DCL - Data Control Language

Creació d'usuaris i Roles

A Postgres els permisos per accedir a objectes de la base de dades s'atorguen als ROLS. Un rol és una entitat que posseeix objectes de la base de dades i que té certs privilegis. Depen de com s'usa un rol es pot interpretar com a un **usuari** o com a un **grup**

CREATE ROLE

```
CREATE ROLE pepito
[ WITH PASSWORD 'password_value'
  | VALID UNTIL 'expiration' ]
  [WITH OPTION SUPERUSER | CREATEDB | CREATEROLE | LOGIN....
];

Per exemple:
CREATE ROLE comercials;
```

http://www.postgresqltutorial.com/postgresql-roles/

Per veure el detall de les opcions:
https://www.postgresql.org/docs/current/sql-createuser.html
https://www.postgresql.org/docs/current/sql-createrole.html

Els usuaris tenen l'opció LOGIN, poden autenticar-se
(CREATE USER és un alias que de fet fa un CREATE ROLE xxx LOGIN)

```
CREATE USER user_name
  [ WITH PASSWORD 'password_value'
  | VALID UNTIL 'expiration' ];

Per exemple:
CREATE USER user_name PASSWORD 'tres45';
És el mateix que:
CREATE ROLE user_name LOGIN PASSWORD 'tres45';
```

https://www.techonthenet.com/postgresql/users/create_user.php

Algunes opcions:

- SUPERUSER|NOSUPERUSER: té permisos per a tot o no
- CREATEDB|NOCREATEDB: pot crear base de dades o no

- CREATEROLE|NOCREATEROLE: pot crear altres rols o no
- INHERIT|NOINHERIT: hereta els privilegis dels rols d'on n'és membre o no
- LOGIN|NOLOGIN: pot fer login o no
- PASSWORD: establim una contrasenya per al rol
- CONNECTION LIMIT num: nombre màxim de connexions simultànies
- VALID UNTIL 'timestamp': després d'aquesta data la contrasenya ja no serà vàlida
- IN ROLE role_name: el rol que creem s'afegeix al rol existent (i tindrà els seus privilegis, sempre i quan hi hagi l'opció INHERIT, que està per defecte).

ALTER I DROP

Per donar de baixa usuaris o modificar-ne els atributs, farem servir les sentències ALTER i DROP, com ja hem vist en altres objectes de la base de dades.

Per exemple.
Recordeu que l'usuari IAMxx el vam crear amb CREATEDB, pot crear bases de dades, però de moment no pot crear usuaris i rols. Per fer-ho l'haurem de modificar:

```
ALTER USER iamxxx WITH CREATEROLE;
```

GRANT

Per assignar un usuari a un grup/role, o per fer que un grup/role incorpori els privilegis d'un altre rol:

```
sintaxi:
grant role_pare [, ...] to role_fill [, ...] ;
A grup fill li atorguem els privilegis definits al grup pare

exemple:
grant vendes to usuari1, usuari2;
usiari1 i usuari2 tindran els provilegis definits a vendes.
```

Canviar de role

Per treballar amb els privilegis d'un rol en concret:

```
set role vendes;
```

Per veure com a quin usuari/role estàs treballant:

```
select session_user, current_user;
```

Per tornar al vostre role normal:

```
reset role;
```

Per treballar des d'un rol concret, podeu iniciar la sessió amb psql amb aquell rol, o des de la sessió on esteu, podeu canviar de rol.

Llistar rols

Per veure la llista de rols existents:

```
\du
```

Veieu per exemple que els usuaris tenen LOGIN i els rols no.

Gestió de privilegis. GRANT / REVOKE

```
GRANT privileges ON object TO role|PUBLIC;
REVOKE privileges ON object FROM role|PUBLIC;
```

On els **privilegis** poden ser:

- SELECT
- INSERT
- UPDATE
- DELETE
- INDEX
- CREATE
- ALTER
- DROP
- GRANT OPTION
- ALL

https://www.techonthenet.com/postgresql/grant_revoke.php

Els objectes poden ser taules, (camps concrets), vistes, seqüències...

https://www.tutorialspoint.com/postgresql/postgresql_privileges.htm
https://www.postgresql.org/docs/current/sql-grant.html

```
GRANT SELECT ON mytable TO PUBLIC;
GRANT SELECT, UPDATE, INSERT ON mytable TO admin;
GRANT SELECT (col1), UPDATE (col1) ON mytable TO miriam_rw;
```

Nota: Les bases de dades tenen **esquemes** i els esquemes tenen les taules. Si no hem creat un esquema explícitament, les nostres taules estan a l'esquema de nom *public*. Els esquemes serveixen per agrupar altres entitats de la base de dades. Ens permeten un nivell més de seguretat. Per exemple a la base de dades training, podríem tenir les dades de dues empreses em dos esquemes diferents i a cada esquema podríem tenir les taules repventa, oficina,... repetides.

Exemples:
Veieu que podem afinar fins a donar certs privilegis sobre certes columnes d'una taula:

```
-- Permetre veure, inserir i esborrar registres de la taula repventa a
l'usuari rrhh:
GRANT SELECT, INSERT, DELETE ON repventas TO rrhh;

-- Permetre la gestió de totes les dades de totes les taules a l'usuari
tworker
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO tworker;

-- Permetre veure, i modificar les columnes objetivo i ventas de la
taula oficina a l'usuari director
GRANT SELECT, UPDATE (objetivo, ventas) ON oficina TO director;
```

Per veure els privilegis:

```
=> \dp
```

Per veure els privilegis sobre una taula:

```
=> \dp mytable
```

on:

```
rolename=xxxx -- privileges granted to a role
        =xxxx -- privileges granted to PUBLIC

            r -- SELECT ("read")
            w -- UPDATE ("write")
            a -- INSERT ("append")
            d -- DELETE
            D -- TRUNCATE
            x -- REFERENCES
            t -- TRIGGER
            X -- EXECUTE
```

```
        U -- USAGE
        C -- CREATE
        c -- CONNECT
        T -- TEMPORARY
    arwdDxt -- ALL PRIVILEGES (for tables, varies for other objects)

      /yyyy -- role that granted this privilege
```

Sintaxi GRANT i REVOKE

Donar permisos permisos a taules:

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES |
TRIGGER }
[, ...] | ALL [ PRIVILEGES ] }
ON { [ TABLE ] table_name [, ...]
| ALL TABLES IN SCHEMA schema_name [, ...] }
TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

Donar permisos a columnes:

```
GRANT { { SELECT | INSERT | UPDATE | REFERENCES } ( column_name [, ...]
)
[, ...] | ALL [ PRIVILEGES ] ( column_name [, ...] ) }
ON [ TABLE ] table_name [, ...]
TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

Treure permisos permisos de taules:

```
REVOKE [ GRANT OPTION FOR ]
{ { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER
}
[, ...] | ALL [ PRIVILEGES ] }
ON { [ TABLE ] table_name [, ...]
| ALL TABLES IN SCHEMA schema_name [, ...] }
FROM { [ GROUP ] role_name | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]
```

Treure permisos a columnes:

```
REVOKE [ GRANT OPTION FOR ]
{ { SELECT | INSERT | UPDATE | REFERENCES } ( column_name [, ...] )
[, ...] | ALL [ PRIVILEGES ] ( column_name [, ...] ) }
ON [ TABLE ] table_name [, ...]
FROM { [ GROUP ] role_name | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]
```

ALTER OWNER

També podem canviar el propietari de la base de dades o d'una taula

```
ALTER DATABASE db_name OWNER TO username;
ALTER TABLE table_name OWNER TO username;

-- Per exemple:

# Donar tots els permisos de la base de dades training a l'usuari
tmanager:
ALTER DATABASE training OWNER TO tmanager;
ALTER TABLE cliente OWNER TO tmanager;
ALTER TABLE oficina OWNER TO tmanager;
ALTER TABLE repventa OWNER TO tmanager;
ALTER TABLE producto OWNER TO tmanager;
ALTER TABLE pedido OWNER TO tmanager;
```

Exemple:
Per tal de tenir tots els permisos en una base de dades el més fàcil és ser propietaris d'aquesta i de totes les seves taules. Si han de tenir permisos dos usuaris, es pot crear un rol (que simuli un grup), fer que aquest rol sigui el propietari de la base de dades i la taula i fer que els dos usuaris tinguin els permisos d'aquest grup.

Donar tots els permisos de la base de dades training a l'usuari tmanager i a l'usuari tadmin:

```
CREATE ROLE tgroup;
CREATE USER tmanager PASSWORD 'man347';
CREATE USER tadmin PASSWORD 'admin174';
ALTER DATABASE training OWNER TO tgroup;
ALTER TABLE clientes OWNER TO tgroup;
ALTER TABLE oficinas OWNER TO tgroup;
ALTER TABLE repventas OWNER TO tgroup;
ALTER TABLE productos OWNER TO tgroup;
ALTER TABLE pedidos OWNER TO tgroup;
GRANT tgroup TO tmanager, tadmin;
```