Usuarios, Roles y Permisos

Bases de Datos 1 (CC3088) - 2026

Usuarios, Roles y Permisos

Semestre 01, 2026

Problemática

Una base de datos de una empresa tiene tablas de clientes, ventas, nómina y auditoría.

El sistema de ventas necesita insertar pedidos.

El contador necesita leer la nómina pero no modificarla.

El analista solo puede ver reportes de ventas.

El administrador de sistemas no debería ver información salarial.

El riesgo de no controlar el acceso

Sin control de acceso, cualquier conexión a la base de datos tiene poder total.

- Un error en una aplicación puede eliminar datos críticos.
- Un atacante que compromete una cuenta puede leer todo.
- Un desarrollador puede accidentalmente modificar datos de producción.

La solución

PostgreSQL implementa un sistema de **usuarios**, **roles** y **permisos**.

Cada actor tiene exactamente los privilegios que necesita y nada más.

Principio de mínimo privilegio: nadie debe tener más acceso del que necesita para hacer su trabajo.

Roles en PostgreSQL

En PostgreSQL, los usuarios y los roles son el mismo concepto.

Todo es un **role**.

La diferencia es si el role puede iniciar sesión o no.

Rol con login (usuario)

```
CREATE ROLE analista LOGIN PASSWORD 'clave123';
```

Puede conectarse a la base de datos como un usuario.

Rol sin login (grupo)

```
CREATE ROLE equipo_ventas;
```

No puede conectarse directamente.

Se usa para agrupar permisos y asignarlos a otros roles.

Diferencia conceptual

| Tipo | ¿Puede hacer login? | Uso principal | | --- | --- | --- | | Role con LOGIN | Sí | Usuario humano o aplicación | | Role sin LOGIN | No | Agrupador de permisos (grupo) |

Ver roles existentes

```
\du
```

```
SELECT rolname, rolcanlogin, rolsuper
FROM pg_roles
ORDER BY rolname;
```

Crear y eliminar roles

CREATE ROLE — opciones principales

```
CREATE ROLE nombre
  LOGIN                     -- puede iniciar sesión
  PASSWORD 'contraseña'     -- contraseña del usuario
  SUPERUSER                 -- privilegios de administrador total
  CREATEDB                  -- puede crear bases de datos
  CREATEROLE                -- puede crear otros roles
  INHERIT                   -- hereda privilegios de roles asignados
  CONNECTION LIMIT 5        -- máximo de conexiones simultáneas
  VALID UNTIL '2027-01-01'; -- la cuenta expira en esa fecha
```

Ejemplos comunes

```
-- Usuario de aplicación: solo login, sin privilegios extras
CREATE ROLE app_ventas LOGIN PASSWORD 'v3nt4s_2026';

-- Administrador de base de datos
```

```
CREATE ROLE dba LOGIN PASSWORD 'dba_2026' CREATEDB CREATEROLE;

-- Cuenta temporal para un auditor externo
CREATE ROLE auditor LOGIN PASSWORD 'audit_tmp'
  VALID UNTIL '2026-07-01'
  CONNECTION LIMIT 1;
```

Modificar un rol existente

```
ALTER ROLE analista PASSWORD 'nueva_clave';
ALTER ROLE analista VALID UNTIL '2027-06-30';
ALTER ROLE analista CONNECTION LIMIT 3;
```

Eliminar un rol

```
DROP ROLE analista;
```

No se puede eliminar un rol que aún posee objetos o tiene privilegios activos.

Primero hay que reasignar o revocar sus permisos.

GRANT

`GRANT` otorga privilegios sobre objetos de la base de datos.

Sintaxis general:

```
GRANT privilegio ON objeto TO rol;
```

Privilegios sobre tablas

```
-- Permiso para leer
GRANT SELECT ON TABLE ventas TO analista;
```

```
-- Permiso para insertar
GRANT INSERT ON TABLE pedidos TO app_ventas;

-- Permiso para actualizar y eliminar
GRANT UPDATE, DELETE ON TABLE clientes TO gestor;

-- Todos los privilegios DML
GRANT ALL PRIVILEGES ON TABLE productos TO dba;
```

Tabla de privilegios disponibles

| Privilegio | Descripción | | --- | --- | | `SELECT` | Leer filas de la tabla | | `INSERT` | Insertar nuevas filas | | `UPDATE` | Modificar filas existentes | | `DELETE` | Eliminar filas | | `TRUNCATE` | Vaciar la tabla completamente | | `REFERENCES` | Crear claves foráneas que referencien la tabla | | `TRIGGER` | Crear triggers sobre la tabla | | `ALL PRIVILEGES` | Todos los anteriores |

GRANT sobre múltiples tablas

```
-- Acceso de lectura a todas las tablas del esquema public
GRANT SELECT ON ALL TABLES IN SCHEMA public TO analista;
```

```
-- Para que también aplique a tablas futuras
ALTER DEFAULT PRIVILEGES IN SCHEMA public
  GRANT SELECT ON TABLES TO analista;
```

GRANT sobre secuencias

Las secuencias controlan los valores de los campos SERIAL / BIGSERIAL.

Sin permiso sobre la secuencia, un INSERT puede fallar aunque se tenga permiso en la tabla.

```
GRANT USAGE, SELECT ON SEQUENCE pedidos_id_seq TO app_ventas;

-- O todas las secuencias del esquema
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO app_ventas;
```

GRANT sobre esquemas

```
-- Sin este permiso, el rol no puede ver los objetos del esquema
GRANT USAGE ON SCHEMA public TO analista;
GRANT USAGE ON SCHEMA reportes TO analista;
```

GRANT sobre base de datos

```
-- Permiso para conectarse a la base de datos
GRANT CONNECT ON DATABASE tienda TO analista;
```

REVOKE

`REVOKE` elimina privilegios previamente otorgados.

Sintaxis general:

```
REVOKE privilegio ON objeto FROM rol;
```

Ejemplos

```
-- Eliminar permiso de lectura
REVOKE SELECT ON TABLE ventas FROM analista;

-- Eliminar todos los permisos sobre una tabla
REVOKE ALL PRIVILEGES ON TABLE clientes FROM gestor;
```

```
-- Revocar acceso a todas las tablas del esquema
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM analista;
```

Revocar acceso a la base de datos

```
REVOKE CONNECT ON DATABASE tienda FROM auditor;
```

El rol ya no puede conectarse, aunque tenga permisos sobre tablas.

GRANT y REVOKE son exactos

Revocar un privilegio elimina exactamente ese permiso.

Si un rol tiene `SELECT` e `INSERT`, revocar `SELECT` deja `INSERT` intacto.

```
GRANT SELECT, INSERT ON TABLE pedidos TO app_ventas;
REVOKE SELECT ON TABLE pedidos FROM app_ventas;
-- app_ventas aún tiene INSERT
```

Roles como grupos

La forma más escalable de manejar permisos es con **grupos de roles**.

Se definen los permisos en el grupo.

Se agregan usuarios al grupo.

El usuario hereda todos los permisos del grupo.

Crear el grupo y asignar permisos

```
-- Grupo para el equipo de ventas
CREATE ROLE equipo_ventas;
```

```
GRANT USAGE ON SCHEMA public TO equipo_ventas;
GRANT SELECT, INSERT, UPDATE ON TABLE pedidos TO equipo_ventas;
GRANT SELECT ON TABLE clientes TO equipo_ventas;
GRANT SELECT ON TABLE productos TO equipo_ventas;
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO equipo_ventas;
```

Agregar usuarios al grupo

```
CREATE ROLE maria LOGIN PASSWORD 'clave_m';
CREATE ROLE juan LOGIN PASSWORD 'clave_j';
CREATE ROLE sofia LOGIN PASSWORD 'clave_s';

GRANT equipo_ventas TO maria;
GRANT equipo_ventas TO juan;
GRANT equipo_ventas TO sofia;
```

Remover un usuario del grupo

```
REVOKE equipo_ventas FROM juan;
```

Juan pierde todos los permisos del grupo. Los demás no se ven afectados.

Ventaja

Sin grupos: si se agrega una nueva tabla, hay que actualizar los permisos de cada usuario.

Con grupos: se actualiza el grupo una sola vez. Todos los miembros lo obtienen.

Herencia de privilegios

Por defecto, un rol hereda los privilegios de los grupos a los que pertenece.

Esto se controla con la opción `INHERIT`.

Con INHERIT (comportamiento por defecto)

```
CREATE ROLE ana LOGIN PASSWORD 'clave' INHERIT;
GRANT equipo_ventas TO ana;
-- Ana puede usar directamente los permisos de equipo_ventas
```

Sin INHERIT

```
CREATE ROLE auditor LOGIN PASSWORD 'clave' NOINHERIT;
GRANT equipo_auditoria TO auditor;
-- auditor NO hereda los permisos automáticamente
-- Debe hacer: SET ROLE equipo_auditoria; para activarlos
```

`NOINHERIT` obliga a cambiar de rol explícitamente, dejando trazabilidad de cuándo se usaron permisos elevados.

SET ROLE

```
-- Activar temporalmente los permisos de otro rol
SET ROLE equipo_auditoria;

-- Realizar operaciones con esos permisos
SELECT * FROM registros_auditoria;

-- Volver al rol original
RESET ROLE;
```

WITH GRANT OPTION

Por defecto, un rol que recibe un permiso no puede otorgárselo a otros.

`WITH GRANT OPTION` permite que el receptor también pueda otorgar ese permiso.

Ejemplo

```
-- El analista puede leer ventas Y puede darle ese permiso a otros
GRANT SELECT ON TABLE ventas TO analista WITH GRANT OPTION;
```

```
-- Ahora el analista puede ejecutar:
GRANT SELECT ON TABLE ventas TO asistente;
```

Cuándo usarlo

Con mucha precaución.

Si el rol otorgante pierde su permiso, los permisos que él otorgó también se revocan.

Solo para casos donde un rol intermedio necesita delegar acceso de forma controlada.

Verificar permisos

Ver los permisos de una tabla

```
\dp ventas
```

```
SELECT grantee, privilege_type, is_grantable
FROM information_schema.role_table_grants
WHERE table_name = 'ventas'
ORDER BY grantee, privilege_type;
```

Ver los roles de un usuario

```
\du analista
```

```
SELECT member::regrole, role::regrole
FROM pg_auth_members
WHERE member = 'analista'::regrole;
```

Ver todos los privilegios del esquema

```
SELECT table_name, grantee, privilege_type
FROM information_schema.role_table_grants
WHERE table_schema = 'public'
ORDER BY table_name, grantee;
```

Ejemplo completo: sistema de tienda

```
-- 1. Crear grupos con sus permisos
CREATE ROLE rol_cajero;
CREATE ROLE rol_inventario;
CREATE ROLE rol_gerente;

-- Permisos del cajero: crear pedidos y leer productos
GRANT USAGE ON SCHEMA public TO rol_cajero;
GRANT SELECT ON TABLE productos, clientes TO rol_cajero;
GRANT SELECT, INSERT ON TABLE pedidos, pedido_detalle TO rol_cajero;
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO rol_cajero;

-- Permisos del inventario: gestionar productos
GRANT USAGE ON SCHEMA public TO rol_inventario;
GRANT SELECT, INSERT, UPDATE ON TABLE productos TO rol_inventario;
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO rol_inventario;

-- Permisos del gerente: leer todo, sin poder modificar
GRANT USAGE ON SCHEMA public TO rol_gerente;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO rol_gerente;
```

```
-- 2. Crear usuarios y asignarlos a sus grupos
CREATE ROLE pedro LOGIN PASSWORD 'pedro_2026';
CREATE ROLE laura LOGIN PASSWORD 'laura_2026';
CREATE ROLE directora LOGIN PASSWORD 'dir_2026';

GRANT rol_cajero TO pedro;
GRANT rol_inventario TO laura;
GRANT rol_gerente TO directora;

-- 3. Permitir conectarse a la base
GRANT CONNECT ON DATABASE tienda TO pedro, laura, directora;
```

Errores comunes

Error 1: permiso en tabla pero no en esquema

```
GRANT SELECT ON TABLE ventas TO analista;
-- analista aún no puede ver la tabla

-- Falta:
GRANT USAGE ON SCHEMA public TO analista;
```

Sin acceso al esquema, el rol no puede "ver" los objetos dentro de él.

Error 2: permiso en tabla pero no en secuencia

```
GRANT INSERT ON TABLE pedidos TO app_ventas;

-- Al insertar:
-- ERROR: permission denied for sequence pedidos_id_seq
```

El campo SERIAL usa una secuencia. Sin permiso sobre la secuencia, el INSERT falla.

```
GRANT USAGE, SELECT ON SEQUENCE pedidos_id_seq TO app_ventas;
```

Error 3: olvidar CONNECT a la base de datos

```
-- El rol tiene todos los permisos, pero no puede conectarse
-- ERROR: permission denied for database tienda

GRANT CONNECT ON DATABASE tienda TO nuevo_rol;
```

Error 4: usar superusuario en aplicaciones

Conectar una aplicación web con el usuario `postgres` o con `SUPERUSER`.

Si la aplicación es comprometida, el atacante tiene control total.

Crear siempre un rol específico con solo los permisos necesarios.

Error 5: no revocar permisos al eliminar un rol

```
DROP ROLE ex_empleado;
-- ERROR: role "ex_empleado" cannot be dropped because some objects depend on it
```

Primero revocar todos sus permisos y reasignar sus objetos:

```
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM ex_empleado;
REVOKE ALL PRIVILEGES ON DATABASE tienda FROM ex_empleado;
DROP ROLE ex_empleado;
```

Buenas prácticas

- Nunca conectar aplicaciones con el usuario `postgres`.
- Crear un rol por aplicación con solo los permisos que necesita.

- Usar grupos de roles para gestionar permisos a escala.
- Aplicar `VALID UNTIL` a cuentas temporales (auditores, consultores externos).
- Revocar `CONNECT` primero cuando se deba deshabilitar una cuenta urgentemente.
- Documentar qué roles existen, qué permisos tienen y por qué.
- Revisar periódicamente quién tiene acceso a qué con `information_schema.role_table_grants`.