Tarea 2

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

NOMBRE: Parra Fernandez Hector Emilio

GRUPO: 01

ASIGNATURA: Bases de Datos

FECHA DE ENTREGA:16/2/26

¿Qué requiero para conectarme a una base de datos?

Acceso a una base de datos en el entorno de PostgreSQL, una vez que la base de datos se encuentra estructurada el acceso a esta se da principalmente por 3 vías, estas dependiendo del nivel de abstracción requerido:

1. Interfaz de Línea de Comandos (CLI): Es por medio del uso del terminal interactivo, como lo es psql, el cual permite la ejecución directa de sentencias de SQL. Esta es la herramienta por excelencia de administradores de sistemas debido a tener una latencia baja y un control total sobre el motor

2. Herramientas Gráficas y Protocolos de Conectividad: Existen GUIs como pgAdmin las cuales son utilizadas como herramientas estándares para la conectividad industrial, como lo son.

- JDBC: Es la API estándar para la conexión de aplicaciones Java con motores de bases de datos.
- ODBC: Estandar para el acceso a bases de datos para aplicaciones en C/C++

3. Abstracción mediante Lenguajes de Programación: Permite el desarrollo de aplicaciones que usan librerías especificas para interactuar con la base de datos de forma programática

Interacción con el Terminal psql

El acceso mediante consola requiere la invocación del binario seguido del identificador del esquema:

$ psql nombre_base_de_datos

Identificación de Roles y Privilegios

Al ingresar al terminal, el símbolo del sistema indica el nivel de autoridad del usuario, siendo los siguientes dos:

- Usuario estándar: Sujeto a las políticas de control de acceso y permisos de objeto.
- Superusuario: Posee privilegios totales y no está restringido por los mecanismos de control de acceso.

Comandos de Control y Consultas de Prueba

Dentro de la interfaz interactiva, se distinguen dos tipos de instrucciones:

- Sentencias SQL: Finalizan con punto y coma y son procesadas por el motor de la base de datos
- Meta-comandos de psql: Instrucciones internas de la herramienta que comienzan con barra invertida (\). No requieren punto y coma.

Permisos a nivel Sistema y Objeto

Privilegios de Objeto Son las acciones que un usuario puede realizar dentro de una base de datos

sobre los objetos, abarcando tablas, vistas, secuencias, procedimientos almacenados, funciones, etc.

– SELECT permite leer datos

– INSERT añade nuevas filas

– UPDATE modifica filas existentes

– DELETE elimina filas de una tabla

– REFERENCES crea restricciones de clave foránea referenciadas a una clave especifica

– EXECUTE ejecuta un proceso de almacenado

– ALTER modifica la estructura de un objeto ya sea añadiendo columnas a una tabla por ejemplo

– INDEX crea los índices de una tabla

La granularidad de estos privilegios es alta, lo que significa que puedes conceder, por ejemplo, permiso

para leer solo una tabla específica a un usuario, o permitir que otro usuario solo inserte datos en otra

tabla, sin que pueda ver los datos existentes.

• Privilegios de Sistema Los privilegios de sistema, por otro lado, controlan la capacidad de un usuario

para realizar operaciones que afectan al SGBD en su conjunto o para crear, modificar o eliminar de

objetos, independientemente de la instancia específica del objeto. Estos privilegios son más potentes y

deben asignarse con extrema precaución.

– Create Table crea nuevas tablas en la base de datos

– CREATE USER crea nuevos usuarios

– DROP ANY TABLE eliminar cualquier tabla en la base de datos

– ALTER ANY PROCEDURE Permite modificar cualquier procedimiento almacenado.

– GRANT ANY PRIVILEGE Permite otorgar cualquier privilegio a otros usuarios (el privilegio

definitivo, reservado para administradores).

– SHUTDOWN Permite apagar el servidor de base de datos.

La gestión de privilegios de sistema suele estar restringida a los administradores de base de datos (DBAS) o

usuarios con roles de alta jerarquía, dado el amplio alcance de las acciones que permiten.

¿Como dar y quitar permisos?

La seguridad y la integridad de la información en un sistema de gestión de bases de datos (DBMS) dependen de una administración de identidades robusta. El proceso de concesión y revocación de permisos no se limita a dar acceso a un individuo, sino que se estructura mediante el modelo RBAC (Role-Based Access Control)

1. Gestión de Identidades: Roles y Usuarios

En PostgreSQL, la distinción entre un "usuario" y un "rol" es mínima; técnicamente ambos son roles. La diferencia radica en el atributo de conexión.

- Creación de un Rol de Grupo: Se utiliza como un contenedor de permisos.
  CREATE ROLE nombre_rol NOLOGIN;

- Creación de un Usuario (Con login):
  CREATE USER nombre_usuario;

- Asignación de Jerarquía: Para que un usuario herede los permisos de un rol:
  GRANT nombre_rol TO nombre_usuario;

2. Concesión de Privilegios (GRANT)

La instrucción GRANT permite definir qué acciones puede realizar un rol sobre un objeto específico (como una tabla).

- Sintaxis Académica: GRANT [TIPO_PERMISO] ON [NOMBRE_OBJETO] TO [NOMBRE_ROL];

- Ejemplo práctico: Para permitir que el rol lea los datos de una tabla:
  GRANT SELECT ON sales_data TO reporting_role;

3. Revocación de Privilegios (REVOKE)

La instrucción REVOKE retira formalmente un permiso previamente otorgado, invalidando el acceso de forma inmediata para todos los miembros de ese rol.

- Sintaxis Académica: REVOKE [TIPO_PERMISO] ON [NOMBRE_OBJETO] FROM [NOMBRE_ROL];

- Ejemplo práctico: Para impedir la lectura de datos:
  REVOKE SELECT ON sales_data FROM reporting_role;

4. Verificación y Auditoría en Consola (psql)

Para asegurar que los cambios se han aplicado correctamente, se utilizan los meta-comandos de la interfaz de línea de comandos:

- \du: Lista los roles existentes, sus atributos (si pueden loguearse o no) y de qué grupos son miembros.
- \z o \dp: Muestra los privilegios de acceso detallados sobre las tablas (SELECT, INSERT, UPDATE, etc.).

Diferencias entre rol y Usuario

En PostgreSQL, la distinción entre un usuario y un rol es fundamentalmente técnica y se resume en el atributo de conexión. Técnicamente, ambos son objetos de tipo Role, pero la diferencia práctica reside en el permiso de LOGIN. Cuando hablamos de un usuario, nos referimos a un rol configurado con la capacidad de autenticarse mediante una contraseña para iniciar una sesión en la base de datos. Por el contrario, un rol propiamente dicho suele crearse con el atributo NOLOGIN, funcionando no como una identidad de acceso, sino como un contenedor lógico de permisos diseñado para agrupar facultades de lectura, escritura o administración.

Desde una perspectiva de diseño y arquitectura de seguridad, un usuario representa la identidad de un sujeto, ya sea una persona o un servicio de software, cuya función principal es permitir la auditoría y el control de quién accede al sistema. El rol, en cambio, representa una función o perfil de trabajo dentro de la organización. La mejor práctica dicta que los permisos nunca deben asignarse directamente a la cuenta del usuario, sino al rol de permisos. De esta manera, el usuario actúa como un miembro que hereda las capacidades de uno o varios roles, lo que facilita enormemente el mantenimiento cuando el personal o las aplicaciones cambian sus responsabilidades.

Este enfoque jerárquico permite aplicar de manera eficiente el principio de menor privilegio (least privilege). Mientras que los usuarios administradores poseen privilegios elevados para tareas de mantenimiento, las aplicaciones y servicios deben utilizar identidades limitadas que solo hereden los roles estrictamente necesarios para su operación, como readonly para consultas o readwrite para procesos transaccionales. Esta separación conceptual asegura que, ante un cambio en la estructura de la base de datos, el administrador solo necesite modificar un único rol en lugar de actualizar individualmente cada cuenta de usuario conectada al sistema.

Finalmente, el uso de roles sobre usuarios directos resuelve el problema de la escalabilidad y el desorden en entornos de producción. Al definir roles que agrupan permisos sobre esquemas específicos y tablas futuras, se garantiza que la gestión de accesos sea predecible y segura. En versiones modernas como PostgreSQL 15, esta distinción es aún más crítica para proteger esquemas sensibles como el public, asegurando que solo los roles autorizados puedan crear o modificar objetos, manteniendo así la integridad del ecosistema de datos frente a accesos no supervisados.

Bibliografía

[1] *PostgreSQL Documentation,* 1.4. Accessing a Database, 12 de febrero de 2026, consultado: 15 de febrero de 2026. [Online]. Available:

https://www-postgresql-org.translate.goog/docs/current/tutorialaccessdb.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

[2] *Lab Ex*. Gestión de Roles y Permisos en PostgreSQL, consultado: 15 de febrero de 2026. [Online]. Available: https://labex.io/es/tutorials/postgresql-postgresql-role-and-permission-management-550960

[3] *Medium* PostgreSQL: usuarios, roles y cómo no arruinar tu modelo de permisos, consultado: 15 de febrero de 2026. [Online]. Available:. https://medium.com/@maraclaudiaprezescalante/postgresql-usuarios-roles-y-c%C3%B3mo-no-arruinar-tu-modelo-de-permisos-5be8099f97c0

[4] Ivan, “Privilegios en Bases de Datos: El Control de Acceso — MySQL YA,” 1 2026, consultado: 15 de febrero de 2026. [Online]. Available: https://mysqlya.com.ar/bases-de-datos/que-son-los-privilegios-en-una-base-de-datos/