

Relational Database Service

Best Practices

Issue 01
Date 2022-09-30



Copyright © Huawei Technologies Co., Ltd. 2022. 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>

Contents

1 RDS for MySQL	1
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
2 RDS for PostgreSQL	4
2.1 Performing Basic Operations Using pgAdmin.....	4
2.1.1 Connecting to an RDS for PostgreSQL Instance.....	4
2.1.2 Creating a Database.....	4
2.1.3 Creating a Table.....	5
2.1.4 Using the Query Tool to Run SQL Statements.....	7
2.1.5 Monitoring Databases.....	9
2.1.6 Backing Up and Restoring Data.....	10
2.2 Viewing Slow Query Logs of RDS for PostgreSQL DB Instances.....	12

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 1 . For details, see Parameter Description .
<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

innodb_flush_log_at_trx_commit	sync_binlog	Description
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 emoji 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.
 - 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.
 - 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**

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

(mysql:~) [mysql] > 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 client management software that designs, maintains, and manages RDS for PostgreSQL databases. It allows you to connect to specific databases, create tables, and run various simple and complex SQL statements. It supports Windows, Linux, Mac OS X, 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.
Visit the [pgAdmin official website](#) and download the pgAdmin installation package for Windows.
- Step 2** Double-click the installation package and complete the installation as instructed.
- Step 3** Start the pgAdmin client after the installation.
- Step 4** Connect pgAdmin to 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.

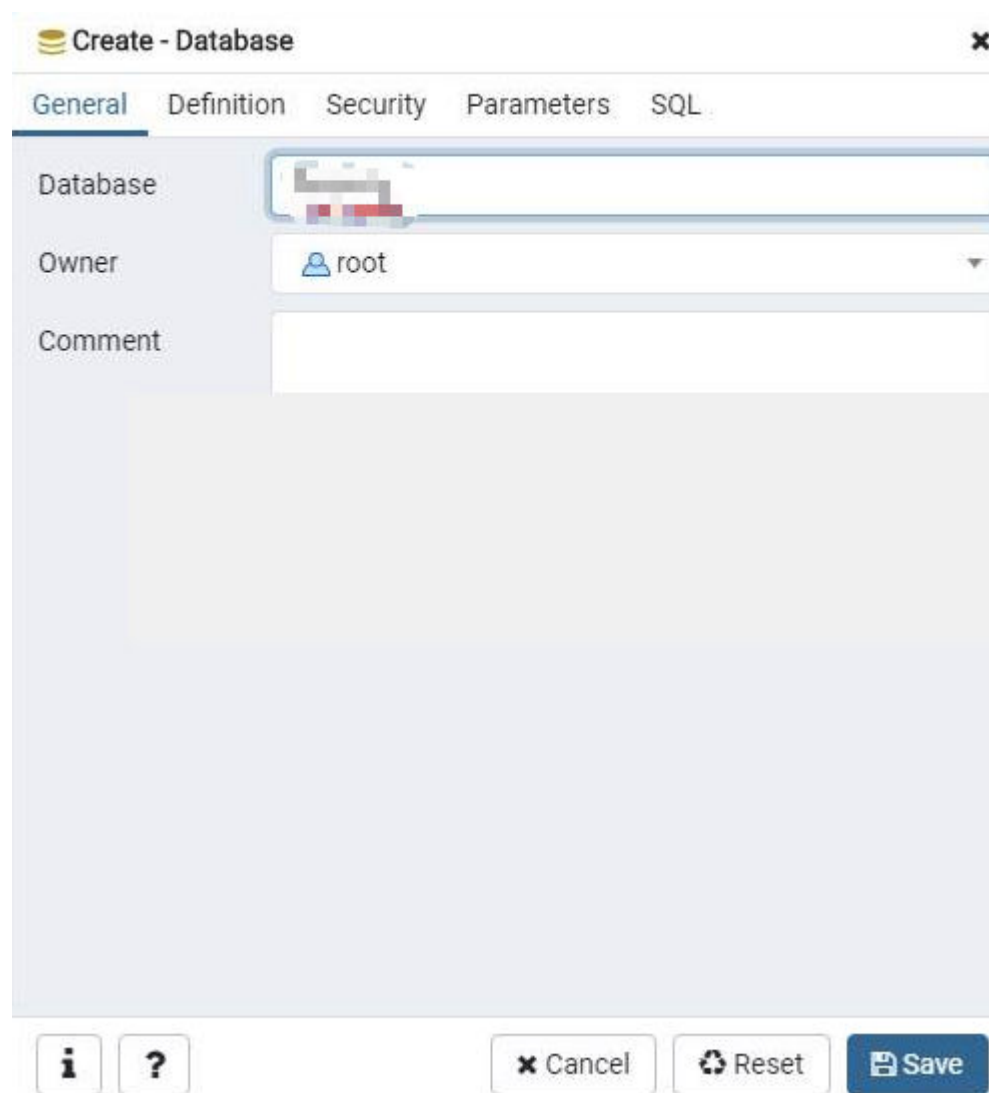
Prerequisites

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

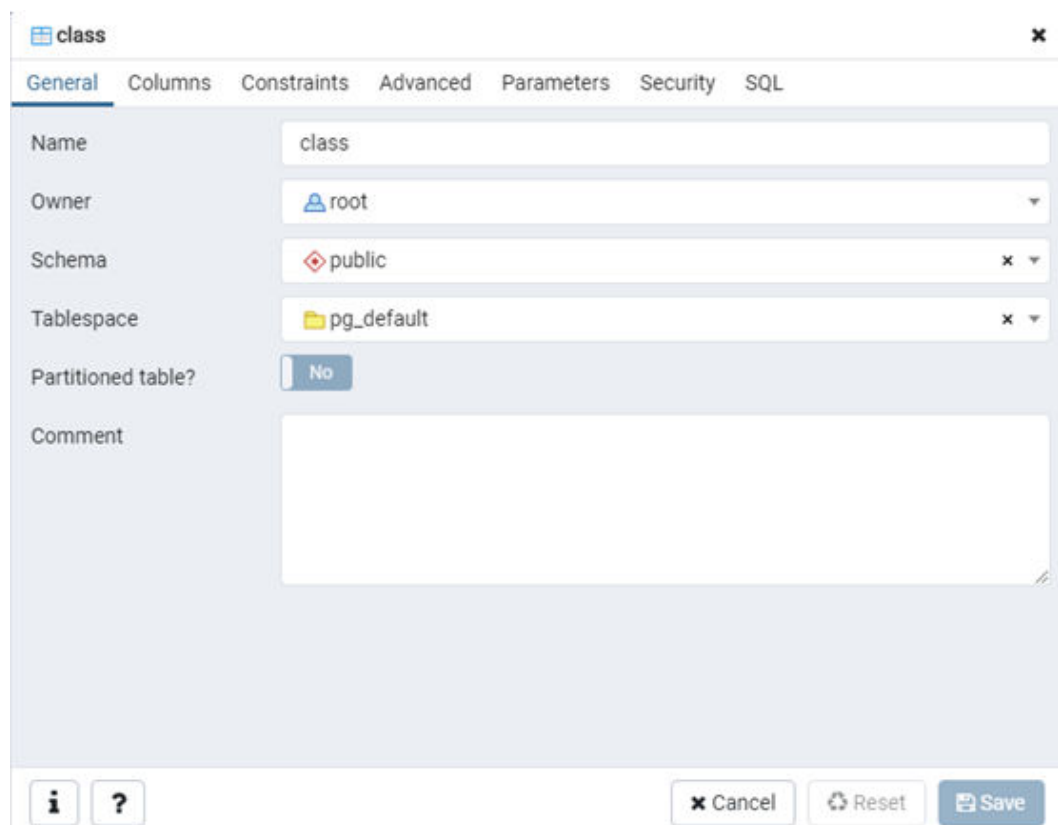
- An RDS for PostgreSQL DB instance has been connected.
- There are tables in the database and schema created by the current user.

Procedure

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

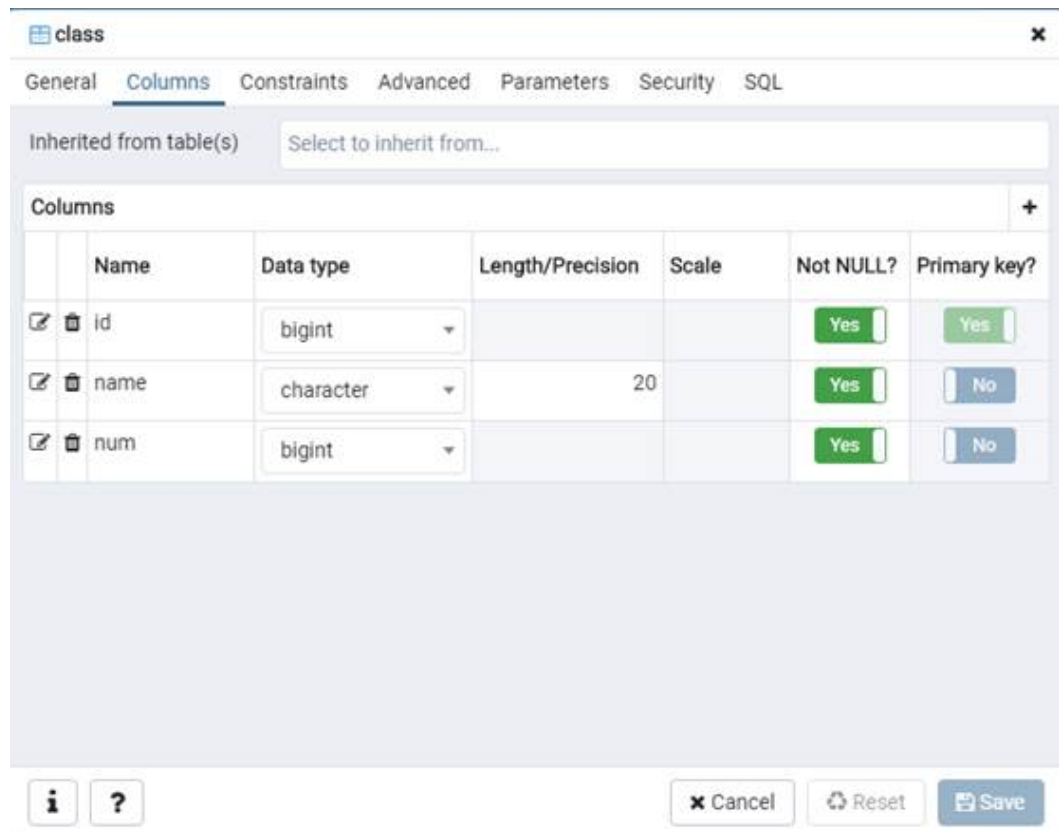
Step 2 On the **General** tab, enter required information and click **Save**.

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

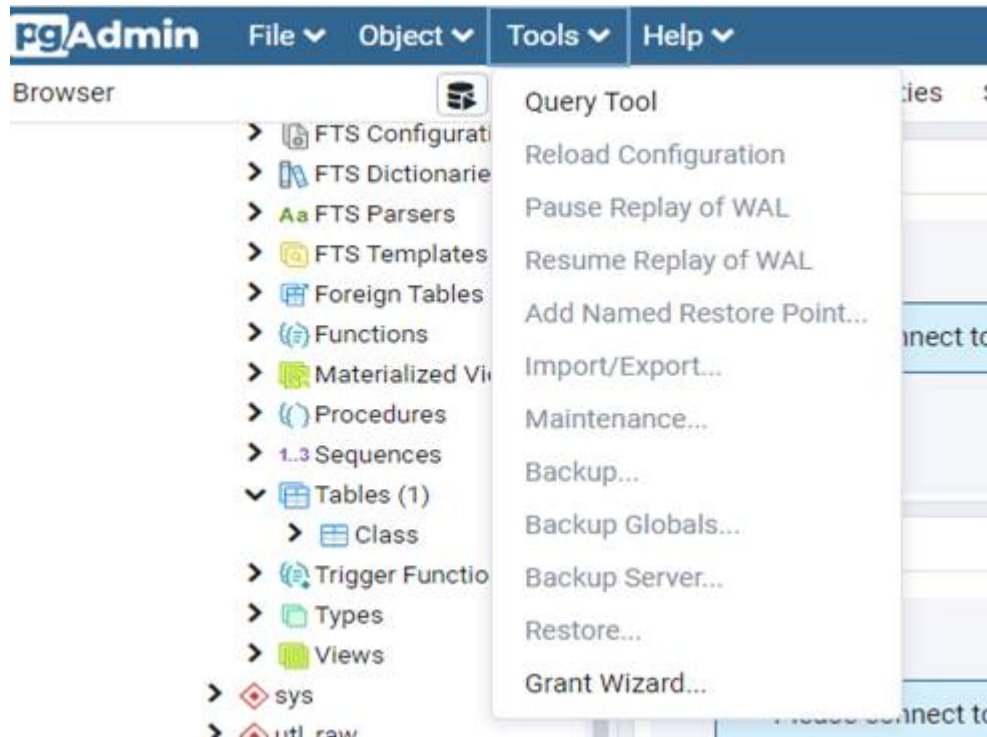
This topic describes how to use Query Tool to run SQL commands.

Prerequisites

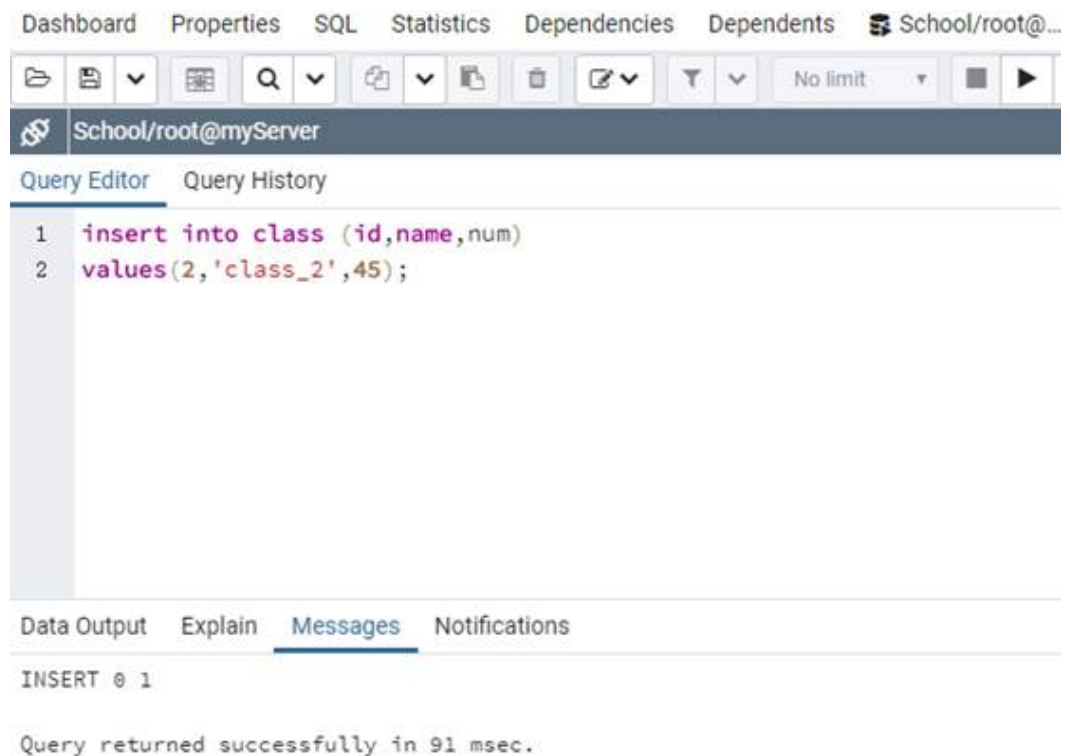
An RDS for PostgreSQL DB instance has been connected.

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.

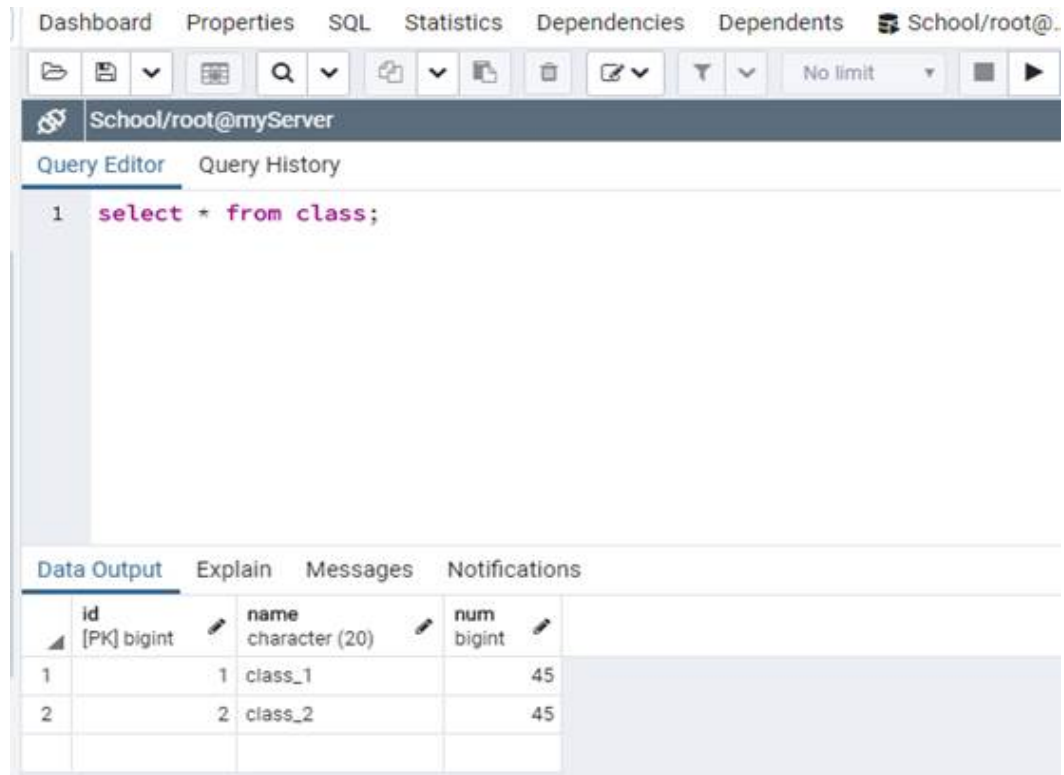


----End

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

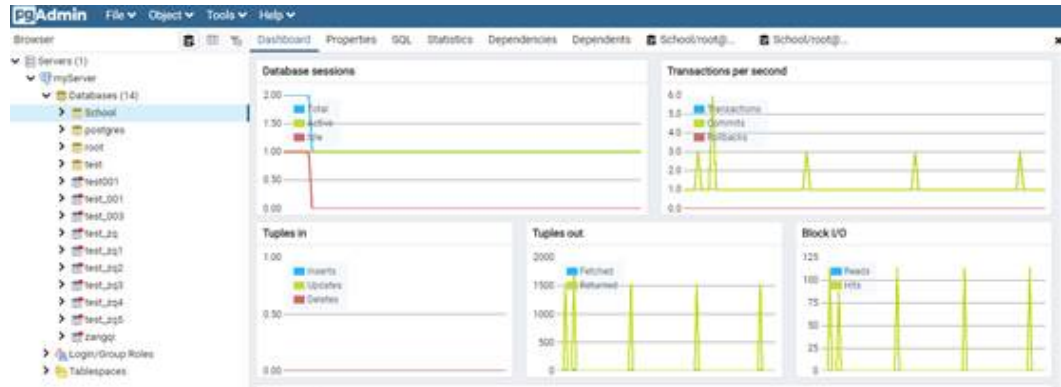
This section describes database monitoring.

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.

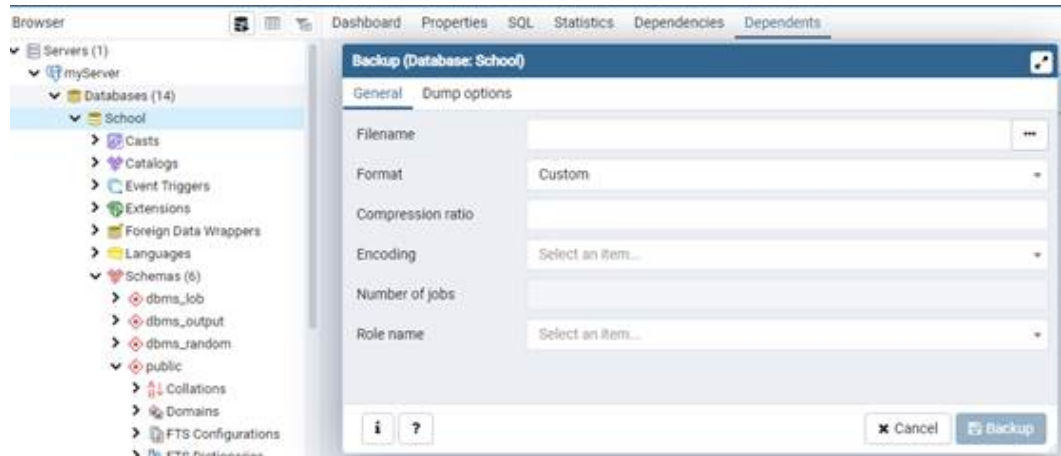
Prerequisites

An RDS for PostgreSQL DB instance has been connected.

pgAdmin Backup and Restoration

Backup

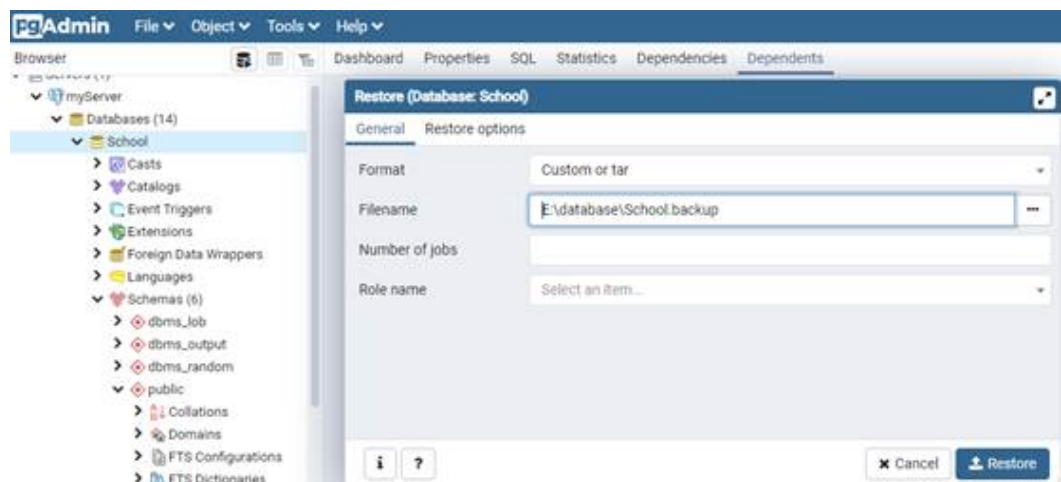
- Step 1** In the navigation pane on the left, right-click the database to be backed up and choose **Backup** from the shortcut menu.
- Step 2** On the **General** tab, enter required information and click **Backup**.

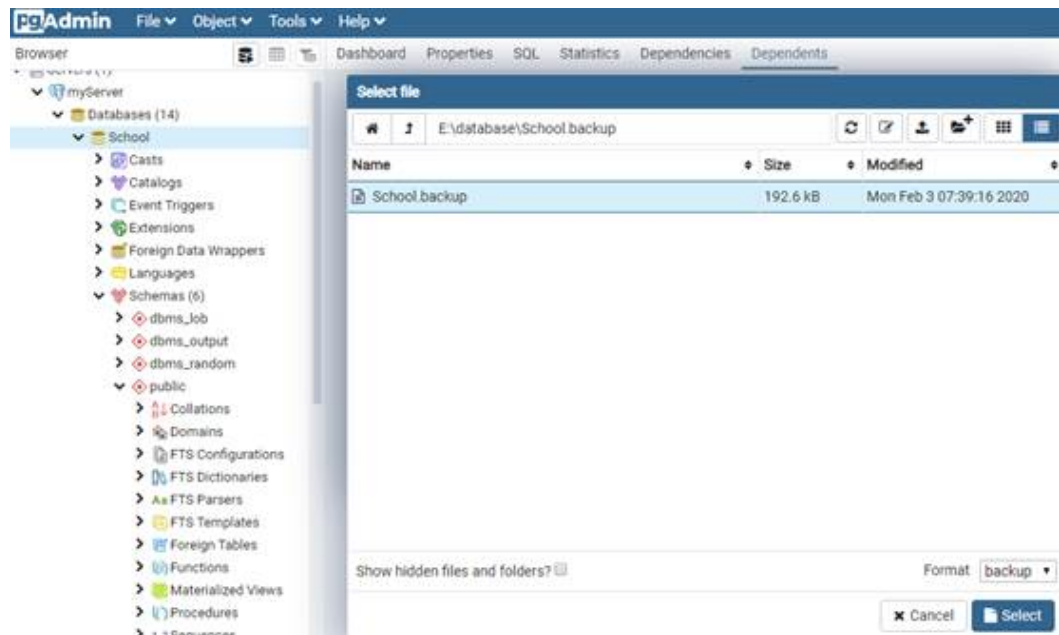


----End

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.
log_statement	Specifies the statement type. The value can be none , ddl , mod , or all . The default value is none . If you change the value to all , the database log format changes and slow query logs fail to be parsed.

Procedure

- Step 1** Log in to the management console.
- Step 2** On the **Instances** page, click the target DB instance.
- Step 3** 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