**Documentación sobre Permisos en la Base de Datos**

**1. ¿Qué son los permisos en una base de datos?**

Los **permisos** en una base de datos se refieren al control de acceso a las diferentes partes de la base de datos (tablas, vistas, funciones, secuencias, etc.). Los permisos son asignados a **roles** (como cliente, empleado, administrador), y determinan qué acciones un usuario puede realizar en la base de datos.

Por ejemplo, un usuario con el rol cliente puede tener permisos para consultar la lista de platos, pero no para modificar los precios. Un usuario con el rol administrador puede tener permisos completos sobre todas las tablas y funciones.

**2. Palabras clave importantes en los permisos de PostgreSQL**

Vamos a desglosar los componentes clave que se utilizan para gestionar los permisos en PostgreSQL:

* **CREATE ROLE**:
  + Se usa para **crear un rol** en la base de datos. Los roles pueden ser de diferentes tipos y pueden tener permisos asignados.

sql

CREATE ROLE cliente;

* **GRANT**:
  + Se usa para **otorgar permisos** a un rol o un usuario en una base de datos. Los permisos pueden ser de lectura (SELECT), escritura (INSERT, UPDATE, DELETE), entre otros.
  + La sintaxis básica es:

sql

GRANT permiso ON objeto TO rol;

* + Donde:
    - **permiso**: El tipo de permiso que otorgamos (por ejemplo, SELECT, INSERT, UPDATE).
    - **objeto**: La base de datos o tabla sobre la cual estamos otorgando el permiso (por ejemplo, Platos, Reservas).
    - **rol**: El rol al cual estamos otorgando el permiso (por ejemplo, cliente, empleado).

Ejemplo:

sql

GRANT SELECT ON Platos TO cliente;

Esto otorga permiso de **lectura** sobre la tabla Platos al rol cliente.

* **REVOKE**:
  + Es el contrario de **GRANT**. Se usa para **revocar permisos** de un rol o usuario sobre un objeto de la base de datos.

sql

REVOKE SELECT ON Platos FROM cliente;

Esto revoca el permiso de **lectura** sobre la tabla Platos para el rol cliente.

* **ALL PRIVILEGES**:
  + Se usa para otorgar **todos los permisos posibles** sobre un objeto (tabla, secuencia, función, etc.). Esto incluye permisos como SELECT, INSERT, UPDATE, DELETE, etc.

sql

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO administrador;

Esto otorga **todos los permisos** sobre **todas las tablas** en el esquema public al rol administrador.

* **SELECT**:
  + Permiso para **leer** los datos de una tabla o vista. Los usuarios con este permiso pueden ejecutar consultas SELECT para ver los datos, pero no pueden modificar nada.

sql

GRANT SELECT ON Platos TO cliente;

* **INSERT**:
  + Permiso para **insertar datos** en una tabla. Los usuarios con este permiso pueden agregar nuevos registros en una tabla.

sql

GRANT INSERT ON Reservas TO cliente;

* **UPDATE**:
  + Permiso para **modificar los datos** en una tabla existente. Los usuarios con este permiso pueden modificar los valores de los registros en una tabla.

sql

GRANT UPDATE ON Reservas TO empleado;

* **DELETE**:
  + Permiso para **eliminar registros** de una tabla. Los usuarios con este permiso pueden borrar registros de una tabla.

sql

GRANT DELETE ON Pedidos TO empleado;

* **REFERENCES**:
  + Permiso para **crear claves foráneas** (relaciones entre tablas). Es necesario para que los usuarios puedan establecer relaciones entre tablas.

sql

GRANT REFERENCES ON Mesas TO empleado;

* **USAGE**:
  + Permiso que se otorga sobre ciertos objetos, como secuencias o esquemas. Permite usar estos objetos sin modificarlos.

sql

GRANT USAGE ON SEQUENCE id\_seq TO administrador;

* **EXECUTE**:
  + Permiso para ejecutar funciones o procedimientos almacenados.

sql

GRANT EXECUTE ON FUNCTION nombre\_funcion() TO administrador;

* **CHECK**:
  + Este no es un permiso directamente, pero es utilizado en el comando CREATE TABLE para asegurarse de que un valor insertado cumpla con una restricción específica. Se utiliza en la definición de columnas y en la creación de roles.

sql

CREATE TABLE Usuarios (

rol VARCHAR(50) CHECK (rol IN ('cliente', 'empleado', 'administrador')) NOT NULL

);

**3. Roles y permisos en el código proporcionado**

* **Rol cliente**:
  + Los clientes solo pueden **ver** los platos, **hacer reservas** y **realizar pedidos**.

sql

GRANT SELECT ON Platos TO cliente;

GRANT INSERT ON Reservas TO cliente;

GRANT INSERT ON Pedidos TO cliente;

* **Rol empleado**:
  + Los empleados pueden **ver** y **actualizar** **solo sus propias reservas** y **pedidos**. Utilizamos vistas (V\_Reservas\_Empleado y V\_Pedidos\_Empleado) para limitar el acceso solo a sus propios datos.

sql

GRANT SELECT, UPDATE ON V\_Reservas\_Empleado TO empleado;

GRANT SELECT, UPDATE ON V\_Pedidos\_Empleado TO empleado;

* **Rol administrador**:
  + Los administradores tienen **todos los permisos** sobre todas las tablas, secuencias y funciones. Esto les da control total sobre la base de datos.

sql

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO administrador;

GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO administrador;

GRANT ALL PRIVILEGES ON ALL FUNCTIONS IN SCHEMA public TO administrador;

**4. Vistas (VIEW)**

Las vistas son **consultas predefinidas** que pueden limitar el acceso a los datos. En este caso, se utilizan para que los empleados solo puedan ver y actualizar sus propias reservas y pedidos:

* **V\_Reservas\_Empleado**:
  + VISTA que permite al empleado ver y actualizar solo las reservas donde el idCliente coincide con el usuario actual.

sql

CREATE VIEW V\_Reservas\_Empleado AS

SELECT \* FROM Reservas WHERE idCliente = current\_user;

* **V\_Pedidos\_Empleado**:
  + VISTA similar que permite al empleado ver y actualizar solo sus pedidos.

sql

CREATE VIEW V\_Pedidos\_Empleado AS

SELECT \* FROM Pedidos WHERE idCliente = current\_user;