TCL - DCL

Lenguaje de control de transacciones.

Lenguaje de control de datos.

¿Qué son las transacciones?

Las transacciones en SQL son unidades o secuencias de trabajo realizadas de forma ordenada y separada en una base de datos. Representan cualquier cambio en la base de datos, y tienen dos objetivos principales:

- Proporcionar secuencias de trabajo fiables que permitan poder recuperarse fácilmente ante errores y mantener una base de datos consistente incluso frente a fallos de sistema.
- Proporcionar aislamiento entre programas accediendo a la vez a la base de datos.

Una transacción es una propagación de uno o más cambios en la BD, ya sea cuando se crea, se modifica o se elimina un registro. En la práctica, consiste en la agrupación de consultas SQL y su ejecución como parte de una transacción.

Propiedades de las transacciones

Bajo el acrónimo de **ACID** las transacciones siguen cuatro propiedades básicas:

- ✓ **Atomicidad**: aseguran que todas las operaciones dentro de la secuencia de trabajo se completen satisfactoriamente. Sino es así, la transacción se abandona en el punto del error y se vuelve a su estado inicial.
- ✓ **Consistencia**: aseguran que la BD cambie de estados en una transacción exitosa.
- ✓ **Aislamiento**: permiten que las operaciones sean aisladas y transparentes unas de otras.
- ✓ **Durabilidad**: aseguran que el resultado o efecto de una transacción completada permanezca en caso de error del sistema.

Control de las transacciones

Existen tres comandos básicos de control en las transacciones SQL:

- **BEGIN** para comenzar una transacción.
- **SAVEPOINT** para establecer un punto de guardado.
- **COMMIT** para guardar los cambios.
- **ROLLBACK** para abandonar la transacción y deshacer los cambios que se hubieran hecho en la transacción.

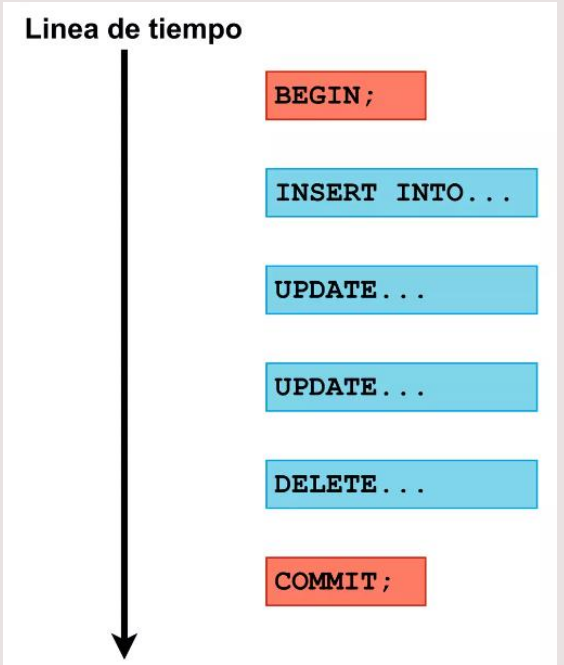
  ROLLBACK TO punto_guardado / ROLLBACK TO SAVEPOINT punto_guardado;

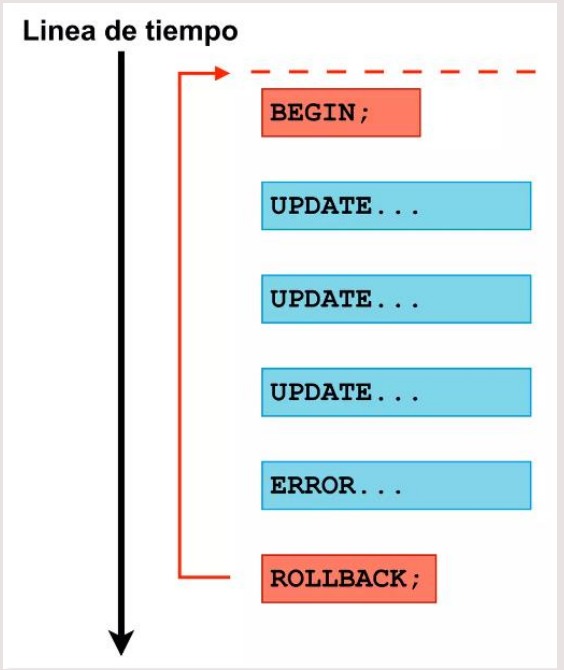
Para poder utilizar estos comandos debemos desactivar la opción **AUTOCOMMIT.** Esta opción está activada por defecto a nivel de cliente.

Ejemplos

```
BEGIN;
INSERT INTO cuentas (n_cuenta, nombre, balance)
VALUES (0679259, 'Pepe', 200);
UPDATE cuentas SET balance = balance - 137.00
WHERE nombre = 'Pepe';
UPDATE cuentas SET balance = balance + 137.00
WHERE nombre = 'Juan';
SELECT nombre, balance FROM cuentas
WHERE nombre = 'Pepe' AND nombre = 'Juan';
ROLLBACK;
COMMIT;
```

```
BEGIN;
"SENTENCIAS SQL"
ROLLBACK;
```

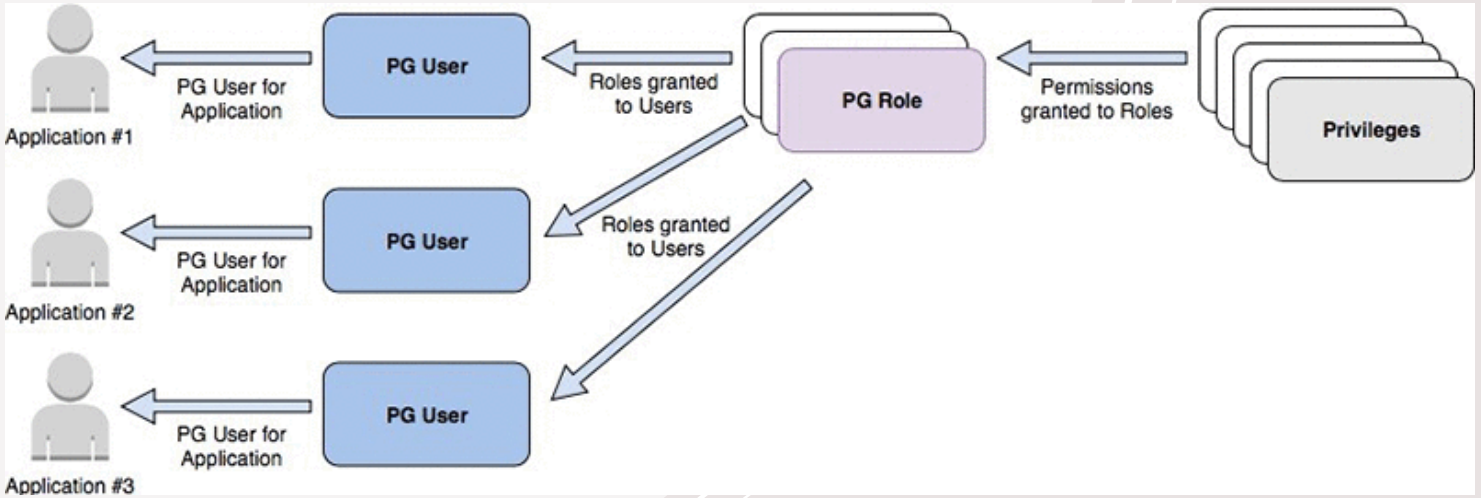




LENGUAJE DE CONTROL DE DATOS (DCL)

Los comandos DCL permiten al Administrador del SGBD controlar el acceso a los objetos.

Con PostgreSQL, podemos crear usuarios y roles con permisos de acceso granulares. Al nuevo usuario o rol se les debe conceder selectivamente los permisos necesarios para cada objeto de base de datos.



LENGUAJE DE CONTROL DE DATOS (DCL)

PostgreSQL permite conceder permisos directamente a los usuarios de la BD. Sin embargo, se recomienda crear varios roles con conjuntos específicos de permisos basados en los requisitos de la aplicación y el acceso. Después, asignar el rol apropiado a cada usuario. Los roles deben utilizarse para aplicar a un modelo de privilegios mínimos para acceder únicamente a los objetos de la BD requeridos.

ROLES, GRUPOS Y USUARIOS

Los usuarios, grupos y roles son lo mismo en PostgreSQL, y la única diferencia es que los usuarios tienen permiso para iniciar sesión de forma predeterminada. Las instrucciones CREATE USER y CREATE GROUP son en realidad alias de la instrucción CREATE ROLE. Para crear un usuario de PostgreSQL, utilizaremos la siguiente instrucción SQL:

```
CREATE USER myuser WITH PASSWORD 'passwd';
```

Este nuevo usuario no tiene ningún permiso, a excepción de los permisos predeterminados disponibles para el rol public. Todos los nuevos usuarios y roles heredan los permisos del rol public.

Los usuarios y roles que deben de poder conectarse a la base de datos o crear objetos en el esquema público, deben recibir los permisos explícitamente antes de revocar los permisos del rol public en el entorno de producción.

CREACIÓN DE ROLES Y USUARIOS

Sintaxis básica para crear roles: CREATE ROLE nombre_rol;

Para recuperar todos los roles en el servidor PostgreSQL actual, puede consultarlos desde el pg_roles catálogo del sistema de la siguiente manera:

SELECT rolname FROM pg_roles;

Los atributos de un rol definen los privilegios para ese rol, incluido el inicio de sesión, el estado de superusuario , la creación de bases de datos, la creación de roles, la administración de contraseñas, etc.

Aquí está la sintaxis para crear un nuevo rol con atributos.

CREATE ROLE name WITH option;

En esta sintaxis, la palabra clave WITH es opcional. Y option puede ser uno o más atributos como SUPERUSER,,, etc.CREATEDBCREATEROLE

EJEMPLOS CREACIÓN DE ROLES

1) Crear roles de inicio de sesión

Por ejemplo, la siguiente declaración crea un rol llamado gestion que tiene privilegios de inicio de sesión y una contraseña inicial:

```
CREATE ROLE gestion

LOGIN

PASSWORD 'securePass1';
```

Tenga en cuenta que coloca la contraseña entre comillas simples (').

2) Crear roles de superusuario

La siguiente declaración crea un rol llamado alicia que tiene el atributo `superuser`.

```
CREATE ROLE alicia

SUPERUSER

LOGIN

PASSWORD 'securePass1';
```

El rol de superusuario tiene todos los permisos dentro del servidor PostgreSQL.

Ten en cuenta que sólo una función de superusuario puede crear otra función de superusuario.

EJEMPLOS CREACIÓN DE ROLES

3) Crear roles con permiso de creación de base de datos.

Si desea crear roles que tengan el privilegio de creación de base de datos, puede utilizar el atributo `CREATEDB`:

```
CREATE ROLE dba

CREATEDB

LOGIN

PASSWORD 'securePass1';
```

4) Crear roles con un período de validez

Para establecer una fecha y hora después de la cual la contraseña del rol ya no es válida, utiliza el atributo

```
VALID UNTIL 'timestamp'
```

Por ejemplo, la siguiente declaración crea un rol llamado `desarrollo` con contraseña válida hasta finales de 2049:

```
CREATE ROLE desarrollo WITH

LOGIN

PASSWORD 'securePass1'

VALID UNTIL '2050-01-01';
```

En 2050, la contraseña de desarrollo ya no es válida.

EJEMPLOS CREACIÓN DE ROLES

5) Crear roles con límite de conexión

Para especificar el número
de conexiones simultáneas que puede realizar un rol,
utilice el atributo CONNECTION LIMIT:

```
CONNECTION LIMIT connection_count
```

Lo siguiente crea un nuevo rol llamado api que puede realizar 1000 conexiones simultáneas:

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

LIMIT -1 SIGNIFICA SIN LIMITE DE CONEXIONES.

LISTADO DE LAS OPCIONES DE CREATE ROLE:

SUPERUSER | NOSUPERUSER

| CREATEDB | NOCREATEDB

| CREATEROLE | NOCREATEROLE

| INHERIT | NOINHERIT

| LOGIN | NOLOGIN

| REPLICATION | NOREPLICATION

| BYPASSRLS | NOBYPASSRLS

| CONNECTION LIMIT ***connlimit***

| [ENCRYPTED] PASSWORD '***password***' | PASSWORD NULL

| VALID UNTIL '***timestamp***'

| IN ROLE ***role_name*** [, ...]

| IN GROUP ***role_name*** [, ...]

PERMISOS Y PRIVILEGIOS

Los comandos para controlar los permisos son:

GRANT: permite otorgar permisos.

REVOKE: elimina los permisos que previamente se le han concedido.

DCL – GRANT

Las tareas sobre las que se pueden conceder o denegar permisos son las siguientes:

- **SELECT**: Permite al usuario ver los datos en una tabla o vista.
- **INSERT**: Permite al usuario agregar nuevos registros a una tabla.
- **UPDATE**: Permite al usuario modificar los datos existentes en una tabla.
- **DELETE**: Permite al usuario eliminar registros de una tabla.
- **ALTER**: Permite al usuario modificar la estructura de una tabla (por ejemplo, agregar o eliminar columnas).
- **CREATE**: Permite al usuario crear nuevos objetos en la base de datos, como tablas, vistas o procedimientos almacenados.
- **DROP**: Permite al usuario eliminar objetos de la base de datos.
- **EXECUTE**: Permite al usuario ejecutar procedimientos almacenados u otros objetos ejecutables.

DCL - GRANT

La sintaxis básica de un comando GRANT en SQL es la siguiente:

GRANT [permisos] ON [objeto] TO [usuario_o_rol];

- **permisos**: Son los permisos específicos que deseas otorgar al usuario o rol. Pueden ser uno o varios permisos separados por comas, como SELECT, INSERT, UPDATE, DELETE, etc.
- **objeto**: Es el objeto de la base de datos en el que deseas otorgar los permisos. Puede ser una tabla, vista, procedimiento almacenado u otro objeto.
- **usuario_o_rol:** Es el nombre del usuario o rol al que deseas otorgar los permisos.

Ejemplo de cómo se usaría la sintaxis:

Conceder permisos de lectura, inserción a un usuario sobre la tabla empleados

GRANT SELECT, INSERT ON empleados TO user1;

DCL - REVOKE

La sintaxis básica de un comando REVOKE es la siguiente:

REVOKE [permisos] ON [objeto] FROM [usuario_o_rol];

- **permisos**: Son los permisos específicos que deseas revocar del usuario o rol. Pueden ser uno o varios permisos separados por comas, como SELECT, INSERT, UPDATE, DELETE, etc.
- **objeto**: Es el objeto de la base de datos del que deseas revocar los permisos. Puede ser una tabla, vista, procedimiento almacenado u otro objeto.
- **usuario_o_rol**: Es el nombre del usuario o rol al que deseas revocar los permisos.

Ejemplo de cómo se usaría la sintaxis:

Revocar permisos de lectura, inserción a un usuario sobre la tabla empleados.

REVOKE SELECT, INSERT ON empleados FROM user1;

EJEMPLOS

GRANT INSERT ON peliculas TO PUBLIC;

Todos los usuarios tendrán permisos de inserción sobre la tabla "peliculas".

GRANT ALL PRIVILEGES ON ciudades TO Manuel;

Concede a Manuel todos los privilegios sobre la tabla "ciudades".

GRANT admin TO Alicia;

Añade al usuario Alicia al rol admin.

REVOKE INSERT ON peliculas FROM PUBLIC;

Todos los usuarios dejan de tener permisos de inserción sobre la tabla "peliculas".

REVOKE ALL PRIVILEGES ON ciudades FROM Manuel;

Quita a Manuel todos los privilegios sobre la tabla "ciudades".

REVOKE admins FROM Alicia;

Quita al usuario Alicia del rol admin.

ROL DE LECTURA

El primer paso consiste en crear un nuevo rol denominado `readonly`: `CREATE ROLE readonly`;

Este es un rol simple sin permisos ni contraseña. No se puede utilizar para iniciar sesión en la base de datos.

Conceder permiso a este rol para conectarse a la BD destino denominada “mydatabase”:

```
GRANT CONNECT ON DATABASE mydatabase TO readonly;
```

Otorgamos acceso al uso de este rol a su esquema. Supongamos que el esquema se llama myschema:

```
GRANT USAGE ON SCHEMA myschema TO readonly;
```

En este paso se concede permiso de rol de `readonly` para realizar alguna actividad dentro del esquema. Sin este paso, el rol `readonly` no puede realizar ninguna acción en los objetos de este esquema, incluso si se han concedido permisos para esos objetos.

El siguiente paso consiste en otorgar acceso al rol `readonly` para ejecutar las consultas en las tablas requeridas.

```
GRANT SELECT ON TABLE mytable1, mytable2 TO readonly;
```

Si el requisito es conceder acceso a todas las tablas y vistas del esquema, puede utilizar el siguiente SQL:

```
GRANT SELECT ON ALL TABLES IN SCHEMA myschema TO readonly;
```

Para garantizar que también se pueda acceder a nuevas tablas y vistas, ejecute la siguiente instrucción para conceder permisos automáticamente:

```
ALTER DEFAULT PRIVILEGES IN SCHEMA myschema GRANT SELECT ON TABLES TO readonly;
```

ROL DE LECTURA Y ESCRITURA

El primer paso es crear un rol: `CREATE ROLE readwrite;`

Permiso a este rol para conectarse a la BD destino: `GRANT CONNECT ON DATABASE mydatabase TO readwrite;`

Privilegios de uso de esquemas: `GRANT USAGE ON SCHEMA myschema TO readwrite;`

Si deseas permitir que este rol cree nuevos objetos como tablas de este esquema, utilice el siguiente SQL en lugar del anterior: `GRANT USAGE, CREATE ON SCHEMA myschema TO readwrite;`

Conceder acceso a las tablas. Como se mencionó en la sección anterior, la concesión puede realizarse en tablas individuales o en todas las tablas del esquema. Para tablas individuales, utilice el siguiente SQL:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE mytable1, mytable2 TO readwrite;
```

Para todas las tablas y vistas del esquema, utilice el siguiente SQL:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA myschema TO readwrite;
```

Para conceder automáticamente permisos sobre tablas y vistas añadidas en el futuro:

```
ALTER DEFAULT PRIVILEGES IN SCHEMA myschema GRANT SELECT, INSERT, UPDATE, DELETE ON TABLES TO readwrite;
```

CREACIÓN DE USUARIOS

Con los roles implementados, se simplifica el proceso de creación de usuarios. Simplemente crearemos el usuario y le concederemos uno de los roles existentes. Estas son las instrucciones SQL para este proceso:

```
CREATE USER user1 WITH PASSWORD 'passwd';

GRANT readonly TO user1;
```

Esta instrucción SQL otorga a *myuser1* los mismos permisos que el rol de solo lectura.

Del mismo modo, podemos conceder acceso de lectura y escritura a un usuario otorgando el rol *readwrite*.

REVOCAR O CAMBIAR PERMISOS

Utilizando el método detallado anteriormente, resulta muy fácil revocar los privilegios de un usuario. Por ejemplo, podemos quitar el permiso de lectura y escritura (*readwrite*) de *user1* utilizando la siguiente instrucción SQL:

```
REVOKE readwrite FROM user1;
```

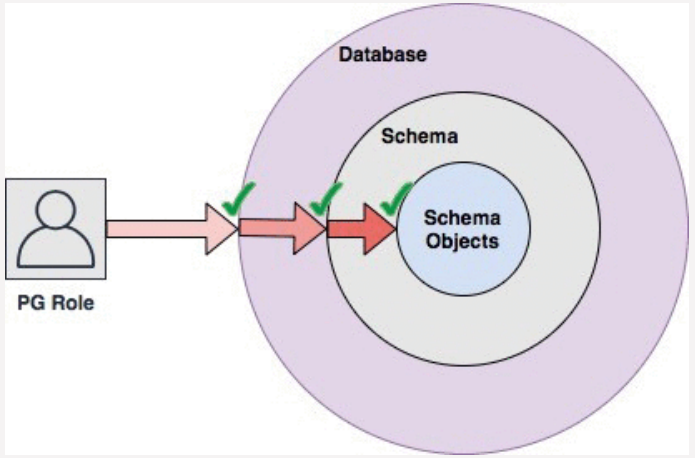
Del mismo modo, podemos otorgar un nuevo rol de la siguiente manera:

```
GRANT readonly TO user1;
```

ESQUEMA PUBLIC Y ROL PUBLIC

Cuando creamos una nueva base de datos, PostgreSQL crea de forma predeterminada un **esquema denominado public** y concede acceso en este esquema a un **rol de backend denominado public**. A todos los usuarios y roles nuevos se les concede de forma predeterminada el rol public y, por lo tanto, pueden crear objetos en el esquema public.

Cuando un usuario intenta crear una nueva tabla sin especificar el nombre del esquema, la tabla se crea en el esquema public. De forma predeterminada, todos los usuarios tienen acceso para crear objetos en el esquema public y esto se convierte en un problema si intentamos crear un usuario de solo lectura. Incluso si restringimos todos los privilegios, los permisos heredados a través del rol public permiten al usuario crear objetos en el esquema public.



ESQUEMA PUBLIC Y ROL PUBLIC

Para solucionarlo, se debe revocar el permiso de creación predeterminado en el esquema public desde el rol public mediante la siguiente instrucción SQL:

```
REVOKE CREATE ON SCHEMA public FROM PUBLIC;
```

Nos aseguraremos de ser el propietario del esquema public o de formar parte de un rol que le permita ejecutar esta instrucción SQL. La siguiente declaración revoca la capacidad del rol público de conectarse a la base de datos:

```
REVOKE ALL ON DATABASE mibasedatos FROM PUBLIC;
```

Esto garantiza que los usuarios no puedan conectarse a la base de datos de forma predeterminada a menos que se conceda explícitamente este permiso. La revocación de los permisos del rol public afecta a todos los usuarios y roles existentes.

Los usuarios y roles que deben de poder conectarse a la base de datos o crear objetos en el esquema público, deben recibir los permisos explícitamente antes de revocar los permisos del rol public en el entorno de producción.

COMPROBACIÓN DE LOS ROLES CONCEDIDOS

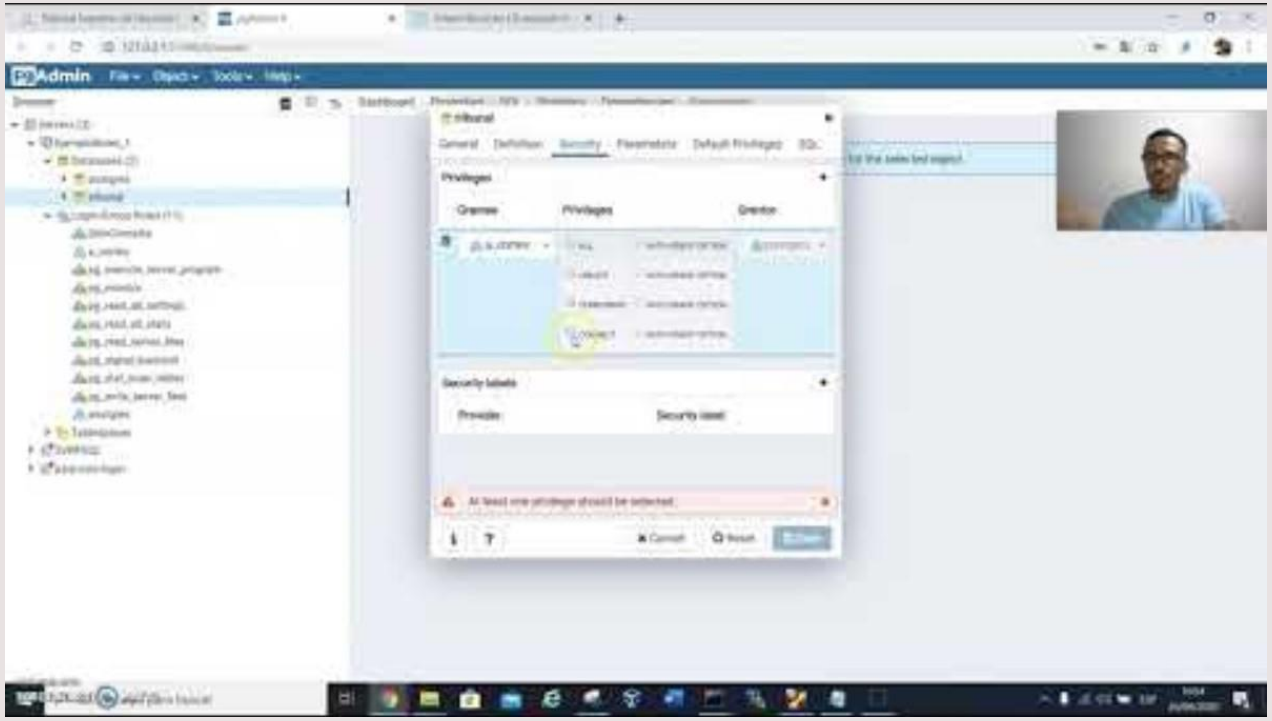
Podemos utilizar la siguiente consulta para obtener una lista de todos los usuarios y roles de la base de datos junto con una lista de roles que se les han concedido:

```
SELECT r.rolname, ARRAY(SELECT b.rolname FROM pg_catalog.pg_auth_members m JOIN pg_catalog.pg_roles b

ON (m.roleid = b.oid) WHERE m.member = r.oid) as memberof

FROM pg_catalog.pg_roles r

WHERE r.rolname NOT IN ('pg_signal_backend','rds_iam', 'rds_replication','rds_superuser', 'rdsadmin','rdsrepladmin')

ORDER BY 1;
```

Debemos tener en cuenta que un usuario puede ser miembro de varios roles con permisos distintos o superpuestos. En este caso, el usuario obtiene una suma de todos los permisos.

También se puede utilizar **la tabla de catálogo_pg_roles** para comprobar atributos como la fecha de caducidad de la contraseña o el número de conexiones paralelas permitidas.

CREAR ROLES Y USUARIOS DESDE pgAdmin



https://www.youtube.com/watch?v=l60FP1gHtJE&t=428s