POSTGRESQL - CREACIÓ D'UNA BASES DE DADES

LA SENTÈNCIA CREATE DATABASE

La primera sentència SQL que veurem serà la que ens permetrà crear una base de dades.

- Serà necessari tenir privilegis per crear bases de dades (CREATEDB) o bé ser l'usuari administrador (postgres).
- El nom de la base de dades no pot existir en el mateix SGBD.
- Postgresql disposa d'una base de dades plantilla anomenada template1 si no s'especifica el contrari les bases de dades es crearan amb les mateixes característiques de template1.

- **OWNER**: Assigna el rol que serà el propietari de la base de dades. Si no s'especifica serà l'usuari de la connexió des d'on s'executi la sentencia CREATE DATABASE.
- **TEMPLATE**: Especifica la base de dades 'model' a partir de la que es crearà la nova base de dades, per defecte template1.
- **ENCODING**: Determina la codificació alfanumèrica que utilitzarà la base de dades Normalment UTF 8 (Unicode 8 bits)
- **LC_COLLATE** i **LC_CTYPE**: Influeixen en l'ordenació de les dades en funció de la codificació alfanumèrica utilitzada i de la configuració regional. Postgresql Collation
- **TABLESPACE**: Especifica el nom de l'espai de treball (tablespace) de la base de dades. Es refereix a l'espai físic de disc on s'emmagatzemen les dades. Per defecte PostgreSQL té dos tablespace:
  - pg_default: on es guarden les dades d'usuari.
  - pg_global: on es guarden les dades globals.
- **CONNECTION_LIMIT**: Especifica el número de connexions concurrents. Per defecte 1, sense límit.
- **ALLOW_CONNECTIONS**: Especifica si s'admeten connexions a la base de dades. Si el valor és *false* llavors no podem connectar a la base de dades.
- **IS_TEMPLATE**: Si el valor és *true* qualsevol usuari podrà utilitzar la base de dades com a plantilla per la creació d'una altra base de dades.

> Exemple:

```
CREATE DATABASE hr
WITH ENCODING = 'UTF8'
OWNER hr
CONNECTION LIMIT 100;
```

ALTRES COMANDES

ALTER DATABASE *database_name* SET *configuration_parameter* = *value*;

ALTER DATABASE *database_name* WITH *option*;

```
ALTER DATABASE testhrdb WITH CONNECTION LIMIT 100;
```

ALTER DATABASE *database_name* RENAME TO *new_database_name*;

```
ALTER DATABASE testdb2 RENAME TO testhrdb;
```

ALTER DATABASE *database_name* OWNER TO *new_owner* | *current_user* | *session_user*;

```
ALTER DATABASE testhrdb OWNER TO hr;
```

ALTER DATABASE *database_name* SET TABLESPACE *new_tablespace*;

```
ALTER DATABASE testhrdb SET TABLESPACE hr_default;
```

DROP DATABASE [IF EXISTS] *database_name*; -- Elimina la base de dades.

```
DROP DATABASE testdb1;
```

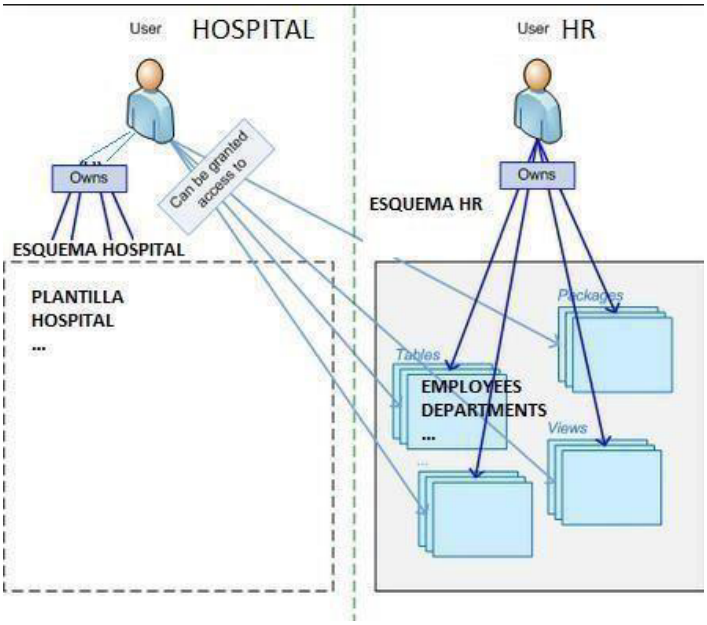
No podem eliminar una base de dades si tenim alguna sessió oberta (usuaris que estan treballant amb la base de dades). Mitjançant sentències de consulta del diccionari de dades podem esbrinar si tenim connexions a la base de dades obertes i, fins hi tot, eliminar totes les sessions actives.

```
SELECT FROM pg_stat_activity WHERE datname ='testdb1';
```

```
SELECT pg_terminate_backend(pg_stat_activity.pid)
FROM pg_stat_activity
WHERE pg_stat_activity.datname ='testdb1';
```

ESQUEMES DE LA BASE DE DADES (SCHEMA)

Un esquema és una col·lecció lògica d'objectes (taules, vistes, seqüències, indexs, etc.), per defecte cada usuari accedeix als objectes del seu esquema i pot accedir als objectes d'un altre usuari si aquest li concedeix privilegis.



Sentència per a crear un nou esquema:

```
CREATE SCHEMA schema_name AUTHORIZATION user_name;
```

AUTHORIZATION serà el propietari (OWNER) del nou esquema.

Postgresql per defecte crea l'esquema *public*. Per referir-nos a una taula d'un determinat schema haurem de posar lo com a prefix <*nom de l'esquema*>.: public.staff Podem consultar en quin esquema estem treballant:

```
SELECT current_schema();
```

Postgresql disposa del paràmetre *search_path* on indiquem els esquemes on ha de cercar els objectes de la base de dades. (Per defecte els esquemes public i els que són propietat de l'usuari que ha iniciat sessió a la base de dades).

```
SHOW search_path;
```

SEGURETAT EN LA BASE DE DADES

La seguretat en la base de dades es classifica en dues categories:

- **Seguretat del sistema**: Controla l'accés i ús de la base de dades a nivell de sistema (accessos, crear, modificar objectes)
- **Seguretat de les dades**: Controla l'accés i ús de la base de dades a nivell d'objectes (permisos de consulta i modificació sobre taules, vistes, etc.)

EL LLENGUATGE DCL (DATA CONTROL LANGUAGE)

El Llenguatge de Control de Dades es el sublenguatge SQL responsable de les tasques administratives de control de la pròpia base de dades, especialment la gestió d'usuaris i/o rols i la concesió i revocació de permisos de base de dades per als usuaris.

GESTIÓ D'USUARIS - ROLS

A diferencia d'altres SGBD Postgresql no treballa amb el concepte d'usuari sinó amb el concepte de **rol**.

Alguns rols permeten realitzar un login a la base de dades, serien els equivalents al usuaris d'altres SGBD.

Els rols que poden contenir altres rols s'anomenen **rols de grup**, serien equivalents als rols d'altres SGBD.

LA SENTÈNCIA CREATE ROLE

```
CREATE ROLE role_name;
```

```
CREATE ROLE role_name
LOGIN
PASSWORD 'role_passwd';
```

Podem consultar els rols de la base de dades mitjançant el diccionari de dades:

```
SELECT rolname FROM pg_roles;
```

Exemples:

```
CREATE ROLE john SUPERUSER LOGIN PASSWORD 'securePass1';
```

```
CREATE ROLE dba CREATEDB LOGIN PASSWORD 'Abcd1234';
```

Quan volguem crear un rol amb permís de d'autenticació a la base de dades (LOGIN) podem utilitzar la sentència CREATE USER (i no serà necessàri indicar la paraula reservada LOGIN).

```
CREATE USER john SUPERUSER PASSWORD 'securePass1';
```

MODIFICAR I ELIMINAR ROLS

Modificar dades d'un usuari o rol:

```
ALTER ROLE role_name [WITH] option;
```

```
ALTER ROLE calf SUPERUSER;
```

```
ALTER ROLE calf PASSWORD '1234';
```

Eliminar un rol:

```
DROP ROLE [IF EXISTS] target_role;
```

```
DROP ROLE calf;
```

LA SENTÈNCIA GRANT

La sentència **GRANT** ens permetrà atorgar permisos (o privilegis) sobre la base de dades, esquemes i objectes de la mateixa.

- Privilegis de sistema: Permisos que s'atorguen sobre bases de dades i esquemes de la base de dades.

```
GRANT { { CREATE | CONNECT | TEMPORARY | TEMP}
| ALL [ PRIVILEGES ] }
ON DATABASE database_name
TO { [ GROUP ] role_name | PUBLIC }
[ WITH GRANT OPTION ]

GRANT { { CREATE | USAGE} | ALL [ PRIVILEGES ] }
ON SCHEMA schema_name
TO { [ GROUP ] role_name | PUBLIC }
[ WITH GRANT OPTION ]
```

Exemples:

```
GRANT USAGE ON SCHEMA hr
TO calf;
```

```
GRANT CONNECT ON DATABASE hr
TO hruser;
```

```
GRANT CREATE, USAGE ON SCHEMA hr
TO alice;
```

```
GRANT ALL PRIVILEGES ON DATABASE hr
TO dba
WITH GRANT OPTION;
```

Clàusula **WITH GRANT OPTION**: Indica que el rol que rep els privilegis podrà atorgar aquests privilegis a d'altres usuaris, sense aquesta clàusula un rol que rep determinats permisos no pot donar-los a altres rols.

- Privilegis sobre les dades: Privilegis que s'atorguen sobre la manipulació i consulta de les dades d'una taula, vista, etc. Es poden donar sobre tot l'objecte (taula, vista, ...), o bé fins hi tot sobre determinats camps d'una taula o vista.

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

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

Exemples:

```
GRANT SELECT
ON candidates
TO joe;
```

```
GRANT INSERT, UPDATE, DELETE
ON candidates
TO joe;
```

```
GRANT ALL
ON ALL TABLES
```

```
IN SCHEMA "public"
TO joe;
```

```
GRANT SELECT
ON TABLE employees
TO PUBLIC;
```

```
GRANT SELECT(first_name),UPDATE(hire_date)
ON employees
TO alice;
```

```
GRANT ALL PRIVILEGES
ON departments
TO manuel;
```

La clàusula **PUBLIC** indica que tothom (tots els rols de la base de dades) tindrà permisos, inclòs els rols que es puguin crear posteriorment.
Per defecte en crear una base de dades s'atorguen permisos públics sobre l'esquema PUBLIC.

LA SENTÈNCIA REVOKE

La sentència **REVOKE** ens permetrà treure permisos (o privilegis) sobre la base de dades, esquemes i objectes de la mateixa.

- Privilegis de sistema:

```
REVOKE [GRANT OPTION FOR]
{ { CREATE | CONNECT | TEMPORARY | TEMP}
| ALL [ PRIVILEGES ] }
ON DATABASE database_name
FROM { [ GROUP ] role_name | PUBLIC }
[ CASCADE | RESTRICT ]

REVOKE [GRANT OPTION FOR]
{ { CREATE | USAGE} | ALL [ PRIVILEGES ] }
ON SCHEMA schema_name
FROM { [ GROUP ] role_name | PUBLIC }
[ CASCADE | RESTRICT ]
```

- Privilegis sobre les dades:

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


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

Les clàusules **CASCADE** o **RESTRICT** (per defecte) indiquen com s'han d'eliminar els possibles permisos que es revoquen a un usuari i que aquest pot haver donat a d'altres usuaris si té el privilegi WITH GRANT OPTION.

- CASCADE: Elimina els permisos en cascada a tots els usuaris a qui se'ls ha donat i als que ells puguin haver donat.
- RESTRICT: No elimina el permis si hi ha d'altres usuaris que l'han rebut de l'usuari al que se li vol treure.

Exemples:

```
REVOKE ALL PRIVILEGES
ON employees
FROM manuel;
```

```
REVOKE INSERT
ON films
FROM PUBLIC;
```

```
REVOKE GRANT OPTION FOR CONNECT
ON DATABASE hr
FROM hruser CASCADE;
```

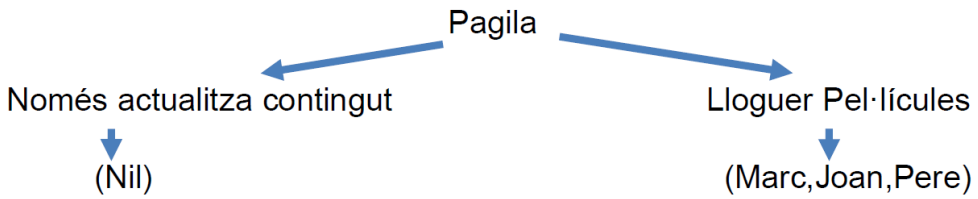
ROLS DE GRUP

És habitual utilitzar rols per agrupar privilegis de diferents usuaris.
Normalment crearem un rol que representarà a un grup al que afegirem altres membres (rols amb login (usuaris) i/o rols de grup)
Per convenció un rol de grup serà aquell que no tindrà permís de Login.

```
CREATE ROLE role_group_name;
```

```
GRANT role_group_name TO user_role;
```

Exemple:




```
CREATE ROLE nomes_contingut;
CREATE ROLE nil LOGIN password '123';

GRANT SELECT ON film
TO nomes_contingut;

GRANT nomes_contingut to nil;
```

```
CREATE ROLE lloguer;
CREATE ROLE Marc LOGIN password '123';
CREATE ROLE Joan LOGIN password '123';
CREATE ROLE Pere LOGIN password '123';

GRANT ALL IN ALL TABLES TO lloguer;

GRANT lloguer to Marc,joan,Pere;
```

Un usuari pot tenir permisos sobre un objecte per si mateix o bé per estar en un rol de grup.

Exemple:

```
Postgresql$ psql -d pagila -U alice

Pagila=> SELECT * FROM film;
...
Pagila=> SELECT * FROM actor;
...
Pagila=>\c pagila postgres;
Pagila=>REVOKE sales FROM alice;

Pagila=>\c pagila alice;
Pagila=>SELECT * FROM actor;
ERROR: Permission denied for relation actor
```