Roles y Seguridad

Checklist	☑
# Día	3
Estado	Completado
Fecha	@21 de diciembre de 2025
Notas Técnicas	**1 . Roles y nivel de alcance:** En PostgreSQL, los roles son entidades globales al cluster. Sus atributos (LOGIN, SUPERUSER, CREATEDB, CREATEROLE) aplican a todo el cluster, mientras que los permisos se asignan a nivel de base de datos y objetos específicos. **2. Atributos de rol vs permisos:** Los atributos de un rol definen capacidades administrativas (como crear bases de datos o roles) y no deben confundirse con los permisos sobre objetos, los cuales se gestionan mediante GRANT y REVOKE. **3. Herencia de roles:** Asignar un rol a un usuario mediante GRANT no implica que sus privilegios se apliquen automáticamente. Para ello, el rol debe tener el atributo INHERIT o el usuario debe activar el rol con SET ROLE. **4. Uso de psql y prompts:** En psql, el prompt `=#` indica que el cliente está listo para ejecutar comandos, mientras que `-#` significa que una instrucción está incompleta y debe cerrarse o cancelarse antes de continuar.

	5. Buenas prácticas de seguridad: No es recomendable otorgar el atributo SUPERUSER a usuarios de aplicación. Es preferible usar roles específicos, permisos mínimos y control explícito del acceso a schemas y tablas.
Tiempo Invertido	6 horas

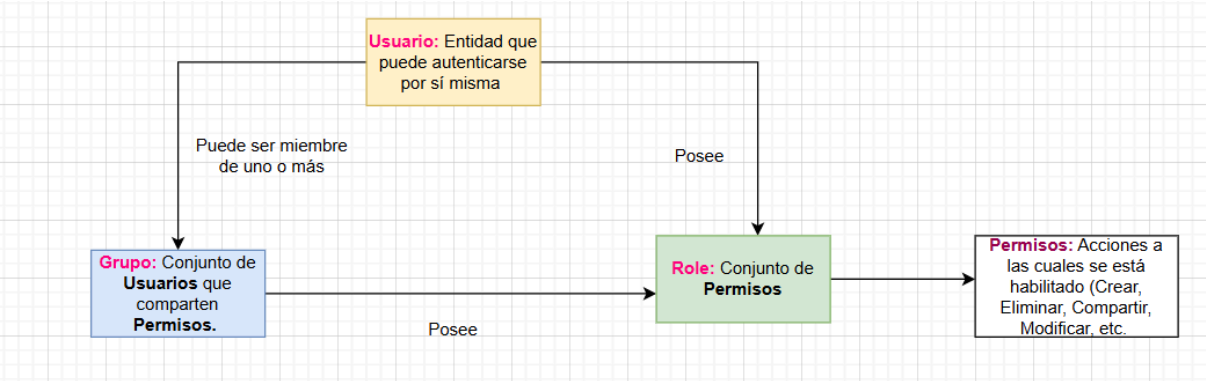
Día 3 – Roles, Usuarios y Permisos en PostgreSQL

🎯 Objetivo del día

Comprender y aplicar el **modelo de seguridad de PostgreSQL**, creando roles y usuarios, asignando permisos correctamente y entendiendo cómo proteger datos en entornos reales.

El **Usuario** se puede entender como un **ROL** con atributo de **Login**, comprendiendo que el **ROL** es una entidad de Seguridad que puede:

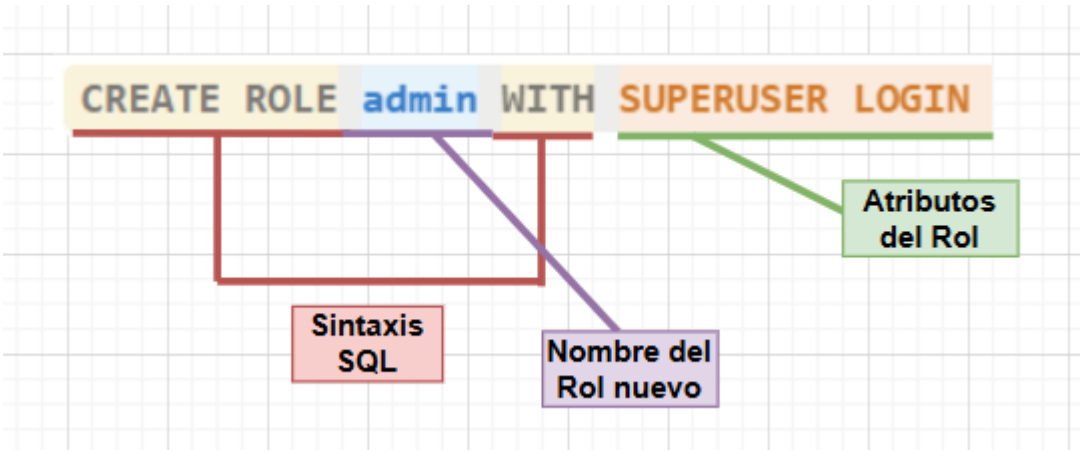
- Representar a una Persona (Usuario)
- Representar una aplicación
- Representar un conjunto de Permisos



Atributos del Rol (Role Attributes):

Los atributos de un Rol, son los que definen las capacidades administrativas de un Rol a nivel de Cluster.

```
CREATE ROLE admin WITH SUPERUSER LOGIN
```



- **SUPERUSER:** Puede sobre escribir todas las clases de acceso a las Bases de Datos, posee un acceso TOTAL al Cluster. (**Default = NOSUPERUSER**)
- **CREATEDB:** Permite la creación de Bases de Datos. (**Default = NOCREATEDB**)
- **CREATEROLE:** Permite la creación de nuevos Roles, además, posee lla capacidad de eliminar o modificar permisos de otros Roles. (**Default = NOCREATEROLE**)
- **INHERIT:** Determina si el nuevo Rol hereda los permisos del grupo al cual pertenece. (**Default = INHERIT)**
- **LOGIN:** Permite que el nuevo Rol pueda hacer Login. (**Default = NOLOGIN**)

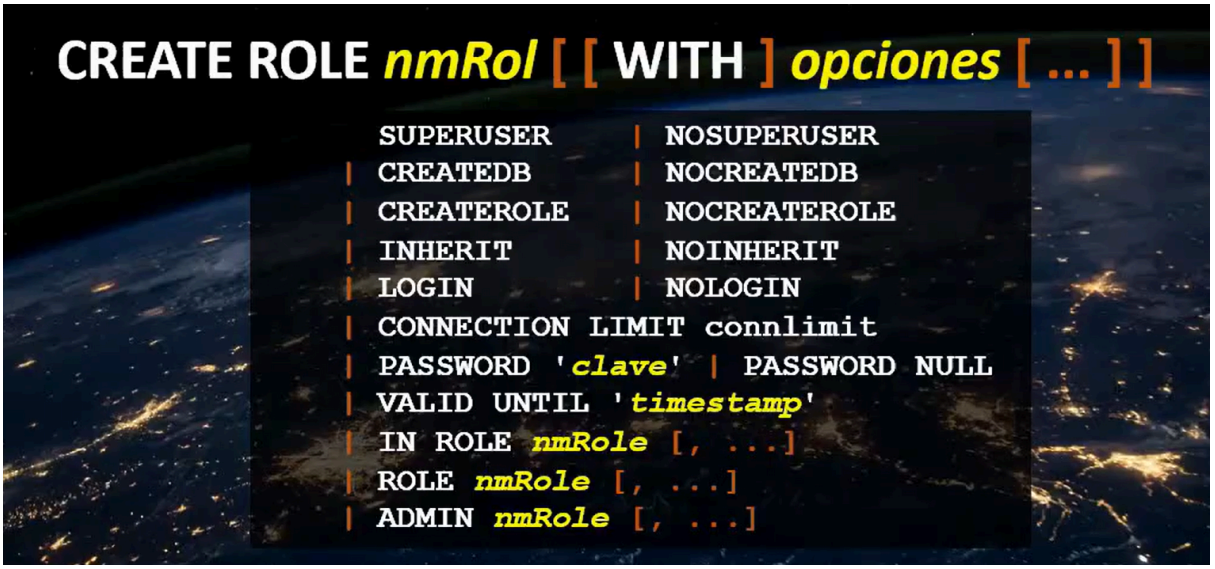
Parámetros de Conexión y Autenticación o Configuraciones del Rol: Estos determinan el comportamiento del Rol y como será su conexión.

- **CONNECTION LIMIT:** Permite configurar el límite de conexiones del Rol, si se establece como -1, signifiga que posee conexiones ilimitadas. (`CONNECTION LIMIT -1`)

- **PASSWORD '.....':** Le asigna una clave de acceso al Rol, para que un Rol pueda tener esta configuración de acceso, necesita poseer el Atributo de Rol **Login**.
- **VALID UNTIL 'aaa-mm-dd hh-mm-ss':** Determina una fecha de expiración para le Password, si este parámetro no se agrega, el Password no vencerá.

Cláusulas de Gestión de Membresía: Definen la relación de los grupos.

- **IN ROLE:** El nuevo Rol será miembro de los Roles listados
- **ROLE:** Los Roles listados en la sentencia serán miembros del nuevo Rol, esta acción hace que el **nuevo Rol** sea un **GRUPO**.
- **ADMIN:** Cumple una función similar a **ROLE**, pero este le concede a los miembros el permiso de también ser administradores del **nuevo Rol**.



En PostgreSQL, los roles pueden tener **atributos administrativos** que definen sus capacidades a nivel del cluster (como **SUPERUSER, CREATEDB, CREATEROLE, LOGIN**) y **opciones de configuración** (como **PASSWORD, CONNECTION LIMIT, VALID UNTIL**).

Estos atributos son distintos de los **permisos sobre objetos**, como Tablas o Schemas, los cuales se gestionan mediante **GRANT** y **REVOKE.**

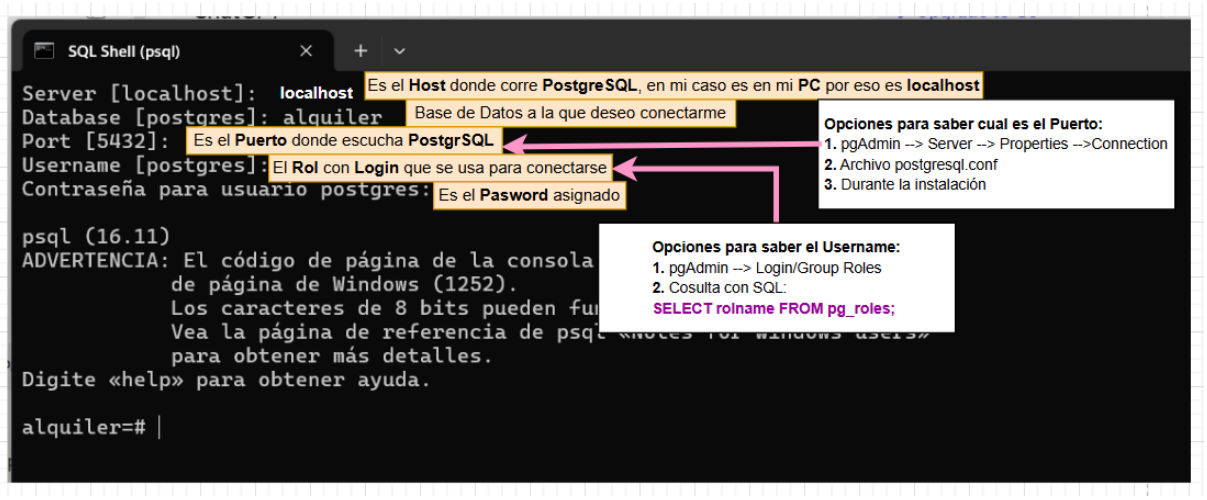
Los principales **Permisos** que hay en PostgreSQL:

- `SELECT`
- `INSERT`
- `UPDATE`
- `DELETE`
- `ALL`

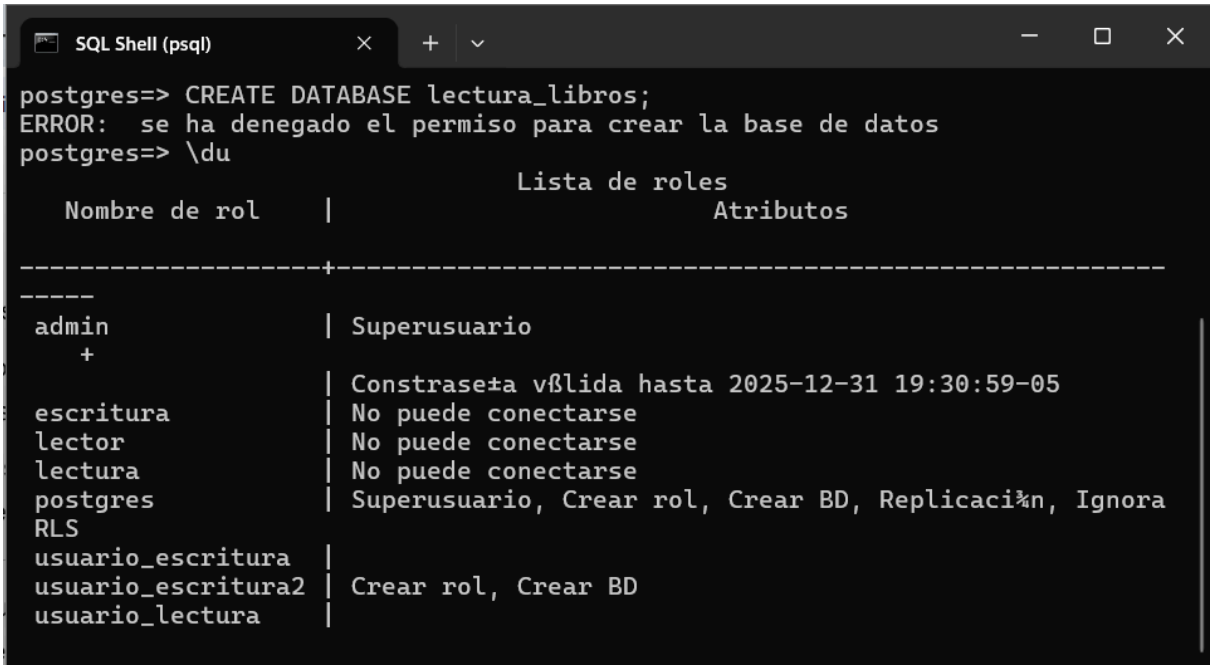
IMPORTANTE: Los **roles** en PostgreSQL son **globales al cluster**, sus **atributos** (LOGIN, SUPERUSER, CREATEDB...) aplican a **todas las bases** y los **permisos** se otorgan por **base de datos, schemas, tablas,** o sea por **objeto** que se defina.

Conexión a la Base de Datos desde la Terminal de SQL Shell, es esta ocación me conecte a la BD creada en el día 2 de estudio Alquiler, pero si no informo la Base de Datos, por defecto se conecta a la BD Postgres

PSQL primero establece la **conexión** al **Servidor** y a la **Base de Datos,** después de eso permite ingresar comandos **SQL.**



Dato curioso: con el comando `\du` en la terminal de PostgrSQL (psql), se puede ver la lista de todos los Roles.



Roles Unificados:

En otros sistemas (como SQL Server o el antiguo Postgres), tienes un **Usuario** para entrar y un **Grupo** para organizar permisos. En PostgreSQL, el sistema solo conoce el objeto `ROLE`.

Un rol puede actuar como usuario, como grupo, o como ambos a la vez, dependiendo de sus **atributos**:

- **Si tiene el atributo `LOGIN`:** Se comporta como un **Usuario** (puede iniciar sesión).
- **Si NO tiene el atributo `LOGIN`:** Se comporta como un **Grupo** (sirve para contener permisos y otros roles).
- **Si tiene miembros:** Se comporta como un **Grupo**.
- **Si es miembro de otro:** Se comporta como un **Usuario** dentro de un grupo.

Comando	Lo que realmente hace el sistema	Resultado Típico
`CREATE USER`	`CREATE ROLE ... WITH LOGIN`	Un rol que puede entrar a la DB.
`CREATE GROUP`	`CREATE ROLE ... NOLOGIN`	Un rol diseñado para agrupar permisos.
`CREATE ROLE`	`CREATE ROLE ... NOLOGIN` (por defecto)	La forma base y flexible.

La verdadera potencia de los roles unificados es la capacidad de crear "árboles" de permisos. Gracias a que son la misma entidad, un rol puede ser "padre" de otro mediante la cláusula `GRANT`.

- **Herencia (`INHERIT`):** Por defecto, si el rol "Juan" es miembro del rol "Ventas", Juan hereda automáticamente todos los permisos de Ventas.
- **Propiedad:** Un rol (ya sea usuario o grupo) puede ser dueño de tablas o bases de datos.

Las ventajas de este modelo son:

1. **Flexibilidad:** Puedes convertir un grupo en un usuario simplemente dándole permiso de `LOGIN`, o viceversa.
2. **Simplicidad:** Solo hay una tabla de sistema (`pg_authid`) para gestionar toda la seguridad del clúster.
3. **Recursividad:** Los grupos pueden estar dentro de otros grupos sin límites complejos, ya que técnicamente todos son la misma clase de objeto.

Autenticación vs Autorización

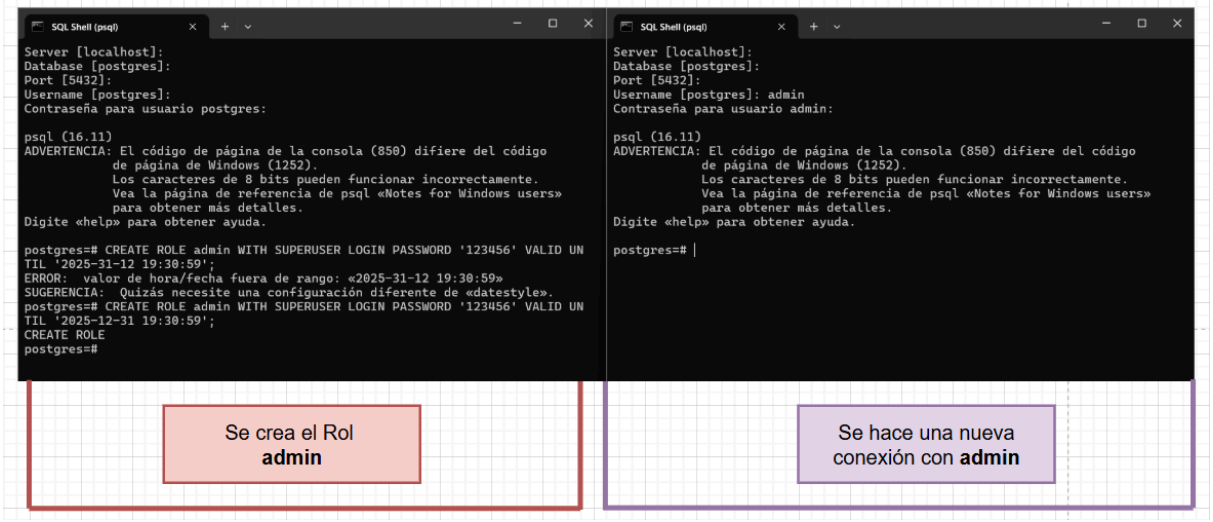
- **Autenticación:** *quién eres*
- **Autorización:** *qué puedes hacer*

🛠️ Actividad práctica:

1️⃣ Crear roles

```
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Contraseña para usuario postgres:

psql (16.11)
ADVERTENCIA: El código de página de la consola (850) difiere del código
             de página de Windows (1252).
             Los caracteres de 8 bits pueden funcionar incorrectamente.
             Vea la página de referencia de psql «Notes for Windows users»
             para obtener más detalles.
Digite «help» para obtener ayuda.

postgres=# CREATE ROLE admin WITH SUPERUSER LOGIN PASSWORD '123456' VALID UNTIL '2025-31-12 19:30:59';
ERROR:  valor de hora/fecha fuera de rango: «2025-31-12 19:30:59»
SUGERENCIA:  Quizás necesite una configuración diferente de «datestyle».
postgres=# CREATE ROLE admin WITH SUPERUSER LOGIN PASSWORD '123456' VALID UNTIL '2025-12-31 19:30:59';
CREATE ROLE
postgres=#
```

Se crea el primer **ROL llamado ADMIN**, el cuál también es un **USUARIO** (porque es una entidad que posee **LOGIN**), pero al momento del su creación se presento un error de sintaxis de la fecha, ya que, PostgreSQL interpreta las fechas según datestyle, por tal motivo espera un formato **YYYY-MM-DD** y se escribió **YYYY-DD-MM,** después de hacer el cambio pertinente el Rol fue creado.



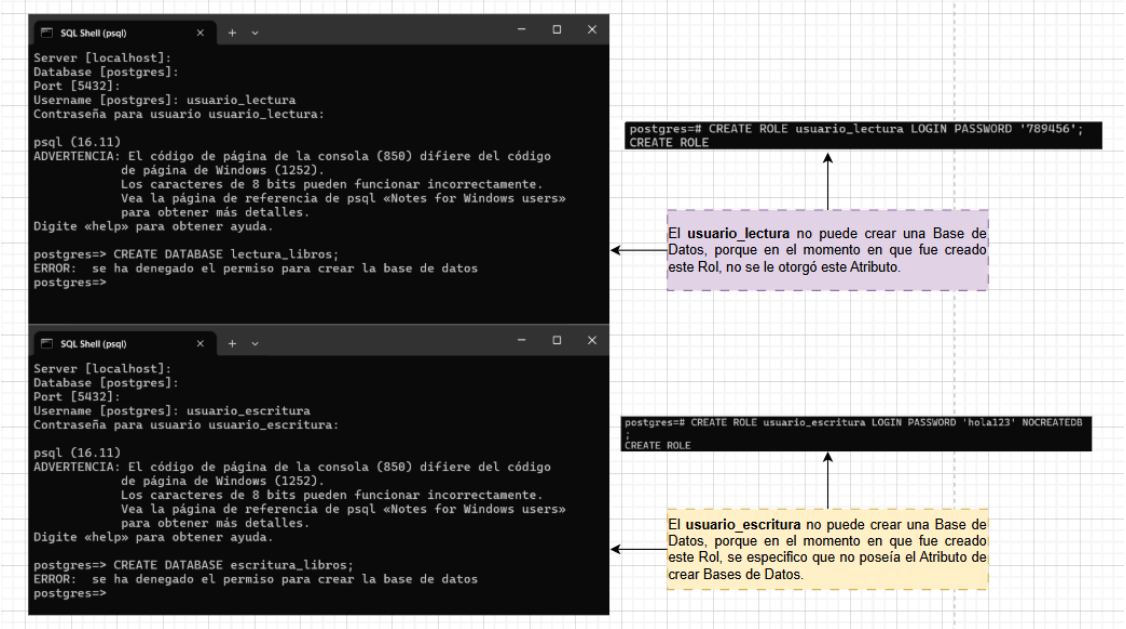

```
-- Rol de solo lectura
CREATE ROLE lector;

-- Rol de escritura
CREATE ROLE escritor;
```

```
postgres=# CREATE ROLE lectura;
CREATE ROLE
postgres=# CREATE ROLE escritura;
CREATE ROLE
postgres=#
```

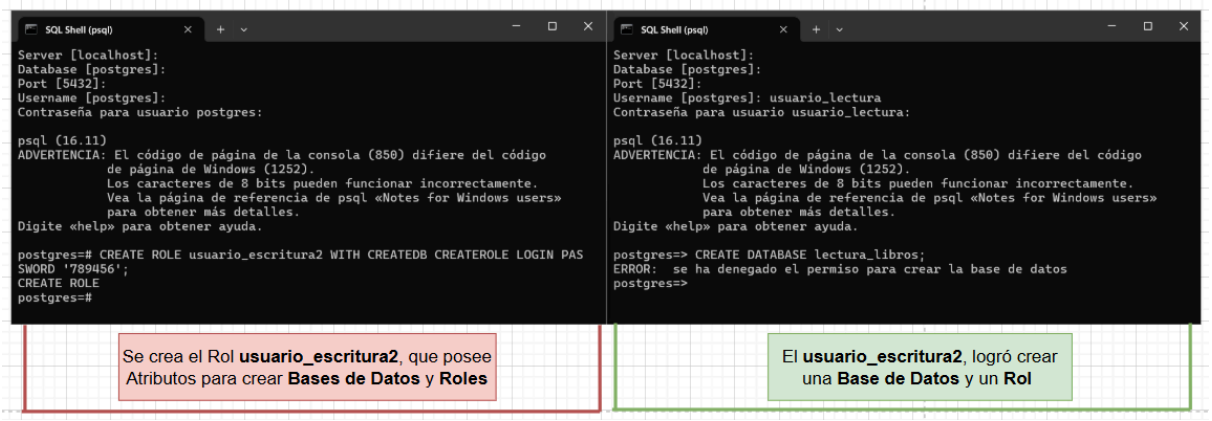
2️⃣ Crear usuarios

```
CREATE ROLE usuario_lectura
LOGIN
PASSWORD'789456';

CREATE ROLE usuario_escritura
LOGIN
PASSWORD'hola123' NOCREATEDB;
```

```
postgres=# CREATE ROLE usuario_lectura LOGIN PASSWORD '789456';
CREATE ROLE
postgres=# CREATE ROLE usuario_escritura LOGIN PASSWORD 'hola123' NOCREATEDB
;
CREATE ROLE
postgres=#
```



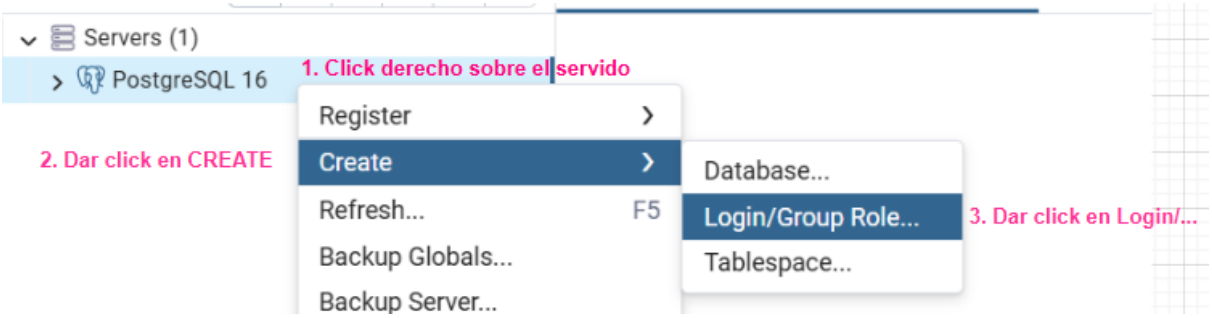

```
--Creación de un Rol (usuario) donde se especifica los Atributos para crear Bases de Datos y Roles
CREATE ROLE usuario_escritura2
WITH CREATEDB CREATEROLE LOGIN PASSWORD '789456';

--El Rol (usuario) crea una Base de Datos y un Rol
CREATE DATABASE libros;
CREATE ROLE lector;
```



Creación de Roles y Usuarios por medio de la herramienta pgAdmin:

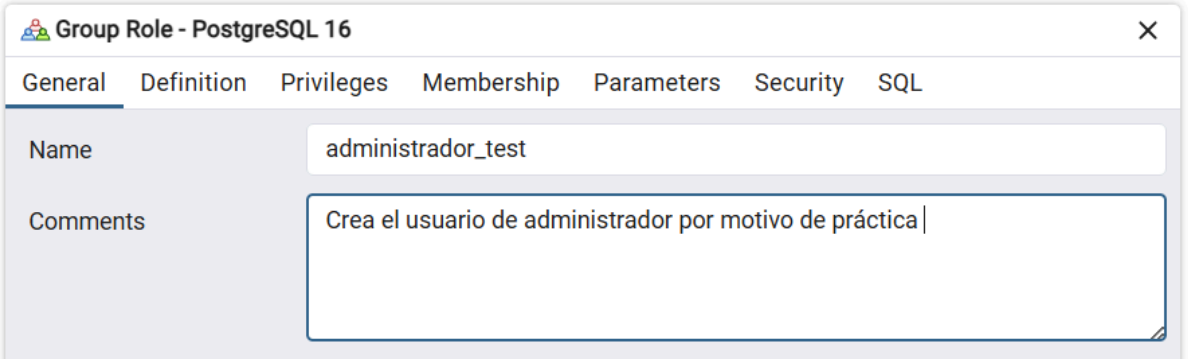
Desde la herramienta de pgAdmin también se pueden crear los Roles y Usuarios, esta es una opción más amigable pues a diferencia de la Terminal está posee una interfaz gráfica.

1. Ubicarse sobre el Servidor
2. Dar clik derecho sobre el Servidor
3. Seleccionar la opción de Create
4. Elegir la opción de Login/Group Role...



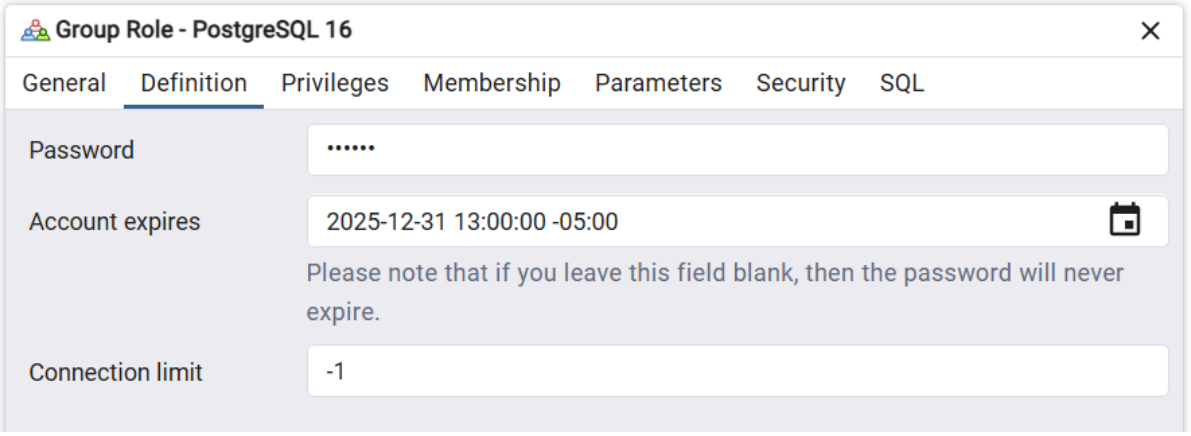
Posteriormente se abrirá una ventana, donde se debe:

5. En la pestaña **General** definir el nombre y agregar un comentario (opcional)

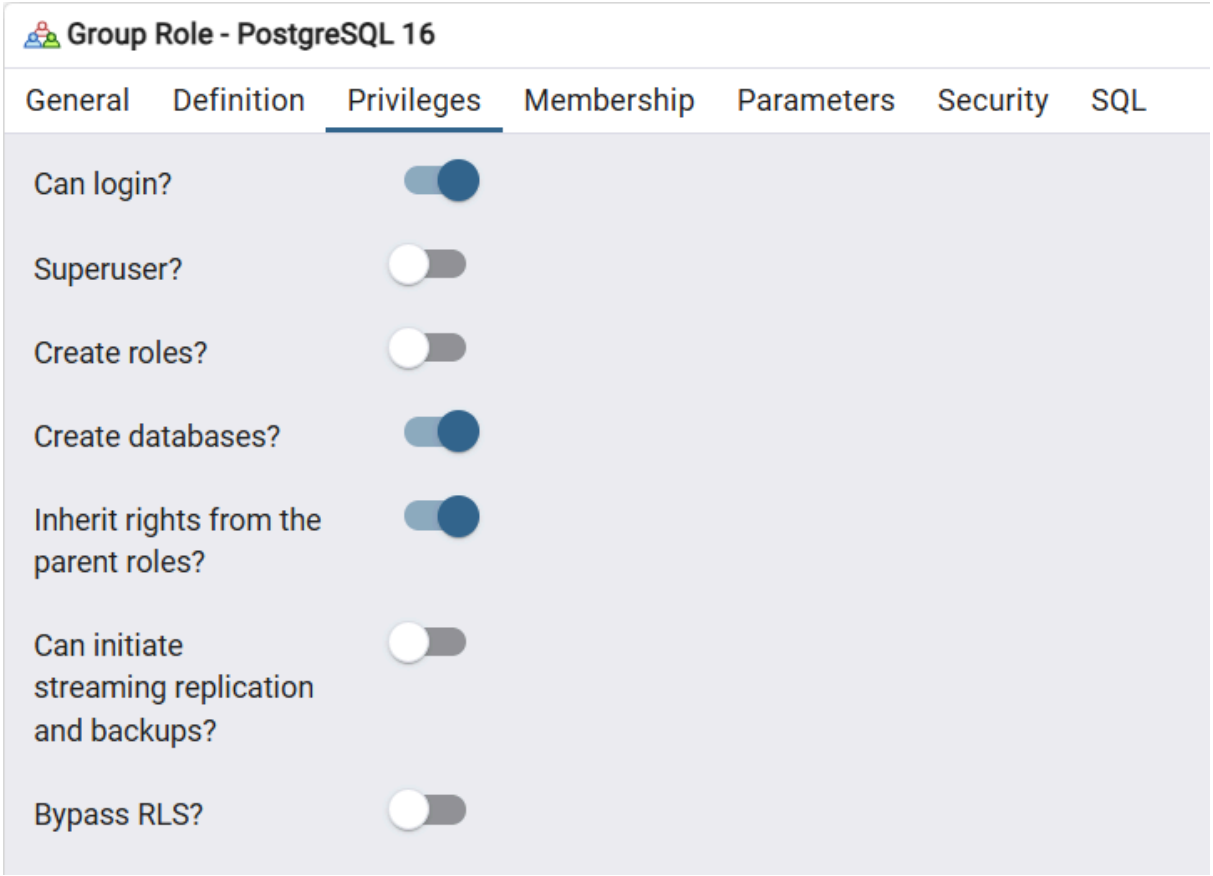


6. Definir las configuraciones de acceso

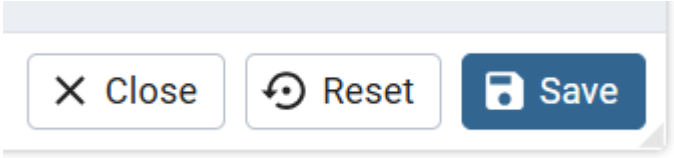
En el ejemplo se agrega un Password, se determina una fecha de vencimiento para la contraseña y se le permite al usuario tener acceso ilimitado



7. Configurar los Privilegios que posee el usuario



8. Guardar la configuración



3️⃣ Asignar roles a usuarios

```
GRANT lectorTO usuario_lectura;
GRANT escritorTO usuario_escritura;
```

Al intentar asignar un Rol al **usuario_escritura2** me sale un error, pues se estaba haciendo desde el **usuario_lectura** y este no cuenta con el privilegio de **CREATEROLE**

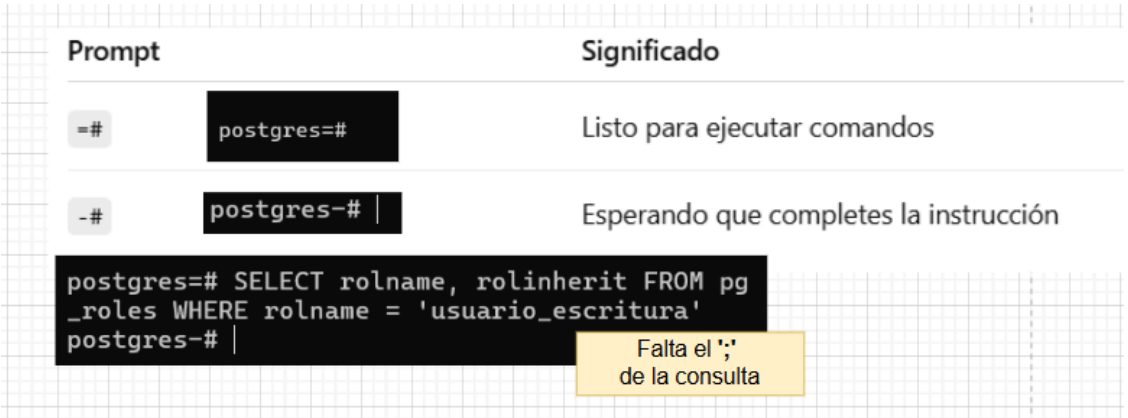
```
postgres=> GRANT admin TO usuario_escritura2;
ERROR:  se ha denegado el permiso para otorgar el rol «admin»
DETALLE:  Sólo los roles con el atributo SUPERUSER pueden otorgar roles con
el atributo SUPERUSER.
postgres=>
```

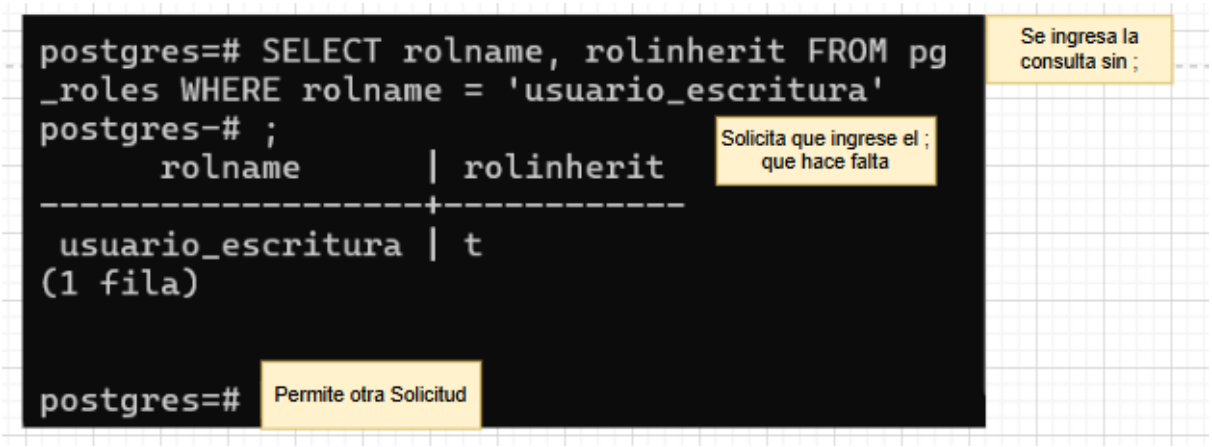
Pero al hacerlo desde el usuario **postgres** si me lo permitió, porque este es un **SUPERUSER** y cuenta con el Privilegio de **CREATEROLE**

```
postgres=# GRANT admin TO usuario_escritura2;
GRANT ROLE
postgres=#
```

Dato curioso:

En psql, el prompt **-#** indica que una instrucción está incompleta; esta se debe cancelar con `Ctrl+C` o completar la sintaxis antes de ejecutar nuevos comandos.





Checklist:

- ~~Estudio del tema~~
- ~~Ejecución de scripts~~
- ~~Evidencia guardada~~
- ~~Notas técnicas escritas~~

Recursos:

- https://www.postgresql.org/docs/current/user-manag.html
- https://youtu.be/JAxgsKy8Q0k?si=o9_UaAieH5DfpoxD
- https://youtu.be/ojEYsse-d2M?si=j0__uYPnmlSYPlrl
-