Autor: Alejandro Martín de León
Contacto: alu0101015941@ull.edu.es

INFORME: "Introducción a PostgreSQL"

INDICE

Introducción

PostgreSQL es un sistema de gestión de bases de datos relacional orientado a objetos. Se trata de un programa *open source*, contando con una comunidad de desarrolladores que trabajan en mejorar el programa de forma desinteresada. Tiene su origen en el año 1982, siendo este proyecto liderado por Michael Stonebraker.

Instalación y acceso

A continuación, se procederá a detallar todos los pasos llevados a cabo para realizar la instalación de PostgreSQL. En este caso, se hará uso de una Máquina Virtual, proporcionada por el IaaS de la ULL.

Una vez iniciada la misma, y después de haber actualizado a través del comando `sudo apt update`, se procede a usar la siguiente sentencia:

```
sudo apt-get install postgresql
```

Una vez terminada la instalación, se procederá a acceder al entorno de postgreSQL. Para ello:

```
sudo su postgres
```

A través de la terminal se podrá visualizar algo parecido a lo siguiente:

```
usuario@adbd:~$ sudo su postgres
postgres@adbd:/home/usuario$
```

De esta forma se cambian las credenciales del usuario al superusuario de postgreSQL, identificado como "postgres". Desde ahí, se procederá a acceder al programa **psql**

```
psql -U <usuario>
```

Sustituyendo en el comando anterior el parámetro *usuario* por *postgres* se accederá al programa desde el superusuario de PostgreSQL. Se mostrará por la terminal el siguiente resultado:

```
postgres@adbd:/home/usuario$ psql -U postgres
psql (12.8 (Ubuntu 12.8-0ubuntu0.20.04.1))
Type "help" for help.

postgres=#
```

Uso de comandos útiles

Una vez realizada apropiadamente la instalación y accedido al programa **psql** se procederá a detallar con claridad algunos de los comandos que permite utilizar PostgreSQL.

Creación de un usuario

Desde un primer momento, en la instalación de PostgreSQL, se crea en el sistema el superusuario **postgre**. Con él, se pueden crear además diferentes usuarios. Para crear un usuario se hace uso del siguiente comando:

```
CREATE USER <usuario> WITH PASSWORD '<contraseña>';
```

Para crearlo sin contraseña:

```
CREATE USER <usuario>;
```

Se sustituirá el parámetro **usuario** por el nombre que identificará al usuario y el parámetro **contraseña**, por la clave de acceso del mismo, contenida entre comillas simples. Un ejemplo del mismo se muestra a continuación:

```
postgres=# CREATE USER user1;
CREATE ROLE
```

Además, por cada usuario se han de poder establecer y modificar roles para los mismos. Aprovechando la creación del usuario anterior, establecido por defecto sin contraseña, se mostrará a continuación el procedimiento para establecerle un credencial al mismo:

```
postgres=# CREATE USER user1;
CREATE ROLE
postgres=# ALTER ROLE user1 WITH PASSWORD 'user1';
ALTER ROLE
postgres=#
```

Los usuarios que han sido creados se pueden listar, para conseguir esto se ha de hacer uso del comando `\du`, el cuál mostrará por la terminal el listado siguiente:

```
postgres=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 miusuario |                                                            | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
 user1     |                                                            | {}
```

En este listado ofrece información adicional sobre cada usuario, como es su lista de roles, así como de las tablas de las que son miembros. Los usuarios además se pueden eliminar, para ello se puede hacer uso de la siguiente sentencia:

```
DROP USER <nombre>;
```

```
postgres=# DROP USER user1;
DROP ROLE
```

Creación de Bases de datos y tablas

Después de conocer los pasos para la creación de usuarios, es de importancia conocer cuáles son los procedimientos llevados a cabo para la creación de las bases de datos, con sus respectivas tablas. En primer lugar, se procede a crear una base de datos, utilizando el siguiente comando:

```
CREATE DATABASE <nombre_bd>
```

Donde se sustituirá **nombre_db** por el nombre que identificará a la base de datos deseada. Esta base de datos por defecto se encontrará vacía, es por ello que para crear las tablas correspondientes a la misma será necesario seleccionar previamente la base de datos deseada. Para ello, se hace uso del comando:

```
\c <nombre_db>
```

El cuál seleccionará la base de datos indicada, pudiendo así hacer cambios sobre la misma. Ahora, es momento de crear la primera tabla de la base de datos, para ello:

```
CREATE TABLE <nombre_tabla> (
    <nombre_atributo> <tipo_atributo>
    . . .
    . . .

);
```

Por último, será necesario insertar las tuplas que complementarán la tabla. PAra realizar esta tarea se hace uso de las sentencias:

```
INSERT INTO <nombre_tabla>(<columna1>, <columna2>, ...) VALUES (<valor1>,
<valor2>, ...);
```

Un ejemplo más detallado de esto se podrá ver en el apartado Ejemplo de pruebas.

Otros comandos útiles

A continuación se detallarán una serie de comandos que pueden resultar de utilidad en el uso de PosgreSQL.

- Listar las bases de datos: Esto se puede lograr con el comando \l:

```
postgres=# \l
                                 List of databases
   Name    |  Owner   | Encoding | Collate |  Ctype  |   Access privileges
-----------+----------+----------+---------+---------+-----------------------
 postgres  | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 pract1    | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 template0 | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
           |          |          |         |         | postgres=CTc/postgres
 template1 | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
           |          |          |         |         | postgres=CTc/postgres
(4 rows)
```

- Conectar a una base de datos: Se puede conectar a una base de datos a través del siguiente comando:

```
\c <nombre_basededatos>
```

```
postgres=# \c pract1
You are now connected to database "pract1" as user "postgres".
```

Esto permitirá poder acceder a las tablas de la misma, así como realizar los cambios pertinentes en la composición de la Base de Datos. Además, se puede especificar un usuario para acceder a la misma con él.

- Listar las tablas en la base de datos actual:

Dentro de una base de datos, se puede listar todas las tablas que la componen, de la siguiente manera:

```
\dt
```

```
postgres=# \c pract1
You are now connected to database "pract1" as user "postgres".
pract1=# \dt
          List of relations
 Schema |   Name   | Type  |  Owner
--------+----------+-------+----------
 public | usuarios | table | postgres
(1 row)
```

- Comando para la ayuda:

El comando `\?` proporcionará un listado de los comandos de psql, con una breve descripción del mismo. Se podrá ver a continuación una pequeña parte de lo que muestra por la terminal:

```
General
  \copyright             show PostgreSQL usage and distribution terms
  \crosstabview [COLUMNS] execute query and display results in crosstab
  \errverbose            show most recent error message at maximum verbosity
  \g [FILE] or ;         execute query (and send results to file or |pipe)
  \gdesc                 describe result of query, without executing it
  \gexec                 execute query, then execute each value in its result
  \gset [PREFIX]         execute query and store results in psql variables
  \gx [FILE]             as \g, but forces expanded output mode
  \q                     quit psql
  \watch [SEC]           execute query every SEC seconds

Help
  \? [commands]          show help on backslash commands
  \? options             show help on psql command-line options
  \? variables           show help on special variables
  \h [NAME]              help on syntax of SQL commands, * for all commands

Query Buffer
  \e [FILE] [LINE]       edit the query buffer (or file) with external editor
  \ef [FUNCNAME [LINE]]  edit function definition with external editor
  \ev [VIEWNAME [LINE]]  edit view definition with external editor
  \p                     show the contents of the query buffer
  \r                     reset (clear) the query buffer
  \s [FILE]              display history or save it to file
  \w FILE                write query buffer to file

Input/Output
  \copy ...              perform SQL COPY with data stream to the client host
```

Además, existe el comando `\h <comando>` que proporcionará una ayuda detallada del comando que se le proporcione, como por ejemplo:

```
pract1=# \h SELECT
Command:     SELECT
Description: retrieve rows from a table or view
Syntax:
[ WITH [ RECURSIVE ] with_query [, ...] ]
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    [ * | expression [ [ AS ] output_name ] [, ...] ]
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY grouping_element [, ...] ]
    [ HAVING condition ]
    [ WINDOW window_name AS ( window_definition ) [, ...] ]
    [ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] select ]
    [ ORDER BY expression [ ASC | DESC | USING operator ] [ NULLS { FIRST | LAST } ] [, ...] ]
    [ LIMIT { count | ALL } ]
    [ OFFSET start [ ROW | ROWS ] ]
    [ FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } ONLY ]
    [ FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE } [ OF table_name [, ...] ] [ NOWAIT | SKIP LO
CKED ] [...] ]
```

- Grabar el histórico de comandos:

Para grabar el histórico de comandos se usará la sentencia: `\s`

- Ejecutar comandos desde un fichero:

Con el comando `\i <ruta_fichero>` se podrán ejecutar scripts.

- Salir: `\q`

Ejemplo de pruebas

Después de esta introducción a PosgreSQL, se va a proponer un pequeño ejemplo de creación de bases de datos y tablas, así como inserciones de valores en las mismas.

En primer lugar, se procede a crear la base de datos de ejemplo. Para ello, se hará uso del comando:

```
CREATE DATABASE <basededatos>;
```

Donde se sustituirá el parámetro **basesdedatos** por el nombre de la base de datos deseada. De esta forma:

```
postgres=# CREATE DATABASE pract1;
CREATE DATABASE
```

Para comprobar que la base de datos ha sido creada se procederá a hacer uso del comando `\l`.

```
postgres=# \l
                                  List of databases
   Name    |  Owner   | Encoding | Collate |  Ctype  |   Access privileges
-----------+----------+----------+---------+---------+-----------------------
 postgres  | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 pract1    | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 template0 | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
           |          |          |         |         | postgres=CTc/postgres
 template1 | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
           |          |          |         |         | postgres=CTc/postgres
(4 rows)

postgres=# \l pract1
                            List of databases
  Name  |  Owner   | Encoding | Collate |  Ctype  | Access privileges
--------+----------+----------+---------+---------+-------------------
 pract1 | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
(1 row)
```

En el ejemplo anterior se muestran dos usos del comando `\l`, donde en el primero se muestran todas las bases de datos creadas. En el segundo ejemplo, el comando se encuentra seguido del nombre de la base de datos a listar.

A continuación, se procederá a crear una tabla dentro de la base de datos creada con anterioridad. Para ello, se ha de seleccionar en primer lugar la base de datos a editar. Para ello, se hace uso del comando anteriormente detallado `\c <nombre_basededatos>`.

Una vez seleccionada, se procede a crear la tabla. Para ello, se hará uso de la sintaxis siguiente:

```
CREATE TABLE <nombre_tabla> (
    <nombre_atributo> <tipo_atributo>
    . . .
    . . .

);
```

Una vez vista la sintaxis, se procede a mostrar el ejemplo:

```
postgres=# \c pract1
You are now connected to database "pract1" as user "postgres".
pract1=# CREATE TABLE usuarios (
pract1(# nombre varchar(30),
pract1(# clave varchar(10)
pract1(# );
CREATE TABLE
pract1=# \d usuarios
                  Table "public.usuarios"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 nombre | character varying(30) |           |          |
 clave  | character varying(10) |           |          |
```

En la imágen anterior se aprecia el uso del comando `\d <nombre_tabla>`, el cuál mostrará toda la información referente a la misma.

Una vez creada la tabla con todos los atributos deseados, se procederá a la introdución de datos en la misma. Para llevar a cabo esta tarea, se hará uso del comando:

```
INSERT INTO <nombre_tabla>(<columna1>, <columna2>, ...) VALUES (<valor1>,
<valor2>, ...);
```

Se mostrará a continuación ejemplo llevado a cabo sobre la tabla **usuarios**

```
pract1=# INSERT INTO usuarios(nombre,clave) values('Isa','asdf');
INSERT 0 1
pract1=# INSERT INTO usuarios(nombre,clave) values('Pablo','jfx344');
INSERT 0 1
pract1=# INSERT INTO usuarios(nombre,clave) values('Ana','tru3fal');
INSERT 0 1
```

Y para mostrar el contenido de la tabla, haciendo uso de SQL:

```
pract1=# SELECT *
pract1-# FROM usuarios;
 nombre |  clave
--------+---------
 Isa    | asdf
 Pablo  | jfx344
 Ana    | tru3fal
(3 rows)
```

Referencias

1. Cheat Sheet Markdown
2. ¿Qué es PostgreSQL?
3. Enlace Wikipedia Michael Stonebraker
4. Insert PSQL
5. Commonly used commands PSQL

VOLVER AL INICIO