling suggestion: Change the table name or return to the object selection page and deselect the tables with the same name. Statement for changing the table name: ALTER TABLE old_table_name RENAME TO new_table_name;</p>
	<p>Failure cause: The destination database contains a table to be synchronized. The table name is the same as the mapping name of the table to be synchronized in the source database.</p> <p>Handling suggestion: Delete the destination database table. Statement for deleting a table: DROP TABLE table_name;</p>

Oracle -> PostgreSQL Synchronization

Table 1-83 Checking whether the names of the source and destination databases are the same

Check Item	Whether the names of the source and destination databases are the same
Description	The migration fails when the names of the source and destination databases are different.
Failure Cause and Handling Suggestion	<p>Failure cause: The source and destination databases must have the same names, except that the destination database must use only lowercase letters.</p> <p>Handling suggestion: Create a database with name in lowercase letters in the destination database.</p>
	<p>Failure cause: The names of the objects to be migrated are the same except for letter cases.</p> <p>Handling suggestion: Select only one database among the databases whose names are the same except for letter cases.</p>

	Failure cause: The names of the tables to be migrated are the same except for letter cases. Handling suggestion: Change table names or do not migrate the tables with the same names.
	Failure cause: The names of the tables to be migrated are the same as those in the destination database and use only lowercase letters. Handling suggestion: Change table names or do not migrate the tables with the same names.
	Failure cause: The destination database contains a table to be synchronized. The table name is the same as the mapping name of the table to be synchronized in the source database. Handling suggestion: Delete the destination database table. Statement for deleting a table: DROP TABLE table_name;

PostgreSQL > PostgreSQL Synchronization

Table 1-84 Checking whether the names of the source and destination databases are the same

Check Item	Whether the names of the source and destination databases are the same
Description	Check whether the source and destination databases have the same names to prevent existing databases from being overwritten. If the source and destination databases have the same name, the migration cannot be performed.
Failure Cause and Handling Suggestion	Failure cause: The destination database names cannot be the same. Handling suggestion: Change the destination database names to prevent the database from being overwritten.

DDM -> Oracle Synchronization

Table 1-85 Checking whether the names of the source and destination databases are the same

Check Item	Whether the names of the source and destination databases are the same
-------------------	--

Description	To synchronize data from DDM to Oracle, you need to create the corresponding database (user) in the destination database in advance. Otherwise, the synchronization fails.
Failure Cause and Handling Suggestion	<p>Failure cause: Some databases cannot be synchronized because the databases with the same names do not exist in the destination databases.</p> <p>Handling suggestion: Create these databases or users in the destination database or do not migrate these databases. Statement for creating a user: CREATE USER user_name IDENTIFIED BY password;</p>

1.10.2 Checking Whether the Same View Names Exist in Both the Source and Destination Databases

Migration from MongoDB to DDS

Table 1-86 Checking whether the same view names exist in both the source and destination databases

Check Item	Whether the same view names exist in both the source and destination databases
Description	Check whether the source and destination databases have the same view names to prevent existing views from being overwritten. If view of the same name exists in the destination database, the migration cannot be performed.
Failure Cause and Handling Suggestion	<p>Failure cause: The same view names exist in both the source and destination databases.</p> <p>Handling suggestions: Delete the destination database views that have the same names as those in the source database. Alternatively, do not migrate the views with the same names.</p>

1.10.3 Checking Whether the Destination Database Contains a Non-Empty Collection with the Same Name As the Source Database

MongoDB Migration

Table 1-87 Checking whether the destination database contains a non-empty collection with the same name as the source database

Check Item	Whether the destination database contains a non-empty collection with the same name as the source database
Description	Check whether the source and destination databases have the same non-empty collections to prevent existing databases from being overwritten. If the source and destination databases have the same collections, the migration cannot be performed.
Failure Cause and Handling Suggestion	Failure cause: This item cannot be checked because the source database failed to be connected. Handling suggestion: Check whether the source database is connected.
	Failure cause: This item cannot be checked because the destination database failed to be connected. Handling suggestion: Check whether the destination database is connected.
	Failure cause: The same non-empty collections exist in both the source and destination databases. Handling suggestion: Determine whether to delete the same non-empty collections from the destination DB instance or specify a new destination DB instance.

1.10.4 Checking Whether Destination Database Contains the Same Table Names As the Synchronization Objects

MySQL Synchronization

Table 1-88 Checking whether destination database contains the same table names as the synchronization objects

Check Item	Whether destination database contains the same table names as the synchronization objects. (table name conflicts)
-------------------	---

Descri ption	<p>The destination database contains objects with the same name as those in the source database. If table of the same name exists in the destination database, the migration cannot be performed.</p> <p>Exceptions: If the names and structures of tables in the source and destination databases are the same, the system determines that no conflict occurs.</p>
Failur e Cause and Handl ing Sugge stion	<p>Failure cause: The source and destination database tables cannot have the same names.</p> <p>Handling suggestions: Check whether the tables with the same names need to be retained. If yes, select another object for synchronization. If no, delete the tables with the same names.</p>
	<p>Failure cause: The destination database contains the same table names as those of the synchronization objects.</p> <p>Handling suggestions: Check whether the tables with the same names need to be retained. If yes, select another object for synchronization. If no, delete the tables with the same names.</p>
	<p>Failure cause: Both the source and destination databases are RDS DB instance and do not have mapped databases.</p> <p>Handling suggestion: Create mappings for the databases that are not mapped.</p>
	<p>Item to be confirmed: The destination database contains tables whose SRIDs are different from those in the source database.</p> <p>Handling suggestion: Check whether the geographic coordinate system used by the destination database table meets requirements. If the attribute of the geographic coordinate system has been specified at the source end, modify the structure of the destination database table to be the same as that at the source end.</p>

MariaDB Synchronization

Table 1-89 Whether destination database contains the same table names as those of the synchronization objects.

Check Item	<p>Whether destination database contains the same table names as the synchronization objects. (table name conflicts)</p>
Descri ption	<p>The destination database contains objects with the same name as those in the source database. If table of the same name exists in the destination database, the migration cannot be performed.</p> <p>Exceptions: If the names and structures of tables in the source and destination databases are the same, the system determines that no conflict occurs.</p>

Failure Cause and Handling Suggestion	<p>Failure cause: The source and destination database tables cannot have the same names.</p> <p>Handling suggestions: Check whether the tables with the same names need to be retained. If yes, select another object for synchronization. If no, delete the tables with the same names.</p>
	<p>Failure cause: The destination database contains the same table names as those of the synchronization objects.</p> <p>Handling suggestions: Check whether the tables with the same names need to be retained. If yes, select another object for synchronization. If no, delete the tables with the same names.</p>
	<p>Failure cause: Both the source and destination databases are RDS DB instance and do not have mapped databases.</p> <p>Handling suggestion: Create mappings for the databases that are not mapped.</p>

1.10.5 Checking Whether the Destination Database Contains Objects with the Same Name As Those in the Source Database

MySQL -> PostgreSQL Synchronization

Table 1-90 Checking whether the destination database contains objects with the same name as those in the source database

Check Item	Whether destination database contains objects with the same name as those in the source database
Description	The destination database contains objects with the same name as those in the source database. If a table of the same name exists in the destination database, the migration cannot be performed.
Failure Cause and Handling Suggestion	<p>Failure cause: The source and destination database tables cannot have the same names.</p> <p>Handling suggestions: Check whether the tables with the same names need to be retained. If yes, select another object for synchronization. If no, delete the tables with the same names.</p>
	<p>Failure cause: The destination database contains the same table names as those of the synchronization objects.</p> <p>Handling suggestions: Check whether the tables with the same names need to be retained. If yes, select another object for synchronization. If no, delete the tables with the same names.</p>

Oracle to PostgreSQL Synchronization

Table 1-91 Checking whether the destination database contains objects with the same name as those in the source database

Check Item	Whether the destination database contains objects with the same name as those in the source database
Description	The destination database contains objects with the same name as those in the source database.
Failure Cause and Handling Suggestion	<p>Failure cause: The destination database contains the data and indexes of the table to be synchronized.</p> <p>Handling suggestion: Delete data and indexes from the destination database table. Otherwise, data inconsistency may occur.</p> <ul style="list-style-type: none"> Statement for clearing data in a table: <code>TRUNCATE TABLE table_name1;</code> Statement for deleting an index: <code>DROP INDEX index_name ;</code>
	<p>Failure cause: The source database contains encrypted objects.</p> <p>Handling suggestion: Go back to the object selection page and select database objects that are not encrypted.</p>
	<p>Failure cause: Source database names are the same except for letter cases.</p> <p>Handling suggestion: Change the table name or return to the object selection page and deselect the tables with the same name. Statement for changing the table name: <code>ALTER TABLE old_table_name RENAME TO new_table_name;</code></p>

Microsoft SQL Server as the Source in Synchronization

Table 1-92 Checking whether the destination database contains objects with the same name as those in the source database

Check Item	Whether the destination database contains objects with the same name as those in the source database
Description	<p>The destination database contains objects with the same name as those in the source database.</p> <ul style="list-style-type: none"> If Table structure is selected, the destination database cannot contain objects with the same name as those in the source database. If Table structure is not selected, create the corresponding table structure in the destination database in advance.

Failure Cause and Handling Suggestion	<p>Failure cause: The destination database table does not exist and cannot be synchronized.</p> <p>Handling suggestion: If you do not synchronize the table structure, create the table to be synchronized in the destination database in advance or synchronize the table structure.</p> <p>Statement for creating a table in the destination database: CREATE TABLE table_name (column_name data_type);</p>
	<p>Failure cause: The table to be synchronized has been mapped to the destination database.</p> <p>Handling suggestion: Return to the Set Synchronization Task page, select the tables that meet the requirements. Alternatively, change the mapping names of the tables to be synchronized.</p>
	<p>Failure cause: The table to be synchronized has not been mapped to the destination database.</p> <p>Handling suggestion: Select Table structure to create a database table, or create the corresponding table structure in the destination database. If the table structure was not missing, check whether the name of the mapped table is correct.</p>

Table-Level Synchronization from PostgreSQL to PostgreSQL

Table 1-93 Checking whether destination database contains objects with the same name as those in the source database

Check Item	Whether destination database contains objects with the same name as those in the source database
Description	The destination database contains objects with the same name as those in the source database. If the same object names exist, the migration cannot be performed.
Failure Cause and Handling Suggestion	<p>Failure cause: The destination database contains objects with the same name as those in the source database.</p> <p>Handling suggestions: Check whether the objects with the same names need to be retained. If yes, select another object for migration. If no, delete the objects with the same names.</p>

1.10.6 Checking Whether Collections in Both the Source and Destination Databases Are Not Capped

MongoDB Migration

Table 1-94 Checking whether collections in both the source and destination databases are not capped

Check Item	Whether collections in both the source and destination databases are not capped
Description	Check whether collections in both the source and destination databases are not capped. If not, the migration fails.
Failure Cause and Handling Suggestion	Failure cause: the destination database has a collection whose name is the same as that of the capped collection of the source database. Handling suggestion: To ensure data consistency, you can delete the collection in the destination database with the same name as the capped collection in the source database. Alternatively, you can choose not to migrate the capped collection that will contradict with that in the destination database.
	Failure cause: The collections to be migrated are capped collections and already exist in the destination database. Handling suggestion: To ensure data consistency, you can delete the collection in the destination database with the same name as the capped collection in the source database. Alternatively, you can choose not to migrate the capped collection that will contradict with that in the destination database.

1.11 SSL Connections

1.11.1 Checking Whether the SSL Connection Is Correctly Configured

Check whether the SSL connection is correctly configured for the source or destination database. If SSL connection is enabled for the database, the database must be connected in SSL mode.

Failure Cause

SSL is enabled for the database. The database must be connected in SSL mode, but no certificate is uploaded.

Handling Suggestion

- On the **Configure Source and Destination Databases** page, enable the SSL connection and upload the certificate.

- Disable the SSL connection.

1.11.2 Checking Whether the SSL Connection Is Enabled for the Source Database

PostgreSQL

Table 1-95 Checking whether the SSL connection is enabled for the source database

Check Item	Whether the SSL connection is enabled for the source database
Description	Check whether the SSL connection is enabled for the source database.
Failure Cause and Handling Suggestion	Failure cause: The source database SSL connection is disabled. Handling suggestion: In the postgresql.conf file, set ssl_ca_file to the directory of an SSL root CA certificate and set ssl to on to enable the SSL connection. Then, restart the database for the modifications to take effect.

1.11.3 Checking Whether the SSL Certificate of the Source Database Exists

MySQL > MySQL

Table 1-96 Checking whether the SSL certificate of the source database exists

Check Item	Whether the SSL certificate of the source database exists
Description	Check whether the SSL certificate type of the source database is correct during MySQL to MySQL synchronization. Otherwise, the synchronization fails.
Failure Cause and Handling Suggestion	Failure cause: The source database uses SSL to encrypt connections but the SSL certificate does not exist. Handling suggestion: On the Configure Source and Destination Databases page, enable SSL connection for the source database and upload an encryption certificate that contains only one beginning tag BEGIN CERTIFICATE and one end tag END CERTIFICATE .

	<p>Failure cause: The SSL certificate type of the source database is not supported.</p> <p>Handling suggestion: On the Configure Source and Destination Databases page, enable SSL connection for the source database and upload an encryption certificate that contains only one beginning tag BEGIN CERTIFICATE and one end tag END CERTIFICATE.</p>
--	---

1.11.4 Checking Whether the SSL Certificate of the Destination Database Exists

MySQL

Table 1-97 Checking whether the SSL certificate of the destination database exists

Check Item	Whether the SSL certificate of the destination database exists
Description	Check whether the SSL certificate type of the destination database is correct during migration. Otherwise, the migration fails.
Failure Cause and Handling Suggestion	<p>Failure cause: The SSL certificate of the destination database does not exist.</p> <p>Handling suggestion: On the Configure Source and Destination Databases page, enable SSL connection for the destination database and upload an encryption certificate that contains only one beginning tag BEGIN CERTIFICATE and one end tag END CERTIFICATE.</p> <p>Failure cause: The SSL certificate type of the destination database is not supported.</p> <p>Handling suggestion: On the Configure Source and Destination Databases page, enable SSL connection for the destination database and upload an encryption certificate that contains only one beginning tag BEGIN CERTIFICATE and one end tag END CERTIFICATE.</p>

1.11.5 Checking Whether Both the Source and Destination Databases Use SSL

MongoDB Migration

Table 1-98 Checking whether both the source and destination databases use SSL to encrypt connections

Check Item	Whether both the source and destination databases use SSL to encrypt connections
Description	Check whether both the source and destination databases use SSL to encrypt connections.
Failure Cause and Handling Suggestion	Failure cause: Both the source and destination databases use SSL to encrypt connections. Handling suggestion: When migrating data to the cloud, disable SSL on the destination database. When migrating data out of the cloud, disable SSL on the source database. To disable SSL, go to the DDS basic information page.

1.12 Object Dependencies

1.12.1 Checking Whether the Objects Referenced by Views Are Selected for Migration

Migration from MongoDB to DDS

Table 1-99 Checking whether the objects referenced by views are selected for migration

Check Item	Whether the objects referenced by views are selected for migration
Description	The views and referenced objects should be migrated together. Otherwise, the migration fails.

Item to Be Confirmed and Handling Suggestion	<p>Failure cause: The views to be migrated have dependencies on the objects that are not to be migrated.</p> <p>Handling suggestion: Select the referenced objects in the migration object list. Alternatively, do not migrate the dependent views.</p>
---	---

1.12.2 Checking Whether Referenced Tables Are Selected for Migration

MySQL Migration and Synchronization

Table 1-100 Checking whether the tables referenced by the foreign key in the table to be migrated are selected for migration.

Check Item	Whether the tables referenced by the foreign key in the table to be migrated are selected for migration.
Description	The tables referenced by the foreign key in the table to be migrated are not selected for migration.
Item to Be Confirmed and Handling Suggestion	<p>Failure cause: Tables referenced by the foreign key in the table to be migrated are not selected for migration.</p> <p>Handling suggestion: Select the referenced tables.</p>

MariaDB Synchronization

Table 1-101 Checking whether the tables referenced by the foreign key in the table to be migrated are selected for migration.

Check Item	Whether the tables referenced by the foreign key in the table to be migrated are selected for migration.
Description	The tables referenced by the foreign key in the table to be migrated are not selected for migration.

Item to Be Confirmed and Handling Suggestion	Failure cause: Tables referenced by the foreign key in the table to be migrated are not selected for migration. Handling suggestion: Select the referenced tables.
---	---

1.13 Source Database Information

1.13.1 Checking Whether the Shards and Mongos Are in the Same Cluster

MongoDB Migration

Table 1-102 Checking whether the shards and mongos are in the same cluster

Check Item	Whether the shards and mongos are in the same cluster
Description	If the shards and mongos are not in the same cluster, the migration will fail.
Item to Be Confirmed and Handling Suggestion	Failure cause: The shards are in different clusters from mongos. Handling suggestion: On the connection test page, enter the shards which are in the same cluster as the mongos.

1.13.2 Checking Whether the Balancers of the Source Database Is Enabled

MongoDB Migration

Table 1-103 Checking whether the balancers of the source database is enabled

Check Item	Whether the balancers of the source database are enabled
-------------------	--

Description	If the source database contains the collections whose balancers are enabled, the migration will fail.
Item to Be Confirmed and Handling Suggestion	Failure cause: Balancers are enabled for the collections in the source database. Handling suggestions: Disable balancers for the collections.

1.13.3 Checking Whether the Source and Destination Database Types Match

MongoDB Migration

Table 1-104 Checking whether the source and destination database types match

Check Item	Whether the source and destination database types match
Description	If the source database type does not match the destination database type, the migration task will fail.
Failure Cause and Handling Suggestion	Failure cause: The source and destination database types do not match. Handling suggestion: If the source DB instance type is cluster, ensure that the object type corresponding to the input IP address and port of the source database cluster is mongos, the source shard database type is replica set, and the destination database type is cluster.

1.13.4 Long Transaction Check in the Source Database

GaussDB Serving as the Source in Incremental Synchronization

Table 1-105 Whether the source database has transactions that have been there for a long time without being submitted

Check Item	Checking Whether the Source Database Has Uncommitted Transactions
Description	If the source database has transactions that have been there for a long time without being submitted, the replication slot fails to be created. As a result, the incremental synchronization task fails.
Handling Suggestion	<ol style="list-style-type: none"> 1. Check whether the source DB instance has long transactions. The following uses 5 minutes as an example. <pre>select datname, pid, xact_start, state, query from pg_stat_activity where xact_start < current_timestamp - interval '300 seconds';</pre> 2. If there are long transactions, wait until the long transactions stop or kill the long transactions and try again. <pre>SELECT pg_terminate backend(281223408887296);</pre> Kill the backend process. 281223408887296 is the PID queried in the pg_stat_activity view. 3. If no long transaction is found, check whether there are prepared transactions. If yes, submit the transactions. <pre>SELECT * FROM pg_prepared_xacts;</pre>

1.14 Pre-Check Timeout

In the pre-check phase, DRS checks whether the prerequisites for data migration or synchronization are met in the source and destination databases to improve the task success rate. When the check duration exceeds the timeout threshold setting on the page, the pre-check result reporting times out.

Possible Causes

- The resources in the source or destination database are insufficient, and the query is slow.
- There are too many objects in the database.
- Too many objects are selected. Some pre-check items are queried by object. For example, check whether the selected table exists in the destination database and whether data exists.
- The DRS task specifications are too low. Due to physical resource restrictions, the pre-check may time out. This issue is more likely to occur when you edit a task.

Handling Suggestion

1. Check the load of the source and destination databases. If the load is heavy, perform the pre-check for the DRS task during off-peak hours.
2. Perform the pre-check again. If the check still times out, [submit a service ticket](#) in the upper right corner of the management console and contact DRS

customer service. If the page times out, the system is still running the pre-check. After the pre-check is complete, contact the customer service to skip the pre-check items that are successful.

2 Failure Cases

2.1 Case Overview

Table 2-1 Overview

Data Flow	Related Documents
Real-Time Migration from MongoDB to DDS	Full Migration Error: Prematurely reached end of stream
	Full Migration Error: not authorized on *** to execute command {***}
	Full Migration Error: GC overhead limit exceeded
	Full Migration Error: The background process is unavailable. Maybe it has been killed manually or by the operating system. Please restart the task if possible or wait for restarting by itself within 5 minutes."
	Full Migration Error: Timed out after 60000 ms while waiting to connect
	Full or Incremental Migration Error: Timed out after 60000 ms while waiting to connect
	Full or Incremental Migration Error: Invalid BSON field name ***
	Incremental Migration Error: Timed out after 60000 ms while waiting for a server that matches com.mongodb.client.internal
	Incremental Migration Error: Command failed with error *** (**):***. The full response is {***}"

Data Flow	Related Documents
Real-Time Migration and Synchronization from MySQL to MySQL	Full Phase Error: Table *** doesn't exist
	Full Phase Error: The background process is unavailable
	Full Phase Error: Communications link failure The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.
	Full Phase Error: Error writing file *** (errno: 28 - No space left on device)
	Full Phase Error: The MySQL server is running with the --super-read-only option so it cannot execute this statement
	Full Phase Error: The table *** is full
	Full Phase Error: Unknown column *** in 'field list'
	Full Phase Error: Lock wait timeout exceeded; try restarting transaction
	Full Phase Error: Java heap space
	Full Phase Error: Table *** already exists
	Full Phase Error: temp table: *** not exist
	Full Phase Error: failed to create new session
	Full Phase Error: load table: *** failed
	Full Phase Error: extract table structure failed!
	Full Phase Error: read table=*** failed
	Full Phase Error: CANNOT UPDATE USER WITH NULL PASSWORD
	Full Phase Error: Access denied for user *** to database ***
	Full Phase Error: The MySQL server is running with the --super-read-only option so it cannot execute this statement
	Full Phase Error: Temporary file write failure.
	Full Phase Error: Incorrect prefix key; the used key part isn't a string, the used length is longer than the key part, or the storage engine doesn't support unique prefix keys
Full Phase Error: Unknown database ***	
Full Phase Error: Access denied; you need (at least one of) the SUPER privilege(s) for this operation	
Full Phase Error: retry structures failed events and Table *** doesn't exist	

Data Flow	Related Documents
	Full Phase Error: shard table=*** failed
	Full Phase Error: error when split table shard occur!
	Full Phase Error: Column name 'AUTO_PK_ROW_ID' is reserved.
	Full Phase Error: transfer account failed, can not find password from src DB
	Full Phase Error: Failed to add the foreign key constraint '****' to system tables
	Full Phase Error: Too many keys specified; max 64 keys allowed
	Full Phase Error: Unknown collation: 'utf8mb4_0900_ai_ci'
	Full Phase Error: Invalid GIS data provided to function st_geometryfromtext'
	Full or Incremental Phase Error: Access denied for user ***
	Full or Incremental Phase Error: binlog is not existed
	Full or Incremental Phase Error: database log download failed
	Full or Incremental Phase Error: Can not read response from server
	Full or Incremental Phase Error: Communications link failure
	Full or Incremental Phase Error: EOF Packet received, master disconnected
	Full or Incremental Phase Error: Extract db create sql failed
	Full or Incremental Phase Error: load database structure failed in source database
	Full or Incremental Phase Error: load table: *** failed
	Full or Incremental Phase Error: Reached end of input stream
	Full or Incremental Phase Error: Read timed out
	Full or Incremental Phase Error: The background process is unavailable
	Full or Incremental Phase Error: Duplicate entry *** for key 'PRIMARY'
	Full or Incremental Phase Error: cause by: Index: ***, Size: ***
	Full or Incremental Phase Error: The offset and file name between src and parser is inconsistency

Data Flow	Related Documents
	Full or Incremental Phase Error: core process is not healthy or crashed
	Full or Incremental Phase Error: table info of table `***` from metadata miss
	Full or Incremental Phase Error: binlog parse fail, data dictionary may be not complete!
	Full or Incremental Phase Error: table *** record field size for insert/delete dml
	Full or Incremental Phase Error: service *** failed, cause by: Unable to connect to DBMS: ***
	Full or Incremental Phase Error: The binlog fetch connection may be interrupted
	Full or Incremental Phase Error: Received error packet: errno - 1047, solstate - HY000 errmsg = Unknown command
	Incremental Phase Error: not equals to target db column count
	Incremental Phase Error: The MySQL server is running with the --super-read-only option
	Incremental Phase Error: you need (at least one of) the SUPER privilege(s) for this operation
	Incremental Phase Error: Can't DROP ***; check that column/key exists
	Incremental Phase Error: Can't find file: *** (errno: 2 - No such file or directory)
	Incremental Phase Error: Data truncation: Data too long for column
	Incremental Phase Error: Failed to read file header from
	Incremental Phase Error: Lock wait timeout exceeded
	Incremental Phase Error: Must seek before attempting to read next event
	Incremental Phase Error: Table *** already exists
	Incremental Phase Error: Table *** doesn't exist
	Incremental Phase Error: Table *** not found in database
	Incremental Phase Error: source has more columns than target
	Incremental Phase Error: Unknown storage engine
	Incremental Phase Error: Unknown table

Data Flow	Related Documents
	<p>Incremental Phase Error: You have an error in your SQL syntax</p> <p>Incremental Phase Error: not illegal for mariaDb gtid position</p> <p>Incremental Phase Error: without PK execute failed</p> <p>Incremental Phase Error: Deadlock found when trying to get lock</p> <p>Incremental Phase Error: current serverUUID not equals to this session</p> <p>Incremental Phase Error: Slave has more GTIDs than the master has, using the master's SERVER_UUID.</p> <p>Incremental Phase Error: Operation not allowed when innodb_force_recovery > 0</p> <p>Incremental Phase Error: filter data in config condition filter error</p>
<p>Real-Time Migration and Synchronization from MySQL to TaurusDB</p>	<p>Full or Incremental Phase Error: Illegal mix of collations (utf8mb4_0900_ai_ci,IMPLICIT) and (utf8mb4_general_ci,IMPLICIT) for operation</p>
<p>Real-Time Synchronization from PostgreS QL to PostgreS QL</p>	<p>Task Startup Error: Initialize logical replication stream failed, the source database may have a long transaction: ****.</p> <p>Full Synchronization Error: function *** does not exist</p> <p>Full Synchronization Error: relation *** does not exist</p> <p>Full Synchronization Error: GC overhead limit exceeded</p> <p>Full Synchronization Error: Java heap space</p> <p>Full Synchronization Error: column *** of relation *** does not exist</p> <p>Full Synchronization Error: column *** does not exist</p> <p>Full Synchronization Error: type 'hstore' does not exist</p> <p>Full Synchronization Error: type 'geometry' does not exist</p> <p>Full Synchronization Error: Check that the hostname and port are correct and that the postmaster is accepting TCP/IP connections</p>

Data Flow	Related Documents
	Full Synchronization Error: invalid locale name
	Full Synchronization Error: password must not equal user name
	Full Synchronization Error: permission denied for schema ***
	Full or Incremental Phase Error: service *** failed, cause by: Unable to connect to DBMS: ***
	Full or Incremental Phase Error: Initialize logical replication stream failed, the source database may have a long transaction
	Full or Incremental Phase Error: memory required is *** MB, maintenance_work_mem is *** MB
	Full or Incremental Phase Error: temporary file size exceeds temp_file_limit
	Incremental Synchronization Error: Table *** not found in target database
	Incremental Synchronization Error: remaining connection slots are reserved
	Incremental Synchronization Error: PL/pgSQL function *** line *** at SQL statement
	Incremental Synchronization Error: The replication slot does not exist and the task is not started for the first time.
Real-Time Synchronization with Oracle Serving as the Source	Full Synchronization Error: has date/datetime: *** which is outside of dest allowed range
	Full or Incremental Phase Error: Got minus one from a read call
	Incremental Synchronization Error: Source supplemental log level is PK/UI. Missing column data at delete+insert on ****
	Incremental Synchronization Error: timeout when get next file log, maybe has been deleted, please check it.
	Incremental Synchronization Error: Failed to construct kafka producer.
	Incremental Synchronization Error: Topic *** not present in metadata after 300000 ms

Data Flow	Related Documents
Real-Time Synchronization with GaussDB Serving as the Source	Task Startup Error: Initialize logical replication stream failed, the source database may have a long transaction: ****.
	Incremental Synchronization Error: The replication slot does not exist and the task is not started for the first time.
Real-Time DR with MySQL Serving as the Source	DR Error: A dml without pk write target db fail
Backup Migration	Backup Migration Failed Because Backup Files Cannot Be Found
	Backup Migration Failed Because a Backup Database Cannot Be Found in the Backup Files
	Backup Migration Failed Because the Database with the Same Name Already Exists
	Backup Migration Failed Because an Incremental Backup File Is Used
	Backup Migration Failed Because a Log Backup File Is Used
	Backup Migration Failed Because the Backup File Verification Failed
	Backup Migration Failed Because of Insufficient Space
	Backup Migration Failed Because Database Names Are Not Specified
	Backup Migration Failed Because a Full Backup File Is Used
	Backup Migration Failed Because the LSNs of Incremental Backup Files Are Inconsecutive
Workload Replay	Parsing Failed, and a Message Is Displayed Indicating That the OBS Connection Failed
Data-Level Comparison	Data-Level Comparison Error: service SDV failed! cause by: the size of records in one shard[***] of target database, exceeds the max size 200000

2.2 Real-Time Migration from MongoDB to DDS

2.2.1 Full Migration Error: Prematurely reached end of stream

Scenarios

During real-time MongoDB-to-DDS migration, an error is reported during full migration. The log information is as follows: service DATAMOVE failed, cause by: Prematurely reached end of stream.

Possible Causes

The number of connections to the source or destination database is insufficient. Check the maximum number of connections to the source or destination database and the number of used connections. Generally, the number of connections used by DRS is about 10 on the source database and 20 on the destination database.

Solution

- Step 1** Adjust the number of connections to the database.
- For DDS, query and adjust the number of connections to a database by referring to [DDS User Guide](#).
 - For MongoDB, adjust the number of connections to a database by referring to the official document.
- Step 2** After the adjustment is complete, click **Resume** in the **Operation** column to resume the task.
- Step 3** If the fault persists, in the upper right corner of the management console, [submit a service ticket](#) and contact DRS customer service.

----End

2.2.2 Full Migration Error: not authorized on *** to execute command {***}

Scenarios

During real-time migration from MongoDB to DDS, an error is reported during full migration. The log information is as follows: service DATAMOVE failed, cause by: apply event=[type=table_data, batch_index_in_shard=1, table_schema=***, table_name=***, record_num=2720] occur error, msg=Command failed with error 13 (Unauthorized): 'not authorized on *** to execute command {***}'.

Possible Causes

The migration account used by DRS does not have the write permission on the destination database.

Solution

Step 1 Grant the destination database write permission to the DRS migration account. For details, see the MongoDB official documents or DDS user guide.

Step 2 After the adjustment is complete, click **Resume** in the **Operation** column to resume the task.

----End

2.2.3 Full Migration Error: GC overhead limit exceeded

Scenarios

During real-time migration from MongoDB to DDS, an error is reported during full migration. The log information is as follows: service DATAMOVE failed, cause by: GC overhead limit exceeded.

Possible Causes

- The size of a single data record in the source database is too large.
- The replication instance specifications are too small.

Solution

Step 1 Check whether the task is normal.

- If the task is normal, this error is recorded in the log and no further action is required.
- If the task is abnormal, go to [Step 2](#).

Step 2 Submit a service ticket in the upper right corner of the management console.

----End

2.2.4 Full Migration Error: The background process is unavailable. Maybe it has been killed manually or by the operating system. Please restart the task if possible or wait for restarting by itself within 5 minutes."

Scenarios

During real-time migration from MongoDB to DDS, an error is reported during full migration. The log information is as follows: service DATAMOVE failed, cause by: The background process is unavailable. Maybe it has been killed manually or by the operating system. Please restart the task if possible or wait for restarting by itself within 5 minutes.

Possible Causes

- The synchronization process is abnormal.

Solution

No further operation is required. The DRS daemon process automatically resumes the task. The migration and synchronization services are not affected, and data is resumed from the breakpoint. If the task is abnormal, click **Resume** in the **Operation** column to resume the task.

2.2.5 Full Migration Error: Timed out after 60000 ms while waiting to connect

Scenarios

During real-time migration from MongoDB to DDS, an error is reported during full migration. The log information is as follows: service DATAMOVE failed, cause by: [reason]:Failed to connect to database due to network, check the network between the DRS and the database or try again later.[message]:apply event=[type=table_data, batch_index_in_shard=144, table_schema=***, table_name=***, record_num=8510] occur error, msg=Timed out after 60000 ms while waiting to connect. Client view of cluster state is {type=UNKNOWN, servers=[{*** type=UNKNOWN, state=CONNECTING, exception={com.mongodb.MongoSocketOpenException: Exception opening socket}, caused by {java.net.ConnectException: Connection refused (Connection refused)}}].

Possible Causes

- The network is unstable. As a result, the connection times out when data is written to the destination database.
- The destination database is busy. As a result, the connection times out.

Solution

1. Check whether the destination database is running properly.
2. Check whether packet loss or retransmission occurs on the network between the DRS replication instance and the destination database.
3. In the upper right corner of the console, submit a service ticket to change the default timeout interval.

2.2.6 Full or Incremental Migration Error: Timed out after 60000 ms while waiting to connect

Scenarios

During real-time migration from MongoDB to DDS, an error is reported during full or incremental migration. The log information is as follows: service LOGMANAGER failed, cause by: [reason]:Failed to connect to database due to network, check the network between the DRS and the database or try again later.[message]:Timed out after 60000 ms while waiting to connect. Client view of cluster state is {type=UNKNOWN, servers=[{***, type=UNKNOWN, state=CONNECTING, exception={com.mongodb.MongoSocketOpenException: Exception opening socket}, caused by {java.net.ConnectException: Connection refused (Connection refused)}}]

Possible Causes

- The network is unstable. As a result, the connection to the source database times out.
- The source database is busy. As a result, the connection times out.

Solution

1. Check whether the source database is running properly.
2. Check whether packet loss or retransmission occurs on the network between the DRS replication instance and the source database.
3. In the upper right corner of the console, submit a service ticket to change the default timeout interval.

2.2.7 Full or Incremental Migration Error: Invalid BSON field name ***

Scenarios

During real-time migration from MongoDB to DDS, an error is reported during full or incremental migration. The log information is as follows: service DATAMOVE failed, cause by: apply event=[type=table_data, batch_index_in_shard={***}, table_schema={***}, table_name={***}, record_num={***}] occur error, msg=Invalid BSON field name {***}

Possible Causes

The field contains invalid characters, such as periods (.) and dollar signs (\$).

Solution

Check and remove invalid symbols in the source database. In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

2.2.8 Incremental Migration Error: Timed out after 60000 ms while waiting for a server that matches com.mongodb.client.internal

Scenarios

During real-time migration from MongoDB to DDS, an error is reported during incremental migration. The log information is as follows: service INCREMENT failed, cause by: [reason]:Failed to connect to database due to network, check the network between the DRS and the database or try again later.[message]:Timed out after 60000 ms while waiting for a server that matches com.mongodb.client.internal.MongoClientDelegate\$1@27105e1a. Client view of cluster state is {type=REPLICA_SET, servers=[{address=***, type=UNKNOWN, state=CONNECTING, exception={com.mongodb.MongoSocketOpenException: Exception opening socket}, caused by {java.net.ConnectException: Connection timed out (Connection timed out)}}, {address=***, type=UNKNOWN,

state=CONNECTING, exception={com.mongodb.MongoSocketOpenException: Exception opening socket}, caused by {java.net.NoRouteToHostException}

Possible Causes

- The network is unstable. As a result, the connection times out when data is written to the destination database.
- The destination database is busy. As a result, the connection times out.

Solution

1. Check whether the destination database is running properly.
2. Check whether packet loss or retransmission occurs on the network between the DRS replication instance and the destination database.
3. In the upper right corner of the console, submit a service ticket to change the default timeout interval.

2.2.9 Incremental Migration Error: Command failed with error *** (***):***. The full response is {***}"

Scenarios

During real-time migration from MongoDB to DDS, an error is reported during incremental migration. The log information is as follows: service INCREMENT failed, cause by: [reason]:The database returns an error.[message]:Command failed with error *** (***):***. The full response is {***}.

Possible Causes

The destination database returns an error. Common error codes are as follows:

- Error 91: The destination database service is abnormal.
- Error 133: The destination database shard is abnormal.
- Error 10107: The primary node of the destination database is abnormal.

Solution

Contact destination database engineers to locate and rectify the fault.

2.3 Real-Time Migration and Synchronization from MySQL to MySQL

2.3.1 Full Phase Error: Table *** doesn't exist

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: Table '***' doesn't exist.

Possible Causes

During the full phase, DDL statements are executed in the source database to delete tables.

Solution

Solution 1

During the full migration and synchronization phases, the DELETE command cannot be performed. For details about how to recreate a task, see [From MySQL to MySQL](#).

Solution 2

Create a table with the same structure as the deleted table in the source database. In the task list on the **Online Migration Management** page, locate the task and click **Resume** in the **Operation** column.

2.3.2 Full Phase Error: The background process is unavailable

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: The background process is unavailable. Maybe it has been killed manually or by the operating system. Please restart the task if possible or wait for restarting by itself within 5 minutes.

Possible Causes

During the full migration or synchronization phase, the DRS process is terminated unexpectedly.

Solution

No further operation is required. The DRS daemon process automatically resumes the task. The migration and synchronization services are not affected, and data is resumed from the breakpoint. If the task is abnormal, click **Resume** in the **Operation** column to resume the task.

2.3.3 Full Phase Error: Communications link failure The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: Unable to connect to DBMS: url=jdbc:mysql://*** user=root, Caused by: Communications link failure The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.

Possible Causes

The connection to the source or destination database fails to be established.

Solution

1. Check whether the source or destination database is running properly.
2. Check whether the network connection between the DRS instance and the source or destination database is normal.
3. Check whether the whitelist of the source or destination database allows access from DRS instance IP addresses.

2.3.4 Full Phase Error: Error writing file *** (errno: 28 - No space left on device)

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: apply data of table=` %s`.` %s` failed: Error writing file '***' (errno: 28 - No space left on device).

Possible Causes

The destination database storage space is insufficient. As a result, data fails to be written to the destination database.

Solution

- Step 1** Adjust the storage space of the destination database.
- If RDS for MySQL is used, see [RDS for MySQL Performance Tuning](#) or contact RDS customer service to adjust the destination database storage space.
 - If an on-premises MySQL database or a MySQL database built on another cloud is used, contact database O&M engineers to check and adjust the destination database storage space.
- Step 2** After the adjustment is complete, click **Resume** in the **Operation** column to resume the task.

----End

2.3.5 Full Phase Error: The MySQL server is running with the --super-read-only option so it cannot execute this statement

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: apply data of table=` %s`.` %s` failed: The MySQL server is running with the --super-read-only option so it cannot execute this statement.

Possible Causes

The destination database is read-only. The possible cause is that the space of the destination database is insufficient.

Solution

Step 1 Adjust the storage space of the destination database and restore the destination database to the Read/Write state.

- If RDS for MySQL is used, see [RDS for MySQL Performance Tuning](#) or contact RDS customer service to adjust the destination database storage space.
- If an on-premises MySQL database or a MySQL database built on another cloud is used, contact database O&M engineers to check and adjust the destination database storage space.

Step 2 After the adjustment is complete, click **Resume** in the **Operation** column to resume the task.

----End

2.3.6 Full Phase Error: The table *** is full

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: apply event=[type=table_data, batch_index_in_shard=1, table_schema=%s, table_name=%s, record_num=%s,] occur error, msg=apply data of table=`%s`.`%s` failed: The table *** is full.

Possible Causes

The destination database storage space is insufficient. As a result, data fails to be written to the destination database.

Solution

Step 1 Adjust the storage space of the destination database.

- If RDS for MySQL is used, see [RDS for MySQL Performance Tuning](#) or contact RDS customer service to adjust the destination database storage space.
- If an on-premises MySQL database or a MySQL database built on another cloud is used, contact database O&M engineers to check and adjust the destination database storage space.

Step 2 After the adjustment is complete, click **Resume** in the **Operation** column to resume the task.

----End

2.3.7 Full Phase Error: Unknown column *** in 'field list'

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: apply event=[type=table_data, batch_index_in_shard=1, table_schema= %s, table_name= %s, record_num=5] occur error, msg=apply data of table=` %s`.` %s` failed: Unknown column ' %s' in'field list'.

Possible Causes

The table structures of the source and destination databases are inconsistent. The possible cause is that DDL is executed on the columns of the destination database table during full synchronization or the table consistency check is skipped during pre-check.

Solution

- Step 1** Contact the O&M personnel to change the table structure of the destination database to be the same as that of the source database.
- Step 2** After the change is complete, click **Resume** in the **Operation** column to resume the task.

----End

2.3.8 Full Phase Error: Lock wait timeout exceeded; try restarting transaction

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: apply event=[type=table_data, batch_index_in_shard=***, table_schema= %s, table_name= %s, record_num=***] occur error, msg=apply data of table=` %s`.` %s` failed: Lock wait timeout exceeded; try restarting transaction.

Possible Causes

- The service connection of the destination database holds the lock for a long time.
- The performance of the destination database is insufficient or the load is heavy, and the execution is slow.

Solution

- Step 1** Contact the O&M personnel to check the lock usage, slow SQL statements, or load status of the destination database.

Step 2 After the destination database is restored, click **Resume** in the **Operation** column to resume the task.

----End

2.3.9 Full Phase Error: Java heap space

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: apply event=[type=table_data, batch_index_in_shard=***, table_schema= %s, table_name= %s, record_num=***] occur error, msg=apply data of table=` %s ` ` %s ` failed: Java heap space.

Possible Causes

The size of a single record exceeds 50 MB.

Solution

Submit a service ticket in the upper right corner of the management console.

2.3.10 Full Phase Error: Table *** already exists

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: apply event=[type=table_rename_or_copy, index=null, schema_name= %s, object_name= %s] occur error, msg=rename table %s. %s error: Table '%s ' already exists.

Possible Causes

The table without a primary key already exists in the destination database.

Solution

Step 1 Contact the O&M engineers to delete the tables that do not have primary keys from the destination database.

Step 2 In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

----End

2.3.11 Full Phase Error: temp table: *** not exist

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: apply

```
event=[type=table_rename_or_copy, index=null, schema_name= %s, object_name= %s] occur error, msg=temp table: %s. %s not exist
```

Possible Causes

An exception occurred when a table without a primary key is being migrated.

Solution

Submit a service ticket in the upper right corner of the management console.

2.3.12 Full Phase Error: failed to create new session

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: com.continuent.tungsten.replicator.ReplicatorException: Unable to connect to DBMS: url=jdbc:mysql://*** user=***, Caused by: failed to create new session.

Possible Causes

The connection to the source or destination database fails to be established.

Solution

1. Check whether the source or destination database is running properly.
2. Check whether the network connection between the DRS instance and the source or destination database is normal.
3. Check whether the whitelist of the source or destination database allows access from the IP address of the DRS instance.

2.3.13 Full Phase Error: load table: *** failed

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: load table: `%s`.`%s` failed.

Possible Causes

The table structure of the source database fails to be loaded. The possible cause is that the user deletes tables during synchronization or the source database user does not have required permissions.

Solution

1. Contact the O&M engineers to check whether the table structure of the source database is normal. The common commands are as follows:

```
SELECT * FROM `%s`.`%s` LIMIT 1  
SHOW CREATE TABLE `%s`.`%s`
```

2. Contact the source database administrator to check whether the source database and tables have been deleted. If they were deleted, recreate the task by referring to [Precautions](#).
3. Check whether the migration account has the SHOW CREATE TABLE permission on the source database tables. If the account does not have the permission, grant the permission to the source database migration account by referring to [Precautions](#). Then, in the task list, click **Resume** in the **Operation** column to resume the task.

2.3.14 Full Phase Error: extract table structure failed!

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by:extract table structure failed! Table is %s. message is %s.

Possible Causes

The table structure of the source database fails to be loaded. The possible cause is that the user deletes tables during synchronization or the source database user does not have required permissions.

Solution

1. Contact the O&M engineers to check whether the table structure of the source database is normal. The common commands are as follows:

```
SELECT * FROM `%s`.`%s` LIMIT 1
SHOW CREATE TABLE `%s`.`%s`
```
2. Contact the source database administrator to check whether the source database and tables have been deleted. If they were deleted, recreate the task by referring to [Precautions](#).
3. Check whether the migration account has the SHOW CREATE TABLE permission on the source database tables. If the account does not have the permission, grant the permission to the source database migration account by referring to [Precautions](#). Then, in the task list, click **Resume** in the **Operation** column to resume the task.

2.3.15 Full Phase Error: read table=*** failed

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: read table=` %s`.` %s` failed.

Possible Causes

Failed to read table data from the source database due to poor source database performance or unstable network connection.

Solution

Step 1 In the upper right corner of the console, submit a service ticket to adjust the timeout interval for accessing the source database.

Step 2 After the adjustment is complete, click **Resume** in the **Operation** column to resume the task.

----End

2.3.16 Full Phase Error: CANNOT UPDATE USER WITH NULL PASSWORD

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: retry structures failed events=the fail structures are [type=account, index=0, schema_name=mysql, object_name='***']reason:[CANNOT UPDATE USER WITH NULL PASSWORD].

Possible Causes

The source database account password is empty.

Solution

Step 1 Contact the O&M engineers to add a password for the account that reports the error in the source database.

Step 2 In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

----End

2.3.17 Full Phase Error: Access denied for user *** to database ***

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: retry structures failed events=the fail structures are [type=account, index=0, schema_name=mysql, object_name='***']reason:[Access denied for user '***' to database '***']

Possible Causes

The DRS migration account does not have sufficient permissions on the destination database.

Solution

Step 1 Contact the O&M engineers to add the schema permission to the migration account in the destination database.

Step 2 In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

----End

2.3.18 Full Phase Error: The MySQL server is running with the --super-read-only option so it cannot execute this statement

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: retry structures failed events=the fail structures are [type=constraint_data, index=0, schema_name= %s, object_name= %s]reason:[The MySQL server is running with the --super-read-only option so it cannot execute this statement]

Possible Causes

When DRS migrates indexes, the destination database is in the read-only state. The possible cause is that the space of the destination database is insufficient.

Solution

Step 1 Contact the O&M engineers to check the status of the destination database and rectify the database fault.

Step 2 After the destination database is restored, click **Resume** in the **Operation** column to resume the task.

----End

2.3.19 Full Phase Error: Temporary file write failure.

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: retry structures failed events=the fail structures are [type=constraint_data, index=0, schema_name= %s, object_name= %s]reason:[Temporary file write failure.]

Possible Causes

The temporary space of the destination database is insufficient when DRS migrates indexes.

Solution

Step 1 Adjust the temporary space of the destination database.

- If an RDS for MySQL instance is used, perform the following operations to adjust the temporary space:
 - a. Scale up the space by referring to [RDS for MySQL Performance Tuning](#).

- b. Check whether the temporary space increases.
 - If yes, go to [Step 2](#).
 - If no, submit a service ticket in the upper right corner of the management console and contact RDS customer service to adjust the temporary space of the destination database.
- If an on-premises MySQL database or a MySQL database built on another cloud is used, contact database O&M engineers to check and adjust the destination database temporary space.

Step 2 After the adjustment is complete, click **Resume** in the **Operation** column to resume the task.

Step 3 If the fault persists, submit a service ticket in the upper right corner of the management console and contact DRS customer service.

----End

2.3.20 Full Phase Error: Incorrect prefix key; the used key part isn't a string, the used length is longer than the key part, or the storage engine doesn't support unique prefix keys

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: retry structures failed events=the fail structures are [type=constraint_data, index=106, schema_name= %s, object_name= %s]reason:[Incorrect prefix key; the used key part isn't a string, the used length is longer than the key part, or the storage engine doesn't support unique prefix keys]

Possible Causes

Table structures in the source and destination databases are inconsistent.

Solution

Step 1 Contact the O&M engineers to change the table structure of the destination database to be the same as that of the source database.

Step 2 In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

----End

2.3.21 Full Phase Error: Unknown database ***

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: retry structures failed events=the fail structures are [type=***, index=***, schema_name=***, object_name=***]reason:[Unknown database '***']

Possible Causes

- The database corresponding to the destination database structure does not exist.
- The database in which objects to be migrated or synchronized reside is not in the object selection list.

Solution

Step 1 Check whether the database specified in the error information exists in the destination database.

- If no, manually create a specified database in the destination database and ensure that the structure of the database is the same as that of the source database. Then, in the task list, locate the row that contains the target task and click **Resume** in the **Operation** column to submit the task again.
- If yes, go to [Step 2](#).

Step 2 Check whether the database specified in the error information exists in the source database.

- If yes, select the database again.
- If no, the database in which objects reside may have been deleted or DRS does not have the permission to read the database. In this case, objects cannot be migrated. Re-create the task and do not select the objects that reside in the deleted database.

----End

2.3.22 Full Phase Error: Access denied; you need (at least one of) the SUPER privilege(s) for this operation

Scenarios

During a full migration, an error is reported, and the log information is as follows:
service DATAMOVE failed, cause by: retry structures failed events=the fail structures are [type=function, index=2, schema_name= %s, object_name= %s]reason:[Access denied; you need (at least one of) the SUPER privilege(s) for this operation]

Possible Causes

The user retained the definer for migration, but the migration account does not have the super permission.

Solution

Grant the super permission to the destination database user and submit the task again. For details, see [RDS FAQs](#). Alternatively, you can choose not to migrate definers when recreating a task.

2.3.23 Full Phase Error: retry structures failed events and Table *** doesn't exist

Scenarios

During a full migration, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: retry structures failed events=the fail structures are [type=view, index=2, schema_name= %s, object_name= %s]reason: [Table ' %s. %s' doesn't exist]

Possible Causes

The user retained the definer for migration, but the definer is abnormal or does not exist.

Solution

Recreate a task and do not migrate definers.

2.3.24 Full Phase Error: shard table=*** failed

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: shard table=`%s`.`%s` failed

Possible Causes

The source database performance is insufficient or the network is unstable. As a result, the source database sharding times out.

Solution

- Step 1** Check whether the task is normal.
- If the task is normal, this error is recorded in the log and no further action is required.
 - If the task is abnormal, go to [Step 2](#).
- Step 2** Submit a service ticket in the upper right corner of the management console.
- End

2.3.25 Full Phase Error: error when split table shard occur!

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: error when split table shard occur! Table is %s .Error code is %s.

Possible Causes

The source database performance is insufficient or the network is unstable. As a result, the source database sharding times out.

Solution

Step 1 Check whether the task is normal.

- If the task is normal, this error is recorded in the log and no further action is required.
- If the task is abnormal, go to [Step 2](#).

Step 2 Submit a service ticket in the upper right corner of the management console.

----End

2.3.26 Full Phase Error: Column name 'AUTO_PK_ROW_ID' is reserved.

Scenarios

An error is reported during full migration or synchronization, and the following log information is as follows: service LOGMANAGER failed, cause by: create table *** error. Column name 'AUTO_PK_ROW_ID' is reserved. Operation 'CREATE' is not permitted.

Possible Causes

The **AUTO_PK_ROW_ID** column name is a reserved column name for the RDS for MySQL database and cannot be created by users.

Solution

- Check the tables whose column names contain **AUTO_PK_ROW_ID** in the source database, change the column names, and resume the task.
- Create a task again and do not select the tables whose column names contain **AUTO_PK_ROW_ID**.

2.3.27 Full Phase Error: transfer account failed, can not find password from src DB

Scenarios

During a full migration, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: transfer account failed, can not find password from src DB.

Possible Causes

The RDS security policy does not allow the user password to be empty. However, if the source database is an on-premises MySQL database, the user password can be empty.

Solution

Step 1 Run the following SQL statement in the source database to query users whose passwords are empty:

- MySQL 5.7 or later versions:

```
SELECT USER,HOST,authentication_string FROM MYSQL.user WHERE authentication_string IS NULL OR authentication_string=";
```
- MySQL 5.6 and earlier versions

```
SELECT USER,HOST,`password` FROM MYSQL.user WHERE `password` IS NULL OR `password`=";
```

Step 2 Run the following SQL statement to delete the users whose passwords are empty from the source database or set passwords for the users:

- Delete a user whose password is empty.

```
DROP USER **@**;
```
- Set a password for a user.

```
ALTER USER **@** IDENTIFIED BY **;
```

----End

2.3.28 Full Phase Error: Failed to add the foreign key constraint '***' to system tables

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: Failed to add the foreign key constraint '***' to system tables

Possible Causes

During the full phase, the destination database has foreign key constraints with the same name.

Solution

1. Run the following SQL statement to delete or rename the foreign key constraints with the same name in the destination database:

```
select * from information_schema.REFERENTIAL_CONSTRAINTS where CONSTRAINT_NAME = "foreign_key_name";
```
2. In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

2.3.29 Full Phase Error: Too many keys specified; max 64 keys allowed

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: retry structures failed events=the fail structures are [type=constraint_data, index=0,

schema_name=DB, object_name=TABLE]reason:[Too many keys specified; max 64 keys allowed]

Possible Causes

A maximum of 64 secondary indexes can be created for a single MySQL table. The total number of new and existing indexes in the destination database exceeds 64.

Solution

1. After manually creating the required indexes in the destination database, contact DRS O&M personnel to skip the migration of secondary indexes in the table.
2. In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

2.3.30 Full Phase Error: Unknown collation: 'utf8mb4_0900_ai_ci'

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: retry structures failed events=the fail structures are [type=table_structure, index=0, schema_name=m825, object_name=t1]reason: [Unknown collation: 'utf8mb4_0900_ai_ci']

Possible Causes

The source database version is later than the destination database version, or the source database is of a special type and supports the utf8mb4_0900_ai_ci collation, but the destination MySQL database does not support this collation. The DRS task fails to synchronize the table structure because the destination database does not support the collation.

Solution

- Step 1** Manually create a table structure in the destination database and modify the collation rule.
- Step 2** Create a DRS task again, deselect the table structure for synchronization, and perform full synchronization again.
- End

2.3.31 Full Phase Error: exist some xa transactions for long times, may lack some data for this Job!

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: exist some xa transactions for long times, may lack some data for this Job!

Possible Causes

There are XA transactions in the source database that have been there for a long time without being submitted, the transaction data may be lost, causing data inconsistency.

Solution

Step 1 Run the **XA RECOVER** statement in the source database to find the XA transaction that has not been submitted. Then, run the **xa commit '{xid}'** command, set **xid** to the data value returned by **xa recover**, and submit the XA transaction.

Step 2 If the preceding operations cannot be performed on the source database and the uncommitted XA transaction does not affect service data, contact DRS O&M personnel.

----End

2.3.32 Full Phase Error: Invalid GIS data provided to function st_geometryfromtext'

Scenarios

During a full migration or synchronization, an error is reported, and the log information is as follows: An error occurred in the process DATAMOVE, caused by: apply event=[type=table_data, shard_id=0, batch_index_in_shard=1, table_name=%s, record_num=1] occur error, msg=apply data of table=%s failed: Data truncation: Invalid GIS data provided to function st_geometryfromtext.

Possible Causes

During cross-version migration of MySQL geographic type data, the rules for checking the valid geographic type in the destination database are inconsistent with those in the source database. This problem usually occurs when MySQL 5.6 is migrated to MySQL 5.7 or 8.0. The geographic type data of the source database is in invalid format in the destination database. As a result, an error is reported when DRS is used to insert data into the destination database.

For example, when POLYGON is used, the number of coordinate points is verified in 5.7 or later. At least three coordinate points are required to form a valid polygon. In version 5.6, no verification is performed, and polygons represented by a single coordinate point can be inserted properly.

Solution

- Solution 1
 - a. Modify the geographic type data that does not meet the format requirements of the destination database version in the source database.
 - b. In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.
- Solution 2
 - a. Create a DRS task again and deselect the table for which the error is reported during object selection.
 - b. After the migration is complete, export the table data, modify the table data to the valid format of the destination database, and manually migrate the table data to the destination database.

2.3.33 Full or Incremental Phase Error: Access denied for user

Scenarios

During a full or incremental migration or synchronization, an error is reported. The log information is as follows: service %s failed, cause by: Unable to connect to DBMS: url=***?useUnicode=true&allowLoadLocalInfile=false&characterEncoding=UTF-8&connectTimeout=5000&useSSL=false&allowPublicKeyRetrieval=true&verifyServerCertificate=false&serverTimezone=UTC user=%s, Caused by: Access denied for user %s

Possible Causes

The connection to the source or destination database fails to be established.

Solution

1. Check whether the source or destination database is running properly.
2. Check whether the password for connecting to the source or destination database is correct.
3. Check whether the network connection between the DRS instance and the source or destination database is normal.
4. Check whether the IP address of the DRS instance is allowed to access the source or destination database.

2.3.34 Full or Incremental Phase Error: binlog is not existed

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service LOGMANAGER failed, cause by: binlog is not existed

Possible Causes

The binlog files in the source database were deleted. DRS cannot obtain logs from the source database.

Solution

Recreate a DRS task.

2.3.35 Full or Incremental Phase Error: database log download failed

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service LOGMANAGER failed, cause by: database log download failed, error code is %s.

Possible Causes

The binlog files in the source database were deleted. DRS cannot obtain logs from the source database.

Solution

Recreate a DRS task.

2.3.36 Full or Incremental Phase Error: Can not read response from server

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service LOGMANAGER failed, cause by: Can not read response from server. Expected to read 4 bytes, read 0 bytes before connection was unexpectedly lost.

Possible Causes

- The network bandwidth between the DRS replication instance and the source database is too small or unstable.
- The source database is overloaded.

Solution

Contact the source database O&M personnel to check the source database load and check whether packet loss occurs on the network between the source database and the replication instance.

2.3.37 Full or Incremental Phase Error: Communications link failure

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service %s failed, cause by: Communications link failure The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.

Possible Causes

The connection to the source or destination database fails to be established.

Solution

1. Check whether the source or destination database is running properly.
2. Check whether the network connection between the DRS instance and the source or destination database is normal.
3. Check whether the IP address of the DRS instance is allowed to access the source or destination database.

2.3.38 Full or Incremental Phase Error: EOF Packet received, master disconnected

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service LOGMANAGER failed, cause by: EOF Packet received, master disconnected

Possible Causes

- The source database is abnormal.
- The binlog file is abnormal.

Solution

Step 1 Contact the source database O&M personnel to check whether the source database is running properly.

Step 2 After the source database is restored, click **Resume** in the **Operation** column to resume the task.

----End

2.3.39 Full or Incremental Phase Error: Extract db create sql failed

Scenarios

During a full or increment migration or synchronization, an error is reported, and the log information is as follows: service LOGMANAGER failed, cause by: Extract db create sql failed, dbName = %s

Possible Causes

- During full migration, the user deleted databases from the source.
- The source database user does not have the permission to perform operations on the source database.

Solution

- Contact the source database administrator to check whether the source database has been deleted. If the source database was deleted, recreate the task by referring to the related section in the product documentation.
- Check whether the source database user has the SHOW CREATE TABLE permission on the source database table. If the user does not have the permission, grant the permission to the user and retry the DRS task.

2.3.40 Full or Incremental Phase Error: load database structure failed in source database

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service LOGMANAGER failed, cause by: load database structure failed in source database, The failed %s is: type=%s, schema_name=%s, object_name=%s, errorcode=%s, message=%s"

Possible Causes

- During full migration, the user deleted databases from the source.
- The source database user does not have the permission to perform operations on the source database.

Solution

- Contact the source database administrator to check whether the source database has been deleted. If the source database was deleted, recreate the task by referring to the related section in the product documentation.
- Check whether the source database user has the SHOW CREATE TABLE permission on the source database table. If the user does not have the permission, grant the permission to the user and retry the DRS task.

2.3.41 Full or Incremental Phase Error: load table: *** failed

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service LOGMANAGER failed, cause by: load table: `%s`. `%s` failed

Possible Causes

- During full migration, the user deleted databases or tables from the source.
- The source database user does not have the permission to perform operations on the source database or table.

Solution

1. Contact the source database administrator to check whether the databases and tables in the source database have been deleted. If they were deleted, recreate the task by referring to [Precautions](#).
2. Check whether the migration account has the SHOW CREATE TABLE permission on the source database tables. If the account does not have the permission, grant the permission to the source database migration account by referring to [Precautions](#). Then, in the task list, click **Resume** in the **Operation** column to resume the task.

2.3.42 Full or Incremental Phase Error: Reached end of input stream

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service LOGMANAGER failed, cause by: Reached end of input stream

Possible Causes

- The source database is abnormal.
- The binlog file is abnormal.

Solution

- Step 1** Contact the source database administrator to check whether the source database is running properly.
- Step 2** After the source database is restored, click **Resume** in the **Operation** column to resume the task.

----End

2.3.43 Full or Incremental Phase Error: Read timed out

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service LOGMANAGER failed, cause by: Read timed out

Possible Causes

The possible causes are as follows:

- Failed to connect to the source database.

Solution

1. Check whether the source database is running properly.
2. Check whether the network connection between the DRS instance and the source database is normal.

2.3.44 Full or Incremental Phase Error: The background process is unavailable

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service %s failed, cause by: The background process is unavailable. Maybe it has been killed manually or by the operating system. Please restart the task if possible or wait for restarting by itself within 5 minutes.

Possible Causes

During the migration, the DRS process stops unexpectedly.

Solution

No further operation is required. The DRS daemon process automatically resumes the task. The migration and synchronization services are not affected, and data is resumed from the breakpoint. If the task is abnormal, click **Resume** in the **Operation** column to resume the task.

2.3.45 Full or Incremental Phase Error: Duplicate entry *** for key 'PRIMARY'

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service CAPTURER failed, cause by: Duplicate entry '120' for key 'PRIMARY'.

Possible Causes

- **binlog_format** in the source database is not set to **ROW**.
- The **binlog_format** setting of the source database does not take effect immediately.

Solution

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

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

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

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

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

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

Method 1

1. Select a non-service period to disconnect all service connections on the current database.
 - a. Run the following command to query all service threads (excluding all binlog dump threads and current threads) in the current database:

```
show processlist;
```
 - b. Stop all the service threads queried in the previous step.

NOTE

Do not create or start a migration task before the preceding operations are complete. Otherwise, data may be inconsistent.

2. To prevent the binlog format of the source database from becoming invalid due to database restart, add or modify the **binlog_format** parameter in the startup configuration file (**my.ini** or **my.cnf**) of the source database and save the modification.

```
binlog_format=ROW
```

Method 2

1. To prevent the binlog format of the source database from becoming invalid due to database restart, add or modify the **binlog_format** parameter in the startup configuration file (**my.ini** or **my.cnf**) of the source database and save the modification.

```
binlog_format=ROW
```
2. Ensure that the **binlog_format** parameter is successfully added or modified. Then, restart the source database at a non-service period.

----End

2.3.46 Full or Incremental Phase Error: cause by: Index: ***, Size: ***

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service CAPTURER failed, cause by: Index: 8, Size: 7

Possible Causes

- **binlog_format** in the source database is not set to **ROW**.
- The **binlog_format** setting of the source database does not take effect immediately.

Solution

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

Step 2 Run the following command for setting global parameters in the source database.
`set global binlog_format = ROW;`

Step 3 Run the following command on the source database and check whether the preceding operation is successful:
`select @@global.binlog_format;`

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

Method 1

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

NOTE

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

Method 2

1. To prevent the binlog format of the source database from becoming invalid due to database restart, add or modify the **binlog_format** parameter in the startup configuration file (**my.ini** or **my.cnf**) of the source database and save the modification.

```
binlog_format=ROW
```

2. Ensure that the **binlog_format** parameter is successfully added or modified. Then, restart the source database at a non-service period.

----End

2.3.47 Full or Incremental Phase Error: The offset and file name between src and parser is inconsistency

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service CAPTURER failed, cause by: The offset and file name between src and parser is inconsistency

Possible Causes

- The source database is abnormal.
- The binlog file is abnormal.

Solution

- Step 1** Contact the source database administrator to check whether the source database is running properly.
- Step 2** After the source database is restored, click **Resume** in the **Operation** column to resume the task.

----End

2.3.48 Full or Incremental Phase Error: core process is not healthy or crashed

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service CAPTURER failed, cause by: core process is not healthy or crashed

Possible Causes

During the migration, the DRS process stops unexpectedly.

Solution

No further operation is required. The DRS daemon process automatically resumes the task. The migration and synchronization services are not affected, and data is resumed from the breakpoint. If the task is abnormal, click **Resume** in the **Operation** column to resume the task.

2.3.49 Full or Incremental Phase Error: table info of table `***` from metadata miss

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service CAPTURER failed, cause by: table info of table `%s`.`%s` from metadata miss

Possible Causes

The table may fail to be created due to DDL syntax incompatibility.

Solution

Step 1 Check whether the task is normal.

- If the task is normal, this error is recorded in the log and no further action is required.
- If the task is abnormal, go to [Step 2](#).

Step 2 Submit a service ticket in the upper right corner of the management console.

----End

2.3.50 Full or Incremental Phase Error: binlog parse fail, data dictionary may be not complete!

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service CAPTURER failed, cause by: binlog parse fail, data dictionary may be not complete! tableName: %s, databaseName:%s

Possible Causes

The table may fail to be created due to DDL syntax incompatibility.

Solution

Submit a service ticket in the upper right corner of the management console.

2.3.51 Full or Incremental Phase Error: table *** record field size for insert/delete dml

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service CAPTURER failed, cause by: table[%s.%s]record field size for insert/delete dml=%s, the column size in dictionary=%s

Possible Causes

Full image is not enabled for the source database binlog.

Solution

For details, see [How Do I Set binlog_row_image=FULL to Take Effect Immediately?](#)

2.3.52 Full or Incremental Phase Error: service *** failed, cause by: Unable to connect to DBMS: ***

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service *** failed, cause by: Unable to connect to DBMS: ***

Possible Causes

The connection to the source or destination database fails to be established.

Solution

1. Check whether the source or destination database is running properly.
2. Check whether the network connection between the DRS instance and the source or destination database is normal.
3. Check whether the IP address of the DRS instance is allowed to access the source or destination database.

2.3.53 Full or Incremental Phase Error: The binlog fetch connection may be interrupted

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: The binlog fetch connection may be interrupted

Possible Causes

DRS is disconnected from the source database to obtain binlogs. The possible cause is that the source database status has changed or the network is abnormal.

Solution

1. Check whether the source database is running properly.
2. Check whether the network connection between the DRS instance and the source database is normal.
3. In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

2.3.54 Full or Incremental Phase Error: Received error packet: errno - 1047, solstate - HY000 errmsg = Unknown command

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: service LOGMANAGER failed, cause by: Received error packet: errno - 1047, solstate - HY000 errmsg = Unknown command

Possible Causes

DRS failed to obtain binlogs. The source database may be a MySQL proxy node whose binlogs cannot be obtained.

Solution

1. Edit the DRS task and replace the source database with a node whose binlogs can be obtained.
2. In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

2.3.55 Incremental Phase Error: not equals to target db column count

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: write table %s. %s failed: table= %s. %s has field list size=[***] not equals to target db column count= %s

Possible Causes

DDL is executed on the destination database table, causing the table structure in the destination database to be inconsistent with that in the source database.

Solution

- Step 1** Contact the destination database O&M engineers to change the table structure of the destination database to be the same as that of the source database.
- Step 2** In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

----End

2.3.56 Incremental Phase Error: The MySQL server is running with the --super-read-only option

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: write table %s. %s failed: record tid: %s,seqno: %s with PK applied failed in table %s. %s, The MySQL server is running with the --super-read-only option so it cannot execute this statement

Possible Causes

The destination database is in the read-only state. Generally, the destination database storage is insufficient.

Solution

- Step 1** Contact the O&M engineers to check the running status and disk space of the destination database.
- Step 2** After the destination database is restored, click **Resume** in the **Operation** column to resume the task.

----End

2.3.57 Incremental Phase Error: you need (at least one of) the SUPER privilege(s) for this operation

Scenarios

During an incremental migration, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Access denied; you need (at least one of) the SUPER privilege(s) for this operation; sql is: CREATE DEFINER= %s

Possible Causes

The user retained the definer for migration, but the definer is abnormal or does not exist.

Solution

Recreate a task and do not migrate definers.

2.3.58 Incremental Phase Error: Can't DROP ***; check that column/key exists

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Can't DROP ' %s'; check that column/key exists; sql is %s

Possible Causes

DDL is executed on the destination database table, causing the table structure in the destination database to be inconsistent with that in the source database.

Solution

- Step 1** Contact the destination database O&M engineers to change the table structure of the destination database to be the same as that of the source database.
- Step 2** In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

----End

2.3.59 Incremental Phase Error: Can't find file: *** (errno: 2 - No such file or directory)

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Can't find file: ' %s' (errno: 2 - No such file or directory); sql is: %s

Possible Causes

The destination database table file is damaged.

Solution

- Step 1** Contact the destination database O&M engineers to check whether the corresponding table exists and is normal.
- Step 2** In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

----End

2.3.60 Incremental Phase Error: Data truncation: Data too long for column

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Data truncation: Data too long for column ' %s' at row %s; sql is: %s

Possible Causes

The DDL statement fails to be executed because data is too long.

Solution

- Step 1** Contact the destination database O&M engineers to check the structure of the table where the synchronization error is reported, adjust the length of the column where the error is reported, and adjust the column data type in the destination database.
- Step 2** In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

----End

2.3.61 Incremental Phase Error: Failed to read file header from

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Failed to read file header from thl.data.0000000011

Possible Causes

The format of the DRS data file is damaged.

Solution

- Step 1** Check whether the task is normal.
 - If the task is normal, this error is recorded in the log and no further action is required.
 - If the task is abnormal, go to [Step 2](#).
- Step 2** Submit a service ticket in the upper right corner of the management console.

----End

2.3.62 Incremental Phase Error: Lock wait timeout exceeded

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Lock wait timeout exceeded; try restarting transaction

Possible Causes

The lock wait times out when the destination database is accessed.

Solution

Step 1 Contact destination database O&M engineers to check the status and load of the destination database.

Step 2 In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

----End

2.3.63 Incremental Phase Error: Must seek before attempting to read next event

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Extract THL file fail! Must seek before attempting to read next event

Possible Causes

The task is interrupted for a long time. Historical DRS data files were deleted and the task cannot be continued.

Solution

Contact the user to re-create the task.

2.3.64 Incremental Phase Error: Table *** already exists

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Table ' %s' already exists

Possible Causes

A table has been created in the destination database. As a result, an error is reported when the DDL statement for creating a table in the source database is executed.

Solution

- Step 1** Contact destination database O&M engineers to delete the corresponding table from the destination database.
- Step 2** In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

----End

2.3.65 Incremental Phase Error: Table *** doesn't exist

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Table ' %s' doesn't exist; sql is: create %s like matches

Possible Causes

Tables are deleted from the destination database. As a result, the synchronization statement reports an error.

Solution

- Step 1** Contact destination database O&M engineers to create a table in the destination database based on the table structure of the source database.
- Step 2** After the table is created, click **Resume** in the **Operation** column to resume the task.

----End

2.3.66 Incremental Phase Error: Table *** not found in database

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Table %s not found in database

Possible Causes

The possible causes are as follows:

- Tables are deleted from the destination database. As a result, the synchronization statement reports an error.

- **Incremental DDLs** is not selected for **Synchronize** during task creation. After a table is created in the source database, DDL statements are filtered out. As a result, an error is reported during synchronization.

Solution

Contact the destination database O&M engineers to create a table in the destination database based on the table structure of the source database. After the table is created, click **Resume** in the **Operation** column to resume the task.

2.3.67 Incremental Phase Error: source has more columns than target

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Check table structure consistency fail! Table %s in source has more columns than target

Possible Causes

The table structure is modified in the destination database. As a result, the synchronization statement reports an error.

Solution

Step 1 Contact destination database O&M engineers to change the table structure of the destination database to be the same as that of the source database.

Step 2 After the change is complete, click **Resume** in the **Operation** column to resume the task.

----End

2.3.68 Incremental Phase Error: Unknown storage engine

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Unknown storage engine 'FEDERATED'; sql is: %s

Possible Causes

When the DDL table creation statement of the source database is replayed in destination database, the source DB engine is not supported by the destination database.

Solution

Step 1 Contact the user to create a table that supports the destination DB engine in the source database.

- Step 2** In the upper right corner of the console, submit a service ticket to skip the DDL statement that reports the error.
- Step 3** In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.
- End

2.3.69 Incremental Phase Error: Unknown table

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Unknown table ' %s'; sql is %s

Possible Causes

The table does not exist in the destination database.

Solution

- Step 1** Contact the user to create the table in the destination database based on the table structure of the source database.
- Step 2** In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.
- End

2.3.70 Incremental Phase Error: You have an error in your SQL syntax

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'START TRANSACTION' at line 39

Possible Causes

During the pre-check, the system skips version check. The syntax of the later version fails to be executed in the earlier version.

Solution

- Step 1** Contact the user to modify the statement based on the destination database syntax and run the statement in the destination database.
- Step 2** In the upper right corner of the management console, submit a service ticket to skip this error.

Step 3 In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

----End

2.3.71 Incremental Phase Error: not illegal for mariaDb gtid position

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: %s not illegal for mariaDb gtid position

Possible Causes

The gtid mode is changed during task creation.

Solution

Contact the user to recreate the task.

2.3.72 Incremental Phase Error: without PK execute failed

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: record tid: %s,seqno: %s without PK execute failed in table %s

Possible Causes

A conflict occurs during data synchronization for tables that do not have primary keys.

Solution

Step 1 Check whether the task is normal.

- If the task is normal, this error is recorded in the log and no further action is required.
- If the task is abnormal, go to [Step 2](#).

Step 2 Submit a service ticket in the upper right corner of the management console.

----End

2.3.73 Incremental Phase Error: Deadlock found when trying to get lock

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: record tid:

%s,seqno: %s with PK applied failed in table %s, Deadlock found when trying to get lock; try restarting transaction

Possible Causes

A deadlock occurs in the destination database.

Solution

Step 1 Contact destination database O&M engineers to check the status and load of the destination database.

Step 2 In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

----End

2.3.74 Incremental Phase Error: current serverUUID not equals to this session

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: write table %s failed: current serverUUID not equals to this session

Possible Causes

The destination database had a switchover.

Solution

Step 1 Contact destination database O&M engineers to check the destination database status.

Step 2 In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.

----End

2.3.75 Incremental Phase Error: Slave has more GTIDs than the master has, using the master's SERVER_UUID.

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service LOGMANAGER failed, cause by: Received error packet: errno = 1236, sqlstate = HY000 errmsg = Slave has more GTIDs than the master has, using the master's SERVER_UUID. This may indicate that the end of the binary log was truncated or that the last binary log file was lost, e.g., after a power or disk failure when sync_binlog != 1. The master may or may not have rolled back transactions that were already replicated to the slave. Suggest to

replicate any transactions that master has rolled back from slave to master, and/or commit empty transactions on master to account for transactions that have been.

Possible Causes

The source database position is rolled back, or the source database position is reset by running the **reset master** command.

Solution

In the task list, locate the target task and click **Reset** in the **Operation** column to reset the task. Alternatively, create a DRS task again.

2.3.76 Incremental Phase Error: Operation not allowed when innodb_force_recovery > 0

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: write table ***.*** failed: Operation not allowed when innodb_force_recovery > 0.

Possible Causes

The destination DB instance is abnormal. When the system variable **innodb_force_recovery** is set to be greater than **0** in the destination database, the INSERT, UPDATE, and DELETE operations are disabled in the destination database.

Solution

- Step 1** Contact destination database O&M engineers to check the destination database status.
 - Step 2** After the destination database is restored, click **Resume** in the **Operation** column to resume the task.
- End

2.3.77 Incremental Phase Error: filter data in config condition filter error

Scenarios

During an incremental migration or synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: filter data in config condition filter error!

Possible Causes

The data filtering conditions of the DRS synchronization task are incorrectly configured. As a result, the incremental data fails to be filtered.

Solution

Filtering rules cannot be modified for tables that have been synchronized. Create a synchronization task again.

2.4 Real-Time Migration and Synchronization from MySQL to TaurusDB

2.4.1 Full or Incremental Phase Error: Illegal mix of collations (utf8mb4_0900_ai_ci,IMPLICIT) and (utf8mb4_general_ci,IMPLICIT) for operation

Scenarios

During a full or incremental migration or synchronization, an error is reported, and the log information is as follows: Illegal mix of collations (utf8mb4_0900_ai_ci,IMPLICIT) and (utf8mb4_general_ci,IMPLICIT) for operation

Possible Causes

The sorting rule of the source MySQL 5.* character set utf8mb4 is utf8mb4_general_ci, and that of the destination TaurusDB character set utf8mb4 is utf8mb4_0900_ai_ci. An error is reported, indicating that the sorting rules are inconsistent.

Solution

- Solution 1
 - a. Run the SQL statement in the destination database to change the character set sorting rule of the corresponding column to utf8mb4_0900_ai_ci. For example, to change the character set sorting rule of column **c1** in table **test_collation_1** to utf8mb4_0900_ai_ci, run the following command:

```
ALTER TABLE test_collation_1 MODIFY COLUMN c1 VARCHAR(16) COLLATE utf8mb4_0900_ai_ci;
```
 - b. In the task list, locate the target task and click **Resume** in the **Operation** column to resume the task.
- Solution 2
 - a. Delete all columns containing collate utf8mb4_general_ci from the source database table.
 - b. In the task list, locate the target task and click **Reset** in the **Operation** column to reset the task. Alternatively, create a DRS task again.
- Solution 3
 - a. Run the SQL statement in the destination database to change the character set sorting rule of the destination database to utf8mb4_0900_ai_ci.

```
SET GLOBAL default_collation_for_utf8mb4='utf8mb4_general_ci';
```

- b. In the task list, locate the target task and click **Reset** in the **Operation** column to reset the task. Alternatively, create a DRS task again.

2.5 Real-Time Synchronization from PostgreSQL to PostgreSQL

2.5.1 Task Startup Error: Initialize logical replication stream failed, the source database may have a long transaction: ****.

Scenarios

An error is reported when a task fails to be started, and the log information is as follows: service LOGMANAGER failed, caused by: Initialize logical replication stream failed, the source database may have a long transaction: ****.

Possible Causes

- If the error information contains **detail:Read timed out:**, after a DRS task starts, the replication slot creation times out due to long transactions blocking or missing consistency position when DRS incremental capture creates a logical replication slot in the source database.
- If the error information contains **slot [***] is active:**, after a DRS task starts, the replication slot fails to be created due to long transaction blocking when DRS incremental capture creates a logical replication slot in the source database. In this case, the status is occupied (active). This error occurs when DRS automatically retries.

Solution

Run the following SQL statement to check whether there is a long transaction in the source database:

```
select datname, pid, xact_start, state, query from pg_stat_activity where xact_start < current_timestamp - interval '300 second'
```

- If yes, wait until the long transaction stops and try again.
- If no, submit a service ticket in the upper right corner of the management console and contact DRS customer service to increase the value of the connection timeout parameter.

2.5.2 Full Synchronization Error: function *** does not exist

Scenarios

During full synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: apply event=[type=table_structure, index=%s, schema_name=%s, object_name=%s] occur error, msg=ERROR: function *** does not exist Hint: No function matches the given name and argument types. You might need to add explicit type casts.

Possible Causes

Functions on which the table structure depends are not created in the destination database in advance. In table-level synchronization from PostgreSQL to PostgreSQL, functions and plugin objects cannot be synchronized. Therefore, you need to manually create functions on which the table structure depends in the destination database.

NOTE

You can log in to the corresponding database in the destination RDS for PostgreSQL and run the following SQL statement to check whether there is the function. In the command, *f_name* indicates the function name.

```
select n.nspname,p.proname,pg_get_functiondef(p.oid) as funcdef from pg_proc p left join pg_namespace n on p.pronamespace=n.oid where proname = 'f_name';
```

Solution

The missing function may belong to a plugin or is a user-defined function. Perform the following steps to check the source of the function in the source database, create the corresponding plugin or function in the destination database, and retry the DRS task.

Step 1 Log in to the source database and run the following SQL statement to query the plugin to which the function belongs (*f_name* indicates the function name):

```
select extname, nspname, proname,pg_get_function_arguments(c.oid) as funcargs from pg_extension e join pg_depend d on (d.refobjid=e.oid) join pg_proc c on (d.objid=c.oid) join pg_namespace n on c.pronamespace=n.oid where proname = 'f_name';
```

- If a query result is displayed, the function belongs to a plugin. The **extname** field in the query result indicates the plugin name. Go to [Step 2](#).
- If no query result is displayed, the function does not belong to any plugin and is a user-defined function. Go to [Step 3](#).

Step 2 If the function belongs to a plugin, click **Plugins** on the destination RDS for PostgreSQL management page and install the plugin.

Step 3 If the function is user-defined function, create the same function in the destination database as that in the source database. For details about the function definition statement, see the execution result of the following SQL statement in the source database. *f_name* indicates the function name.

```
select n.nspname,p.proname,pg_get_functiondef(p.oid) as funcdef from pg_proc p left join pg_namespace n on p.pronamespace=n.oid where proname = 'f_name';
```

Step 4 Retry the DRS task.

----End

2.5.3 Full Synchronization Error: relation *** does not exist

Scenarios

During full synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: ERROR: relation '%s' does not exist
Position: 15

Possible Causes

During table-level synchronization, objects with dependencies are not synchronized. For example, the source database contains tables A and B and table A depends on table B, but only table A is synchronized.

Solution

- Step 1** Clear data in the destination database.
 - Step 2** Create a synchronization task again and select the objects to be synchronized and all dependent objects.
 - Step 3** Start the synchronization task.
- End

2.5.4 Full Synchronization Error: GC overhead limit exceeded

Scenarios

During full synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: GC overhead limit exceeded.

Possible Causes

Too many large objects exist in the service. As a result, the memory usage of the synchronization task exceeds the threshold.

Solution

Submit a service ticket in the upper right corner of the management console.

2.5.5 Full Synchronization Error: Java heap space

Scenarios

During full synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: Java heap space.

Possible Causes

A large number of fields exist in the service. As a result, the memory usage of the synchronization task exceeds the threshold.

Solution

- Step 1** Check whether the task is normal.
 - If the task is normal, this error is recorded in the log and no further action is required.
 - If the task is abnormal, go to [Step 2](#).

Step 2 Submit a service ticket in the upper right corner of the management console.

----End

2.5.6 Full Synchronization Error: column *** of relation *** does not exist

Scenarios

During full synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: apply event=[type=table_data, batch_index_in_shard= %s, table_schema= %s, table_name= %s, record_num= %s] occur error, msg=apply table %s data failed: %s: ERROR: column ' %s' of relation ' %s' does not exist Position: 1043 Call getNextException to see other errors in the batch.

Possible Causes

- During the full synchronization, DDL operations are executed in the destination database. As a result, the table structure in the destination database is inconsistent with that in the source database.
- During the full synchronization, DDL operations are executed in the source database. As a result, the table structure in the destination database is inconsistent with that in the source database.

Contact the customer to confirm whether they executed DDL operations.

Solution

Create a synchronization task again. During the full synchronization, ensure that no DDL operation is executed on the source database and no data is written to the destination database. Otherwise, data may be inconsistent or the synchronization may fail.

2.5.7 Full Synchronization Error: column *** does not exist

Scenarios

During full synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: retry structures failed events=the fail structures are [type=function, index=0, schema_name= %s, object_name=' %s']reason:[ERROR: column ' %s' does not exist Position: %s].

Possible Causes

- During the full synchronization, DDL operations are executed in the destination database. As a result, the table structure in the destination database is inconsistent with that in the source database.
- During the full synchronization, DDL operations are executed in the source database. As a result, the table structure in the destination database is inconsistent with that in the source database.

Contact the customer to confirm whether they executed DDL operations.

Solution

Create a synchronization task again. During the full synchronization, ensure that no DDL operation is executed on the source database and no data is written to the destination database. Otherwise, data may be inconsistent or the synchronization may fail.

2.5.8 Full Synchronization Error: type 'hstore' does not exist

Scenarios

During full synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: retry structures failed events=the fail structures are [type=operator, index=2, schema_name=public, object_name=?]reason:[ERROR: type 'hstore' does not exist].

Possible Causes

The hstore plug-in is not installed on the destination database.

NOTE

Run the following SQL statement in the destination RDS PostgreSQL database:

```
select * from pg_extension where extname = 'hstore';
```

Solution

Extensions are not synchronized. Before synchronization, install the corresponding extension in the destination database. Perform the following steps to install the extension and retry the DRS task:

Step 1 Log in to the destination RDS PostgreSQL database as the **root** user.

Step 2 Run the following SQL statements to install hstore:

```
create extension "hstore";
```

Step 3 Retry the DRS task.

----End

2.5.9 Full Synchronization Error: type 'geometry' does not exist

Scenarios

During full synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: retry structures failed events=the fail structures are [type=operator, index=2, schema_name=public, object_name=?]reason:[ERROR: type 'geometry' does not exist].

Possible Causes

The postgis plug-in is not installed on the destination database.

 **NOTE**

Run the following SQL statement in the destination RDS for PostgreSQL database:

```
select * from pg_extension where extname = 'postgis';
```

Solution

Extensions are not synchronized. Before synchronization, install the corresponding extension in the destination database. Perform the following steps to install the extension and retry the DRS task:

Step 1 Log in to the destination RDS for PostgreSQL database as the **root** user.

Step 2 Run the following SQL statements to install postgis:

```
create extension "postgis";
```

Step 3 Retry the DRS task.

----End

2.5.10 Full Synchronization Error: Check that the hostname and port are correct and that the postmaster is accepting TCP/IP connections

Scenarios

During full synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: source engine postgresql client initialize failed, detail: Unable to connect to DBMS: url= %s/position3.0? client_encoding=UTF-8&ssl=false&sslmode=prefer user= %s, Caused by: Connection to %s refused. Check that the hostname and port are correct and that the postmaster is accepting TCP/IP connections.

Possible Causes

The connection to the source or destination database fails to be established.

Solution

Perform the following operations:

1. Check whether the source or destination database is running properly.
2. Check whether the DRS instance IP address is allowed by the listening port of the source or destination database.
 - For DRS tasks performed over a public network, the source database must allow access from the DRS instance EIP, and the destination database must allow access from the private IP address of the DRS instance.
 - For DRS tasks performed in a VPC, VPN, or Direct Connect network, both the source and destination databases must allow access from the private IP addresses of DRS instance.

2.5.11 Full Synchronization Error: invalid locale name

Scenarios

During full synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: target engine postgresql client initialize failed, detail: Unable to connect to DBMS: url= %s/position3.0? client_encoding=UTF-8&autosave=always&stringtype=unspecified&ssl=false&sslmode=prefer user= %s, Caused by: ERROR: invalid locale name:'Chinese (Simplified)_China.936'.

Possible Causes

The source database region type is not supported by the destination database.

Solution

Contact the customer to check whether the region type can be changed to another one (UTF-8 by default). The region type may affect the sorting rules of different languages. If it can be changed to UTF-8, submit a service ticket in the upper right corner of the management console.

2.5.12 Full Synchronization Error: password must not equal user name

Scenarios

During full synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: retry structures failed events=the fail structures are [type=account, index=0, schema_name=dummy, object_name=%s]reason:[ERROR: password must not equal user name].

Possible Causes

For RDS for PostgreSQL synchronization, the password cannot be the same as the username. If the password is the same as the username in the source database, an error will be reported when data is synchronized to the destination database.

Solution

Manually create the user in the destination database and click **Resume** on the DRS task management page to continue the synchronization.

2.5.13 Full Synchronization Error: permission denied for schema ***

Scenarios

During full synchronization, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: retry structures failed events=the fail

structures are [type=table_structure, index=0, schema_name=%s, object_name=%s]reason:[ERROR: permission denied for schema %s]

Possible Causes

The destination database user does not have the create permission on the schema.

Solution

Step 1 Run the following SQL statement in the destination database to grant the create permission on the schema to which the table owner belongs:

```
grant create on schema <schema_name> to <table_owner_in_source>;
```

Step 2 On the **Data Synchronization Management** page, click **Resume** to resume the synchronization task.

----End

2.5.14 Full or Incremental Phase Error: service *** failed, cause by: Unable to connect to DBMS: ***

Scenarios

During a full or incremental synchronization, an error is reported, and the log information is as follows: service *** failed, cause by: Unable to connect to DBMS: ***

Possible Causes

Failed to connect to the source or destination database.

Solution

1. Check whether the source or destination database is running properly.
2. Check whether the network connection between the DRS instance and the source or destination database is normal.
3. Check whether the IP address of the DRS instance is allowed to access the source or destination database.

2.5.15 Full or Incremental Phase Error: Initialize logical replication stream failed, the source database may have a long transaction

Scenarios

During a full or incremental synchronization, an error is reported, and the log information is as follows: service LOGMANAGER failed, cause by: Initialize logical replication stream failed, the source database may have a long transaction: ***

Possible Causes

A logical replication slot fails to be created in the source database.

Solution

Step 1 Check whether the number of replication slots in the source database reaches the upper limit. If yes, delete replication slots that are no longer used from the source database or increase the value of **max_replication_slots** and restart the source database.

- Run the following command to query the number of logical replication slots:

```
select count(1) from pg_replication_slots;
```
- Run the following command to query the maximum number of logical replication slots:

```
select setting as number from pg_settings where name = 'max_replication_slots';
```

Step 2 Check whether the source database has long transactions that are not submitted. If yes, slot creation times out. As a result, the task fails.

- Run the following command to query a transaction status:

```
select pid, datname, state, backend_xid, xact_start, (now() - xact_start) as cost from pg_stat_activity where backend_xid is not null order by xact_start;
```
- Run the following command to stop a long transaction:

```
select pg_terminate_backend(pid);
```

----End

2.5.16 Full or Incremental Phase Error: memory required is *** MB, maintenance_work_mem is *** MB

Scenarios

During a full or incremental synchronization, an error is reported, and the log information is as follows: retry structures failed events=the fail structures are [type=index, index=***, schema_name=***, object_name=***]reason:[ERROR: memory required is *** MB, maintenance_work_mem is *** MB]

Possible Causes

When an index is created in the destination database, the required memory is greater than the value of **maintenance_work_mem** configured for the database.

Solution

Step 1 Change the value of **maintenance_work_mem** in the destination database RDS for PostgreSQL to an appropriate value. For details, see [Modifying RDS for PostgreSQL Instance Parameters](#).

Step 2 Restart the database to apply the change. Then, On the **Data Synchronization Management** page, locate the target task and click **Resume** in the **Operation** column to resume the task.

----End

2.5.17 Full or Incremental Phase Error: temporary file size exceeds temp_file_limit

Scenarios

During a full or incremental synchronization, an error is reported, and the log information is as follows: retry structures failed events=the fail structures are [type=index, index=0, schema_name=fossbot, object_name=scan_mr5_file_union]reason:[ERROR: temporary file size exceeds temp_file_limit (20000000kB)]

Possible Causes

The size of the temporary table generated during SQL execution exceeds the upper limit of the temporary tablespace in the system.

Solution

- Step 1** Increase the value of the **temp_file_limit** parameter in the destination database by referring to [Modifying RDS for PostgreSQL Instance Parameters](#).
 - Step 2** Restart the database to apply the change. Then, On the **Data Synchronization Management** page, locate the target task and click **Resume** in the **Operation** column. After the synchronization task is complete, change the value to the original value. Otherwise, the DB instance disk may be full due to large temporary tablespace.
- End

2.5.18 Incremental Synchronization Error: Table *** not found in target database

Scenarios

During incremental synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Check table structure consistency fail! Table %s not found in target database

Possible Causes

- The user did not select to synchronize DDL, so the CREATE TABLE statement is not synchronized to the destination database.
- The user selected to synchronize DDL, but the source database uses the DDL statement that is not supported by DRS to create a table.
- The table is deleted from the destination database.

Solution

- Method 1: Create a task again and comply with the following DRS usage rules:
 - If you select to synchronize DDL, do not execute DDL statements that are not supported by DRS in the source database.

- If you do not synchronize DDL, do not execute DDL statements in the source database, or execute DDL statements in the destination database before executing the same DDL statements in the source database.
- During full and incremental synchronization, do not write data to the destination database. Otherwise, data may be inconsistent or the synchronization may fail.
- Method 2: Perform the following operations to restore the DRS task:
 - **Possible cause:** The user did not select to synchronize DDL, so the CREATE TABLE statement in the source database is not synchronized to the destination database.
Solution: Create a table in the destination database based on the table structure of the source database and retry the DRS task.
 - **Possible cause:** The user selected to synchronize DDL, but the source database uses the DDL statement that is not supported by DRS to create a table.
Solution: Create a table in the destination database based on the table structure of the source database and retry the DRS task.
 - **Possible cause:** The table is deleted from the destination database.
Solution: Re-create the table in the destination database based on the structure of the deleted table and retry the DRS task.

NOTE

If both the table and the data in the table are deleted, re-creating the table may lead to data inconsistency or cause the task to fail again.

2.5.19 Incremental Synchronization Error: remaining connection slots are reserved

Scenarios

During incremental synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Unable to connect to DBMS: url= %s user= %s, Caused by: FATAL: remaining connection slots are reserved for non-replication superuser connections.

Possible Causes

The number of user connections to the destination database reaches the upper limit. As a result, the connection to the destination database fails to be established.

NOTE

Log in to the destination RDS PostgreSQL database and run the following SQL statement:

- View max_connections.

```
show max_connections;
```
- Check the current number of connections.

```
select count(*) from pg_stat_activity;
```

Solution

On the destination RDS PostgreSQL database console, change the value of **max_connections** to a larger value and make it take effect. Each DRS task requires about 100 connections.

2.5.20 Incremental Synchronization Error: PL/pgSQL function *** line *** at SQL statement

Scenarios

During incremental synchronization, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: insert %s failed:tid: %s, sqno: %s, ERROR: %s' PL/pgSQL function %s line %s at SQL statement.

Possible Causes

The destination database **session_replication_role** is not set to **replica**, and the destination database trigger is not disabled.

NOTE

You can log in to the destination database RDS PostgreSQL and run the following SQL statement to check the value of **session_replication_role**:

```
show session_replication_role;
```

Solution

Step 1 On the RDS PostgreSQL console, change the value of **session_replication_role** to **replica** and apply the changes.

Step 2 Retry the DRS task.

----End

2.5.21 Incremental Synchronization Error: The replication slot does not exist and the task is not started for the first time.

Scenarios

During incremental synchronization, an error is reported, and the log information is as follows: service LOGMANAGER failed, caused by: The replication slot does not exist and the task is not started for the first time.

Possible Causes

The logical replication slot of the source database has been deleted. DRS cannot obtain logs connecting to the current synchronization position from the source database.

Solution

Create a DRS task for synchronization again.

2.6 Real-Time Synchronization with Oracle Serving as the Source

2.6.1 Full Synchronization Error: has date/datetime: *** which is outside of dest allowed range

Scenarios

During a full synchronization from Oracle to MySQL, an error is reported, and the log information is as follows: service DATAMOVE failed, cause by: java.lang.InterruptedExce^{ption}: Database: ***, Table: ***, Column: *** has date/datetime: *** 0:0:0 which is outside of dest allowed range.

Possible Causes

Oracle and MySQL heterogeneous databases support different time types. You can run the following SQL statement in the Oracle database to view data:

```
select to_char(column_name, 'SYYYY-MM-DD') from table_name;
```

Solution

In the upper right corner of the management console, submit a service ticket. After the fault is rectified, DRS writes data based on the following rules:

- If the destination database stores DATE data, 0000-01-01 00:00:00 is written.
- If the destination database stores TIMESTAMP data, 1970-01-01 00:00:01 is written.

2.6.2 Full or Incremental Phase Error: Got minus one from a read call

Scenarios

During a full or incremental synchronization with Oracle serving as the source, an error is reported, and the log information is as follows: service LOGMANAGER failed, cause by: Unable to connect to DBMS: url=jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=*.***.***)(PORT=1521))(CONNECT_DATA=*)) user=*, Caused by: IO Error: Got minus one from a read call.

Possible Causes

- The source database server rejects access from the IP address of the DRS task.
- The source database connection information has changed.
- The number of connections to the source database has reached the upper limit.

Solution

- Step 1** Modify the `sqlnet.ora` file in `$ORACLE_HOME/network/admin` to allow the IP address of the DRS task to access the source database.
- If a whitelist is used, `TCP.INVITED_NODES` must contain the DRS task IP address.
 - If a blacklist is used, `TCP.EXCLUDED_NODES` cannot contain the DRS task IP address.
- Step 2** Check whether the source database information (such as the IP address, port number, or service name/sid) is modified. If the source database information is modified, perform the following operations:
- Restore the modified source database information. The DRS task will automatically retry to continue the synchronization task.
 - Create a synchronization task again.
- Step 3** Run the following commands to check whether the number of connections to the source database has reached the upper limit.
- Run the following command to check the current number of connections:
`select count(*) from v$process;`
 - Run the following command to check the maximum number of connections:
`select value from v$parameter where name ='processes';`

If the number of connections to the source database has reached the upper limit, run the following command to change the maximum number of connections allowed by the database:

```
alter system set processes = 300 scope = spfile;
```

Restart the database for the modification to take effect.

----End

2.6.3 Incremental Synchronization Error: Source supplemental log level is PK/UI. Missing column data at delete+insert on ***"

Scenarios

During an incremental synchronization from Oracle to PostgreSQL or GaussDB, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Source supplemental log level is PK/UI. Missing column data at delete +insert on ***

Possible Causes

The supplemental log level of the source Oracle database is PK/UI. The **update** operation of the source database is not hit in the workload replay of the destination database. DRS converts the update operation to **delete** and **insert** operations by default. During the **insert** operation, the log does not contain data of other columns. As a result, an error is reported.

Solution

Change the supplemental log level of the source database to **ALL**. Then, in the task list, click **Reset** in the **Operation** column to submit the task again.

2.6.4 Incremental Synchronization Error: timeout when get next file log, maybe has been deleted, please check it.

Scenarios

During an incremental synchronization with Oracle serving as the source, an error is reported, and the log information is as follows: service CAPTURE failed, cause by: get next Oracle log file error. The next file is: 1.log, errorcode = 'code': '01300', 'name': 'LOGS_NOT_EXIST', 'retry': false, 'reset': false, 'level': 3, message = timeout when get next file log, maybe has been deleted, please check it.

Possible Causes

1. The source Oracle database is a physical standby database. The logs of the source database where the incremental startup position is located are not archived. As a result, DRS cannot obtain the logs.
2. There are problems about nodes and logs in the source database. As a result, DRS fails to obtain logs and reports an error.
3. The network is unstable, affecting the speed of obtaining logs from the source database. As a result, reading logs times out.

Solution

Step 1 After a DRS task is started, wait for about 10 minutes, click the task name, and check whether the error log is displayed on the **Synchronization Logs** page.

- If no, DRS has obtained logs.
- If yes, go to **Step 2**.

Step 2 If an error is reported for the LOGMANAGER process on the **Synchronization Logs** page, submit a service ticket in the upper right corner of the management console.

----End

2.6.5 Incremental Synchronization Error: Failed to construct kafka producer.

Scenarios

During an incremental synchronization from Oracle to Kafka, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Failed to construct kafka producer.

Possible Causes

If the destination Kafka instance is deployed across multiple AZs and an AZ fails, the preceding error may be reported when the Kafka client produces or consumes messages.

Solution

Step 1 Check and restore the Kafka partitioning.

Step 2 In the task list, locate the target task and click **Reset** in the **Operation** column to submit the task again.

----End

2.6.6 Incremental Synchronization Error: Topic *** not present in metadata after 300000 ms

Scenarios

During an incremental synchronization from Oracle to Kafka, an error is reported, and the log information is as follows: service INCREMENT failed, cause by: Topic *** not present in metadata after 300000 ms

Possible Causes

If the destination Kafka instance is deployed across multiple AZs and an AZ fails, the preceding error may be reported when the Kafka client produces or consumes messages.

Solution

Step 1 Check and restore the Kafka partitioning.

Step 2 In the task list, locate the target task and click **Reset** in the **Operation** column to submit the task again.

----End

2.7 Real-Time Synchronization with GaussDB Serving as the Source

2.7.1 Task Startup Error: Initialize logical replication stream failed, the source database may have a long transaction: ****.

Scenarios

An error is reported when a task fails to be started, and the log information is as follows: service LOGMANAGER failed, caused by: Initialize logical replication stream failed, the source database may have a long transaction: ****.

Possible Causes

- If the error information contains **detail:Read timed out:**, after a DRS task starts, the replication slot creation times out due to long transactions blocking or missing consistency position when DRS incremental capture creates a logical replication slot in the source database.
- If the error information contains **slot [***] is active:**, after a DRS task starts, the replication slot fails to be created due to long transaction blocking when DRS incremental capture creates a logical replication slot in the source database. In this case, the status is occupied (active). This error occurs when DRS automatically retries.

Solution

Run the following SQL statement to check whether there is a long transaction in the source database:

```
select datname, pid, xact_start, state, query from pg_stat_activity where xact_start < current_timestamp - interval '300 second'
```

- If yes, wait until the long transaction stops and try again.
- If no, submit a service ticket in the upper right corner of the management console and contact DRS customer service to increase the value of the connection timeout parameter.

2.7.2 Incremental Synchronization Error: The replication slot does not exist and the task is not started for the first time.

Scenarios

During incremental synchronization, an error is reported, and the log information is as follows: service LOGMANAGER failed, caused by: The replication slot does not exist and the task is not started for the first time.

Possible Causes

The logical replication slot of the source database has been deleted. DRS cannot obtain logs connecting to the current synchronization position from the source database.

Solution

Create a DRS task for synchronization again.

2.8 Real-Time DR with MySQL Serving as the Source

2.8.1 DR Error: A dml without pk write target db fail

Scenarios

During a DR task with MySQL serving as the source, an error is reported, and the log information is as follows: A dml without pk write target db fail

Possible Causes

- If a table does not have a primary key to uniquely identify every row, value comparison cannot be performed. Also, if the network connection is unstable, data written to the table without a primary key may be inconsistent with that in the source database.
- The source is RDS for MySQL of an earlier version (5-5.7.23). Tables that have no primary key contain hidden primary keys in the source database. As a result, the DRS task reports an error indicating that the update or delete operation is not hit.

Solution

- If the table does not have a primary key, create a primary key for the table and create a DRS DR task again.
- If the source is RDS for MySQL of an earlier version (5-5.7.23) and there are hidden primary keys in the tables having no primary key, perform the following steps:
 - a. Use an account with the process permission to run the following SQL statement at the source end to query table information. In the statement, *database/table* indicates the database name and table name of a table without a primary key. If the table is a partition table, use the **like** statement.
select * from information_schema.INNODB_SYS_TABLES where name = 'database/table';

```
mysql> select * from information_schema.INNODB_SYS_TABLES where name = 'test1/nopk1';
+-----+-----+-----+-----+-----+-----+-----+-----+
| TABLE_ID | NAME          | FLAG | N_COLS | SPACE | FILE_FORMAT | ROW_FORMAT | ZIP_PAGE_SIZE | SPACE_TYPE |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 44        | test1/nopk1  |      | 7       | 29    | Barracuda   | Dynamic    | 0              | Single     |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

- b. Run the following SQL statement to query the column information of the table without a primary key based on *TABLE_ID* obtained in **a**:
select * from information_schema.INNODB_SYS_COLUMNS where TABLE_ID = 44;

```
mysql> select * from information_schema.INNODB_SYS_COLUMNS where TABLE_ID = 44;
+-----+-----+-----+-----+-----+-----+
| TABLE_ID | NAME          | POS | MTYPE | PRTYPE | LEN |
+-----+-----+-----+-----+-----+-----+
| 44        | name1         | 0   | 12    | 2949135 | 128 |
| 44        | name3         | 1   | 12    | 2949135 | 128 |
| 44        | AUTO_PK_ROW_ID | 2   | 6     | 1288    | 8   |
| 44        | name4         | 3   | 12    | 2949135 | 128 |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

- c. According to the query result, the third column whose **POS** is **2** is the hidden auto-increment primary key column. If the hidden primary key column in binlog is not the last column, DRS synchronization will fail.
- d. Log in to the RDS console and **upgrade the minor kernel version** or contact RDS customer service to upgrade the version.
- e. Create a DRS task again.

2.9 Backup Migration

2.9.1 Backup Migration Failed Because Backup Files Cannot Be Found

Scenarios

When you migrate full backups from self-built OBS buckets to clouds, the following error message is displayed: restore:null.

Possible Causes

The possible causes are as follows:

- Backup files are deleted after you submit a backup migration task.
- When you upload backup files to a self-built OBS bucket, you select **Archive** for **Storage Class**. OBS archive storage offers cloud storage for rarely accessed data. An archive file uploaded for the first time is in the **Not restored** status. As a result, a Microsoft SQL Server DB instance cannot download the file.

Solutions

Based on the previous analysis, solutions are provided as follows:

Solution 1

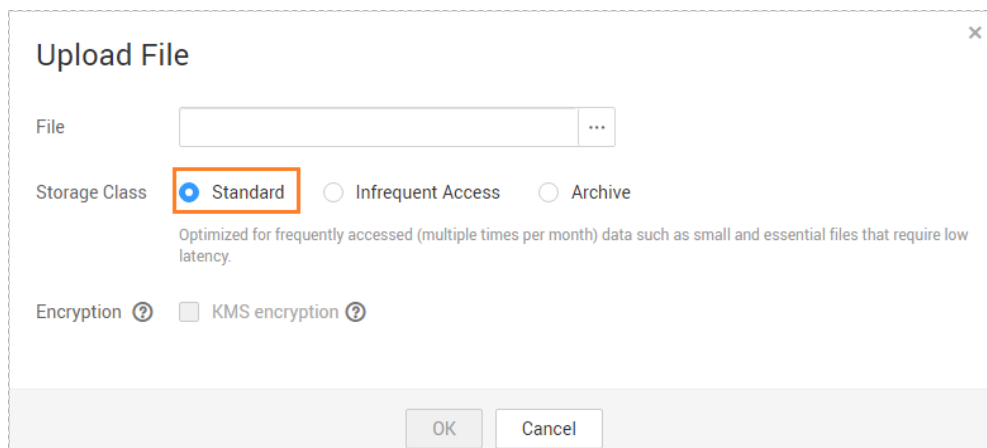
If the migration fails because you delete the backup files, you can upload the deleted backup files again to a self-built OBS bucket and select **Standard** for **Storage Class**. For details, see **Uploading a File** in *Object Storage Service Console Operation Guide*.

Solution 2

- If the migration failed because the storage class of your backup files is **Archive**, perform the following steps. If the size of backup files is small, upload the backup files again to an OBS bucket and select **Standard** for **Storage Class**.

For details, see "Uploading a File" in the *Object Storage Service Console Operation Guide*.

Figure 2-1 Uploading a file



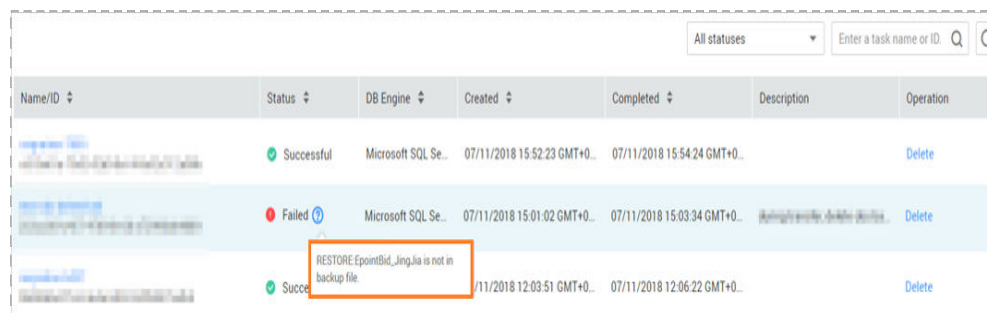
- If the backup files are large in size, log in to the OBS console and click the bucket to which the backup files are uploaded. On the displayed page, choose **Objects** in the navigation pane on the left. On the **Objects** page, select the object to be restored and click **Restore** above the file list. After the status of the backup files becomes **Restored**, submit an offline migration task again. For details, see the [Restoring an Archive File on OBS](#) section in the *Object Storage Service Console Operation Guide*.

2.9.2 Backup Migration Failed Because a Backup Database Cannot Be Found in the Backup Files

Scenarios

When you migrate full backups from self-built OBS buckets to clouds, the system displays an error message indicating that the migration failed because the source database cannot be found in the backup files.

Figure 2-2 Backup migration



Name/ID	Status	DB Engine	Created	Completed	Description	Operation
	Successful	Microsoft SQL Se...	07/11/2018 15:52:23 GMT+0...	07/11/2018 15:54:24 GMT+0...		Delete
	Failed	Microsoft SQL Se...	07/11/2018 15:01:02 GMT+0...	07/11/2018 15:03:34 GMT+0...	RESTORE: EpointBid_JingJia is not in backup file	Delete
	Successful		/11/2018 12:03:51 GMT+0...	07/11/2018 12:06:22 GMT+0...		Delete

Possible Cause

The name of a .bak backup file uploaded to a self-built OBS bucket is too long.

Solution

Based on the previous analysis, a solution is provided as follows:

- Step 1** Check the name of the backup file to be uploaded to an OBS bucket by referring to [Preparing Backup Files](#) in the *Backup Migration*.
- Step 2** Change the name of the backup file in the local database and upload the file to a self-built OBS bucket again.

----End

2.9.3 Backup Migration Failed Because the Database with the Same Name Already Exists

Scenarios

When you migrate full backup data to the cloud, the following error message is displayed: The restore database already exists in the destination DB instance

Possible Causes

To ensure data security, RDS for SQL Server does not support migrating databases with the same name to the cloud.

Solution

If you want to overwrite the data in the existing database, back up the existing data and delete the database with the same name. Alternatively, set **Overwrite Data** to **Yes** when creating a backup migration task, and then migrate the data again.

2.9.4 Backup Migration Failed Because an Incremental Backup File Is Used

Scenarios

When you migrate full backup data to the cloud, the following error message is displayed: In full mode, incremental file restoration is not supported. To restore incremental files, perform full restoration first.

Possible Causes

The selected backup file is an incremental backup file instead of a full backup file. Only full backup files can be migrated to the cloud at a time. Differential backup is not supported.

Solution

Incremental files cannot be used for full data restoration. To restore incremental files, perform full restoration first or use full backup files to migrate data.

2.9.5 Backup Migration Failed Because a Log Backup File Is Used

Scenarios

When you migrate full backup data to the cloud, the following error message is displayed: Target database has been restored,can not restore for transaction log

Possible Causes

The backup file selected during task creation is a log backup file instead of a full backup file. Only full backup files can be migrated to the cloud at a time. Log backup is not supported.

Solution

Log files cannot be used for full data restoration. Select a full backup file for data migration.

2.9.6 Backup Migration Failed Because the Backup File Verification Failed

Scenarios

When you create a backup migration task, the following error message is displayed: Failed to obtain the restoration file information

Possible Causes

The backup file is damaged or incomplete, and the backup file verification fails.

Solution

Select a complete full backup file and perform the migration again.

2.9.7 Backup Migration Failed Because of Insufficient Space

Scenarios

When you create a backup migration task, a message is displayed indicating that the space is insufficient. The following error information may be displayed:

1. The disk space of the target database is insufficient.
2. The disk space of the destination database must be 1.5 times larger than the size of the backup file.
3. The disk space of the destination database is insufficient. Check whether the backup is compressed.

Possible Causes

- The remaining space of the destination database must be greater than 1.5 times the size of the backup file.
- The backup file is compressed. As a result, the storage space of the destination database is insufficient.

Solution

Scale up the storage space by referring to [Scaling up Storage Space](#), or contact RDS customer service to change the destination database space and perform the migration again.

2.9.8 Backup Migration Failed Because Database Names Are Not Specified

Scenarios

If you choose to restore some databases, the following error message is displayed: If you choose to restore a partial database, specify the database name.

Possible Causes

If the restoration file contains multiple databases and you select some databases for restoration on the GUI, the names of the databases to be restored are not specified.

Solution

If you select some databases for restoration, specify the names of the databases to be restored and perform the migration again.

2.9.9 Backup Migration Failed Because a Full Backup File Is Used

Scenarios

When you migrate incremental backup data to the cloud, the following error message is displayed: Full files cannot be restored in incremental mode.

Possible Causes

During an incremental backup, after the full backup file is restored, only transaction log backup files can be used. If you select a full backup file again, this error is reported.

Solution

Full files cannot be used for incremental data restoration. Select a backup file and a backup mode based on site requirements.

2.9.10 Backup Migration Failed Because the LSNs of Incremental Backup Files Are Inconsecutive

Scenarios

When you migrate incremental backup data to the cloud, the following error message is displayed: In incremental restoration mode, the incremental .bak file is not continuous with the previous full restoration file.

Possible Causes

In a SQL Server database, the LSN of the differential backup or log backup must continuously follow the LSN of the backup file restored last time. Otherwise, this error is reported.

Solution

Select the corresponding LSN backup file for incremental backup based on the backup time sequence. Ensure that the LSN of the selected backup file can continuously follow the LSN of the last restoration file.

2.9.11 Backup Migration Failed Because the Number of Databases to Be Restored Exceeds the Destination Database Threshold

Scenarios

When you migrate backup data to the cloud, the following error message is displayed: The number of the recovery database exceeds the threshold of the target database.

Possible Causes

The number of databases to be restored exceeds the threshold of the destination database.

Solution

Select another RDS for SQL Server DB instance as the destination database, or delete unnecessary databases from the destination database and then perform the migration.

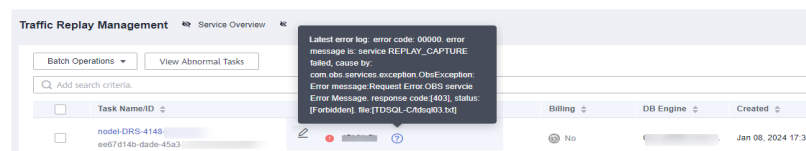
2.10 Workload Replay

2.10.1 Parsing Failed, and a Message Is Displayed Indicating That the OBS Connection Failed

Scenarios

When you create a to-the-cloud workload replay task and obtain workload files in the OBS bucket using an AK/SK, workload files failed to be parsed and a message is displayed indicating that the OBS connection failed.

Figure 2-3 Parsing failed



Possible Causes

The possible causes are as follows:

1. The AK, SK, bucket name, or endpoint is incorrect.
2. You do not have the permission to read files in the OBS bucket.

Solution

Based on the previous analysis, a solution is provided as follows:

Step 1 Click the task name. The **Basic Information** page is displayed.

Step 2 In the **Connection Information** area, check whether the AK, SK, bucket name, and endpoint information is correct.

If a **temporary AK/SK** is used, you also need to check the permissions and validity period of the temporary AK/SK and security token.

- If the connection information is correct, go to **Step 3**.
- If the connection information is incorrect, click **Pause** in the **Operation** column of the task. After the task is paused, click **Edit** in the **Operation** column. On the **Configure Source and Destination Databases** page, enter information about the source database and task settings, and start the task.

Step 3 Check whether you have the permission to read files in the OBS bucket. For details about how to grant users the permission to read files in OBS buckets, see **OBS Permissions Management**.

----End

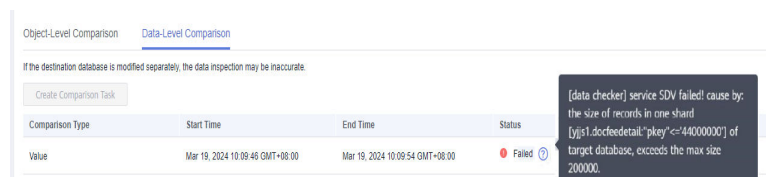
2.11 Data-Level Comparison

2.11.1 Data-Level Comparison Error: service SDV failed! cause by: the size of records in one shard[***] of target database, exceeds the max size 200000

Scenarios

During a value comparison task, an error is reported, and the log information is as follows: service SDV failed! cause by: the size of records in one shard[***.***] of target database, exceeds the max size 200000

Figure 2-4 Comparison failed



The screenshot shows a web interface for 'Data-Level Comparison'. It includes a 'Create Comparison Task' button and a table with columns for 'Comparison Type', 'Start Time', 'End Time', and 'Status'. A single row is visible with 'Value' as the comparison type, start and end times of 'Mar 19, 2024 10:09:46 GMT+08:00', and a 'Failed' status. A tooltip displays the error message: '[data checker] service SDV failed! cause by: the size of records in one shard [yjs1.docfsdetail:"pk"<=44000000] of target database, exceeds the max size 200000.'

Comparison Type	Start Time	End Time	Status
Value	Mar 19, 2024 10:09:46 GMT+08:00	Mar 19, 2024 10:09:54 GMT+08:00	Failed

Possible Causes

The data model of the table to be compared is special. As a result, automated sharding cannot be performed for the comparison task, and the task fails.

Solution

Create a value comparison task again and deselect the table for which the error is reported.

- Step 1** Click the task name. The **Basic Information** page is displayed.
- Step 2** Take a synchronization task as an example. Choose **Synchronization Comparison**.
- Step 3** Click the **Data-Level Comparison** tab and click **Create Comparison Task**.
- Step 4** Deselect the table where the error is reported and click **OK** to submit the comparison task again.
- Step 5** If the fault persists, choose [Service Tickets > Create Service Ticket](#) in the upper right corner of the management console and contact DRS customer service.

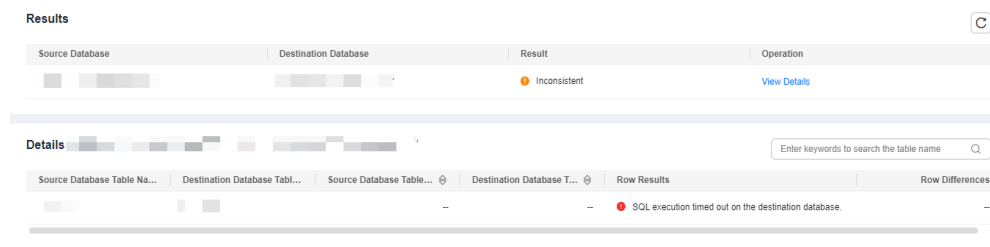
----End

2.11.2 Row Comparison Is Inconsistent, and SQL Execution in the Destination Database Times Out

Scenarios

Row comparison of data-level comparison is inconsistent, and a message is displayed, indicating that the SQL statement execution of the destination database times out.

Figure 2-5 Inconsistent comparison



Possible Cause

There is a lot of data to be compared in the table, and the row comparison times out. The default timeout interval is 1 hour.

Solution

1. Perform row comparison again.
2. Split the table to be compared and compare data in batches.
3. If the fault persists, choose [Service Tickets > Create Service Ticket](#) in the upper right corner of the management console and contact DRS customer service.