Administración de usuarios

Jesús Reyes Carvajal

Esquemas en Postgres

En Postgres, una base de datos contiene uno o más schemas, los cuales contienen tablas. Pero también pueden contener otros tipos de objetos como tipos de datos, funciones y operadores. Los nombres se pueden repetir entre schemas, por ejemplo la tabla "usuario" puede aparecer en dos schemas.

- Los schemas se utilizan generalmente para que diferentes usuarios puedan utilizar una misma base de datos sin interferir entre ellos.
- Se utilizan para organizar los objetos de una base de datos en grupos lógicos que permitan mejorar su administración
- Evitar conflictos de nombre con aplicaciones de terceros (poniéndolas en schemas separados); etc.

Listar los esquemas de una base de datos

```
ciclismo=# select nspname from
pg_catalog.pg_namespace;
      nspname
--------------------
 pg_toast
 pg_temp_1
 pg_toast_temp_1
 pg_catalog
 public
 information_schema
```

Listar relaciones del esquema public

```
ciclismo=# \dt public.
              List of relations
 Schema |      Name       | Type  |  Owner
--------+-----------------+-------+----------
 public | ciclistas       | table | postgres
 public | contratos       | table | postgres
 public | equipos         | table | postgres
 public | naciones        | table | postgres
 public | participaciones | table | postgres
 public | pruebas         | table | postgres
(6 rows)
```

Crea esquema tienda **y autoriza a** jhon
CREATE SCHEMA tienda AUTHORIZATION jhon;

Acceder al esquema tienda
SET search_path TO tienda;

Volver al esquema public
SET search_path TO public;

Administración de usuarios.

Una vez terminada la instalación de postgres, este crea un usuario postgres en el sistema operativo y ademas crea un usuario postgres en PostgreSQL, es el primer usuario creado que posee todos los privilegios para crear bases datos y nuevos usuarios.

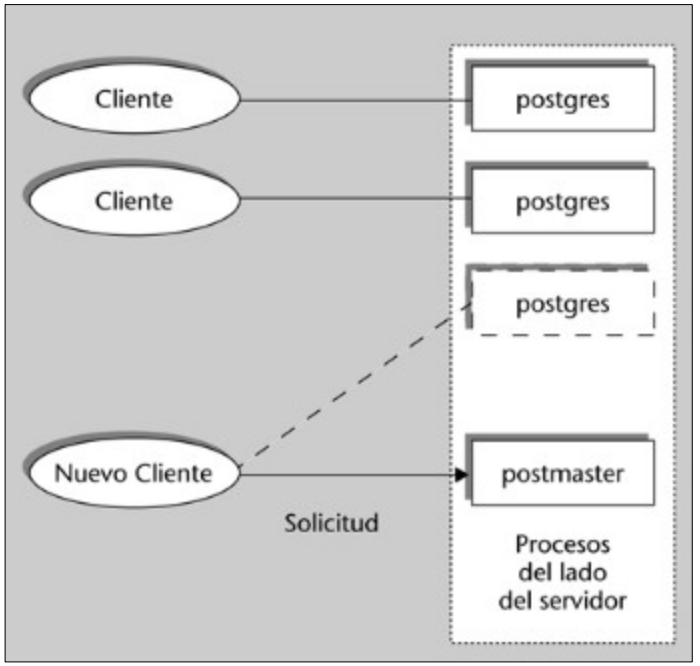
El usuario postgres del sistema operativo no tiene acceso al shell, ni posee una contraseña, por tanto debemos acceder como usuario root del sistema para después acceder como usuario postgres y realizar tareas en su nombre.

Arquitectura Cliente/servidor

El programa servidor se llama **postgres** y los clientes son pgaccess(cliente grafico) y psql(cliente modo texto)

Un proceso servidor postgres solo puede atender a un cliente.

El proceso postmaster es el encargado de ejecutar un nuevo proceso postgres, el cual atenderá a un cliente.



Roles, usuarios y grupos en PostgreSQL

La creación de los usuarios son responsabilidad del DBA(Administrador de la base de datos), como también para la conexión de aplicaciones (web, de escritorio, etc).

Los permisos de acceso en PostgreSQL se administran definiendo roles. Dentro de este concepto se incluyen tanto los usuarios como los grupos. La diferencia principal entre ambos es que, mientras el rol de tipo usuario se usa para acceder y trabajar con la base de datos, los grupos agregan a usuarios y definen permisos sobre esquemas y objetos de la estructura.

Es fundamental indicar que los usuarios en Postgres son independientes de los usuarios creados en el sistema operativo. Por ejemplo, al instalar PostgreSQL se crea por defecto el rol usuario postgres con un perfil de superusuario o administrador y como tal en el sistema operativo se crea un usuario postgres.

Crear un rol usuario

Para crear un nuevo rol de tipo usuario podemos usar tanto CREATE ROLE como CREATE USER. La diferencia entre ambos es que si usamos la opción de USER añadimos automáticamente el atributo LOGIN. Este atributo es el que nos va a servir para indicar si es un rol usuario o un rol de grupo.

Los roles pueden ser(LOGIN, SUPERUSER, CREATEDB, CREATEROL…) que van a definir sus privilegios.

Así se crea un nuevo usuario:

```
CREATE ROLE juan WITH
ENCRYPTED PASSWORD '123456'
LOGIN
VALID UNTIL 'INFINITY';
```

Oh también así:

```
CREATE USER juan WITH PASSWORD '123456';
```

En este caso el usuario juan ya tiene el rol de LOGIN por defecto

Ejemplo de roles

```
CREATE ROLE alice
LOGIN
PASSWORD 'securePass1';
```

```
CREATE ROLE john
SUPERUSER
LOGIN
PASSWORD 'securePass1';
```

```
CREATE ROLE dba
CREATEDB
LOGIN
PASSWORD 'Abcd1234';
```

```
CREATE ROLE dev_api WITH
LOGIN
PASSWORD 'securePass1'
VALID UNTIL '2030-01-01';
```

```
CREATE ROLE api
LOGIN
PASSWORD 'securePass1'
CONNECTION LIMIT 1000;
```

GRANT & REVOKE

Los Sistemas Gestores de Base de Datos Relacionales implementan una serie de comandos SQL que permiten al administrador controlar el acceso a los objetos de una Base de Datos.

Los comandos GRANT & REVOKE, podemos otorgar o revocar privilegios a uno o más objetos.

Los objetos sobre los cuales podemos otorgar o revocar privilegios son los siguientes:

- Tablas semaforo-grant
- Columnas
- Vistas
- Tablas externas
- Secuencias
- Base de datos
- Contenedor de datos externos (FDW)
- Servidor externo
- Funciones
- Procedimientos
- Lenguaje de programación
- Esquemas
- Espacio de tablas

Sintaxis:
GRANT privileges ON object TO user;
REVOKE privileges ON object FROM user;

Ejemplos:

Crea el usuario jtorres
postgres=# CREATE USER jtorres WITH PASSWORD 'alguna clave' ;

Muestra el listado de usuarios
postgres=#\du

Lista los privilegios otorgados sobre objetos.
postgres=#\dp

```
Schema  |  Name        | Type  |     Access privileges     | Column access privileges
---------+--------------+-------+---------------------------+-------------------------
 public   | equipos        | table | jtorres=r/postgres                |
```

jtorres=r significa que tiene permiso de lectura sobre la tabla equipos

Estos son los posibles permisos:

- a: Permiso de INSERT o append
- r: Permiso de SELECT o read
- w: Permiso de UPDATE o write
- d: Permiso de DELETE o delete

Asignación de Privilegios de Superuser
El usuario que los reciba puede ejecutar comandos DDL (Data Definition Language), es decir podrá crear, modificar y eliminar objetos en la base de datos, por lo que se sugiere no asignar este nivel tan alto de permisos a un usuario común Por lo general este privilegio lo posee el personal DBA.

postgres=# ALTER USER jtorres SUPERUSER ;

Mas ejemplos:

Creación de un Usuario de solo lectura (Read Only User)
Es un caso muy común que se requiera un usuario con solo lectura, por lo general es utilizado por aplicaciones de minería de datos, es decir que solo necesitan extraer data por medio de consultas.

postgres=# ALTER USER jtorres SET default_transaction_read_only = on;

Asigna permiso de conexión al usuario:
GRANT CONNECT ON DATABASE nombre_BD TO jtorres;

Permiso de acceso al esquema en donde están las tablas que el usuario necesita consultar:
GRANT USAGE ON SCHEMA nombre_esquema TO jtorres;

Asignación de permiso SELECT en una tabla especifica:
GRANT SELECT ON nombre_tabla TO nombre_usuario ;

Asignación de permiso SELECT en todas las tablas de un esquema:
GRANT SELECT ON ALL TABLES IN SCHEMA nombre_esquema TO nombre_usuario

Asigna todos los permisos a una tabla
GRANT ALL ON nombre_table TO nombre_usuario;

Revoca todos los permisos a una tabla
REVOKE ALL ON nombre_table FROM nombre_usuario;

Revoca el permiso de borrado y actualización del usuario jtorres en la tabla productos;
REVOKE DELETE, UPDATE ON productos FROM jtorres;

CREACIÓN DE GRUPOS

La creación de grupos de usuarios hacen la administración mucho más fácil. Cada grupo puede incluir usuarios. Cada usuario puede llegar a pertenecer a uno o más grupos. Cuando se otorga o revocan privilegios para un objeto, se puede identificar un usuario específico o un grupo de usuarios.

Cada usuario es automáticamente miembro de un grupo PUBLIC. PUBLIC es realmente un grupo virtual, no puede agregar o remover miembros y no puede borrar este grupo, pero se permite asociar privilegios con PUBLIC.

CREATE GROUP name [[WITH] option [...]]

Donde option puede ser:

```
    SUPERUSER | NOSUPERUSER
  | CREATEDB | NOCREATEDB
  | CREATEROLE | NOCREATEROLE
  | INHERIT | NOINHERIT
  | LOGIN | NOLOGIN
  | [ ENCRYPTED ] PASSWORD 'password'
  | VALID UNTIL 'timestamp'
  | IN ROLE role_name [, ...]
  | IN GROUP role_name [, ...]
  | ROLE role_name [, ...]
  | ADMIN role_name [, ...]
  | USER role_name [, ...]
  | SYSID uid
```

Nota: Los usuarios ya deben estar creados cuando se quieren asociar a un grupo.

Ejemplos:

Se puede asignar un miembro a un grupo de varias maneras:

- Usando la opción IN GROUP en el comando CREATE USER
- Usando el comando ALTER GROUP

CREATE GROUP desarrolladores USER Bernardo, lety;

Adicionar o remover un miembro de un grupo:
ALTER GROUP groupname [ADD|DROP] USER username [, ...]

Ver los grupos creados:
template1=# SELECT * FROM pg_group;

Ver los miembros de un grupo:
template1=#select usename from pg_user, (select grolist from pg_group where groname = 'ventas') as groups where usesysid = ANY(grolist);

Borrar un grupo:
template1=#DROP GROUP desarrolladores;

Ejemplo de creación de un usuario y conexión a una base de datos sobre tabla especifica

$su postgres
postgres$psql -U postgres -d template1
template1=#create user juan with password ‘juan’; //Ya tiene el rol de LOGIN
template1=#create database tienda with owner juan;
template1=#\c tienda;
tienda=#create table clientes(codigo serial not null primary, nombre varchar(30));
tienda=#gran select on clientes to juan;

Nota: Si necesita que el usuario “juan” ingrese a Postgres por consola a la base de datos tienda, debe primero que todo crear en el sistema operativo al usuario “juan” (adduser juan). Ya con esto puede ingresar al sistema operativo como usuario juan (su juan) y luego ingresar a Postgres (psql -U juan -d tienda)

Para ingresar por HTTP a la base de datos seria:

Archivo: conexiondb.php

```
<?php
      $conn_string = "dbname=tienda host=localhost port=5432 user=juan password=juan";
      $dbconn = pg_connect($conn_string) or die('Error de conexion: ' . pg_last_error());
 ?>
```

Archivo: index.php

```
<?php include('conexiondb.php');
 $query = "SELECT * FROM clientes";
 $query = pg_query($query);
while($row= pg_fetch_array($query,NULL,PGSQL_ASSOC)){
  echo "Codigo: ".$row['codigo']."<br/>";
  echo "Nombre: ".$row['nombre']."<br/>";
}
?>
<html>
<head>
</head>
<body background="#fdfdfd">
<h4>Listado de clientes</h4>
</body>
</html>
```

Desde el browser

http://localhost/app-tiendas/index.php

Nota: Instalar antes en el sistema operativo postgres, apache2 y PHP5 o superior.