06 - PosgreSQL

PostgreSQL, or Postgres, is an object-relational database management system that uses the SQL language. It's free, open-source, reliable, robust, and performant. PostgreSQL is also one of the most popular & used relational databases.

`psql` is an interface you can access through the terminal to interact with Postgres databases. You can use it to connect to a database, add & read & modify data, check the available databases & fields, run commands from a file, and so on.

If you want to learn about Postgres commands or refresh your memory, you are in the right place! This article will teach you the top `psql` commands and flags you need to know when working with PostgreSQL.

1. Connect to a database - `psql -d`

The first step involves learning how to connect to a database. There are two ways to connect to a PostgreSQL database, depending on where the database resides.

Same host database

If the database is on the same host as your machine, you can use the following command:

```
psql -d <db-name> -U <username> -W


// example
psql -d tutorials_db -U admin -W
```

The above command includes three flags:

- `-d` - specifies the name of the database to connect to
- `-U` - specifies the name of the user to connect as
- `-W` - forces psql to ask for the user password before connecting to the database

In this example, the command connects you to the `tutorials_db` under the `admin` user.

Different host database

In the cases where your database is hosted somewhere else, you can connect as follows:

```
psql -h <db-address> -d <db-name> -U <username> -W


//example
psql -h my-psql-db.cloud.neon.tech -d tutorials_db -U admin -W
```

The `-h` flag specifies the host address of the database.

SSL mode

There might be cases where you want to use SSL for the connection.

```
psql "sslmode=require host=<db-address> dbname=<db-name> user=<username>"


//example
psql "sslmode=require host=my-psql-db.cloud.neon.tech dbname=tutorials_db user=admin"
```

The above command opens an SSL connection to the specified database.

2. List all databases - `\l`

In many cases, you will work with more than one database. You can list all the available databases with the following command:

```
\l
```

List all Postgres databases with the psql command \l

The above image illustrates what happens when you run the command. You get a table with all databases and their name, owner, access privileges, and other information.

3. Switch to another database - `\c`

You can also switch to another database with the following command:

```
\c <db-name>
```

```
// example
\c tutorials_db
```

The below image illustrates the result after running the command.

Switch to another Postgres database with the psql command \c

The command switches to the specified database under the user you logged in previously.

4. List database tables - \dt

Let's consider you want to see all the tables from the database. You can list all database tables as follows:

```
\dt
```

List all tables from a database with the psql command \dt

The `\dt` psql command returns the tables alongside:

- the schema they belong to
- their type
- their owner

5. Describe a table - \d

psql also has a command that lets you see the table's structure.

```
\d <table-name>


// example
\d tutorials
```

Describe a table with the psql command \d

The `\d` command returns all the columns, their types, collection, whether they are nullable or not, and their configured default value.

If you want more information about a table, you can use the command:

```
\d+ <table-name>
```

Get extra information about a table with the psql command \d+

Image not found or type unknown

Now, you get extra information such as storage, compression, stats target, and a description.

6. List all schemas - \dn

The `\dn` psql command lists all the database schemas.

List all schemas with the psql \dn command

Image not found or type unknown

It returns the name of the schemas and their owners.

7. List users and their roles - \du

Sometimes, you might need to change the user. Postgres has a command that lists all the users and their roles.

```
\du
```

List all users and their roles with the psql command \du

Image not found or type unknown

As the image shows, the command returns all the users.

8. Retrieve a specific user - \du

You can also retrieve information about a specific user with the following command:

```
\du <username>


//example
/du postgres
```

Retrieve information about a specific user with psql command \du

Image not found or type unknown

Now, you can see the roles of the specified user, and whether the user is a member of a group or not.

9. List all functions - `\df`

You can list all the functions from your database with the `\df` command.

List all available functions with the psql command \df

The command returns all functions and the:

- schema they belong to
- names
- result data type
- argument data types
- type

10. List all views - `\dv`

The `psql` interface enables you to list all the database views with the `\dv` command.

11. Save query results to a file - `\o`

There might be cases where you want to analyze the result of a query at a later time. Or two compare two query results. The `psql` interface allows you to do that.

You can save query results in a file as follows:

```
\o <file-name>


// example
\o query_results
...run the psql commands...
\o - stop the process and output the results to the terminal again
```

Let's save the following query results in a file:

- the available database tables
- the result of describing a database table
- the list of all users
- the details of the user "postgres"

Save query results to a file

Note: To stop saving results to the file, you need to run the `\o` command again without the file name.

Query results in a file

Image not found or type unknown

The above image illustrates the file containing all the query results.

12. Run commands from a file - `\i`

It's also possible to run commands from a file. For simple commands, it might not be the best solution. But when you want to run multiple commands and complex SQL statements, it helps a lot.

Create a `txt` file with the following content:

```
\l
\dt
\du
```

When you run the file, it should return a list of all:

- databases
- database tables
- users

You can run commands from a file with the following `psql` command:

```
\i <file-name>


// example
\i psql_commands.txt
```

Run psql commands from a file

Image not found or type unknown

The command returned all the databases, tables, and users as expected.

Quit psql - `\q`

You quit the `psql` interface with the `\q` command.

Get APIs instantly for PostgreSQL

All the above commands with psql are great to connect to PostgreSQL on the commandline for direct access and execution of operations. But in case you are looking to build APIs for data access on Postgres, you should check out Hasura. Hasura connects to your existing or new PostgreSQL database, introspects the schema(s) and lets you generate CRUD APIs in GraphQL and REST declaratively in minutes. Read more here to get started with Hasura and PostgreSQL.

Note: Any flavor of Postgres which is wire compatible will work with Hasura.

Summary

The `psql` interface is powerful and allows you to do many things, including running SQL statements. If you want to get started with PostgreSQL or deepen your knowledge, check out our PostgreSQL Tutorial. You can also check out the Postgre articles on our blog.

> If you are looking for options to host your Postgres database, check these PostgreSQL hosting options.

Revision #2
Created 7 March 2024 14:40:12 by Sandi Wibowo
Updated 7 March 2024 14:41:39 by Sandi Wibowo