INGENIERÍA UNIVERSIDAD DE LOS ANDES MÉRIDA VENEZUELA | DEPARTAMENTO DE COMPUTACIÓN ESCUELA DE SISTEMAS

Laboratorio de Base de Datos
Práctica Nro. 1
Instalaciones y Configuraciones

Prof. Solazver Solé
Prep. Yenifer Ramírez
Semestre B-2018

1. Instalación de PostgreSQL

PostgreSQL[1] es un poderoso sistema de gestión de bases de datos (SGBD) objeto-relacional de código abierto. Durante el curso se utilizará en todas las prácticas que utilicen el modelo de datos relacional. A continuación se explica la instalación y configuración de conexiones de este sistema.

1.1. Proceso de instalación

Se debe ejecutar el siguiente comando para realizar la instalación:[2]

```
root@pc:/# apt-get install postgresql-x.x
```

donde x.x corresponde a la versión de Postgresql que será instalada, al día de hoy la última versión de este proyecto es la 10.5.[3]

Después que se instala el SGBD Postgresql es recomendable cambiar la contraseña del usuario 'postgres' que se crea durante la instalación, esto se logra de la siguiente manera:

```
root@pc:/# passwd postgres
```

Para acceder al servidor de BD cambie el usuario actual a 'postgres' y luego acceda al servidor:

[1] http://www.postgresql.org
[2] Para Sistemas derivados de Arch Linux https://walkingsources.blogspot.com/2013/07/instalar-postgres-en-manjaro-linux-086.html
[3] Si ya tienes instalado PostgreSQL puedes verificar tu versión con psql –version

```
root@pc:/# su postgres
```

```
postgres@pc:/$ psql postgres
```

Se debe cambiar la contraseña del usuario 'postgres' en el SGBD PostgreSQL:

```
postgres=# ALTER ROLE postgres PASSWORD 'CONTRASENA';
```

Para salir del servidor PostgreSQL:

```
postgres=# \q
```

Para salir del usuario postgres:

```
postgres@pc:/$ exit
```

1.2. Configuración de Acceso Local

La configuración del acceso que tendrán los usuarios localmente se encuentra en un archivo que podemos modificar:

```
root@pc:/# gedit /etc/postgresql/x.x/main/postgresql.conf
```

Busque la siguiente línea y ponga el valor de 'localhost' a la variable listen_addresses, debe quedar de la siguiente manera:

```
#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------

# - Connection Settings -

listen_addresses = 'localhost'          # what IP address(es) to listen on;
```

También vamos a modificar el siguiente archivo:

```
root@pc:/# gedit /etc/postgresql/x.x/main/pg_hba.conf
```

En la siguiente línea si la última columna tiene como valor 'peer' se sustituye con la palabra 'password', debe quedar de la siguiente manera:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
# "local" is for Unix domain socket connections only

  local   all             all                                     password
```

Para que los cambios que hemos hecho en la configuración tengan efecto se debe reiniciar el servicio de PostgreSQL con el siguiente comando:

```
root@pc:/# service postgresql restart
```

1.3. Configuración de Acceso Remoto

Para habilitar el acceso remoto a clientes PostgreSQL desde otras máquinas se deben aplicar las siguientes configuraciones:

1. Se tiene que cambiar el archivo de configuración del servidor PostgreSQL, para esto se debe ejecutar el siguiente comando:

```
root@pc:/# gedit /etc/postgresql/x.x/main/postgresql.conf
```

2. Se necesita encontrar la linea listen_addresses = 'localhost' esta linea la debemos modificar, principalmente se tienen dos opciones:

   a) Colocar el siguiente valor para permitir la comunicación desde cualquier dirección IP

   ```
   listen_addresses = '*'
   ```

   b) Especificar las direcciones IP que tendrán acceso al servidor, esto se logra de la siguiente manera:

   ```
   listen_addresses='192.168.56.124 192.168.56.123'
   ```

3. Modificar el archivo de configuración del cliente PostgreSQL, esto se logra con la siguiente instrucción:

   ```
   root@pc:/# gedit /etc/postgresql/x.x/main/pg_hba.conf
   ```

   En este archivo se pueden configurar los computadores de una red que pueden acceder a los datos almacenados en el servidor PostgreSQL y los usuarios que tendrán acceso. Se debe agregar debajo de la linea "# IPv4 local connections:" la siguiente instrucción:

   ```
   # IPv4 local connections:
     host nombreBaseDatos  usuarioPostgresql  192.168.2.3/32   md5
   ```

   donde 'nombreBaseDatos' y 'usuarioPostgresql' es el nombre de la base de datos y el usuario de PostgreSQL y 'md5' es el método de envío de la contraseña del usuario PostgreSQL por la red.

Para que los cambios realizados tengan efecto se reinicia el servicio de PostgreSQL con el siguiente comando:

```
root@pc:/# service postgresql restart
```

1.4. Creación de un Usuario

Es necesario acceder al usuario postgres:

```
root@pc:/# su postgres
```

Se crea al nuevo usuario indicando los privilegios que tendrá:

```
postgres@pc:/$ createuser -D -S -R -l usuario
```

Se debe asignar una contraseña para que acceda al servidor PostgreSQL:

```
postgres@pc:/$ psql postgres
```

Desde el servidor se asigna una contraseña cifrada al nuevo usuario:

```
postgres=# ALTER USER usuario WITH ENCRYPTED PASSWORD '123456';
```

Se puede consultar los usuarios para observar si tienen contraseña asignada:

```
postgres=# SELECT usename, passwd FROM pg_shadow;
```

Para salir del servidor postgreSQL:

```
postgres=# \q
```

1.5. Creación de la Base de Datos

Es necesario acceder al usuario postgres:

```
root@pc:/# su postgres
```

Se crea la base de datos con el siguiente comando:

```
postgres@pc:/$ createdb -Ttemplate0 -O usuario -EUTF-8 bd_b2018
```

Se accede al servidor postgresql:

```
postgres@pc:/$ psql postgres
```

Se asignan los privilegios correspondientes al usuario para manejar la base de datos recién creada:

```
postgres=# GRANT ALL PRIVILEGES ON DATABASE bd_b2018 TO usuario;
```

1.6. Operaciones con la Base de Datos

Se debe conectar a la base de datos, utilizando el siguiente comando:

```
user@pc:/$ psql -d bd_b2018 -U usuario
```

Para verificar que todo funciona, se puede crear la siguiente tabla de ejemplo con la cual realizaremos un par de operaciones, supongamos que queremos guardar la cédula, el nombre y el apellido de unas cuantas personas, para crear esa tabla se debe ejecutar la siguiente instrucción:

```
postgres=# CREATE TABLE persona (ci varchar (15), nombre varchar (40), apellido varchar (40));
```

Luego se puede probar insertando un par de registros en la tabla, para eso se ejecutan la instrucciones siguientes:

```
postgres=#  INSERT INTO persona VALUES ('23456234','Andres','Iniesta');
```

```
postgres=#  INSERT INTO persona VALUES ('18764123','Jordi','Alba');
```

Para consultar los registros recién insertados, se ejecuta:

```
postgres=# SELECT * FROM persona;
```

Para desconectarse del servidor, se ejecuta:

```
postgres=# \q
```

1.7. Conexión remota en Postgresql

Para conectarse de manera remota a la base de datos se utiliza el siguiente comando:

```
user@pc:/$ psql -h ipServidor -U miUsuario -d miBD
```

2. Instalación de SQLite

SQLite[4] es un SGBD relacional monousuario, contenido en una relativamente pequeña (aprox. 275 kiB) *biblioteca* escrita en lenguaje **C**. A diferencia de los SGBD cliente-servidor (como Postgresql), el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo.

[4] http://www.sqlite.org

2.1. Proceso de instalación

Para instalar se debe ejecutar el siguiente comando:

```
root@pc:/# apt-get install sqlite3
```

2.2. Operaciones con SQLite3

Para utilizar SQLite3, primero se debe crear un directorio y acceder a este, luego se procede a crear un archivo donde estará contenida nuestra base de datos, esto se realiza con la siguiente instrucción:

```
user@pc:/$ sqlite3 bd_b2018.db
```

Para verificar que todo funciona correctamente, se puede tomar el ejemplo anterior de la tabla persona, entonces se procede a crear la tabla:

```
sqlite> CREATE TABLE persona (ci varchar (15), nombre varchar (40), apellido varchar (40));
```

Probemos insertando un par de registros en la tabla, para eso se ejecuta la siguiente instrucción:

```
sqlite>  INSERT INTO persona VALUES ('23456234','Maria Altagracia','Iniesta');
```

```
sqlite>  INSERT INTO persona VALUES ('18764123','Tahi','Alba');
```

Para consultar los registros recien insertados, se ejecuta:

```
sqlite> SELECT * FROM persona;
```

SQlite utiliza una serie de comandos llamados comandos de punto(**dot-commands**). Para visualizar cada uno de los comandos y su función utilizamos:

```
sqlite> .help
```

Para observar la configuración por defecto de **SQlite** utilizamos el comando:

```
sqlite> .show
```

Listing 1: comando show

```
sqlite> .show
echo: off
eqp: off
explain: auto
headers: on
mode: list
nullvalue: ""
output: stdout
colseparator: "|"
rowseparator: "\n"
stats: off
width:
```

Para abrir una base de datos existente utilizamos el comando:

```
sqlite> .open FILENAME
```

Para ejecutar un archivo SQL en la base de datos utilizamos el comando **.read**.

```
sqlite> .read FILENAME.sql
```

Probemos ejecutando los comandos el archivo **insert-data.sql** en la base de datos **bd_a2017.db**, luego una vez mas, debemos consultar la base de datos.

```
sqlite> .read insert-data.sql
```

Ahora abrimos una nueva base de datos ubicada en la carpeta ~**/bd sqlite** donde se encuentra la base de datos de nombre **chinook.db**

```
sqlite> .open chinook.db[5]
```

Para ver todas las bases de datos que se han cargado utilizamos el comando:

```
sqlite> .databases
```

```
seq  name             file
---  -----  ----------------------------------------
0    main    /home/yeniferr/Documents/DB/chinook.db
```

Para observar las tablas de la base de datos utilizamos el comando **.tables**

```
sqlite> .tables
```

Listing 2: Tablas de la base de datos chinook.db

```
sqlite> .tables
albums          employees       invoices        playlists
artists         genres          media_types     tracks
customers       invoice_items   playlist_track
```

Para darle formato a las consultas en SQlite podemos utilizar los siguientes comandos:

```
sqlite> .header on|off
```

```
sqlite> .mode csv|column|html|line|list|tabs|others ...
```

```
sqlite> .timer on|off
```

Listing 3: Ejemplo de salida sin formato

```
sqlite> select * from albums;
1|For Those About To Rock We Salute You|1
2|Balls to the Wall|2
3|Restless and Wild|2
4|Let There Be Rock|1
5|Big Ones|3
6|Jagged Little Pill|4
7|Facelift|5
8|Warner 25 Anos|6
9|Plays Metallica By Four Cellos|7
10|Audioslave|8
```

Listing 4: Ejemplo de salida con formato

```

sqlite> select * from albums;
AlbumId  Title                                ArtistId
----     ---------------------------------    ----------
1        For Those About To Rock We Salute You  1
2        Balls to the Wall                      2
3        Restless and Wild                      2
4        Let There Be Rock                      1
5        Big Ones                               3
6        Jagged Little Pill                     4
7        Facelift                               5
8        Warner 25 Anos                         6
9        Plays Metallica By Four Cellos         7
```

Para enviar la salida a un archivo utilizamos el comando:

```
sqlite> .output FILENAME
```

Como ejemplo puedes enviar la salida de la consulta anterior a un **archivo.txt** ejecutando el siguiente comando y consultando la base de datos nuevamente.

```
sqlite> .output salida.txt
```

Para restablecer la salida la configuramos a la salida estandar **stdout**

```
sqlite> .output stdout
```

Si queremos observar el *esquema* de la base de datos usamos el comando:

```
sqlite> .schema
```

Finalmente, para salir de la cónsola de SQLite, se ejecuta:

```
sqlite> .quit
```

POSTGRESQL 8.3 PSQL CHEAT SHEET

psql is located in the bin folder of the PostgreSQL install and PgAdmin III install.

This is psql 8.3.5, the PostgreSQL interactive terminal.

Usage: **psql** [OPTIONS]... [DBNAME [USERNAME]]

General options:

-c COMMAND	run only single command (SQL or internal) and exit
-d, --dbname=NAME	specify database name to connect to (default: "*logged in username here*")
-f, --file=FILENAME	execute commands from file, then exit
--help	show this help, then exit
-l, --list	list available databases, then exit
-v NAME=VALUE	set psql variable NAME to VALUE
--version	output version information, then exit
-X	do not read startup file (~/.psqlrc)

Interactive Console:

TYPE: \copyright	for distribution terms
\h for help with SQL commands	for help with SQL commands
\? for help with psql commands	for help with psql commands
\g or terminate with semicolon to execute query	or terminate with semicolon to execute query
\q to quit	to quit
GENERAL:	
\c[onnect] [DBNAME|- USER|- HOST|- PORT|-]	connect to new database
\cd [DIR]	change the current working directory
\encoding [ENCODING]	show or set client encoding
\h [NAME]	help on syntax of SQL commands, * for all commands
\set [NAME [VALUE]]	set internal variable, or list all if no parameters
\timing	toggle timing of commands (currently off)
\unset NAME	unset (delete) internal variable
\prompt [TEXT] NAME	prompt user to set internal variable
\! [COMMAND]	execute command in shell or start interactive shell
QUERY BUFFER:	
\e [FILE]	edit the query buffer (or file) with external editor
\g [FILE]	send query buffer to server (and results to file or |pipe)
\p	show the contents of the query buffer
\r	reset (clear) the query buffer
\w FILE	write query buffer to file
INPUT/OUTPUT:	
\echo [STRING]	write string to standard output
\i FILE	execute commands from file
\o [FILE]	send all query results to file or |pipe
\qecho [STRING]	write string to query output stream (see \o)
INFORMATIONAL:	
\d [NAME]	describe table, index, sequence, or view
\d{t|i|s|v|S} [PATTERN] (add "+" for more detail)	list tables/indexes/sequences/views/system tables
\da [PATTERN]	list aggregate functions
\db [PATTERN]	list tablespaces (add "+" for more detail)
\dc [PATTERN]	list conversions
\dC	list casts
\dd [PATTERN]	show comment for object
\dD [PATTERN]	list domains
\df [PATTERN]	list functions (add "+" for more detail)
\dF [PATTERN]	list text search configurations (add "+" for more detail)
\dFd [PATTERN]	list text search dictionaries (add "+" for more detail)
\dFt [PATTERN]	list text search templates
\dFp [PATTERN]	list text search parsers (add "+" for more detail)
\dg [PATTERN]	list groups
\dn [PATTERN]	list schemas (add "+" for more detail)
\do [NAME]	list operators
\dl	list large objects, same as \lo_list
\dp [PATTERN]	list table, view, and sequence access privileges
\dT [PATTERN]	list data types (add "+" for more detail)
\du [PATTERN]	list users
\l	list all databases (add "+" for more detail)
\z [PATTERN]	list table, view, and sequence access privileges (same as \dp)
FORMATTING	
\a	toggle between unaligned and aligned output mode
\C [STRING]	set table title, or unset if none
\f [STRING]	show or set field separator for unaligned query output
\H	toggle HTML output mode (currently off)
\pset NAME [VALUE]	set table output option (NAME := {format|border|expanded|fieldsep|footer|null|numericlocale|recordsep |tuples_only|title|tableattr|pager})
\t	show only rows (currently off)
\T [STRING]	set HTML <table> tag attributes, or unset if none
\x	toggle expanded output (currently off)
COPY, LARGE OBJECT	
\copy ...	perform SQL COPY with data stream to the client host
\lo_export LOBOID FILE	LOBOID FILE
\lo_import FILE [COMMENT]	FILE [COMMENT]
\lo_list	
\lo_unlink LOBOID	large object operations

Connection options:

-h, --host=HOSTNAME	database server host or socket directory
-p, --port=PORT	database server port number
-U, --username=NAME	connect as specified database user
-W, --password	force password prompt (should happen automatically)
-e, --exit-on-error	exit on error, default is to continue
-d DBNAME	some database

psql automated shell examples

restore w hole server

```
psql --host=localhost --username=someuser -f /path/to/pgdumpall.sql
```

Run an sql batch script against a database

```
psql -h localhost -U someuser -d somedb -f /path/to/somefile.sql
```

Run an sql batch script against a database and send output to file

```
psql -h localhost -U someuser -d somedb -f /path/to/scriptfile.sql  -o /path/to/outputfile.txt
```

Run a single statement against a db

```
psql -U postgres -d pagila -c "CREATE TABLE test(some_id serial PRIMARY KEY, some_text text);"
```

Output data in html format

```
psql -h someserver -p 5432 -U someuser -d somedb -H -c "SELECT * FROM sometable" -o mydata.html
```

psql Interactive mode

Launch Interactive session

```
psql -h localhost -U postgres -d somedb
```

View help for SELECT * LIMIT

```
\h  SELECT * LIMIT
```

List all tables in db w ith descriptions

```
\dt+
```

List all tables in db w ith **s** in the name

```
\dt *s*
```

Cancel out of MORE screen

```
:q
```

http://www.sqlitetutorial.net/sqlite-sample-database
https://www.sqlite.org/cli.html