Relational Database Service

Troubleshooting

Issue	01
Date	2022-09-30

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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

Notice

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

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

Huawei Technologies Co., Ltd.

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

Website: https://www.huawei.com

Email: support@huawei.com

Contents

1 RDS for MySQL

1.1 Backup and Restoration Issues

1.1.1 RDS for MySQL Backup Job Failure

Scenario

When a user ran the mysqldump command to back up RDS for MySQL data to an ECS that is in a different subnet from RDS, the backup job failed after running for 300 seconds.

Possible Causes

Replace the ECS where the backup job is executed with an ECS that is in the same subnet as RDS. The backup job is successfully executed.

- Network: There are no differences on latency and bandwidth between the two ECSs.
- Database: The **net_write_timeout** parameter is set to **300** on the RDS for MySQL database. The connection between the ECS and RDS for MySQL is interrupted after 300s regardless of whether data writes have been completed.

```
| net_write_timeout             | 300      |
```

Procedure

Step 1 Understand the backup data flow, protocol, and port.

mysqldump uses TCP to connect to port 8635 of RDS. After the connection is established, the backup job starts.

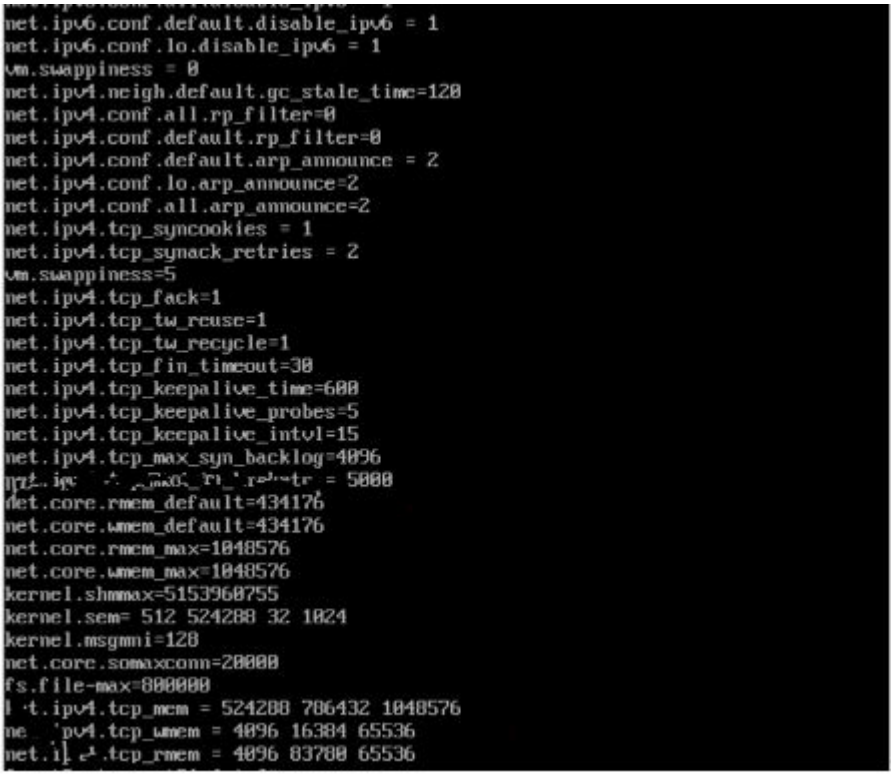
Step 2 Compare the hardware configuration and OS version of the two ECSs.

1. Both of them use the same hardware configuration: two cores and 6 GB of memory.

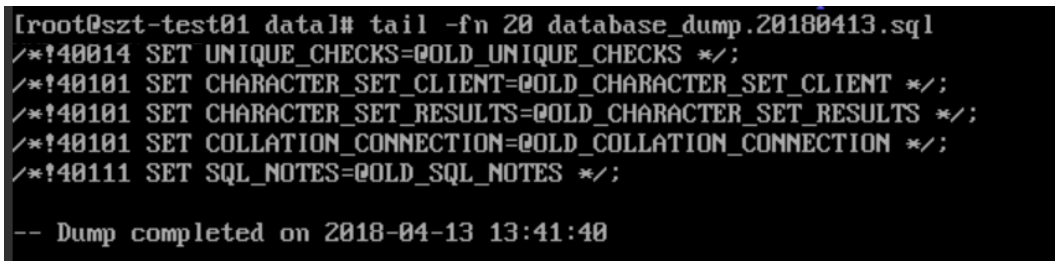
2. Both of them use the same OS version: CentOS 7.4.

Step 3 Check whether the NIC rates are the same.

Step 4 Check whether the kernel parameter settings are the same. The result shows that the network parameters on the ECS where the backup job failed are not optimized.



Step 5 Set the kernel parameters of the ECS where the backup job failed to the same as those of the ECS where the backup job succeeded. Start a backup job again. The backup job is successful.



----End

Solution

There is a large volume of data writes during the backup process across networks. The data write capability and TCP buffer on the backup end do not match the sending capability of the RDS. When the timeout period reaches the preset threshold (300s), the backup job failed. You can increase the TCP buffer by modifying the ECS kernel parameters to resolve this issue.

1.1.2 Automated Incremental Backup Failed Due to Full Storage

Scenario

Automated incremental backup failed.

Possible Causes

As your service data grows, the load on your DB instance is too heavy and the original storage space may be insufficient. When the DB instance status is **Storage full**, data cannot be written to databases, causing the failure of incremental backup.

Solution

Scale up storage space for your RDS instance. On the **Instance Management** page, locate the target DB instance and choose **More** > **Scale Storage Space** in the **Operation** column. Perform a full backup after the storage space is successfully scaled up and the incremental backup will be successful.

1.1.3 Precautions for Exporting Large Tables Through mysqldump

If the **–q** or **--quick** parameter is added when you use mysqldump to export data, the results of SELECT statements are not buffered in memory but directly exported. If this parameter is disabled, the results of SELECT statements are buffered in memory and then sent to the client.

- If you use mysqldump to back up only a small amount of data which can be stored in the idle memory buffer, disabling **-q** increases the export speed.
- Buffering a large amount of data may consume a large amount of memory, causing a memory swapping. If you use mysqldump to back up a large amount of data which cannot be stored in the memory buffer, enable **-q**. If **-q** is not enabled, a large amount of memory will be consumed and may even cause the database to break down due to out of memory.

Therefore, you are advised to enable the **-q** parameter when using mysqldump to back up data.

Example command:

mysqldump -uroot -p-P*8635* **-h***192.168.0.199* **--set-gtid-purged=OFF --single-transaction --flush-logs -q** *test t1*>t1.**sql**

1.1.4 Commands for Exporting Data Through mysqldump

Background

mysqldump is the most commonly used tool for importing and exporting MySQL data.

mysqldump Options

Table 1-1 Option description

Option Name	Description
add-drop-table	Adds the DROP TABLE statement before each data table is created.
events, E	Exports events.
routines, R	Exports stored procedures and customized functions.
flush-logs	Updates logs before the logs are exported.
no-create-db, n	Exports only data without adding of the CREATE DATABASE statement.
add-drop-database	Adds the DROP DATABASE statement before each database is created.
no-create-info, t	Exports only data without adding of the CREATE TABLE statement.
no-data, d	Exports only the database table structure.
set-gtid-purged=OFF	Does not export GTID statements.
hex-blob	Exports binary string fields in hexadecimal format.

Examples for Using mysqldump

1. Export all data of databases **db1** and **db2**.

   >db12.sql*db1 db2***--hex-blob --set-gtid-purged=OFF --single-transaction --order-by-primary --flush-logs -q --databases** *192.168.0.199* **-h***8635***mysqldump -uroot -p -P**

2. Export the **t1** and **t2** tables of database **db1**.

   .sql*t1_t2* **>***t1 t2* **--tables** *db1* **--hex-blob --set-gtid-purged=OFF --single-transaction --order-by-primary --flush-logs -q --databases** *192.168.0.199* **-h***8635***mysqldump -uroot -p -P**

3. Export data whose id equals **1** from table **t1** in database **db1**.

   .sql*t1_id***'>***id=1* **--where='***t1* **--tables** *db1* **--hex-blob --set-gtid-purged=OFF --single-transaction --order-by-primary --flush-logs -q --databases** *192.168.0.199* **-h***8635***mysqldump -uroot -p -P**

4. Export all table structures in database **db1** without exporting data.

   _table.sql*db1* >db1**set-gtid-purged=OFF --single-transaction --order-by-primary -n --flush-logs -q --databases --no-data --***192.168.0.199* **-h***8635***mysqldump -uroot -p -P**

5. Export all data excluding the tables and data in database **db1**.

   > others.sql *db1* **--set-gtid-purged=OFF -F -n -t -d -E -R***8635* **-P***192.168.0.199***mysqldump -uroot -p -h**

1.1.5 SQL Statements Such as SET @@SESSION.SQL_LOG_BIN Displayed After You Run mysqldump

Scenario

When you run mysqldump on a database, the following code is displayed:

Figure 1-1 Code

```
backup_procedure.sql
1  -- MySQL dump 10.13  Distrib 5.7.24, for Linux (x86_64)
2  --
3  -- Host: 192.168.1.64    Database: rptdb
4  -- ------------------------------------------------------
5  -- Server version   5.7.31-2-log
6
7  /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
8  /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
9  /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
10 /*!40101 SET NAMES utf8 */;
11 /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
12 /*!40103 SET TIME_ZONE='+00:00' */;
13 /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
14 /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
15 /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
16 /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
17 SET @MYSQLDUMP_TEMP_LOG_BIN = @@SESSION.SQL_LOG_BIN;
18 SET @@SESSION.SQL_LOG_BIN= 0;
19
20 --
21 -- GTID state at the beginning of the backup
22 --
23
24 SET @@GLOBAL.GTID_PURGED='7f47edf7-2e4d-11eb-9a43-fa163eacf6f0:1-36975382';
25
26 --
27 -- Dumping routines for database 'rptdb'
28 --
29 /*!50003 DROP FUNCTION IF EXISTS `f_sys_get_partition` */;
30 /*!50003 SET @saved_cs_client      = @@character_set_client */ ;
31 /*!50003 SET @saved_cs_results     = @@character_set_results */ ;
32 /*!50003 SET @saved_col_connection = @@collation_connection */ ;
33 /*!50003 SET character_set_client  = utf8 */ ;
34 /*!50003 SET character_set_results = utf8 */ ;
35 /*!50003 SET collation_connection  = utf8_general_ci */ ;
36 /*!50003 SET @saved_sql_mode       = @@sql_mode */ ;
```

Fault Analysis

The parameter **gtid-mode** is set to **ON**.

If GTID is enabled for a database, you can use mysqldump to back up or dump all Global Transaction Identifiers (GTIDs) in the database or even to back up the whole RDS for MySQL database.

Solution

When the primary and standby RDS for MySQL databases are exported for backup and restoration, check whether GTID is enabled.

If GTID is enabled, add **–set-gtid-purged=OFF** to the **mysqldump** command during data dump.

1.2 Primary/Standby Replication Issues

1.2.1 Abnormal Replication Between Primary and Standby RDS DB Instances

Scenario

The replication relationship between primary and standby RDS DB instances is abnormal. A possible cause is that the default security group rule is deleted.

Solution

Step 1 Log in to the management console.

Step 2 On the **Instances** page, click the target DB instance.

Step 3 On the **Basic Information** page, click the security group name.

Step 4 On the **Inbound Rules** tab, click **Add Rule**. In the displayed dialog box, select **All** for **Protocol & Port** and **Security group** for **Source**. In the drop-down list, select the same security group as the displayed one on the **Basic Information** page.

Step 5 Wait until the rule is added. Check that the replication relationship between primary and standby DB instances is restored.

----End

1.3 Parameter-related Issues

1.3.1 [ERROR] 1071 Reported When an Index Fails to Be Created for RDS for MySQL

Scenario

The index failed to be created because its length exceeds the upper limit. The following error was reported:

[ERROR] 1071 - Specified key was too long; max key length is 3072 bytes

This problem may occur in MySQL-8.0.20.5.

Fault Analysis

The InnoDB table engine has a length limit on index prefixes.

By default, an index prefix can contain a maximum of 767 bytes, but if **innodb_large_prefix** is set to **ON**, the index prefix length is increased to 3,072 bytes.

```
SHOW VARIABLES LIKE '%innodb_large_prefix%';
```



```
mysql> show variables like '%innodb_large_prefix%';
+---------------------+-------+
| Variable_name       | Value |
+---------------------+-------+
| innodb_large_prefix | ON    |
+---------------------+-------+
1 row in set (0.02 sec)
```

The length of an index prefix also depends on the InnoDB page size. If the parameter **innodb_page_size** is set to its default value 16 KB, the maximum index prefix length is 3,072 bytes, but if this parameter is set to 8 KB, the maximum index prefix length is 1,536 bytes. If this parameter is set to 4 KB, the maximum length of the index prefix is 768 bytes.

```
SHOW VARIABLES LIKE '%innodb_page_size%';
```

```
mysql> show variables like 'innodb_page_size';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| innodb_page_size | 16384 |
+------------------+-------+
1 row in set (0.01 sec)
```

Check the structure of the problem table and query all supported character sets and their byte usage:

```
SHOW CHARACTER SET;
```

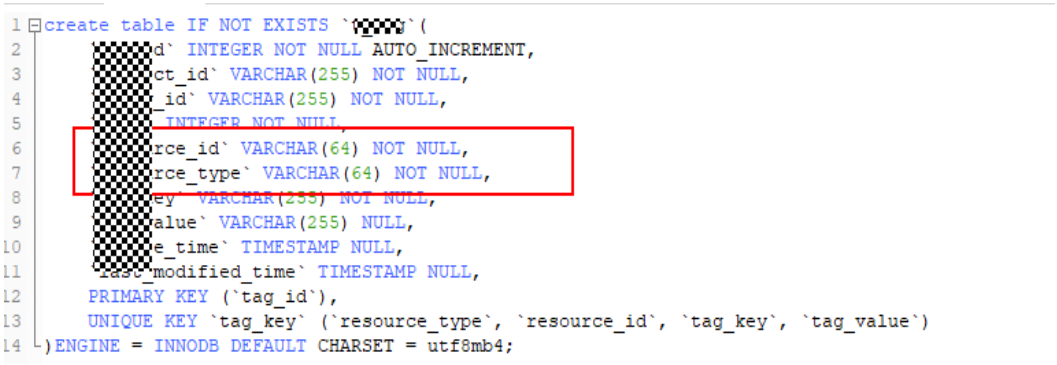


```
mysql> SHOW CHARACTER SET;
+----------+---------------------------------+---------------------+--------+
| Charset  | Description                     | Default collation   | Maxlen |
+----------+---------------------------------+---------------------+--------+
| armscii8 | ARMSCII-8 Armenian              | armscii8_general_ci |      1 |
| ascii    | US ASCII                        | ascii_general_ci    |      1 |
| big5     | Big5 Traditional Chinese        | big5_chinese_ci     |      2 |
| binary   | Binary pseudo charset           | binary              |      1 |
| cp1250   | Windows Central European        | cp1250_general_ci   |      1 |
| cp1251   | Windows Cyrillic                | cp1251_general_ci   |      1 |
| cp1256   | Windows Arabic                  | cp1256_general_ci   |      1 |
| cp1257   | Windows Baltic                  | cp1257_general_ci   |      1 |
| cp850    | DOS West European               | cp850_general_ci    |      1 |
| cp852    | DOS Central European            | cp852_general_ci    |      1 |
| cp866    | DOS Russian                     | cp866_general_ci    |      1 |
| cp932    | SJIS for Windows Japanese       | cp932_japanese_ci   |      2 |
| dec8     | DEC West European               | dec8_swedish_ci     |      1 |
| eucjpms  | UJIS for Windows Japanese       | eucjpms_japanese_ci |      3 |
| euckr    | EUC-KR Korean                   | euckr_korean_ci     |      2 |
| gb18030  | China National Standard GB18030 | gb18030_chinese_ci  |      4 |
| gb2312   | GB2312 Simplified Chinese       | gb2312_chinese_ci   |      2 |
| gbk      | GBK Simplified Chinese          | gbk_chinese_ci      |      2 |
| geostd8  | GEOSTD8 Georgian                | geostd8_general_ci  |      1 |
| greek    | ISO 8859-7 Greek                | greek_general_ci    |      1 |
| hebrew   | ISO 8859-8 Hebrew               | hebrew_general_ci   |      1 |
| hp8      | HP West European                | hp8_english_ci      |      1 |
| keybcs2  | DOS Kamenicky Czech-Slovak      | keybcs2_general_ci  |      1 |
| koi8r    | KOI8-R Relcom Russian           | koi8r_general_ci    |      1 |
| koi8u    | KOI8-U Ukrainian                | koi8u_general_ci    |      1 |
| latin1   | cp1252 West European            | latin1_swedish_ci   |      1 |
| latin2   | ISO 8859-2 Central European     | latin2_general_ci   |      1 |
| latin5   | ISO 8859-9 Turkish              | latin5_turkish_ci   |      1 |
| latin7   | ISO 8859-13 Baltic              | latin7_general_ci   |      1 |
| macce    | Mac Central European            | macce_general_ci    |      1 |
| macroman | Mac West European               | macroman_general_ci |      1 |
| sjis     | Shift-JIS Japanese              | sjis_japanese_ci    |      2 |
| swe7     | 7bit Swedish                    | swe7_swedish_ci     |      1 |
| tis620   | TIS620 Thai                     | tis620_thai_ci      |      1 |
| ucs2     | UCS-2 Unicode                   | ucs2_general_ci     |      2 |
| ujis     | EUC-JP Japanese                 | ujis_japanese_ci    |      3 |
| utf16    | UTF-16 Unicode                  | utf16_general_ci    |      4 |
| utf16le  | UTF-16LE Unicode                | utf16le_general_ci  |      4 |
| utf32    | UTF-32 Unicode                  | utf32_general_ci    |      4 |
| utf8     | UTF-8 Unicode                   | utf8_general_ci     |      3 |
| utf8mb4  | UTF-8 Unicode                   | utf8mb4_0900_ai_ci  |      4 |
+----------+---------------------------------+---------------------+--------+
```

When the problem table uses the utf8mb4 character set, each character occupies 4 bytes. This means that the index prefix only contains 768 (3072/4 = 768) characters if the index prefix has 3072 bytes. Therefore, you only need to set the index prefix length in the CREATE TABLE statement to **768** or modify the index field to keep it less than 3072 bytes.

Solutions

Change the length of the index field.



```
create table IF NOT EXISTS `[redacted]`(
    [redacted]d` INTEGER NOT NULL AUTO_INCREMENT,
    [redacted]ct_id` VARCHAR(255) NOT NULL,
    [redacted]_id` VARCHAR(255) NOT NULL,
    [redacted]_INTEGER NOT NULL,
    [redacted]rce_id` VARCHAR(64) NOT NULL,
    [redacted]rce_type` VARCHAR(64) NOT NULL,
    [redacted]ey` VARCHAR(255) NOT NULL,
    [redacted]alue` VARCHAR(255) NULL,
    [redacted]e_time` TIMESTAMP NULL,
    [redacted]modified_time` TIMESTAMP NULL,
    PRIMARY KEY (`tag_id`),
    UNIQUE KEY `tag_key` (`resource_type`, `resource_id`, `tag_key`, `tag_value`)
)ENGINE = INNODB DEFAULT CHARSET = utf8mb4;
```

1.3.2 Tables Failed to Be Found After Case-Sensitivity Setting Changes for RDS for MySQL

Scenario

After the RDS for MySQL parameter **lower_case_table_names** was set to case sensitive, a table containing uppercase letters was created. However, after the parameter was set to case insensitive, the table containing uppercase letters, such as tbl_newsTalking, cannot be found.

Case: When a backup is restored to a new instance, restoration will fail if the parameter case-sensitive setting of the new instance is different from that of the original instance.

Solution

Step 1 Change the value of **lower_case_table_names** to **0**, indicating that table names are case sensitive.

Step 2 Reboot the database.

Step 3 Change uppercase letters in table names to corresponding lowercase letters.

Step 4 Change the value of **lower_case_table_names** to **1**, indicating that table names are case insensitive.

Step 5 Reboot the database, or export the entire database and import it again.

----End

NOTE

- The names of databases, tables, and variables, and table alias must be case sensitive.
- Column names and aliases are always case insensitive. Field values are case insensitive by default.
- For RDS for MySQL 5.7, you can specify case sensitivity for table names when creating an instance on the console or using APIs, or set the **lower_case_table_names** parameter after an instance is created.
- For RDS for MySQL 8.0, you can only specify case sensitivity for table names when creating an instance on the console or using APIs.

1.4 Performance Issues

1.4.1 Why Is My SQL Query Running So Slow?

1. You can view the slow SQL logs to check whether any slowly executed SQL queries exist and view their performance characteristics (if any) to locate the cause.
2. View the CPU usage metrics of RDS DB instance to facilitate troubleshooting.
3. Create read replicas to offload read pressure on primary DB instances.
4. When multiple associated tables are queried, indexes must be created for the associated fields.
5. Do not use the SELECT statement to scan all tables. You can specify fields or add the WHERE condition.

1.4.2 Native Error 1461 Reported by an RDS for MySQL DB Instance

Scenario

The following error information is displayed when there are large amounts of concurrent read and write requests, large amounts of SQL statements, or in data migration scenarios:

mysql_stmt_prepare failed! error(1461)Can't create more than max_prepared_stmt_count statements (current value: 16382)

Fault Analysis

The **max_prepared_stmt_count** value ranges from 0 to 1048576. The default value is **16382**. This parameter limits the total number of prepared statements in all sessions on mysqld. The current value exceeds the value range of this parameter.

Solution

Set **max_prepared_stmt_count** to a larger value. The recommended value is **65535**.

1.4.3 System Inaccessible After Field Addition to an RDS for MySQL Database Table

Description

After a field was added to an RDS for MySQL database table, the system becomes inaccessible.

Solution

The database performance is affected due to the addition of table fields. A possible reason is that indexes are not added to the new table fields. As a result, a large amount of data consumes a large number of CPU resources. You are advised to:

- Add indexes and primary keys.
- Optimize slow SQL statements.

1.5 SQL Issues

1.5.1 ERROR[1451] Reported When a Table with Foreign Keys Cannot Be Deleted

Scenario

The **root** user does not have the permissions required to delete or modify tables in the database. Error message is as follows:

ERROR[1451] -Cannot deleteorupdatea parent row:

aforeignkeyconstraintfails (...)

Fault Analysis

The FRM file also exists in sys_tables. This table has foreign key relationships with other tables and cannot be deleted directly.

The foreign key association has been set in the RDS for MySQL DB instance. As a result, data cannot be updated or deleted. You can set **foreign_key_checks** to avoid this problem.

Solution

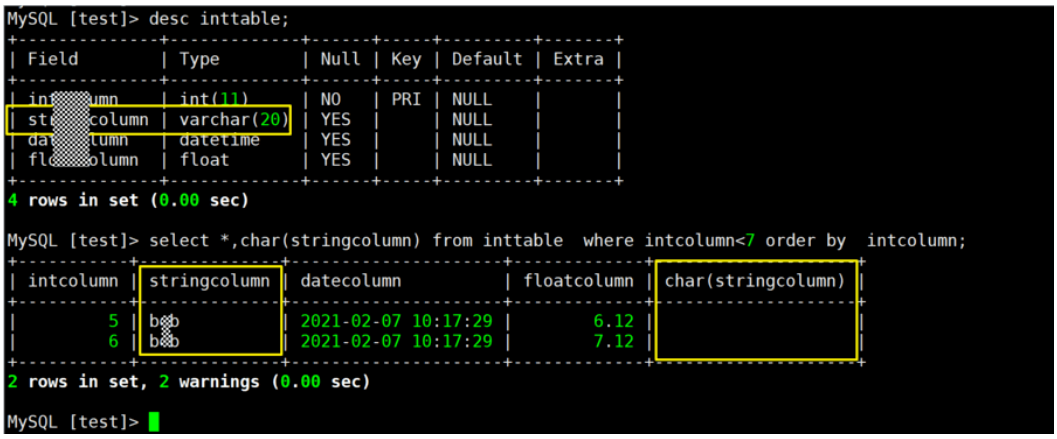
```
set session foreign_key_checks=off;
drop table table_name;
```

To delete the table, set **foreign_key_checks** to **off**.

1.5.2 Solution to the Failure of Converting the Field Type

Scenario

The **varchar** field is read using the char data type, and cannot be converted through the char() function.



Fault Analysis

The char() function cannot be used to convert data types.

Solution

The CAST() function and CONVERT() function of RDS for MySQL can be used to obtain values of one type and generate values of another type. The syntax is as follows:

```
CAST(value as type);
CONVERT(value, type);
```

That is, CAST(xxx AS type) and CONVERT(xxx, type).

NOTE

The data types that can be converted are limited. The supported data types are as follows:

- BINARY: the same as adding a binary prefix to a data string
- CHAR(): characters. You can specify the length of the characters.
- DATE: calendar date
- Time: time of day
- DATETIME: date and time
- DECIMAL: floating point number
- SIGNED: integer
- UNSIGNED: unsigned integer

1.5.3 "Row size too large" Reported When an RDS for MySQL Table Failed to Be Created

Scenario

An RDS for MySQL table failed to be created and the following information is displayed:

Row size too large. The maximum row size for the used table type, not counting BLOBs, is 65535. This includes storage overhead, check the manual. You have to change some columns to TEXT or BLOBs

Fault Analysis

The total length of the **varchar** fields exceeds 65535, resulting in a table creation failure.

Solution

1. Reduce the length.
   ```
   CREATE TABLE t1 (a VARCHAR(10000),b VARCHAR(10000),c VARCHAR(10000),d VARCHAR(10000),e VARCHAR(10000),f VARCHAR(10000) ) ENGINE=MyISAM  CHARACTER SET latin1;
   ```
2. Change a column to **TEXT** by referring to the **official document**.

1.5.4 ERROR [1412] Reported by an RDS for MySQL DB Instance

Scenario

The following error information is displayed:

ERROR[1412]:Table definition has changed, please retry transaction

This problem may occur in MySQL-5.7.31.2.

Fault Analysis

Cause 1: A transaction is started using START TRANSACTION WITH CONSISTENT SNAPSHOT.

Scenario 1

```
mysql> start transaction with consistent snapshot;
Query OK, 0 rows affected (0.00 sec)
```

Scenario 2

```
mysql> alter table t_sec_user add test int;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Scenario 3

```
ysql> select count(*) from t_sec_user;
ERROR 1412 (HY000): Table definition has changed, please retry transaction
ysql>
```

Cause 2: DDL operations are performed for binlog files.

```
# at 13793936
#210325 18:44:06 server id 1605802263  end_log_pos 13794065 CRC32 0x4df2db06    Query   thread_id=19314688      exec_time=1     error_code=0
use `zzkj`/*!*/;
SET TIMESTAMP=1616669046/*!*/;
SET @@session.sql_mode=1346371584/*!*/;
/*!\C utf8mb4 *//*!*/;
SET @@session.character_set_client=45,@@session.collation_connection=45,@@session.collation_server=33/*!*/;
ALTER TABLE bas_cabinet_box  ADD lock_status tinyint(4)
/*!*/;
```


Solution

If the error is caused by any of the preceding causes, the fault needs to be rectified on the service side.

1.6 Connection Issues

1.6.1 Login Failed After the authentication_string Field Is Changed to Display the Password for RDS for MySQL

Scenario

After you use Navicat to modify the field **authentication_string** of the root account in the user table to display the password, the user cannot log in to the client.

This problem may occur in MySQL-8.0.20.6.

Possible Causes

The password changing method is incorrect. The hash key of the **authentication_string** field in the user table is directly changed. The password of the **root** user should be reset on the console.

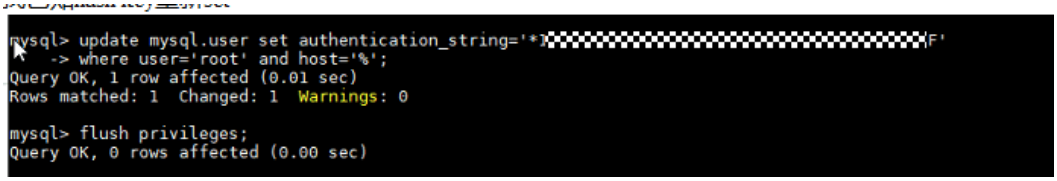
Solution

MySQL 8.0 does not support the password function. Therefore, you should perform the following steps to rectify the fault:

Step 1 Locate the **authentication_string** field of account **rdsAdmin** and run the following command:

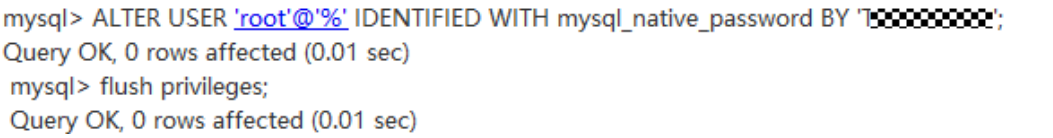
```
update mysql.user set authentication_string='XXX'
```

XXX indicates the new password.



Step 2 Reset the password of account **root**.

```
ALTER USER 'root'@'%' IDENTIFIED WITH mysql_native_password BY 'XXX';
flush privileges;
```



After the password is changed, the user can log in to the system as user **root**.

----End

1.6.2 MySQL-server Connection Failure After a Version Upgrade of RDS for MySQL

Scenario

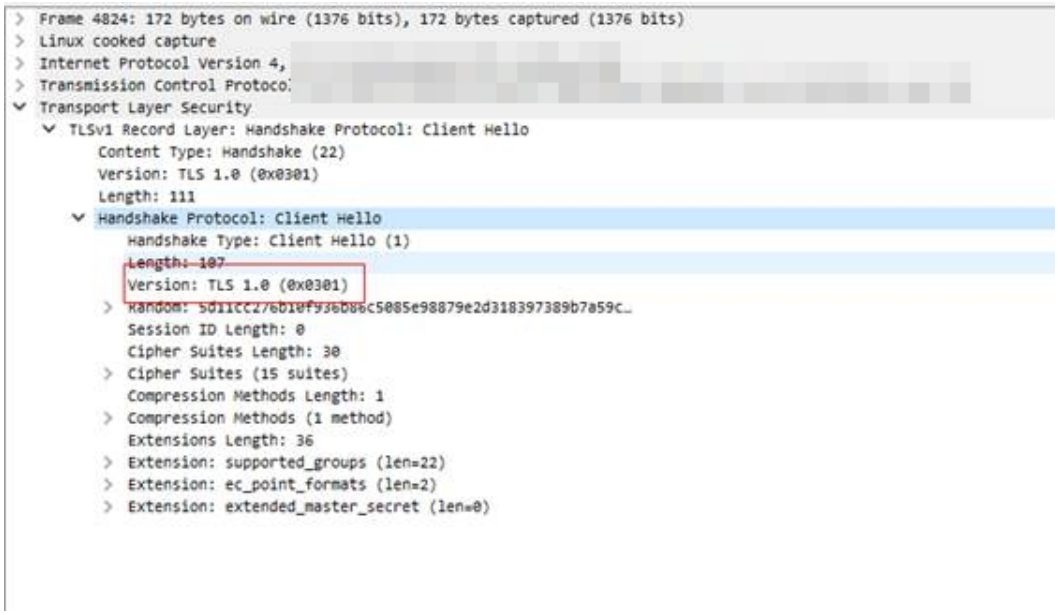
The following error message is displayed when a database is connected through commands:

Caused by: javax.net.ssl.SSLException: Received fatal alert: protocol_version

MySQL-server connection failed after RDS for MySQL 5.7.23 is upgraded to 5.7.25. **Figure 1-2** shows the captured packet.

The TLS version sent from the client to the server during the TLS handshake is 1.0. A total number of 15 supported cipher suites are provided.

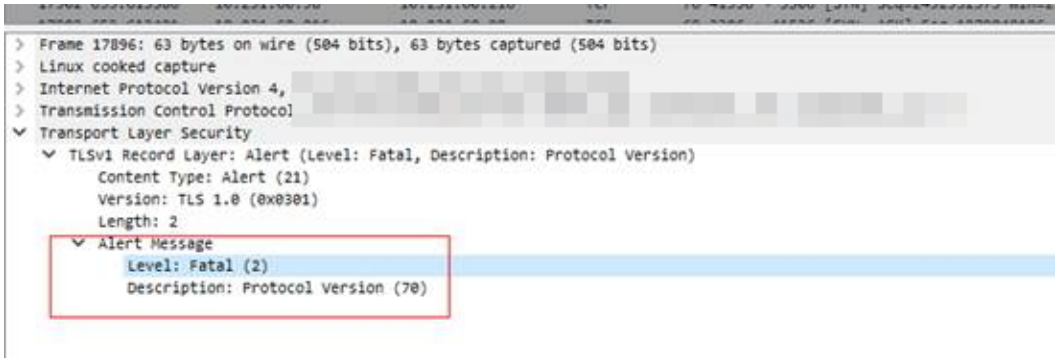
Figure 1-2 Packets captured when connection failed



Fault Analysis

As shown in the MySQL-server response in **Figure 1-3**, the server rejects the client connection because OpenSSL has been upgraded to 1.1.1a on MySQL 5.7.25, resulting in the rejection of the insecure TLS version and password suite.

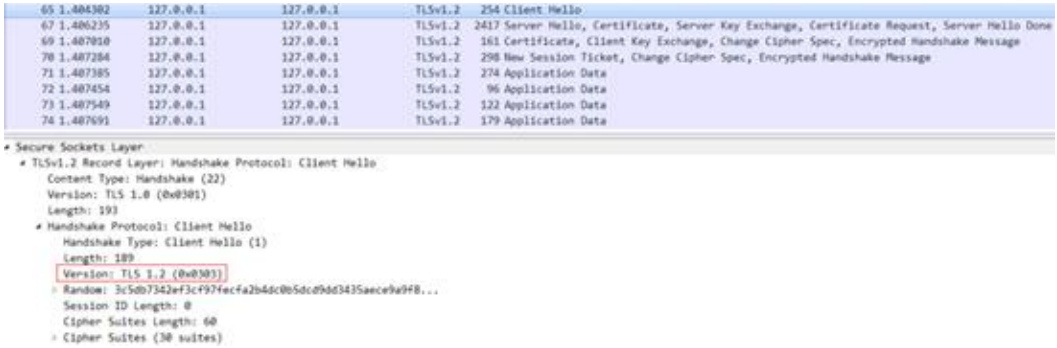
Figure 1-3 MySQL-server response



Solution

Upgrade your JDK client to **JDK 8 or a later version**. By default, TLS 1.2 is supported and 30 cipher suites are provided. **Figure 1-4** shows a normal captured packet.

Figure 1-4 Packets captured when connection is normal



A Change History

Released On	Description
2022-09-30	This issue is the first official release.

