

# Relational Database Service

## Best Practices

**Issue** 01  
**Date** 2024-07-15



**Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2024. 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 Cloud Computing Technologies Co., Ltd.

## **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are the property 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 Cloud 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.

---

# Contents

---

|   |          |
|---|----------|
| <b>1 RDS for MySQL</b>  | <b>1</b> |
| 1.1 Description of innodb_flush_log_at_trx_commit and sync_binlog                           | 1        |
| 1.2 How Do I Use the utf8mb4 Character Set to Store Emojis in an RDS for MySQL DB Instance? | 3        |
| <b>2 RDS for PostgreSQL</b>   | <b>5</b> |
| 2.1 Performing Basic Operations Using pgAdmin   | 5        |
| 2.1.1 Connecting to an RDS for PostgreSQL Instance  | 5        |
| 2.1.2 Creating a Database   | 5        |
| 2.1.3 Creating a Table  | 6        |
| 2.1.4 Using the Query Tool to Run SQL Statements  | 8        |
| 2.1.5 Monitoring Databases  | 10       |
| 2.1.6 Backing Up and Restoring Data   | 11       |
| 2.2 Viewing Slow Query Logs of RDS for PostgreSQL DB Instances                              | 13       |
| 2.3 Security Best Practices   | 14       |

# 1 RDS for MySQL

## 1.1 Description of `innodb_flush_log_at_trx_commit` and `sync_binlog`

The `innodb_flush_log_at_trx_commit` and `sync_binlog` are key parameters for controlling the disk write policy and data security of RDS for MySQL. Different parameter values have different impacts on performance and security.

**Table 1-1** Parameter description

| Parameter                                   | Allowed Values        | Description  |
|---|-----------------------|--|
| <code>innodb_flush_log_at_trx_commit</code> | 0, 1, and 2           | Controls the balance between strict ACID compliance for commit operations, and higher performance that is possible when commit-related I/O operations are rearranged and done in batches. The default value is <b>1</b> . For details, see <a href="#">Parameter Description</a> . |
| <code>sync_binlog</code>                    | 0 to 4, 294, 967, 295 | Sync binlog (RDS for MySQL flushes binary logs to disks or relies on the OS).  |

### Parameter Description

- **`innodb_flush_log_at_trx_commit`:**
  - **0:** The log buffer is written out to the log file once per second and the flush to disk operation is performed on the log file, but nothing is done at a transaction commit.
  - **1:** The log buffer is written out to the log file at each transaction commit and the flush to disk operation is performed on the log file.

- **2**: The log buffer is written out to the file at each commit, but the flush to disk operation is not performed on it. However, the flushing on the log file takes place once per second.

 **NOTE**

- A value of **0** is the fastest choice but less secure. Any mysqld process crash can erase the last second of transactions.
  - A value of **1** is the safest choice because in the event of a crash you lose at most one statement or transaction from the binary log. However, it is also the slowest choice.
  - A value of **2** is faster and more secure than **0**. Only an operating system crash or a power outage can erase the last second of transactions.
- **sync\_binlog=1 or N**  
By default, the binary log is not every time synchronized to disk. In the event of a crash, the last statement in the binary log may get lost.  
To prevent this issue, you can use the **sync\_binlog** global variable (**1** is the safest value, but also the slowest) to synchronize the binary log to disk after N binary log commit groups.

## Recommended Configurations

**Table 1-2** Recommended configurations

| <b>innodb_flush_log_at_trx_commit</b> | <b>sync_binlog</b> | <b>Description</b>  |
|---------------------------------------|--------------------|---|
| 1                                     | 1                  | High data security and strong disk write capability   |
| 1                                     | 0                  | High data security and insufficient disk write capability. Standby lagging behind or no replication is allowed. |
| 2                                     | 0/N (0 < N < 100)  | Low data security. A small amount of transaction log loss and replication delay is allowed.                     |
| 0                                     | 0                  | Limited disk write capability. No replication or long replication delay is allowed.                             |

 **NOTE**



- When both **innodb\_flush\_log\_at\_trx\_commit** and **sync\_binlog** are set to **1**, the security is the highest but the write performance is the lowest. In the event of a crash you lose at most one statement or transaction from the binary log. This is also the slowest choice due to the increased number of disk writes.
- When **sync\_binlog** is set to *N* (*N* > 1) and **innodb\_flush\_log\_at\_trx\_commit** is set to **2**, the RDS for MySQL write operation achieves the optimal performance.

## 1.2 How Do I Use the utf8mb4 Character Set to Store Emojis in an RDS for MySQL DB Instance?

To store emojis in an RDS for MySQL DB instance, ensure that:

- The client outputs the utf8mb4 character set.
- The connection supports the utf8mb4 character set. If you want to use a JDBC connection, download MySQL Connector/J 5.1.13 or a later version and leave **characterEncoding** undefined for the JDBC connection string.
- Configure the RDS DB instance as follows:
  - Setting **character\_set\_server** to **utf8mb4**

| Parameter Name       | Effective upon Reboot | Value   | Allowed Values             | Description                         |
|----------------------|-----------------------|---------|----------------------------|-------------------------------------|
| character_set_server | Yes                   | utf8mb4 | utf8, latin1, gbk, utf8mb4 | The server's default character set. |

- Log in to the management console.
  - Click  in the upper left corner of the page and choose **Databases > Relational Database Service**.
  - On the **Instances** page, click the instance name.
  - In the navigation pane on the left, choose **Parameters**. On the **Parameters** tab page, locate **character\_set\_server** and change its value to **utf8mb4**.
  - Click **Save**. In the displayed dialog box, click **Yes**.
- Selecting **utf8mb4** for database character set
    - Log in to the management console.
    - Click  in the upper left corner of the page and choose **Databases > Relational Database Service**.
    - On the **Instances** page, click the instance name.
    - On the **Databases** page, click **Create Database**. In the displayed dialog box, enter a database name and remarks, select the character set **utf8mb4**, and authorize permissions for users. Then, click **OK**.
  - Setting the character set of the table to **utf8mb4**

```
( [ ] ) [ ]> create table emoji_01 (id int auto_increment primary key, content varchar(255)) default charset utf8mb4;
Query OK, 0 rows affected (0.01 sec)

( [ ] ) [ ]> show create table emoji_01 \G
***** 1. row *****
      Table: emoji_01
Create Table: CREATE TABLE 'emoji_01' (
  'id' int(11) NOT NULL AUTO_INCREMENT,
  'content' varchar(255) DEFAULT NULL,
  PRIMARY KEY ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
1 row in set (0.00 sec)
```

### FAQs

If you have set **characterEncoding** to **utf8** for the JDBC connection string, or the emoji data cannot be inserted properly after you have performed the above operations, you are advised to set the connection character set to **utf8mb4** as follows:

```
String query = "set names utf8mb4";  
stat.execute(query);
```

# 2 RDS for PostgreSQL

---

## 2.1 Performing Basic Operations Using pgAdmin

### 2.1.1 Connecting to an RDS for PostgreSQL Instance

pgAdmin is an administration and development tool for PostgreSQL. Using pgAdmin, you can connect to specific databases from your clients, create tables, and run simple and complex SQL statements. It can be used on Windows, Linux, macOS, and other operating systems. The latest version of pgAdmin is based on the browser/server (B/S) architecture.

This section describes how to use pgAdmin to connect to an RDS for PostgreSQL instance and create databases and tables. pgAdmin4-4.17 is used as an example in this section.

For more information, see the [pgAdmin documentation](#).

#### Procedure

- Step 1** Obtain the pgAdmin installation package.  
Download the pgAdmin installation package from the [pgAdmin official website](#).
- Step 2** Double-click the installation package and complete the installation as instructed.
- Step 3** Start the pgAdmin client after the installation.
- Step 4** Use pgAdmin to access your RDS for PostgreSQL instance by referring to [Connecting to a DB Instance Through pgAdmin](#).

----End

### 2.1.2 Creating a Database

#### Scenarios

This section describes how to create a database.



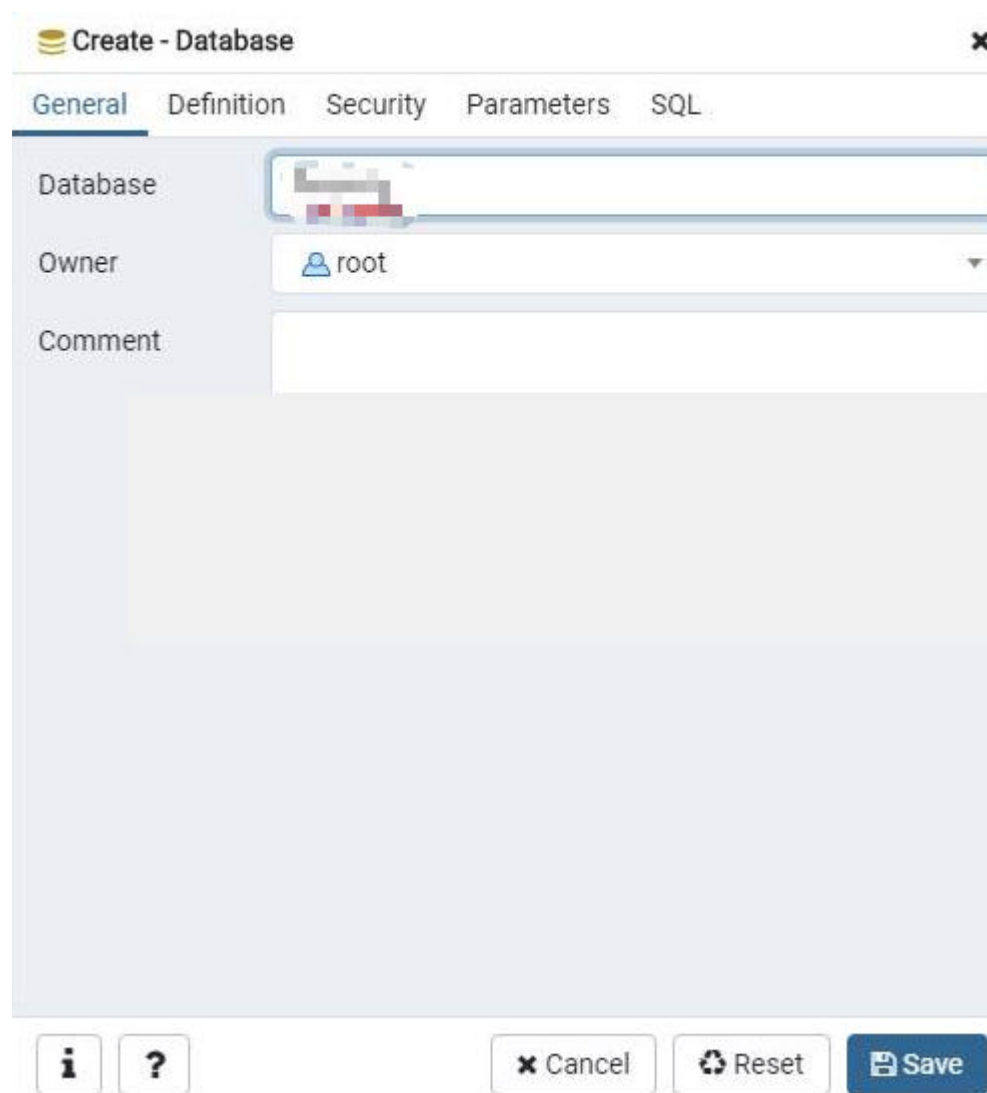
## Prerequisites

An RDS for PostgreSQL DB instance has been connected.

## Procedure

- Step 1** In the navigation pane on the left, right-click the target instance node and choose **Create > Database** from the shortcut menu.
- Step 2** On the **General** tab, specify **Database** and click **Save**.

**Figure 2-1** Creating a database



----End

## 2.1.3 Creating a Table

### Scenarios

This section describes how to create a table.

## Prerequisites

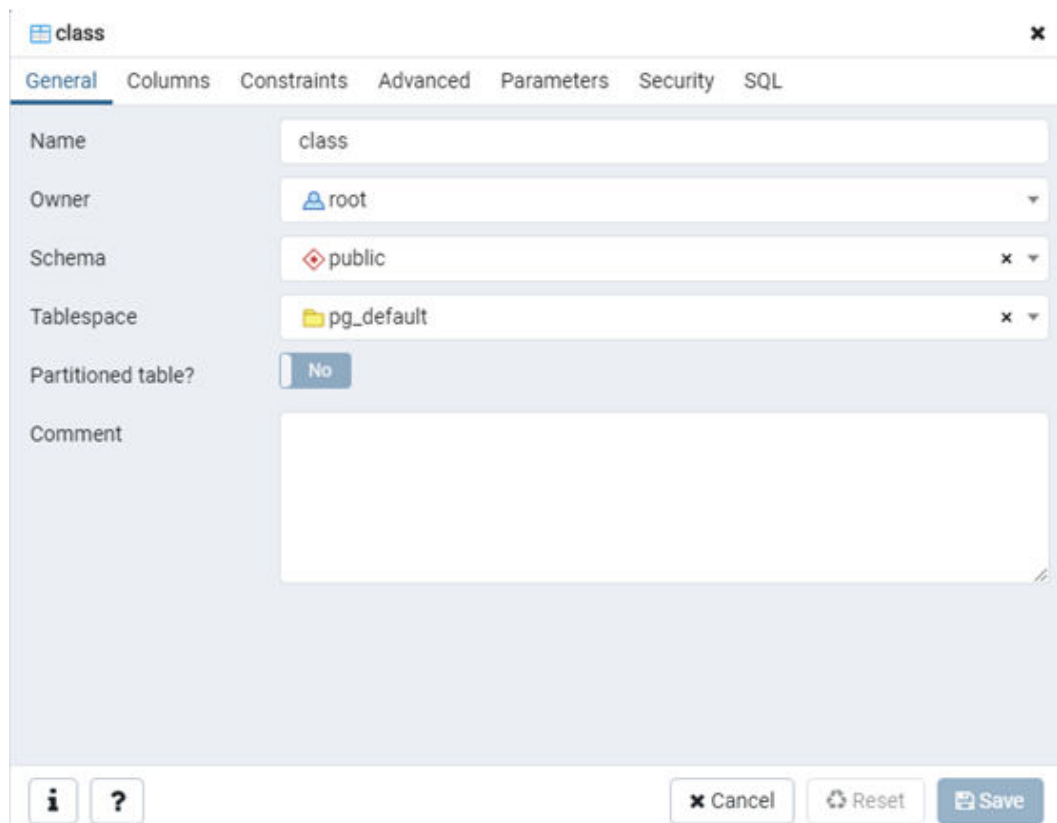
- An RDS for PostgreSQL DB instance has been connected.
- There are tables in the database and schema created by the current user.

## Procedure

**Step 1** In the navigation pane on the left, right-click the target table and choose **Create > Table** from the shortcut menu.

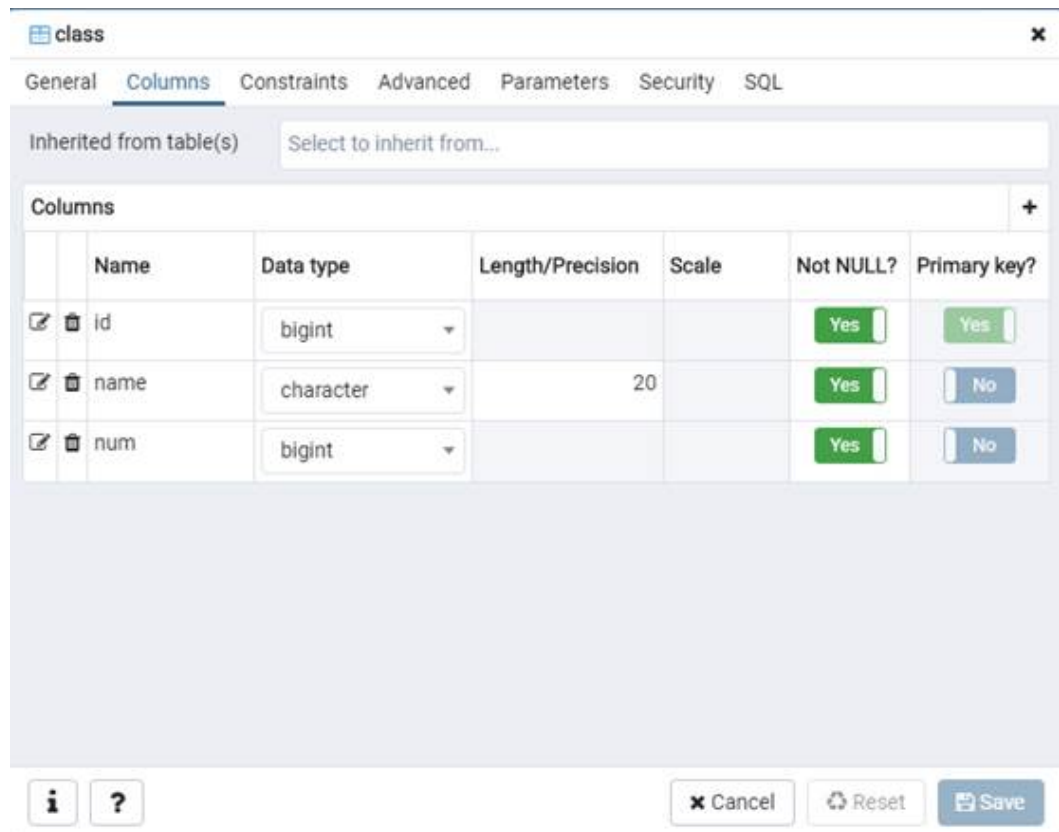
**Step 2** On the **General** tab, enter required information and click **Save**.

**Figure 2-2** Basic information



The screenshot shows a dialog box titled 'class' with a close button (x) in the top right corner. The dialog has several tabs: 'General', 'Columns', 'Constraints', 'Advanced', 'Parameters', 'Security', and 'SQL'. The 'General' tab is selected. The 'Name' field contains 'class'. The 'Owner' dropdown menu shows 'root'. The 'Schema' dropdown menu shows 'public'. The 'Tablespace' dropdown menu shows 'pg\_default'. The 'Partitioned table?' field has a 'No' button. The 'Comment' field is empty. At the bottom of the dialog, there are three buttons: 'Cancel', 'Reset', and 'Save'. There are also information (i) and help (?) icons on the bottom left.

**Step 3** On the **Columns** tab, add table columns and click **Save**.



----End

## 2.1.4 Using the Query Tool to Run SQL Statements

### Scenarios

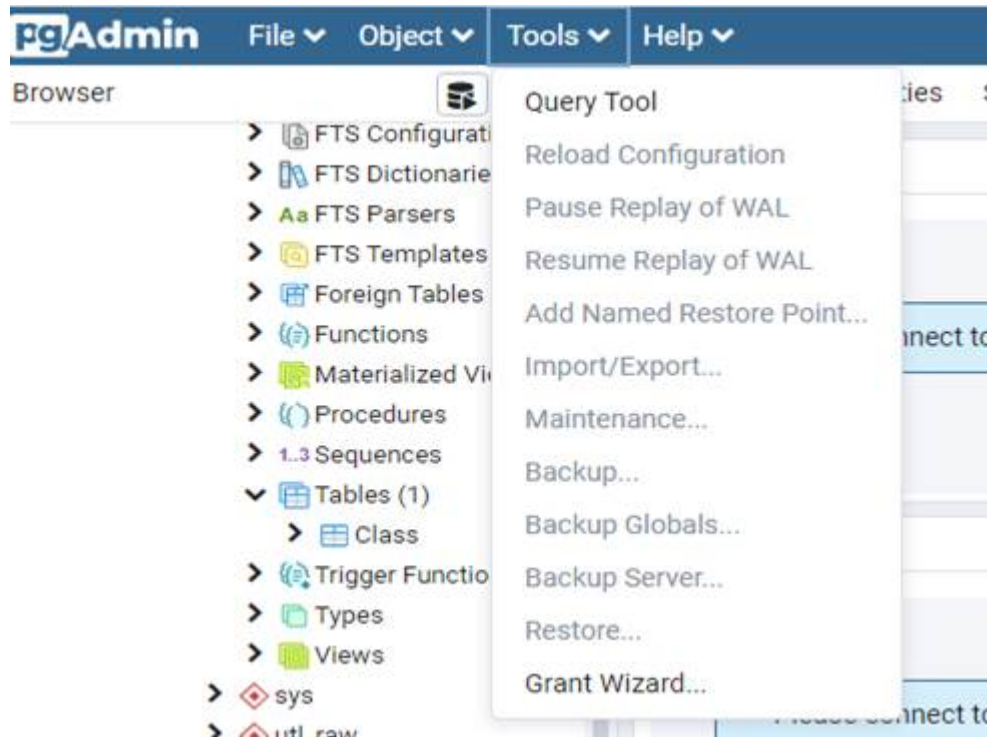
This topic describes how to use Query Tool to run SQL commands.

### Prerequisites

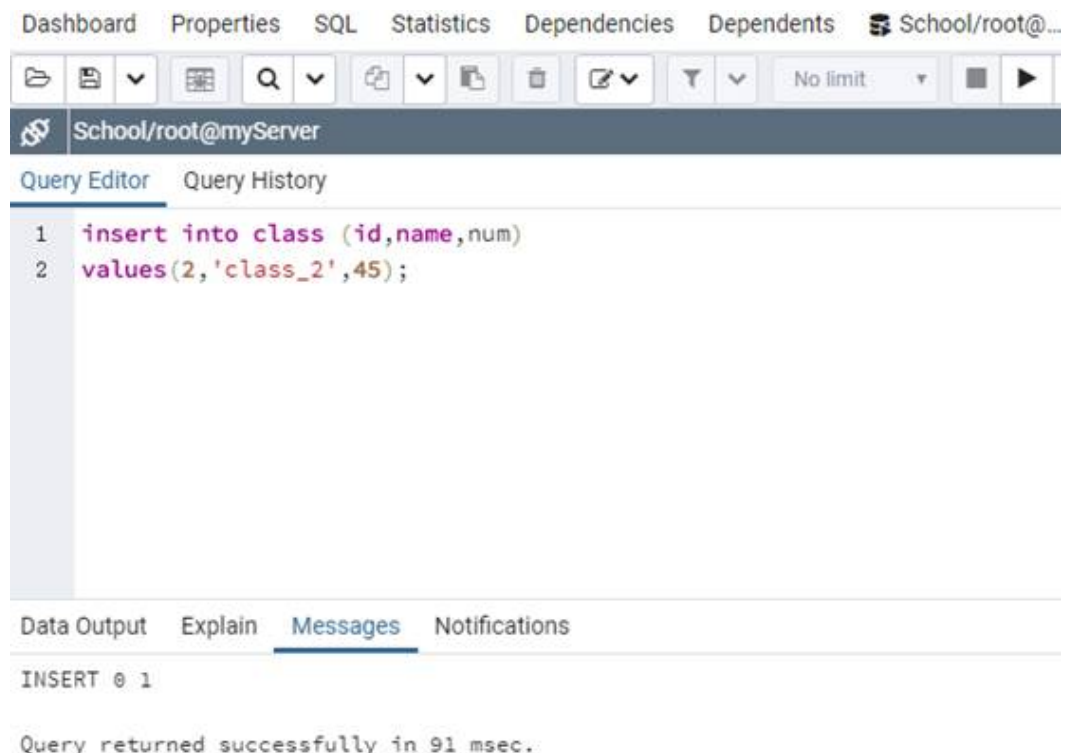
An RDS for PostgreSQL DB instance has been connected.

### Inserting Data

**Step 1** On the top menu bar, choose **Tools > Query Tool**. The SQL CLI is displayed.



**Step 2** On the SQL CLI, enter an **INSERT** command and click **Execute** to insert data into the corresponding table.

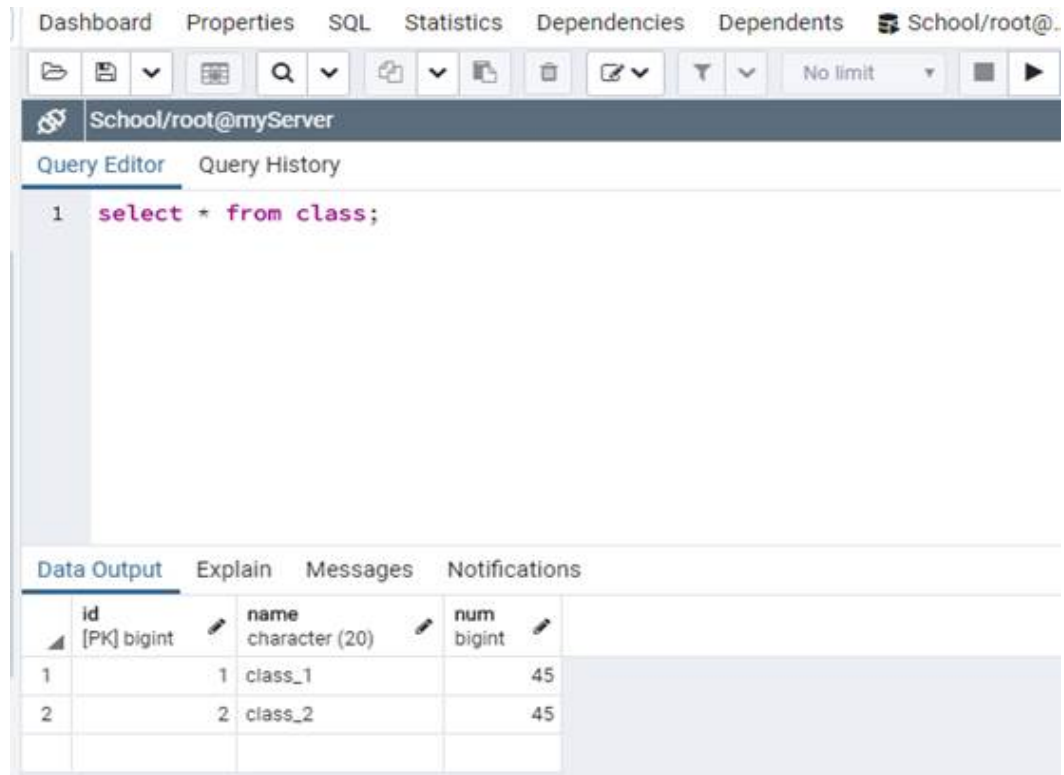


----End

## Querying Data

**Step 1** On the top menu bar, choose **Tools > Query Tool**. The SQL CLI is displayed.

**Step 2** On the SQL CLI, enter an **SELECT** command and click **Execute** to query data in the corresponding table.



----End

## 2.1.5 Monitoring Databases

### Scenarios

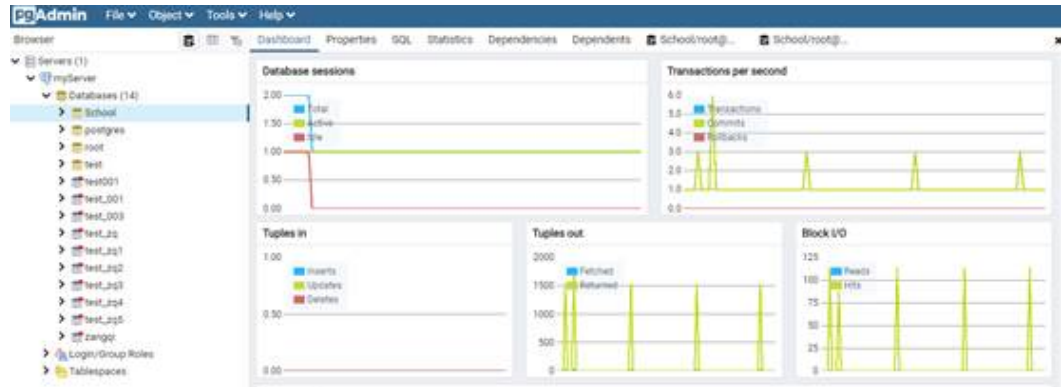
This section describes database monitoring.

### Prerequisites

An RDS for PostgreSQL DB instance has been connected.

### pgAdmin Monitoring

**Step 1** In the navigation pane on the left, select a database and click the **Dashboard** tab on the right pane to monitor database metrics, including Database sessions, Transactions per second, Tuples in, Tuples out, and Block I/O.



----End

## RDS Monitoring

The Cloud Eye service monitors operating statuses of RDS DB instances. You can view RDS database metrics on the management console. For more information, see [Viewing Monitoring Metrics](#).

## 2.1.6 Backing Up and Restoring Data

### Scenarios

This section describes how to back up and restore data.

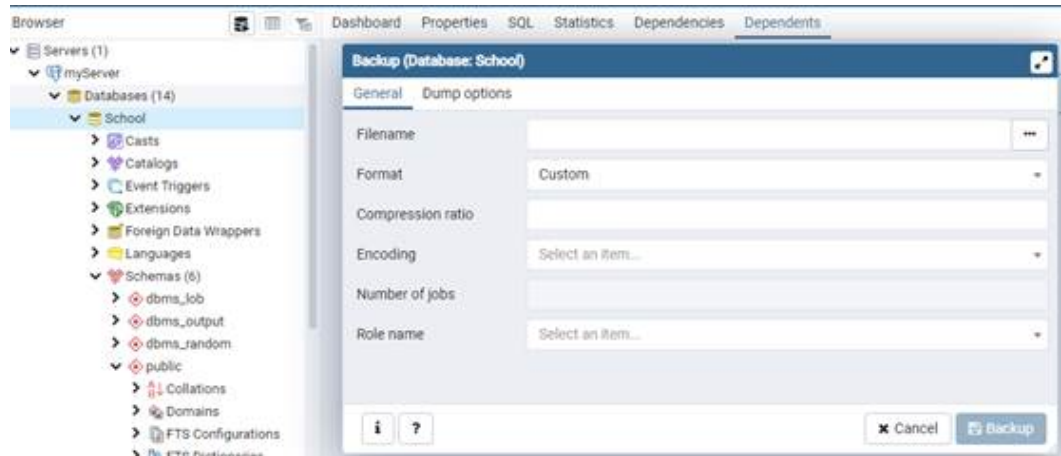
### Prerequisites

An RDS for PostgreSQL DB instance has been connected.

## pgAdmin Backup and Restoration

### Backup

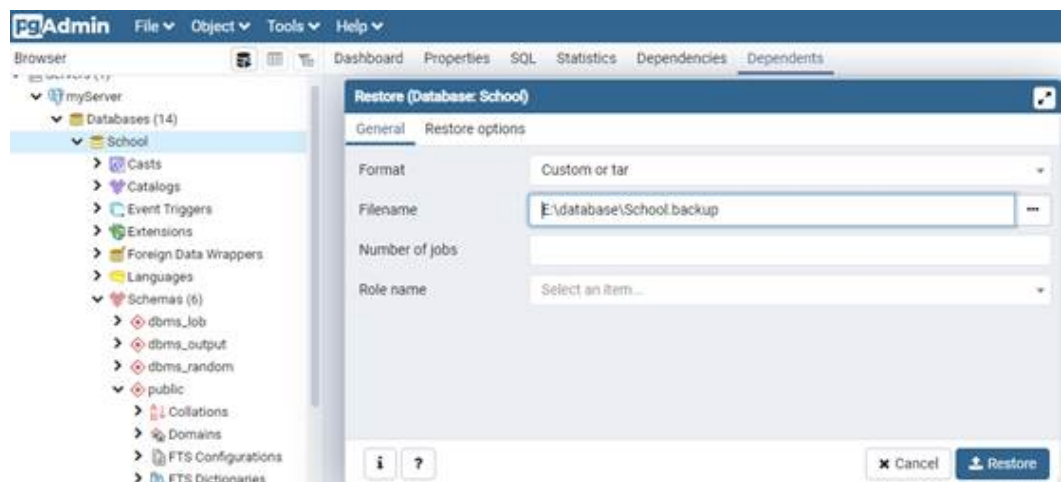
- Step 1** In the navigation pane on the left, right-click the database to be backed up and choose **Backup** from the shortcut menu.
- Step 2** On the **General** tab, enter required information and click **Backup**.

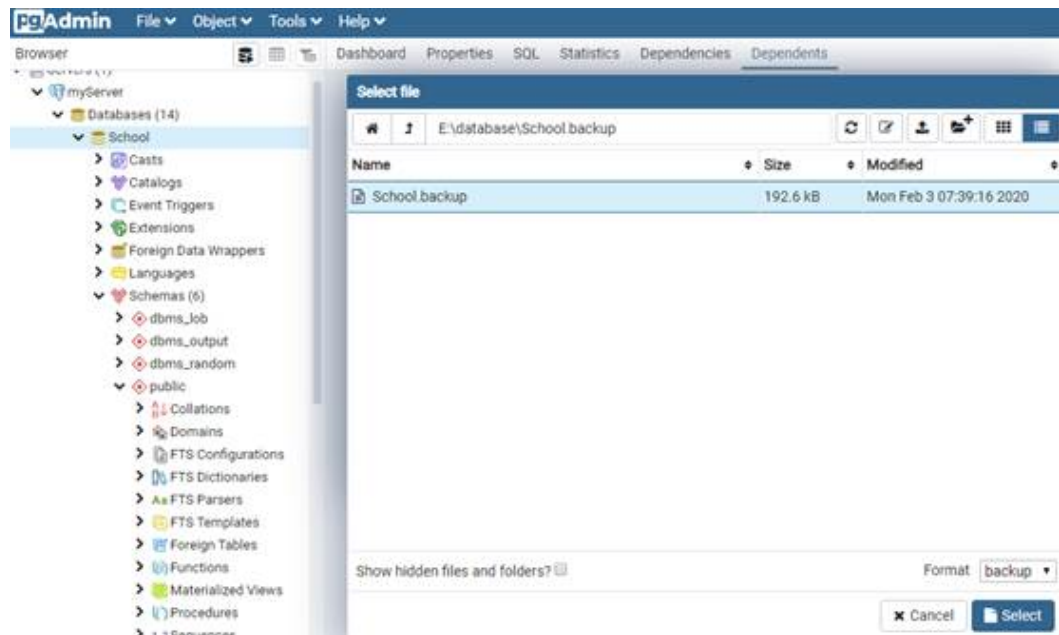


----End

### Restoration

- Step 1** In the navigation pane on the left, right-click the database to be restored and choose **Restore** from the shortcut menu.
- Step 2** On the displayed tab in the right pane, select a file name and click **Restore**.





----End

## RDS Backup and Restoration

RDS supports DB instance backup and restoration to ensure data reliability. For more information, see [Creating a Manual Backup](#).

## 2.2 Viewing Slow Query Logs of RDS for PostgreSQL DB Instances

### Scenarios

Slow query logs record statements that exceed the **log\_min\_duration\_statement** value (1 second by default). You can view log details and statistics to identify statements that are slowly executed and optimize the statements. RDS for PostgreSQL supports the following statement types:

- SELECT
- INSERT
- UPDATE
- DELETE
- CREATE
- DROP
- ALTER
- DO
- CALL
- COPY




## Parameter Description

**Table 2-1** Parameters related to RDS for PostgreSQL slow queries

| Parameter                  | Description  |
|----------------------------|--|
| log_min_duration_statement | Specifies the minimum execution time. The statements whose execution time is greater than or equal to the value of this parameter are recorded.<br><br>If this parameter is set to a smaller value, the number of log records increases, which increases the disk I/O and deteriorates the SQL performance.  |
| log_statement              | Specifies the statement type. The value can be <b>none</b> , <b>ddl</b> , <b>mod</b> , or <b>all</b> .<br><br>The default value is <b>none</b> . If you change the value to <b>all</b> : <ul style="list-style-type: none"> <li>• The database disk I/O increases, and the SQL performance deteriorates.</li> <li>• The log format changes, and you cannot view slow query logs on the console.</li> </ul> |
| log_statement_stats        | Specifies whether to output performance statistics to server logs.<br><br>The default value is <b>off</b> . If you change the value to <b>on</b> : <ul style="list-style-type: none"> <li>• The database disk I/O increases, and the SQL performance deteriorates.</li> <li>• The log format changes, and you cannot view slow query logs on the console.</li> </ul>                                       |

## Procedure

**Step 1** Log in to the management console.

**Step 2** Click  in the upper left corner of the page and choose **Databases > Relational Database Service**.

**Step 3** On the **Instances** page, click the target DB instance.

**Step 4** In the navigation pane on the left, choose **Logs**. On the **Slow Query Logs** page, click **Log Details**.

You can view the slow query log records of a specified statement type in a specified time period.

----End

## 2.3 Security Best Practices

PostgreSQL has earned a reputation for reliability, stability, and data consistency, and has become the preferred choice as an open-source relational database for

many enterprises. RDS for PostgreSQL is a cloud-based web service that is reliable, scalable, easy to manage, and immediately ready for use.

Make security configurations from the following dimensions to meet your service needs.

- [Configuring the Maximum Number of Connections to the Database](#)
- [Configuring the Timeout for Client Authentication](#)
- [Configuring SSL and Encryption Algorithm](#)
- [Configuring Password Encryption](#)
- [Disabling the Backslash Quote](#)
- [Periodically Checking and Deleting Roles That Are No Longer Used](#)
- [Revoking All Permissions on the public Schema](#)
- [Setting a Proper Password Validity Period for a User Role](#)
- [Configuring the Log Level to Record SQL Statements That Cause Errors](#)
- [Enabling Data Backup](#)
- [Avoiding Access to Your RDS for PostgreSQL Instance over the Internet](#)
- [Configuring the Delay for Account Authentication Failures](#)

## Configuring the Maximum Number of Connections to the Database

The `max_connections` parameter specifies the maximum concurrent connections allowed in a database. If the value of this parameter is large, the RDS for PostgreSQL database may request more System V shared memory or semaphore. As a result, the requested shared memory or semaphore may exceed the default value on the OS. Set `max_connections` based on service complexity. For details, see [Instance Usage Suggestions](#).

## Configuring the Timeout for Client Authentication

The `authentication_timeout` parameter specifies the maximum duration allowed for client authentication, in seconds. This parameter prevents the client from occupying the connection channel for a long time. The default value is 60s. If the authentication is not complete within the specified period, the connection is forcibly closed. This configuration can enhance the security of your RDS for PostgreSQL instance.

## Configuring SSL and Encryption Algorithm

SSL is recommended for TCP/IP connections because SSL ensures that all communications between the client and server are encrypted, preventing data leakage and tampering and ensuring data integrity. When configuring SSL encryption, you need to configure a secure TLS protocol and encryption algorithm on the server. TLSv1.2 and ECDH+ECDSA+AESGCM:ECDH+aRSA+AESGCM:EDH+aRSA+AESGCM:EDH+aDSS+AESGCM:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!SRP:!RC4 are recommended. For details, see [SSL Connection](#).

To configure the TLS protocol and encryption algorithm, change the values of `ssl_min_protocol_version` and `ssl_ciphers`, respectively.

## Configuring Password Encryption

Passwords must be encrypted. When you use **CREATE USER** or **ALTER ROLE** to change a password, the password is encrypted by default. The password encryption method **scram-sha-256** is recommended. To change the password encryption method, change the value of **password\_encryption**.

The **MD5** option is used only for compatibility with earlier versions. New DB instances use **scram-sha-256** by default.

---

### NOTICE

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

---

## Disabling the Backslash Quote

The **backslash\_quote** parameter specifies whether a single quotation mark (') in a string can be replaced by a backslash quote (\'). The preferred, SQL-standard way to represent a single quotation mark is by doubling it ("). If client-side code does escaping incorrectly then an SQL-injection attack is possible. You are advised to set **backslash\_quote** to **safe\_encoding** to reject queries in which a single quotation mark appears to be escaped by a backslash, preventing SQL injection risks.

## Periodically Checking and Deleting Roles That Are No Longer Used

Check whether all roles are mandatory. Any unknown role must be reviewed to ensure that it is used properly. If any role is no longer used, delete it. To query roles, run the following command:

```
SELECT rolname FROM pg_roles;
```

## Revoking All Permissions on the public Schema

The **public** schema is the default schema. All users can access objects in it, including tables, functions, and views, which may cause security vulnerabilities. You can run the following command as user **root** to revoke the permissions:

```
revoke all on schema public from public;
```

## Setting a Proper Password Validity Period for a User Role

When creating a role, you can use the **VALID UNTIL** keyword to specify when the password of the role becomes invalid. If this keyword is ignored, the password will be valid permanently. You are advised to change the password periodically, for example, every three months. To configure a password validity period, run the following command:

```
CREATE ROLE name WITH PASSWORD 'password' VALID UNTIL 'timestamp';
```

To check whether a password validity period is configured, run the following command:

```
SELECT rolname,rolvaliduntil FROM pg\ Roles WHERE rolsuper = false AND
rolvaliduntil IS NULL;
```

## Configuring the Log Level to Record SQL Statements That Cause Errors

The `log_min_error_statement` parameter specifies which SQL statements that cause errors can be recorded in server logs. The SQL statements of the specified level or higher are recorded in logs. Valid values include **debug5**, **debug4**, **debug3**, **debug2**, **debug1**, **info**, **notice**, **warning**, **error**, **log**, **fatal**, and **panic**. The value of `log_min_error_statement` must be at least **error**.

## Enabling Data Backup

When you create an RDS DB instance, an automated backup policy is enabled by default with the retention period set to seven days. You can change the backup retention period as required. RDS for PostgreSQL DB instances support **automated backups** and **manual backups**. You can periodically back up your instance. If the instance fails or data is damaged, **restore it using backups** to ensure data reliability. For details, see .

## Avoiding Access to Your RDS for PostgreSQL Instance over the Internet

Do not deploy your instance on the Internet or in a demilitarized zone (DMZ). Instead, deploy it on a Huawei Cloud private network and use routers or firewalls to protect it. Do not bind an EIP to your instance for access over the Internet to prevent unauthorized access and DDoS attacks. If you have bound an EIP to your instance, you are advised to unbind it. If you do need an EIP, **configure security group rules** to restrict the source IP addresses that can access your instance.

## Configuring the Delay for Account Authentication Failures

By default, RDS for PostgreSQL instances have a built-in `auth_delay` extension. `auth_delay` causes the server to stop for a short period of time before an authentication failure message is returned, making it more difficult to crack the database password. To configure the delay for account authentication failures, change the value of the **auth\_delay.milliseconds** parameter (which indicates the number of milliseconds to wait before reporting an authentication failure) by referring to **Modifying Parameters of an RDS for PostgreSQL Instance**. The default value of this parameter is **0**.