Tutorial básico de PostgreSQL

Article · April 2020

CITATIONS
0

READS
6,137

1 author:

Jorge Domínguez Chávez
Universidad Politécnica Territorial del Estado Aragua
68 PUBLICATIONS **42** CITATIONS

SEE PROFILE

REPÚBLICA BOLIVARIANA DE VENEZUELA
MINISTERIO DEL PODER POPULAR PARA LA EDUCACION UNIVERSITARIA
UNIVERSIDAD POLITÉCNICA TERRITORIAL DEL ESTADO ARAGUA
"FEDERICO BRITO FIGUEROA"
LA VICTORIA- ESTADO ARAGUA
Departamento de Informática

Paradigmas de programación
Trayecto I, Fase I

Nombre y sección: _____
Profesor: _____

Fecha: _____

1. Tutorial básico de PostgreSQL

Para iniciar nuestro trabajo en PosgreSQL ejecutamos:

```
jorge@jodocha:~$ su
Contraseña:
root@jodocha:/home/jorge# su postgres
postgres@jodocha:/home/jorge$ psql
psql (9.6.17)
Digite 'help' para obtener ayuda.

postgres=#
```

Empezamos ingresando la clave del root y presionamos enter, luego su postgres y nuevamente enter. Por último, escribimos psql y presionamos enter, listos para trabajar...!!!.

2. Creando una base de datos

Debemos crear una base de datos, por lo tanto necesitamos crearla.

```
postgres=# CREATE DATABASE zoologico;
psql (9.6.17)
Digite 'help' para obtener ayuda.
```

Para utilizar la base de datos escribimos:

```
postgres=# \c zoologico y presionamos enter
```

Ahora está conectado a la base de datos 'zoologico'con el usuario 'postgres'.

```
zoologico=#
```

Bajo el sistema operativo Unix, los nombres de las bases de datos son sensibles al uso de mayúsculas y minúsculas (no como las palabras clave de SQL), por lo tanto tendremos cuidado de escribir correctamente el nombre de la base de datos. También se aplica para los nombres de las tablas.

Al crear una base de datos, ésta no se selecciona de manera autómatica; debemos hacerlo explícitamente, por ello usamos el comando \c en el ejemplo anterior.

La base de datos se crea sólo una vez, pero deberemos seleccionarla cada vez que iniciamos una sesión con psql para trabajar con ella. Por ello, es recomendable que indique la base de datos sobre la que vamos a trabajar al momento de invocar al cliente psql. Así:

```
shell>psql -h localhost -U posgres -d zoologico
Contraseña para usuario posgres:
```

o escriba la sentencia donde 'jorgeés superusuario:

```
psql -h localhost -U jorge -d zoologico
psql (9.6.17)
conexión SSL (protocolo: TLSv1.2, cifrado: ECDHE-RSA-AES256-GCM-SHA384,
    bits: 256, compresión: desactivado)
Digite 'help' para obtener ayuda.

zoologico=#
```

Observe que "zoologico" no es la contraseña que se está proporcionando desde la línea de comandos, sino el nombre de la base de datos a la que accedemos.

```
Opciones de conexión:
-h, --host=NOMBRE    nombre del anfitrión o directorio de socket
(por omisión: /var/run/postgresql)
-p, --port=PUERTO    puerto del servidor (por omisión: 5432)
-U, --username=NOMBRE
nombre de usuario  (por omisión: jorge)
-w, --no-password    nunca pedir contraseña
-W, --password       forzar petición de contraseña
(debería ser automático)
```

Sin embargo, escribir nuestra contraseña desde la línea de comandos no es recomendado, ya que es inseguro.

3. Creando una tabla

Crear la base de datos es la parte más fácil, pero en este momento la base de datos está vacía, como lo indica el comando \dt:

```
zoologico=#  \dt
No se encontraron relaciones.
zoologico=#
```

La parte un tanto complicada es decidir la estructura que tendrá nuestra base de datos: qué tablas se necesitan y qué columnas estarán en cada tabla.

En principio, necesitamos una tabla que contenga un registro para cada una de nuestras mascotas. Ésta puede ser una tabla llamada mascotas, y debe contener por lo menos el nombre de cada uno de nuestros animalitos. Ya que el nombre en sí no es interesante, la tabla debe contener alguna otra información. Como si más de una persona tiene más de una mascota, es probable que guardemos la información acerca de quien es el dueño de cada mascota. Así mismo, también es interesante contar con alguna información más descriptiva tal como la especie, y el sexo de cada mascota.

¿Y que sucede con la edad?. Esto también es de interés, pero no es una buena idea almacenar este dato en la base de datos. La edad cambia conforme pasa el tiempo, lo cual significa que debemos de actualizar los registros frecuentemente. En vez de esto, es una mejor idea guardar un valor fijo, tal como la fecha de nacimiento. Entonces, cuando necesitemos la edad, la calculamos como diferencia entre la fecha actual y la fecha de nacimiento. Estas funciones las proporciona psql para hacer operaciones entre fechas, así que no hay problema.

Al almacenar la fecha de nacimiento en lugar de la edad tenemos otras ventajas:

Usamos la base de datos para tareas tales como generar recordatorios para cada cumpleaños próximo de las mascotas. Calculamos la edad en relación a otras fechas que la fecha actual.

Si almacenamos la fecha en que murió la mascota en la base de datos, es fácil calcular que edad tenía el animalito cuando falleció. Es probable que estemos pensando en otro tipo de información que sería igualmente útil en la tabla "mascotas", pero será suficiente por ahora contar con información de nombre, propietario, especie, nacimiento y fallecimiento.

Usaremos la sentencia CREATE TABLE para indicar como estarán conformados los registros de nuestras mascotas.

```
zoologico=# create table mascotas(
zoologico(# nombre varchar(20),
zoologico(# propietario varchar(60),
zoologico(# especie varchar(20),
zoologico(# sexo char(1),
zoologico(# nacimiento date,
zoologico(# fallecimiento date);
CREATE TABLE
zoologico=#
```

VARCHAR es una buena elección para los campos nombre, propietario, y especie, ya que los valores que almacena son de longitud variable. No es necesario que la longitud de estas columnas sea la misma, ni tampoco que sea de 20. Se puede especificar cualquier longitud entre 1 y 255, lo que se considere más adecuado. Si resulta que la elección de la longitud de los campos hecha no resultó adecuada, psql proporciona una sentencia ALTER TABLE que nos ayuda a solventar este problema.

El campo sexo es representado en una variedad de formas, "m" y "f", o tal vez "masculino" y "femenino", aunque resulta más simple la primera opción.

El uso del tipo de dato DATE para los campos nacimiento y fallecimiento resulta obvio.

Ahora que hemos creado la tabla, la sentencia $\backslash dt$ produce:

```
zoologico=# \dt
          Listado de relaciones
 Esquema |  Nombre  | Tipo  |  Dueño
---------+----------+-------+----------
 public  | mascotas | tabla | postgres
(1 fila)

zoologico=#
```

o utilizamos:

```
zoologico=# \dt+
                      Listado de relaciones
 Esquema |  Nombre  | Tipo  |  Dueño   | Tamaño  | Descripción
```

```
---------+----------+-------+----------+---------+-------------
 public  | mascotas | tabla | postgres | 0 bytes |
(1 fila)

zoologico=#
```

Para verificar que la tabla fué creada como esperabamos, usaremos la sentencia $\backslash d$:

```
zoologico=# \d mascotas
                Tabla 'public.mascotas'
    Columna    |          Tipo          | Modificadores
---------------+------------------------+---------------
 nombre        | character varying(20) |
 propietario   | character varying(60) |
 especie       | character varying(20) |
 sexo          | character(1)          |
 nacimiento    | date                  |
 fallecimiento | date                  |

zoologico=#
```

Hacemos uso de la sentencia $\backslash d$ mascotas en cualquier momento, si olvidamos los nombres o el tipo de las columnas en la tabla.

4. Cargando datos en una tabla

Después de crear la tabla, incorporamos algunos datos en ella, para lo cual usaremos las sentencias INSERT y COPY.

Supongamos que los registros de las mascotas son descritos por los datos mostrados en la siguiente tabla.

```
Nombre Propietario especie    Sexo Nacimiento  Fallecimento
Fluffy Arnoldo     Gato        f   1999-02-04
Mau    Juan        Gato        m   1998-03-17
Buffy  Arnoldo     Perro       f   1999-05-13
FanFan Benito      Perro       m   2000-08-27
Kaiser Diana       Perro       f   1998-08-31
Chispa Omar        Ave         m   1998-09-11
Wicho  Tomás       Ave         m   2000-02-09
Skim   Benito      Serpiente   f   2001-04-29
```

El archivo debe estar así:

```
Fluffy,Arnoldo,Gato,f,1999-02-04
Mau,Juan,Gato,m,1998-03-17
Buffy,Arnoldo,Perro,f,1999-05-13
FanFan,Benito,Perro,m,2000-08-27
Kaiser,Diana,Perro,f,1998-08-31
Chispa,Omar,Ave,m,1998-09-11
Wicho,Tomás,Ave,m,2000-02-09
Skim,Benito,Serpiente,f,2001-04-29
```

Observamos que psql recibe las fechas en el formato YYYY-MM-DD, que es diferente a lo acostumbrado.

Ya que tenemos una tabla vacía, la manera más fácil de poblarla es crear un archivo de texto que contenga un registro por línea para cada uno de los animalitos para que posteriormente carguemos el contenido del archivo en la tabla únicamente con una sentencia.

Por tanto, creamos un archivo de texto "mascotas.txt" que contiene un registro por línea con valores separados por tabuladores, cuidando que el orden de las columnas sea el mismo que utilizamos en la sentencia CREATE TABLE. Para valores que no conozcamos usamos nulos (NULL).

5. El archivo mascotas.sql

Para cargar el contenido del archivo en la tabla mascotas, usamos el siguiente comando:

```
zoologico=# COPY mascotas(nombre,propietario,especie,sexo,nacimiento)
   from '/home/postgresql_labpostgresql_lab/mascotas.txt' USING
   DELIMITERS ',';
COPY 8
zoologico=# select * from mascotas;
 nombre | propietario |  especie  | sexo | nacimiento | fallecimiento
--------+-------------+-----------+------+------------+---------------
 Fluffy | Arnoldo     | Gato      | f    | 1999-02-04 |
 Mau    | Juan        | Gato      | m    | 1998-03-17 |
 Buffy  | Arnoldo     | Perro     | f    | 1999-05-13 |
 FanFan | Benito      | Perro     | m    | 2000-08-27 |
 Kaiser | Diana       | Perro     | f    | 1998-08-31 |
 Chispa | Omar        | Ave       | m    | 1998-09-11 |
 Wicho  | Tomás       | Ave       | m    | 2000-02-09 |
 Skim   | Benito      | Serpiente | f    | 2001-04-29 |
(8 filas)

zoologico=#
```

o sin nombres de columnas, queda:

```
zoologico=# COPY mascotas from '/home/postgresql_labpostgresql_lab/
   mascotas.txt' USING DELIMITERS ',';
```

En la sentencia COPY especificamos cuál es el separador de columnas, y el de registros, por omisión el tabulador es el separador de columnas (campos), y el salto de línea es el separador de registros, que en este caso son suficientes para que la sentencia COPY lea correctamente el archivo "mascotas.txt".

Si lo que desea es añadir un registro a la vez, entonces hacemos uso de la sentencia INSERT. En la manera más simple, debemos proporcionar un valor para cada columna en el orden en el cual fueron listados en la sentencia CREATE TABLE. Supongamos que Diana compra un nuevo hamster nombrado Pelusa. Usamos la sentencia INSERT para agregar su registro en la base de datos.

```
zoologico=# INSERT INTO mascotas VALUES('Pelusa','Diana','Hamster','f',
   '2000-03-30',NULL);
INSERT 0 1
zoologico=#
```

Note que los valores de cadenas y fechas deben estar encerrados entre comillas. También, con la sentencia INSERT insertamos el valor NULL directamente para representar un valor nulo, un valor que no conocemos.

De este caso, debemos ser capaces de ver que es un poco más la tarea que se tiene que realizar si

inicialmente cargamos los registros con varias sentencias INSERT en lugar de una única sentencia COPY.

6. Recuperando información de una tabla

La sentencia SELECT es usada para obtener la información guardada en una tabla. La forma general de esta sentencia es:

```
SELECT LaInformaciónQueDeseamos FROM DeQueTabla WHERE Condició
    nASatisfacer
```

Aquí, LaInformaciónQueDeseamos es la información que queremos ver. Esta puede ser una lista de columnas, o un * para indicar "todas las columnas". DeQueTabla indica el nombre de la tabla de la cual vamos a obtener los datos. La claúsula WHERE es opcional. Si está presente, la CondiciónASatisfacer especifica las condiciones que los registros deben satisfacer para que puedan ser mostrados.

7. Seleccionando todos los datos

La manera simple de la sentencia SELECT es cuando se recuperan todos los datos de una tabla:

```
zoologico=# select * from mascotas;
 nombre | propietario |  especie  | sexo | nacimiento | fallecimiento
--------+-------------+-----------+------+------------+---------------
 Fluffy | Arnoldo     | Gato      | f    | 1999-02-04 |
 Mau    | Juan        | Gato      | m    | 1998-03-17 |
 Buffy  | Arnoldo     | Perro     | f    | 1999-05-13 |
 FanFan | Benito      | Perro     | m    | 2000-08-27 |
 Kaiser | Diana       | Perro     | f    | 1998-08-31 | 1997-07-29
 Chispa | Omar        | Ave       | m    | 1998-09-11 |
 Wicho  | Tomás       | Ave       | m    | 2000-02-09 |
 Skim   | Benito      | Serpiente | f    | 2001-04-29 |
 Pelusa | Diana       | Hamster   | f    | 2000-03-30 |
(9 filas)

zoologico=#
```

Esta forma del SELECT es útil si deseamos ver los datos completos de la tabla; para asegurarnos de que están todos los registros después de la carga de un archivo.

En este caso que estamos tratando, al consultar los registros de la tabla, nos damos cuenta de que hay un error en el archivo de datos (mascotas.txt): parece que Kaiser ha nacido después de que ha fallecido!. Al revisar el pedigree de Kaiser encontramos que la fecha correcta de nacimiento es el año 1989, no 1998.

Hay por lo menos un par de maneras de solucionar este problema:

Editar el archivo "mascotas.txt" para corregir el error, eliminar los datos de la tabla mascotas con la sentencia DELETE, y cargar los datos nuevamente con el comando COPY:

```
zoologico=# delete from mascotas;
DELETE 9
zoologico=#
```

```
zoologico=# select * from mascotas;
 nombre | propietario |  especie  | sexo | nacimiento | fallecimiento
--------+-------------+-----------+------+------------+---------------
 Fluffy | Arnoldo     | Gato      | f    | 1999-02-04 |
 Mau    | Juan        | Gato      | m    | 1998-03-17 |
 Buffy  | Arnoldo     | Perro     | f    | 1999-05-13 |
 FanFan | Benito      | Perro     | m    | 2000-08-27 |
 Kaiser | Diana       | Perro     | f    | 1998-08-31 |
 Chispa | Omar        | Ave       | m    | 1998-09-11 |
 Wicho  | Tomás       | Ave       | m    | 2000-02-09 |
 Skim   | Benito      | Serpiente | f    | 2001-04-29 |
 Pelusa | Diana       | Hamster   | f    | 2000-03-30 |
(9 filas)

zoologico=#
```

Sin embargo, si hacemos esto, debemos ingresar los datos de Pelusa, la mascota de Diana. La segunda opción consiste en corregir sólo el registro erróneo con una sentencia UPDATE:

```
zoologico=# UPDATE mascotas SET nacimiento='1989-08-31' WHERE nombre='
   Kaiser';
UPDATE 1
zoologico=#

zoologico=# select * from mascotas;
 nombre | propietario |  especie  | sexo | nacimiento | fallecimiento
--------+-------------+-----------+------+------------+---------------
 Fluffy | Arnoldo     | Gato      | f    | 1999-02-04 |
 Mau    | Juan        | Gato      | m    | 1998-03-17 |
 Buffy  | Arnoldo     | Perro     | f    | 1999-05-13 |
 FanFan | Benito      | Perro     | m    | 2000-08-27 |
 Chispa | Omar        | Ave       | m    | 1998-09-11 |
 Wicho  | Tomás       | Ave       | m    | 2000-02-09 |
 Skim   | Benito      | Serpiente | f    | 2001-04-29 |
 Pelusa | Diana       | Hamster   | f    | 2000-03-30 |
 Kaiser | Diana       | Perro     | f    | 1989-08-31 |
(9 filas)

zoologico=#
```

Como se mostró, es fácil recuperar los datos de una tabla completa. Pero no deseamos hacer esto, particularmente cuando las tablas son demasiado grandes. En vez de ello, estaremos más interesados en responder preguntas particulares, en cuyo caso debemos especificar algunas restricciones para la información que deseamos ver.

8. Seleccionando registros particulares

Para seleccionar sólo registros particulares de una tabla o verificar el cambio que hicimos a la fecha de nacimiento de Kaiser, seleccionamos sólo el registro de Kaiser de la siguiente manera:

```
zoologico=# SELECT * FROM mascotas WHERE nombre='Kaiser';
 nombre | propietario | especie | sexo | nacimiento | fallecimiento
```

```
--------+-------------+---------+------+------------+---------------
 Kaiser | Diana       | Perro   | f    | 1989-08-31 |
(1 fila)

zoologico=#
```

La salida confirma que el año ha sido corregido de 1998 a 1989.

La comparación de cadenas es normalmente no sensitiva, así que especificamos el nombre como ”kaiser”, ”KAISER”, etc. El resultado de la consulta será el mismo.

Además, especificamos condiciones sobre cualquier columna, no sólo el ”nombre”. Si deseamos conocer qué mascotas nacieron después del 2000, tendríamos que usar la columna ”nacimiento”:

```
zoologico=# SELECT * FROM mascotas WHERE nacimiento >= '2000-1-1';
 nombre | propietario |  especie  | sexo | nacimiento | fallecimiento
--------+-------------+-----------+------+------------+---------------
 FanFan | Benito      | Perro     | m    | 2000-08-27 |
 Wicho  | Tomás       | Ave       | m    | 2000-02-09 |
 Skim   | Benito      | Serpiente | f    | 2001-04-29 |
 Pelusa | Diana       | Hamster   | f    | 2000-03-30 |
(4 filas)

zoologico=#
```

También, combinamos condiciones para localizar a los perros hembras:

```
zoologico=# SELECT * FROM mascotas WHERE especie='Perro' AND sexo='f';
 nombre | propietario | especie | sexo | nacimiento | fallecimiento
--------+-------------+---------+------+------------+---------------
 Buffy  | Arnoldo     | Perro   | f    | 1999-05-13 |
 Kaiser | Diana       | Perro   | f    | 1989-08-31 |
(2 filas)

zoologico=#
```

La consulta anterior usa el operador lógico AND. Hay un OR:

```
zoologico=# SELECT * FROM mascotas WHERE especie = 'Ave' OR especie = '
   Gato';
 nombre | propietario | especie | sexo | nacimiento | fallecimiento
--------+-------------+---------+------+------------+---------------
 Fluffy | Arnoldo     | Gato    | f    | 1999-02-04 |
 Mau    | Juan        | Gato    | m    | 1998-03-17 |
 Chispa | Omar        | Ave     | m    | 1998-09-11 |
 Wicho  | Tomás       | Ave     | m    | 2000-02-09 |
(4 filas)

zoologico=#
```

El operador AND y OR son intercambiables. Si hacemos esto, es buena idea usar paréntesis para indicar como deben ser agrupadas las condiciones:

```
zoologico=# SELECT * FROM mascotas WHERE (especie = 'Gato' AND sexo = '
   m') OR (especie = 'Perro' AND sexo = 'f');
 nombre | propietario | especie | sexo | nacimiento | fallecimiento
--------+-------------+---------+------+------------+---------------
```

```
 Mau     | Juan         | Gato     | m     | 1998-03-17 |
 Buffy   | Arnoldo      | Perro    | f     | 1999-05-13 |
 Kaiser | Diana        | Perro    | f     | 1989-08-31 |
(3 filas)

zoologico=#
```

9. Seleccionando columnas particulares

Si no deseamos ver los registros completos de una tabla, usamos los nombres de las columnas en las que estamos interesados separándolas por coma. Si queremos conocer la fecha de nacimiento de nuestras mascotas, seleccionamos la columna ”nombre” y ”nacimiento”:

```
zoologico=# SELECT nombre, nacimiento FROM mascotas;
 nombre | nacimiento
--------+------------
 Fluffy | 1999-02-04
 Mau    | 1998-03-17
 Buffy  | 1999-05-13
 FanFan | 2000-08-27
 Chispa | 1998-09-11
 Wicho  | 2000-02-09
 Skim   | 2001-04-29
 Pelusa | 2000-03-30
 Kaiser | 1989-08-31
(9 filas)

zoologico=#
```

Para conocer quién tiene alguna mascota, usaremos la siguiente consulta:

```
zoologico=# SELECT propietario FROM mascotas;
 propietario
-------------
 Arnoldo
 Juan
 Arnoldo
 Benito
 Omar
 Tomás
 Benito
 Diana
 Diana
(9 filas)

zoologico=#
```

Sin embargo, notamos que la consulta recupera el nombre del propietario de cada mascota, y algunos de ellos aparecen más de una vez. Para minimizar la salida, agregaremos la palabra clave DISTINCT:

```
zoologico=# SELECT DISTINCT propietario FROM mascotas;
 propietario
-------------
```

```
 Diana
 Benito
 Omar
 Tomás
 Arnoldo
 Juan
(6 filas)

zoologico=#
```

Se puede usar una claúsula WHERE para combinar selección de filas con la de columnas. Para obtener la fecha de nacimiento de los perritos y los gatitos, usaremos la siguiente consulta:

```
zoologico=# SELECT nombre, especie, nacimiento FROM mascotas WHERE
   especie = 'Perro' OR especie = 'Gato';
 nombre | especie | nacimiento
--------+---------+------------
 Fluffy | Gato    | 1999-02-04
 Mau    | Gato    | 1998-03-17
 Buffy  | Perro   | 1999-05-13
 FanFan | Perro   | 2000-08-27
 Kaiser | Perro   | 1989-08-31
(5 filas)

zoologico=#
```

10. Ordenando registros

Se debe notar en los ejemplos anteriores que las filas regresadas son mostradas sin orden en particular. Sin embargo, es fácil examinar la salida de una consulta cuando las filas son ordenadas en alguna forma útil. Para ordenar los resultados, tenemos que usar una cláusula ORDER BY.

Aquí aparecen los datos ordenados por fecha de nacimiento:

```
zoologico=# SELECT nombre, nacimiento FROM mascotas ORDER BY nacimiento
   ;
 nombre | nacimiento
--------+------------
 Kaiser | 1989-08-31
 Mau    | 1998-03-17
 Chispa | 1998-09-11
 Fluffy | 1999-02-04
 Buffy  | 1999-05-13
 Wicho  | 2000-02-09
 Pelusa | 2000-03-30
 FanFan | 2000-08-27
 Skim   | 2001-04-29
(9 filas)

zoologico=#
```

En las columnas de tipo caracter, el ordenamiento es ejecutado normalmente de forma no sensitiva, es decir, no hay diferencia entre mayúsculas y minúsculas. Sin embargo, se puede forzar un ordenamiento

sensitivo al usar el operador BINARY.

Para ordenar en orden inverso, debemos agregar la palabra clave DESC al nombre de la columna que estamos usando en el ordenamiento:

```
zoologico=# SELECT nombre, nacimiento FROM mascotas ORDER BY nacimiento
    DESC;
 nombre | nacimiento
--------+------------
 Skim   | 2001-04-29
 FanFan | 2000-08-27
 Pelusa | 2000-03-30
 Wicho  | 2000-02-09
 Buffy  | 1999-05-13
 Fluffy | 1999-02-04
 Chispa | 1998-09-11
 Mau    | 1998-03-17
 Kaiser | 1989-08-31
(9 filas)

zoologico=#
```

Podemos ordenar múltiples columnas. Para ordenar por tipo de animal, y poner al inicio los animalitos más pequeños de edad, usaremos la siguiente consulta:

```
zoologico=# SELECT nombre, especie, nacimiento FROM mascotas ORDER BY
   especie, nacimiento DESC;
 nombre |  especie  | nacimiento
--------+-----------+------------
 Wicho  | Ave       | 2000-02-09
 Chispa | Ave       | 1998-09-11
 Fluffy | Gato      | 1999-02-04
 Mau    | Gato      | 1998-03-17
 Pelusa | Hamster   | 2000-03-30
 FanFan | Perro     | 2000-08-27
 Buffy  | Perro     | 1999-05-13
 Kaiser | Perro     | 1989-08-31
 Skim   | Serpiente | 2001-04-29
(9 filas)

zoologico=#
```

Note que la palabra clave DESC aplica sólo a la columna nombrada que le precede.

11. Salida

Salimos de la sesión con:

```
zoologico=# \q
```

Examen SQL Para realizar este examen deberemos crear la tabla Persona y la tabla Idiomas en la base de datos _prueba.

El script SQL que crea las tablas con sus datos es:

```sql
CREATE TABLE IF NOT EXISTS Idiomas (
Id SERIAL,
Etiqueta varchar(20) NOT NULL,
PRIMARY KEY (Id)
);
-- Contenido de la tabla `idioma`
INSERT INTO Idiomas (Id, Etiqueta) VALUES
(1, 'Francés'),
(2, 'Inglés'),
(3, 'Alemán'),
(4, 'Ruso'),
(5, 'Castellano');
-- --------------------------------------------------------
-- Estructura de la tabla `persona`

CREATE TABLE IF NOT EXISTS Persona (
Id_person SERIAL,
Nombre varchar(20) NOT NULL,
Apellidos varchar(20) NOT NULL,
Edad int(11) NOT NULL,
Id_idioma int(11) NOT NULL,
PRIMARY KEY (Id)
) ;
-- Contenido de la tabla `persona`
INSERT INTO Persona (Id, Nombre, Apellidos, Edad, Id_idioma) VALUES
(1, 'Nanie', 'Morales HonHon', 55, 1),
(2, 'David', 'Manrique Adán', 60, 2),
(3, 'María', 'Malasaña Agora', 35, 3),
(4, 'Roberto', 'Magalán', 23, 1),
(5, 'Manuel', 'Olis De Las Heras', 20, 2),
(6, 'Margarita', 'Germán', 26, 1)
```

12. Examen

Objetivo: Evaluar las competencias descritas en la teoría presentada previamente referente a aspectos básicos de PostgreSQL, identificación y uso de comandos SQL e instrucciones del cliente psql en problemas aplicados a contextos reales.

Instrucciones Generales: Lee tu examen cuidadosamente. Pon atención a los detalles. Llena tu hoja de respuestas (no olvides marcar también la respuesta en tu examen).

- Tienes 2 dias para contestar este examen.
- Puedes utilizar tu computador o laptop o tablet. Si no dispones de alguno de ellos y tiene internet vista http://ideone.com o si no tienes computadora, deberás esperar a que alguien algún compañero termine y entregue su examen, para que te la preste.
- Cuando termines, no olvides entregar enviar tu examen a *zevcha56@gmail.com*.

A continuación escriba las sentenciast SQL sin datos.

Enunciados

1. (1 Puntos) Cree una consulta que muestre los apellidos, el nombre y la edad de las personas mayores de 50 años. (muy fácil)
2. (1.5 Puntos) Cree una consulta que muestre los apellidos, el nombre y la edad de las personas que hablen francés y cuyos apellidos contengan "Ma" (fácil).
3. (1.5 Puntos) Cree una consulta que muestre los apellidos, el nombre y la edad de las tres primeras personas con Nombre en orden alfabético (fácil).
4. (2 Puntos) Cree una consulta que muestre la edad media y el idioma de las personas agrupadas por idioma (dificultad media).
5. (2 Puntos) Cree una consulta que muestre la edad media redondeada a la unidad de las personas agrupadas por idioma que tienen la cadena de caracteres "es" contenida en su idioma (dificultad media).
6. (2 Puntos) Cree una consulta que muestre la hora actual si el nombre contiene seis caracteres y la hora actual más una hora en otros casos (dificultad media).
7. (3 Puntos) Cree una consulta que muestre los idiomas que no habla David Manrique Adán (difícil).
8. (3 Puntos) Cree una consulta que muestre las tres primeras letras del nombre concatenadas con un espacio concatenado a su vez con las tres primeras letras de los apellidos de las personas mayores de 30 años. Y además, las tres últimas letras del nombre concatenadas con un espacio concatenado a su vez con las tres últimas letras de los apellidos de las personas menores de 30 años (difícil).
9. (4 Puntos) Cree una consulta que muestre la suma de edades agrupadas por idioma de las personas que hablan los idiomas que no hablan las personas de 18 a 25 años (muy difícil).