Universidad Nacional Autónoma de México
Facultad de Ingeniería
Semestre 2026-2

Bases de Datos

Grupo 01

Ing. Fernando Arreola Franco

Tarea 2

Fecha de entrega: 20 de febrero de 2026

Parra Vera Héctor Jesús

Calificación: _____

1. Requisitos para conectarse a una base de datos

Para establecer una conexión exitosa hacia un servidor de PostgreSQL, se deben cumplir condiciones tanto en la configuración del servidor como en el cliente que realiza la petición. Los elementos necesarios son los siguientes [1]:

- **Configuración del servidor (`pg_hba.conf` y `postgresql.conf`):** Por defecto, PostgreSQL solo escucha conexiones locales. Para acceso remoto, el parámetro `listen_addresses` debe estar configurado en `postgresql.conf`, y el archivo `pg_hba.conf` (Host-Based Authentication) debe tener una regla explícita que permita la dirección IP del cliente.
- **Dirección del servidor y puerto:** La IP o dominio donde reside el SGBD. PostgreSQL utiliza por defecto el puerto lógico `5432`.
- **Credenciales de acceso:** Un rol de PostgreSQL que posea el atributo `LOGIN` y su respectiva contraseña.
- **Nombre de la base de datos:** El catálogo específico al que el rol intentará conectarse.
- **Controlador o adaptador:** Un middleware que permita la comunicación entre el lenguaje de programación y el protocolo nativo de PostgreSQL. Por ejemplo, al desarrollar el backend de una aplicación web con frameworks como Django, se suele utilizar el adaptador `psycopg2` o `psycopg` [2].
- **URI de conexión:** La cadena estandarizada que el driver utiliza para conectarse, típicamente con el formato: `postgresql://usuario:password@host:5432/nombre_bd`.

2. Permisos a nivel sistema y a nivel objeto

El modelo de seguridad en PostgreSQL separa claramente los permisos globales (atributos del rol) de los permisos específicos sobre los objetos de la base de datos [1].

2.1. Permisos a nivel sistema (atributos de rol)

En PostgreSQL, los permisos a nivel sistema no se otorgan con el comando `GRANT`, sino que se definen como atributos al momento de crear o alterar un rol [3]. Estos determinan privilegios administrativos:

- `SUPERUSER`: Otorga control total sobre el clúster, ignorando cualquier otra restricción de permisos.
- `CREATEDB`: Permite al rol crear nuevas bases de datos.
- `CREATEROLE`: Permite crear, modificar o eliminar otros roles.
- `LOGIN`: Permite que el rol pueda iniciar una sesión (lo que tradicionalmente define a un üsuario").
- `REPLICATION`: Otorga permisos para iniciar flujos de replicación física o lógica.

2.2. Permisos a nivel objeto

Son los privilegios específicos (DCL) otorgados sobre las estructuras de datos dentro de una base de datos (tablas, esquemas, vistas, secuencias, funciones). En PostgreSQL, un nivel jerárquico muy importante es el esquema (`SCHEMA`), típicamente `public` por defecto. Los permisos incluyen [4]:

- `USAGE`: Permite acceder a los objetos dentro de un esquema específico.
- `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE`: Operaciones sobre tablas y vistas.
- `EXECUTE`: Permite la ejecución de funciones o procedimientos almacenados.

3. ¿Cómo dar y quitar permisos?

PostgreSQL utiliza los comandos estándar de SQL (`GRANT` y `REVOKE`), pero su jerarquía requiere que primero se otorgue acceso al esquema antes que a las tablas [1].

3.1. Otorgar permisos (`GRANT`)

Para otorgar permisos, primero se debe asegurar el acceso al esquema y luego a los objetos.

```
-- 1. Permitir el uso del esquema
GRANT USAGE ON SCHEMA public TO backend_role;

-- 2. Otorgar permisos sobre una tabla específica
GRANT SELECT, INSERT, UPDATE ON public.empleados TO backend_role;

-- 3. Otorgar permisos sobre todas las tablas actuales (específico de Postgres)
GRANT SELECT ON ALL TABLES IN SCHEMA public TO backend_role;
```

3.2. Revocar permisos (`REVOKE`)

El comando `REVOKE` funciona a la inversa, retirando privilegios específicos.

```
-- Quitar el permiso de inserción y actualización
REVOKE INSERT, UPDATE ON public.empleados FROM backend_role;

-- Quitar todos los privilegios sobre todas las tablas del esquema
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM backend_role;
```

4. Diferencia entre rol y usuario

A diferencia de otros motores de bases de datos, **en PostgreSQL no existe una diferencia interna real entre un rol y un usuario**. A partir de la versión 8.1, PostgreSQL unificó el concepto de usuarios y grupos en una sola entidad llamada Role" [1].
La única diferencia radica en los comandos utilizados como alias por convención:

- **Usuario:** Cuando se ejecuta `CREATE USER nombre;`, PostgreSQL internamente ejecuta `CREATE ROLE nombre LOGIN;`. Es decir, un üsuario.[es] simplemente un rol que tiene habilitado el atributo para iniciar sesión.
- **Rol o grupo:** Cuando se ejecuta `CREATE ROLE nombre;`, se crea la entidad sin el atributo `LOGIN`. Por convención, estos roles se utilizan como "grupos"para agrupar permisos lógicos (ej. `rol_lectura`) y luego asignar estos grupos a los roles que sí pueden iniciar sesión mediante la sintaxis `GRANT rol_lectura TO usuario_app;`.

Referencias

[1] The PostgreSQL Global Development Group, *PostgreSQL 16.0 Documentation*, 2023.

[2] The Psycopg Team, *Psycopg 3 – PostgreSQL database adapter for Python*, 2023.

[3] C. J. Date, *An Introduction to Database Systems*. Addison-Wesley, 8th ed., 2003.

[4] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*. Pearson, 7th ed., 2015.