RDS for PostgreSQL

Troubleshooting

Issue 01

Date 2025-06-24

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Contents

1 A Large Number of Schemas Whose Owner Is rdsadmin

Scenario

There are a large number of schemas whose owner is **rdsadmin** in an RDS for PostgreSQL instance.

Possible Causes

Temporary tables in an RDS for PostgreSQL instance are classified into session-level temporary tables and transaction-level temporary tables.

- In a session-level temporary table, data exists throughout the lifecycle of a session. The default temporary table is on a per-session basis.
- In a transaction-level temporary table, data exists only in the lifecycle of a transaction.

RDS for PostgreSQL temporary tables are special tables generated in a schema named **pg_temp_***n*, where *n* indicates a number that varies by session.

In this scenario, a large number of temporary tables are used by user services. These temporary tables cannot be deleted. If they are deleted, they will be re-created soon.

2 Index Name Containing a Schema Name During Index Creation

Scenario

In the official PostgreSQL syntax, an index name cannot contain a schema name. For example, CREATE UNIQUE INDEX fee_code_desc_uni_idx is a correct index name. If a schema name is added to it, like CREATE UNIQUE INDEX "isp-1".fee_code_desc_uni_idx, a syntax parsing error is reported.

However, when you create an index on an RDS for PostgreSQL 11 instance, the index name can contain a schema name. But RDS for PostgreSQL 12 does not support schema names in index names.

Possible Causes

In RDS for PostgreSQL 11, the syntax parsing of **create index** is modified. Schema names in an index name can be parsed.

But such modification is not made to RDS for PostgreSQL 12.

3 Authentication Not Supported When a DB Instance Is Accessed Through an Application

Scenario 1

- Description

  If an RDS for PostgreSQL instance is connected through a PostgreSQL application and the client does not support the scram-sha-256 authentication method, the following error message is displayed:

  ```
  Authentication method not supported (Received: 10)
  ```
- Possible causes

  The client version is too old and incompatible with the encryption algorithm used by the DB instance.
- Solution

  a. Check the client or client driver (such as the JDBC driver) and update it to the latest version to ensure that the latest authentication method is supported.

     If the latest authentication is still not supported, go to the next step.

  b. Change the value of **password_encryption** to **md5**.

     NOTICE

     The modification of **password_encryption** takes effect only after the password is reset.

  c. If the fault persists, **modify pg_hba.conf**. Check whether the authentication method is set to **scram-sha-256**. If yes, change it to **md5** and try again.

Scenario 2

- Description

After data is migrated from an instance of an earlier version to an instance of a later version, the new instance cannot be connected and the following error message is displayed:

```
unsupported authentication method requested by the server: 10
```

- Possible causes

  The value of **password_encryption** is **md5** on the original instance. After data is migrated to the new instance, the default value of **password_encryption** becomes **scram-sha-256**. This value is changed to **md5**, but the password is not reset after the change, so the password authentication fails.

- Solution

  Reset the password after the value of **password_encryption** is changed.

Scenario 3

- Description

  When an RDS for PostgreSQL instance is connected using a JDBC application, the following error message is displayed in the console error logs because the JDBC version does not support the scram-sha-256 authentication:

  ```
  unsupported frontend protocol 1234.5680: server supports 2.0 to 3.0
  ```

- Possible causes

  The JDBC version is too early.

- Solution

  Download and use a JDBC of the latest version.

4 Error Reported When a Request Is Executed Through an Existing Connection

Scenario

When a client executes a request through an existing connection, the following error information is displayed:

- Error 1
```
org.postgresql.util.PSQLException: An I/O error occurred while sending to the backend.
```
- Error 2
```
org.postgresql.util.PSQLException: The connection attempt failed
...
Caused by: java.net.SocketException: Connection reset
```

Solution to Error 1

Possible causes:

- There are too many parameters in the SQL statement.
- The connection has been released.

Solution:

- If there are too many parameters in the SQL statement, split the SQL statement.
- If the connection has been released, check the client connection parameters, such as the connection timeout interval. Add an automatic retry mechanism for your client.

Solution to Error 2

This error is reported when a connection that has been released is used. The possible causes are as follows:

- There are problems with network connectivity.
- The instance is being rebooted or the backend process crashes.
- Idle connections are released upon timeout.

Solution:

1. Check the network connectivity and determine whether the disconnection is caused by network link factors (such as high packet loss rate and retransmission rate).
2. If there are no network problems, check for other errors.
3. If no, check the connection timeout parameters (such as **sockettimeout** and **connecttimeout** for the JDBC connection pool). If the values are too small, connections can be released automatically.

5 Slow Instance Reboot Due to Too Many Inodes

Excessive inodes are often caused by stacked temporary files or excessive table objects. If there are too many inodes, the instance can reboot slowly.

Scenario 1

- Description

  When a large number of complex SQL statements are executed on an RDS for PostgreSQL instance, temporary files are stacked and an out of memory (OOM) exception occurs. In this case, the instance reboots slowly and services are unavailable for a long time.
- Possible causes

  If the SQL statements involve operations such as sorting, hash joins, and aggregation and the memory usage exceeds the value of **work_mem**, temporary files are generated. If a large number of such SQL statements are executed, an OOM exception occurs and the OS kills the database process. The kernel does not clear the temporary files, causing stacked temporary files. Too many temporary files slow down the database startup. That is because all temporary files need to be deleted before the database process can start.
- Solution

  Optimize the SQL statements or increase the value of **work_mem** to reduce the number of temporary files generated.

Scenario 2

- Description

  There are a large number of tables in an RDS for PostgreSQL instance. At a certain time, connections to the instance and workloads increase sharply. The database process memory is used up and an OOM issue occurs. Then the instance reboots slowly, and services are unavailable for a long time.
- Possible causes

  During the reboot, the kernel process traverses all tables and flushes data in the OS cache to disks (fsync). If there are too many table objects, the traversal takes a long time and slows down the reboot.

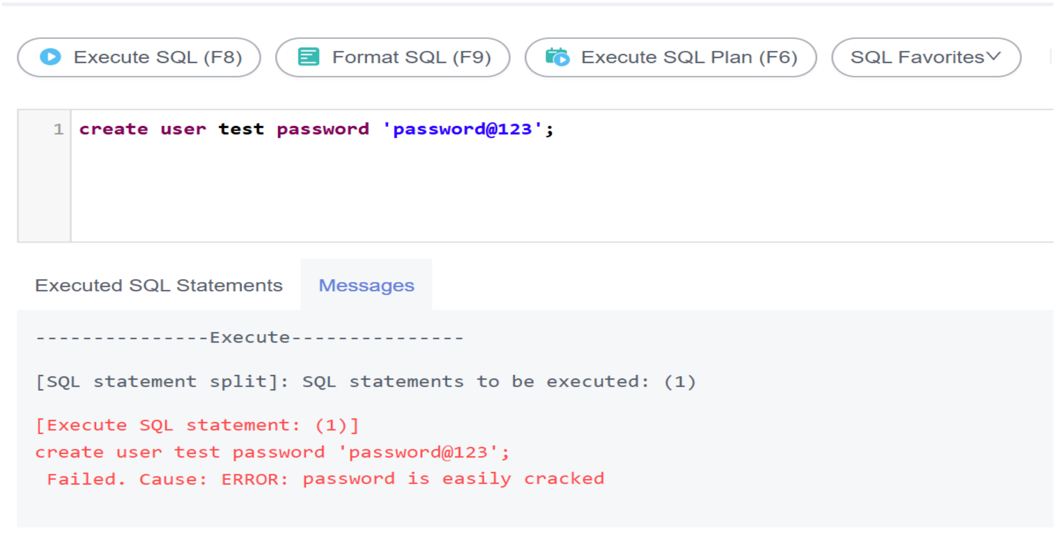
- Solution
  - Ensure that there are no more than 20,000 tables in a single instance and no more than 4,000 tables in a single database. For details, see **Instance Usage Suggestions**.
  - Configure memory monitoring on the application side. If the memory is about to run out, increase the memory to avoid OOM issues. Pay attention to the value of metric **Inodes** and control the number of objects.

6 "password is easily cracked" Displayed When an RDS for PostgreSQL User Is Created

Scenario

The error message "password is easily cracked" is displayed when a database user is created for an RDS for PostgreSQL instance.

Figure 6-1 Error message



```
create user test password 'password@123';
```

```
---------------Execute---------------
[SQL statement split]: SQL statements to be executed: (1)
[Execute SQL statement: (1)]
create user test password 'password@123';
 Failed. Cause: ERROR: password is easily cracked
```

Possible Causes

This is because a weak password is intercepted. In RDS for PostgreSQL 11.22, 12.22, 13.18, 14.15, 15.10, 16.6, and later versions, weak password detection is enabled by default. If a weak password is used to create a user, an error will be reported.

Solutions

- Use a complex password to prevent brute-force attacks.
- Disable the detection of weak passwords. To disable it, set **passwordcheck.rds_enable_cracklib** to **off** so that the system only verifies a

password according to the basic requirements. For details, see **Modifying Parameters of an RDS for PostgreSQL Instance**. Then the password is checked only according to the basic rules.

The basic password requirements are as follows:

- The password must contain at least eight characters.
- The password must consist of letters and other characters.
- The password cannot contain the username.