

```
postgres=> select user;
user
-----
oracle
(1 row)
```

ver los usuarios

```
postgres=> \du

               List of roles
   Role name   | Attributes                                   | Member of
-----+-----+-----
\xB\xB[3-dúkjvkljvéeb |                                     | {}
oracle         |                                     | {}
postgres       | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
```

```
postgres=> \l

               List of databases
   Name   | Owner   | Encoding | Collate | Ctype   | Access privileges
-----+-----+-----+-----+-----+-----
postgres | postgres | UTF8     | gl_ES.UTF-8 | gl_ES.UTF-8 | postgres=Ctc/postgres+
          |          |          |          |          | oracle=Ctc/postgres  +
          |          |          |          |          | =c/postgres           +
template0 | postgres | UTF8     | gl_ES.UTF-8 | gl_ES.UTF-8 | =c/postgres           +
          |          |          |          |          | postgres=Ctc/postgres +
          |          |          |          |          | =c/postgres           +
template1 | postgres | UTF8     | gl_ES.UTF-8 | gl_ES.UTF-8 | =c/postgres           +
          |          |          |          |          | postgres=Ctc/postgres +
(3 rows)
```

```
postgres=> \c postgres postgres
Password for user postgres:
psql (14.5 (Ubuntu 14.5-0ubuntu0.22.04.1), server 12.12 (Ubuntu 12.12-0ubuntu0.20.04.1))
You are now connected to database "postgres" as user "postgres".
postgres=#
```

cambiar de usuarios entre oracle → postgres

```
postgres=# create database dam1;
CREATE DATABASE
postgres=#
```

crear una base de datos

```
postgres=# \c dam1 oracle
Password for user oracle:
psql (14.5 (Ubuntu 14.5-0ubuntu0.22.04.1), server 12.12 (Ubuntu 12.12-0ubuntu0.20.04.1))
You are now connected to database "dam1" as user "oracle".
dam1=> █
```

nos conectamos a la base de datos 'dam1' que acabamos de crear cambiandonos al usuario oracle

```
postgres=# revoke connect on database dam1 from public;
REVOKE
postgres=#
```

le quitamos los permisos a oracle (REVOKE) mediante public

```
postgres=# \c dam1 oracle
Password for user oracle:
connection to server on socket "/var/run/postgresql/.s.PGSQL.5432" failed: FATAL: permission denied for database "dam1"
DETAIL: User does not have CONNECT privilege.
Previous connection kept
postgres=#
```

comprobamos el anterior comando y efectivamente no nos podemos conectar a la base de datos dam1 desde oracle

```
postgres=# \c dam1 oracle
Password for user oracle:
psql (14.5 (Ubuntu 14.5-0ubuntu0.22.04.1), server 12.12 (Ubuntu 12.12-0ubuntu0.20.04.1))
You are now connected to database "dam1" as user "oracle".
dam1=>
```

```
dam1=> \dn
List of schemas
Name | Owner
-----+-----
public | postgres
(1 row)
```

es un espacio dentro de un\dn base de datos

```
dam1=> \d
```

para ver las bases que hay

```
dam1=# create schema esquema3;
CREATE SCHEMA
dam1=# \dn
List of schemas
Name | Owner
-----+-----
esquema3 | postgres
public | postgres
(2 rows)
```

creamos un esquema

```
dam1=# create user user3 with password 'user3';  
CREATE ROLE
```

creamos un usuario llamado user3 con contraseña user3

```
dam1=# alter schema esquema3 owner to user3;  
ALTER SCHEMA  
dam1=# \dn  
      List of schemas  
      Name      | Owner  
-----+-----  
esquema3 | user3  
public   | postgres  
(2 rows)
```

ahora el esquema3 pertenece a user3 en vez de a postgres (alterar un esquema)

```
postgres=CTc/postgres+  
oracle=CTc/postgres +
```

la C significa que el usuario puede crear esquemas/tablas la c significa que se puede conectar

```
dam1=# grant connect on database dam1 to user3;  
GRANT  
dam1=#
```

damos permisos a user3 sobre dam1

le damos permisos (mediante GRANT) a oracle de nuevo solamente para que se conecte a la base de datos de dam1 y lo comprobamos

```
dam1=# \c dam1 user3  
Password for user user3:  
psql (14.5 (Ubuntu 14.5-0ubuntu0.22.04.1), server 12.12 (Ubuntu 12.12-0ubuntu0.20.04.1))  
You are now connected to database "dam1" as user "user3".  
dam1=>
```

nos conectamos a dam1 mediante user3

```

dam1=> create table esquema3.prueba1(codigo integer);
CREATE TABLE
dam1=> \d
          List of relations
 Schema | Name      | Type  | Owner
-----+-----+-----+-----
 public | prueba1   | table | user3
(1 row)

dam1=> \d esquema3
Did not find any relation named "esquema3".
dam1=> \d esquema3.*
          Table "esquema3.prueba1"
 Column | Type      | Collation | Nullable | Default
-----+-----+-----+-----+-----
 codigo | integer   |           |          |

```

creamos una tabla en el esquema3 llamada prueba1

```

dam1=> alter user user3 in database dam1 set search_path to esquema3;
ALTER ROLE
dam1=> █

```

hace que cuando crees una tabla se te cree automáticamente en el esquema3

```

dam1=> create table prueba2(codigo integer);
CREATE TABLE
dam1=> \d
          List of relations
 Schema | Name      | Type  | Owner
-----+-----+-----+-----
esquema3 | prueba1   | table | user3
esquema3 | prueba2   | table | user3
(2 rows)

```

comprobamos que el anterior comando ha funcionado

```

dam1=# alter schema esquema4 owner to user4;
ALTER SCHEMA
dam1=# \dn
      List of schemas
      Name          | Owner
      -----+-----
      esquema3      | user3
      esquema4      | user4
      public         | postgres
(3 rows)

```

cambiar el propietario de un esquema

```

Crear usuario user4 contraseña user4
\c dam1 postgres
create user user4 with password 'user4';

Darle permisos de conexión al usuario user4 a la base dam1
grant connect on database dam1 to user4;

crear un nuevo esquema dentro de la base dam1 llamado esquema4
create schema esquema4 ;

hacer propietario del esquema 4 al usuario user4
alter schema esquema4 owner to user4;

por como esquema de búsqueda por defecto del usuario user4 al esquema esquema4
alter user user4 in database dam1 set search_path to esquema4;

```

alter \

- mirar bases de datos: \l
- ver las tablas que hay: \d
- mirar los esquemas de la bd: \dn
- cambiar de base: \c nombre_base usuario
- dar permisos: **grant connect on database nombrebase to usuario**
- quitar permisos: **revoke connect on database nombrebase to usuario**
- crear usuario: **create user nombreusuario with password 'contraseña'**
- cambiar de propietario un esquema: **alter schema nombreesquema owner to nombreusuario;**

Ejercicio Ejemplo:

- Crear usuario udam2

Create user udam2 with password 'udam2'

- Crear esquema e2, propietario udam2

Create schema e2

Alter schema e2 owner to udam2

- Crear usuario udam2 debe crear una tabla llamada "tudam2e2", con los campos codigo y nombre de tipo Integer y Varchar(20)

Create table e2.tudam2e2(codigo integer, nombre varchar(20))

- Conectarse como usuario udam2

\c dam1 udam2

- Insertamos en la tabla las filas:

- 1,'casa'

inser into e2.tudam2e2 values (1, 'casa')

- 2,'arbol'

inser into e2.tudam2e2 values (2, 'arbol')

```
dam1=# create schema e2 authorization udam2;  
CREATE SCHEMA
```

Cambia el propietario del esquema directamente a 'udam2'

\z e2.*

mira los permisos y propietario del esquema e2 además de ver la tabla que tiene dentro

```
dam1=# \z e2.*
```

Schema	Name	Type	Access privileges		Column privileges	Policies
			Access privileges			
e2	tudam2e2	table	udam2=arwdDxt/udam2+			
			udam1=ar/udam2			

```
udam2=arwdDxt/udam2+  
udam1=ar/udam2
```

a = permiso de insertar

r = permiso de leer

```
dam1=> alter table e1.tudam1e1 add column nome varchar(20);  
ALTER TABLE
```

le añadimos una nueva columna a una tabla que ya tenía una columna
alter

```
dam1=> update e1.tudam1e1 set nome='pedro' where codigo=2;  
UPDATE 1  
dam1=> select * from e1.tudam1e1 ;  
codigo | nome  
-----+-----  
      3 |  
      2 | pedro  
(2 rows)
```

```
dam1=> update e1.tudam1e1 set nome='ana' where codigo=3;  
UPDATE 1  
dam1=> select * from e1.tudam1e1 ;  
codigo | nome  
-----+-----  
      2 | pedro  
      3 | ana  
(2 rows)
```

insertamos un valor a la tabla seleccionando la fila que queremos

```
dam1=> grant select(nome) on e1.tudam1e1 to udam2;  
GRANT  
dam1=> \z e1.*  
Schema | Name | Type | Access privileges | Column privileges | Policies  
-----+-----+-----+-----+-----+-----  
e1      | tudam1e1 | table | | nome: + |  
      | | | | udam2=r/udam1 |  
(1 row)
```

darle permiso a usuario udam2 para que pueda hacer select en la columna 'nome'
nos cambiamos de usuario y probamos el select

```
dam1=> select nome from e1.tudam1e1 ;  
nome  
-----  
pedro  
ana  
(2 rows)
```

solamente podemos ver la columna nome, por el permiso que le dimos anteriormente

le damos permiso para insertar a la columna 'nome'

```
dam1=> grant insert(nome) on e1.tudam1e1 to udam2;  
GRANT
```

```
dam1=> insert into e1.tudam1e1 (nome)values('juan');  
INSERT 0 1
```

añadimos solamente datos a la columna de 'nome'

hacemos que la clave primaria de e1.tudam1e1 sea 'codigo'(por su puesto no puede haber ningun nulo, en la columna 'codigo')

```
dam1=> alter table e1.tudam1e1 add primary key(codigo);  
ALTER TABLE
```

sabiendo que estoy conectado como postgres, sabiendo que la tabla es u1, que orden hay

creamos base de datos
y quitamos permisos de connect de schemas a public y a partir de ahí funcionamos

como crear un rol:

```
basex=# create role readonly;  
CREATE ROLE
```

```
basex=# create schema sa authorization ua;
```

crea un schema sa con ua desde postgres dandole como propietario a ua

```
basex=> grant usage on schema sa to readonly;  
GRANT  
basex=> █
```

le damos el permiso para usar el schema sa de solo leer.

```
basex=# grant readonly to ub;  
GRANT ROLE  
basex=# █
```

le damos el rol al usuario b


```
basex=# grant connect ON database basex TO readonly ;
GRANT
```

gracias al rol todos los usuarios que tengan ese rol pueden conectarse a la basex

```
que privilegios sobre tablas tiene el rol readonly ?
SELECT * FROM information_schema.table_privileges where grantee='readonly';
```

que privilegios tiene sobre las tablas el rol readonly, se hace la manera anterior

```
basex=# alter user postgres in database basex set search_path to public,sa,sb;
ALTER ROLE
basex=#
```

postgres busca en los 3 esquemas

hacemos una copia de seguridad

```
pg_dump -U postgres -Fp basex > basex.dump
```

```
oracle@oracle-VirtualBox:~$ pg_dump -U postgres --inserts -Fp basex > basexcompleta.dump
Password:
oracle@oracle-VirtualBox:~$
```

-t exporta solo tabla esa

-T exporta todo menos la tabla esa

```
oracle@oracle-VirtualBox:~$ pg_dump -U postgres -T 'sa.probauat1' --inserts -Fp basex > uat1.d
ump
Password:
```

Hay que tener cuidado con los permisos de las tablas y a quien pertenecen porque igual hay algún usuario que no puede hacer nada con una de las tablas que le pedimos

-n [Para exportar **esquemas** se usa la n]

```
oracle@oracle-VirtualBox:~$ pg_dump -U postgres -n 'sa' --inserts -Fp basex > uat1.dump
Password:
```

Funciona igual que con las T y la t

para ver esas cosas se hace con [**gedit nombre_fichero ;**]

```
oracle@oracle-VirtualBox:~$ pg_dump -U postgres -n 'sa' -t 's*.p*' --inserts -Fp basex > uat1.
dump
Password:
```

el esquema empieza por S y la tabla empieza por p, el . “separa”

```
oracle@oracle-VirtualBox:~$ pg_dump -U postgres -Fc basex > uat1.dump
```

Si cambiamos la p por la c se hace un formato comprimido

```
oracle@oracle-VirtualBox:~$ dropdb -U postgres basex
Password:
```

borra la basex entera

```
oracle@oracle-VirtualBox:~$ pg_restore -U postgres -C -d postgres basexcomprimida.dump
```

restauramos la base de datos basex mediante la copia que habíamos hecho llamada basexcomprimida.dump

-C

el postgres en blanco es la base de datos postgres, se pone postgres porque es la base de datos que tiene como referencia con los privilegios, usuarios, etc. [valdría usar en vez de postgres otra como por ejemplo: template0]

- crear una base de datos con un nombre a partir de la base comprimida, imaginar que la nueva base se debe llamar basez.

```
createdb -U postgres -T template0 basez
pg_restore -U postgres -d basez basexcomprimida.dump
```

esto sería todo junto, se usa template0 porque la usa como referencia para crear la nueva base de datos

restaurar parte de la información del fichero comprimido [partiendo de que la base fue borrada entera]:

```
oracle@oracle-VirtualBox:~$ pg_restore -l basexcomprimida.dump > listado.txt
oracle@oracle-VirtualBox:~$ gedit listado.txt
```

-l hace un listado que lo mete en "listado.txt"

lo editamos con gedit

hay que ir quitando mediante el gedit lo que no nos interes

creas la base basex desde postgres y después pones el comando siguiente

```
oracle@oracle-VirtualBox:~$ pg_restore -U postgres -d basex -L listado.txt basexcomprimida.dump
```

listado.txt tiene lo que queremos restaurar y se lo "metemos a basexcomprimida.dump" que era donde estaba todo, para decirle que solamente queremos lo que está en listado.txt

```
oracle@oracle-VirtualBox:~$ pg_restore -U postgres -C -d postgres -L listado.txt basexcomprimida.dump
Password:
```

con este comando no haría falta crear la basex y luego lanzar el comando

```

oracle@oracle-VirtualBox:~$ psql basex postgres
Password for user postgres:
psql (14.7 (Ubuntu 14.7-0ubuntu0.22.04.1), server 12.12 (Ubuntu 12.12-0ubuntu0.20.04.1))
Type "help" for help.

basex=# \dn
      List of schemas
  Name | Owner
-----+-----
 public | postgres
  sa    | ua
(2 rows)

```

en este caso solo queriamos lo de sa y como se puede ver fue lo único que se quedó ahí

si la base no fue borrada de todo, si solo hemos perdido parte de la información:

```

oracle@oracle-VirtualBox:~$ pg_restore -l basexcomprimida.dump > listado.txt
oracle@oracle-VirtualBox:~$ gedit listado.txt
oracle@oracle-VirtualBox:~$ pg_restore -U postgres -d basex -L listado.txt basexcomprimida.dump

```

```

futbol4=# create role venta;
CREATE ROLE
futbol4=# create role conta;
CREATE ROLE
futbol4=#

```

```

futbol4=# create user alex with password 'alex';
CREATE ROLE
futbol4=# create user berto with password 'berto';

```

Mostrar aplicaciones

```

CREATE ROLE
futbol4=# grant venta to alex;
GRANT ROLE
futbol4=#

```

```

GRANT ROLE
futbol4=# grant conta to berto;
GRANT ROLE
futbol4=#

```

```
futbol4=# \dp facturas;
```

Schema	Name	Type	Access privileges	Access privileges	Column privileges	Policies
public	facturas	table	postgres=arwdDxt/postgres+			
			alex=arwd/postgres			

```
(1 row)
```

```
futbol4=#
```

```
futbol4=# grant select,insert,delete,update on facturas to alex;
GRANT
futbol4=#
```

damos permiso de seleccion en facturas a cuenta, en las columnas id, finalized y details.

```
futbol4=# GRANT SELECT (id, finalized, details) on facturas to conta;
GRANT
futbol4=#
```

NUEVO:

activa la seguridad a nivel de fila.

```
You are now connected to database futbol4 as user postgres .
futbol4=# alter table facturas enable row level security;
ALTER TABLE
futbol4=#
```

siempre y cuando el campo finalized está en true si es falso no se muestra

```
You are now connected to database futbol4 as user postgres .
futbol4=# Create policy only_finalized on facturas to conta using(finalized = true);
CREATE POLICY
futbol4=#
```

puede ver todas las filas

```
futbol4=# create policy all_rows on facturas to venta using(true);
CREATE POLICY
futbol4=#
```

se le aplica a todos los usuarios, solo cuando efectuen operaciones de borrado sobre la tabla facturas.

```
futbol4=# create policy cant_delete_finalized on facturas as restrictive for delete to public using(finalized = false);
;
CREATE POLICY
futbol4=#
```

solo elimina 1 que es la false

```
futbol4=> select * from facturas ;
  id  | finalized |      venta_notes      |      details
-----+-----+-----+-----
 202345 | t         | repeat customer      | {"details": "..."}
 202346 | t         | volume discount      | {"details": "..."}
 202347 | f         | call to be scheduled | {"details": "..."}
(3 rows)

futbol4=> delete from facturas;
DELETE 1
futbol4=>
```

para exportar en formato plano

```
pg_dump -U postgres -Fp futbol20 > copiarFutbol20.dump
```

para exportar como texto comprimido

```
pg_dump -U postgres -Fc futbol20>copiFutbol20.dump
```

para ver la copia de la base que acabamos de crear, hacemos:

```
gedit nombre_base
```

para borrar la base desde fuera

```
dropdb -U postgres nombre_base
```

recuperar la base de datos

