



Objectius

És el moment d'administrar un SGBD. Ací veurem sols els conceptes bàsics que considere que vos pot anar bé conèixer.

Pense que és un SGBD molt potent i flexible, i a més és software lliure.

Els objectius concrets d'aquest tema seran :

- Instal·lar el servidor PostgreSQL.
- Arrancar i parar la Base de Dades.
- Crear usuaris
- Fer còpies de seguretat de la Base de Dades, i saber recuperar-la en cas de necessitat.



Coneixements previs

En aquest tema instal·larem i administrarem PostgreSQL sobre Linux. Faran falta, per tant, unes nocions bàsiques sobre Linux.

Qui vulga també pot instal·lar el servidor en Windows o en un altre Sistema Operatiu. En aquest sentit també es veurà la instal·lació en Windows 7. Caldran evidentment unes nocions bàsiques sobre el Sistema Operatiu triat per a instal·lar el servidor.

Tanmateix, es recomana utilitzar una màquina virtual per evitar problemes.

Ara que anem a administrar PostgreSQL, podríem plantejar-nos la conveniència o no d'utilitzar PostgreSQL, sobretot comparant-lo amb l'altre SGBD més utilitzat en Linux (i que abans era totalment software lliure), MySQL

Definint de forma molt ràpida, PostgreSQL és un SGBD Relacional que incorpora conceptes d'objectes. Anem a refinar un poc açò.

El primer de tot és que és un SGBD Relacional, que ofereix suport pràcticament total a l'estàndard SQL 92 / SQL 3, i que implementa perfectament la integritat referencial, les restriccions, disparadors, transaccions, ...

Intenta ser molt flexible a l'extensibilitat, i així permet a l'usuari definir tipus de dades nous, operadors nous, ...

I sobretot, incorpora també conceptes de B.D. orientades a objectes. No vol dir que siga un SGBD orientat a objectes, sinó que incorpora coses d'aquestos, com són les classes i l'herència.

Alguns autors anomenen a aquest tipus com SGBD objecte-relacionals (ORDBMS)

L'origen de PostgreSQL va ser un projecte de la Universitat de Berkeley. Li van donar el nom de Postgres i va començar en 1986. En 1987 es va llançar la versió 1, que va anar millorant fins la versió 4.2.

En 1994 va sorgir un descendent de Postgres de domini públic i codi obert, anomenat Postgres95, que a més millorava coses i era més eficient. A més, el llenguatge de consultes va passar de l'original *Postqual* a *SQL*.

Posteriorment, com que el nom no s'aguantava, en 1996 es va passar a **PostgreSQL** per reflectir la relació entre el SGBD original i les versions més recents amb capacitats SQL. Les versions van partir de la 6.0, per a tornar amb la seqüència original.

En Linux, o millor dit, en el món del software lliure, s'utilitzen principalment dos SGBD: **MySQL** i **PostgreSQL**.

- En el primer s'intenta per damunt de tot que siga molt ràpid i que gaste pocs recursos. Per aconseguir açò fins i tot sacrifica coses impensables en altres entorns (integritat referencial, subconsultes, transaccions, ...). S'ha imposat en la creació de pàgines web (formant la trilogia Apache - Php - MySQL), on es busca sobretot velocitat. Tanmateix en les últimes versions comença a incorporar alguns dels conceptes abans esmentats, sobretot a partir de la versió 5.
- PostgreSQL podríem definir-lo com a "més seriós", i encara que perd velocitat respecte a MySQL, pot ser perfectament un SGBD per a llocs on la seguretat i la solvència és una cosa més que anecdòtica.

Hi ha moltes comparatives en Internet dels dos SGBD, i normalment la conclusió no és que un siga millor que l'altre, sinó que depèn del l'ús que es vulga fer, contestant més bé la pregunta: **velocitat o potència?**

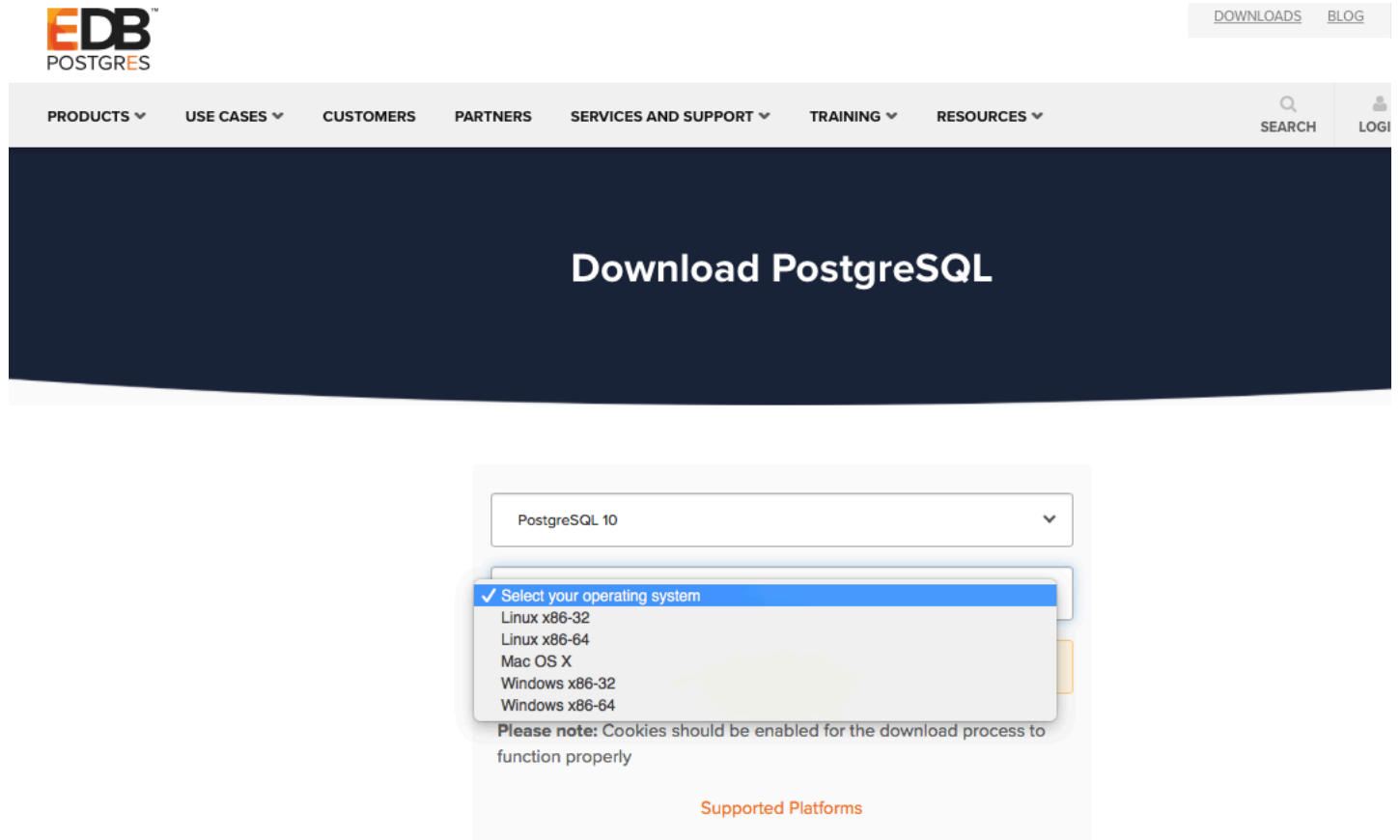
La instal·lació es pot realitzar a més d'un nivell: instal·lació senzilla (en Linux i Windows), instal·lació per paquets (també molt senzilla) i la instal·lació completa, compilant les fonts (només en Linux, evidentment).

Nosaltres sols anem a veure la instal·lació senzilla en el Sistema Operatiu triat. En cas de dubte, **us aconselle que instal·leu en una màquina virtual.**

2.1 Instal·lació automàtica en Linux

A partir de les últimes versions, PostgreSQL ofereix la possibilitat de fer una instal·lació automàtica, a l'estil de Windows, executant un programa d'instal·lació.

De fet, si accedim a la pàgina de PostgreSQL, en la secció de Downloads (www.postgresql.org/download) és el primer que ens ofereix, en les versions de Sistemes Operatius més habituals.



El següent vídeo mostra tots el procés (per a la versió 9.4.4 que era la última en el moment de realitzar aquest vídeo.) on tot es pot fer per defecte excepte la contrasenya de l'usuari de la Base de Dades **postgres**, que serà l'Administrador de la Base de Dades (DBA). Us proposa la contrasenya **postgres**.

Per a la versió 9.6:

El lloc on per defecte es guarden els programes i configuracions és **/opt/PostgreSQL/9.6** (9.6 o la versió que instal·lem)

La Base de Dades creada (Base de Dades gran, a l'estil d'Oracle, no a l'estil d'Access) per defecte s'instal·la en **/opt/PostgreSQL/9.6/data**

I durant el procés d'instal·lació s'ha creat un usuari de Sistem Operatiu anomenat **postgres**, que serà el propietari de tot el de PostgreSQL

En la Base de Dades només hi haurà un usuari, anomenat també **postgres**, que com hem comentat abans serà el DBA

Nosaltres instal·larem la versió indicada en el nostre curs.

2.2 Instal·lació automàtica en Windows

PostgreSQL es pot instal·lar perfectament en qualsevol versió de Windows XP, 7, 8

En Windows Vista s'ha de desactivar un servei primer, per a que no hi hagen problemes. Si algú l'està utilitzant (cosa molt improbable) i té problemes, hauria de desactivar UAC (User Account Control), encara que siga temporalment.

En aquestos apunts mostraré la instal·lació en Windows 7, on no ha d'haver cap problema.









La instal·lació en Windows és pràcticament igual que la de Linux, i per tant els comentaris seran similars. Agafarem el fitxer d'instal·lació del mateix lloc (<http://www.postgresql.org/download>). La versió disponible és pràcticament la mateixa que en Linux

Durant la instal·lació, com en Linux, tot ho posarem per defecte, excepte la contrasenya de l'usuari **postgres**, que posarem **postgres**.

PostgreSQL Installers

These enterprise-class 64-bit PostgreSQL binaries are always free and Open Source. They are tested to run on Centos 6+, Ubuntu 12.04+, OSX 10.9+, Windows 7+ and Windows Server 2008+.

Featured! Command line Package Manager [usage instructions](#) for all versions of Postgres.

PostgreSQL 9.6.2 - Stable (09-Feb-17)	PostgreSQL 9.5.6 - Proven (09-Feb-17)
 postgresql-9.6.2-win64.exe	 postgresql-9.5.6-win64.exe
 postgresql-9.6.2-osx64.dmg	 postgresql-9.5.6-osx64.dmg
 postgresql-9.6.2-x64.rpm	 postgresql-9.5.6-x64.rpm
 postgresql-9.6.2-x64.deb	 postgresql-9.5.6-x64.deb

El següent vídeo mostra tots el procés (per a la versió 9.4.5 que era la última en el moment de realitzar aquest vídeo.)

Nosaltres instal·larem la versió indicada en el nostre curs.

Per a la versió 9.6:

El lloc on per defecte es guarden els programes i configuracions és **C:\Program Files (x86)\PostgreSQL\9.6** (o la seua traducció: **C:\Archivos de Programa (x86)\PostgreSQL\9.6**)

La Base de Dades creada (Base de Dades gran, a l'estil d'Oracle, no a l'estil d'Access) per defecte s'instal·la en **C:\Program Files (x86)\PostgreSQL\9.6\data**

I en la Base de Dades només hi ha un usuari de moment, el DBA, anomenat també **postgres** (amb contrasenya **postgres**)

Els entorns de treball seran els programes que ens permetran connectar i fins i tot administrar la Base de Dades.

- El **PSQL** va incorporat en PostgreSQL, i per tant sempre està disponible. És un editor de comandos SQL en mode línia.

Els altres dos entorns són gràfics.

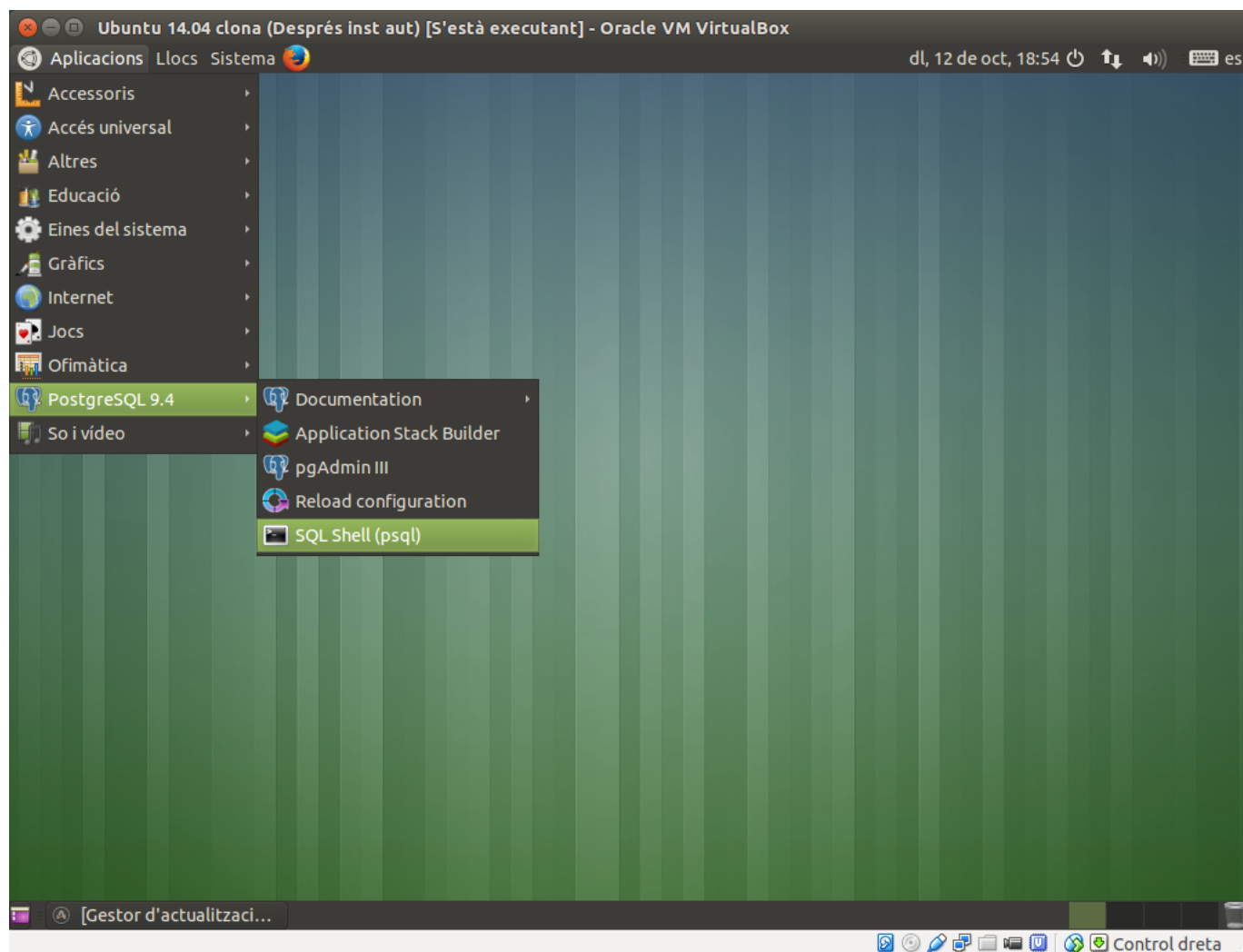
- El **PGADMIN** és el més habitual per administrar PostgreSQL.
- El **PHPPGADMIN** és un conjunt de pàgines PHP. Permet per tant administrar des de qualsevol lloc (si en el servidor també tenim un servidor web). És pràcticament igual que el PHPMYADMIN, que s'utilitza moltíssim per a administrar MySQL, i per tant és molt conegut.

El programa més senzill que podem utilitzar per a connectar-nos amb el servidor i executar les sentències SQL és el **PSQL**, que senzillament és un intèrpret de sentències SQL on anirem posant els comandos i ens anirà donant els resultats. És, per tant, prou incòmode, però com que forma part de PostgreSQL ja està instal·lat i sempre funciona.

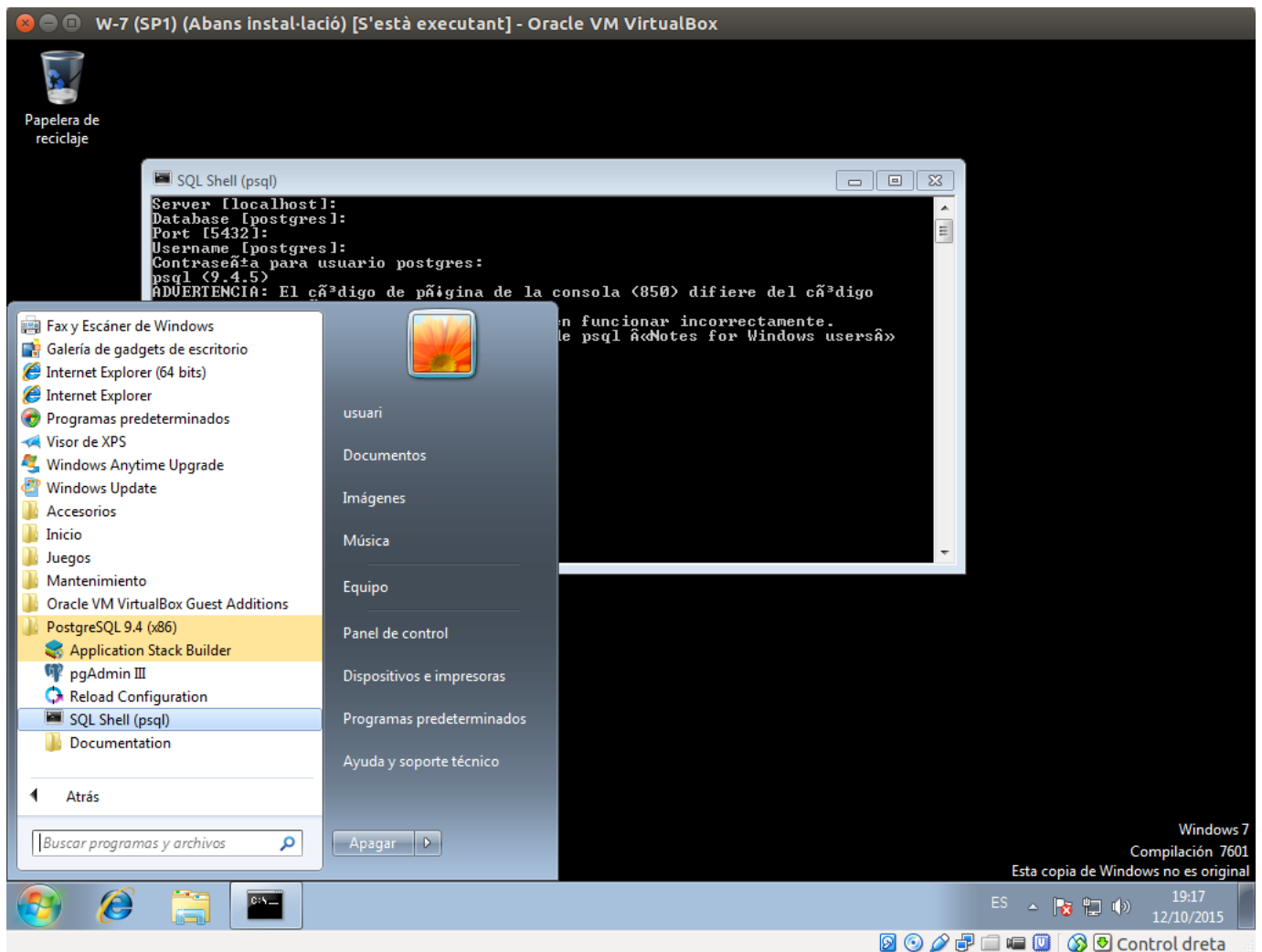
Nota

Observeu com ara ens estem connectant des del mateix servidor

En la instal·lació que havíem fet tenim en el menú un accés directe.



I des de Windows es faria de forma totalment similar:



En realitat no és un accés directe a psql, sinó a un script que demana uns paràmetres (usuari, Base de Dades, servidor, ...) i després crida a psql. Ho podem deixar tot en blanc, excepte la contrasenya (postgres)

```

• runpsql.sh wait
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:
psql bin (9.4.4)
Type "help" for help.

No entry for terminal type "xterm";
using dumb terminal settings.
postgres=#

```

De tota manera pot ser molt convenient conèixer un poc tant els paràmetres de psql com els comandos, una vegada hem connectat. Recordeu que el lloc on està situat és:

- En Linux: `/opt/PostgreSQL/9.6/bin/psql`
- En Windows: `c:\Program Files (x86)\PostgreSQL\9.6\bin\psql.exe`

Des d'un terminal

Per a connectar directament com a usuari postgres:

```
$ /opt/PostgreSQL/9.6/bin/psql -U postgres
```

tal i com ens mostra la següent imatge:

```
~$ /opt/PostgreSQL/9.6/bin/psql -U postgres
Password for user postgres:
psql.bin (9.6.2)
Type "help" for help.

Cannot read termcap database;
using dumb terminal settings.
postgres=#
```

Per a connectar-nos com distints usuaris, o a bases de dades distintes, haurem de jugar amb les opcions

El format total és el següent:

```
psql [opcions]...[base-de-dades [usuari]]
```

En les opcions podem posar, entre altres:

-?	ajuda
-d nom_bd	nom de la base de dades (no cal posar -d)
-c	per a executar tan sols un comando
-f fitxer	per a executar les sentències del fitxer especificat
-l	per a llistar les Bases de Dades existents
-h nom	nom o adreça del servidor a què ens connectem (per defecte local)
-p port	port del servidor a través del qual ens connectem (per defecte 5432)
-U nom	nom de l'usuari
-W	per a que ens demane obligatòriament la contrasenya.

A partir d'ara ja podrem posar les sentències SQL, que poden ocupar més d'una línia i que han de finalitzar per ;

Amb les tecles de moviment de cursor podem anar a les sentències anteriors (però només una línia) per a poder modificar alguna cosa i tornar a executar.

A banda podem posar alguns comandos (ja dins de **psql**):

\c nom_bd	per a connectar a una altra Base de Dades
\e	invoca un editor (per defecte vi en Linux i notepad en Windows) per editar l'última sentència
\l	llista de les Bases de Dades existents
\d	llista de les taules
\d taula	descripció de la taula
\g [fitxer]	executa l'actual sentència i, en tot cas, envia el resultat al fitxer
\i fitxer	executa la sentència (o sentències) del fitxer
\w fitxer	guarda al fitxer la sentència del buffer
\q	eixir de psql

Encara que normalment en una consulta SQL s'ha de posar la taula o taules que proporcionen les dades, PostgreSQL permet no posar cap origen de dades. Això ens permetrà fer consultes per a fer càlculs utilitzant els distints operadors i funcions.

Per exemple, per a que ens done la data d'avui:

```
SELECT NOW() ;
```

La realització d'aquesta pràctica és obligatòria

En aquesta pràctica guiada anem a introduir unes dades de prova, a partir d'un fitxer anomenat **dades_geo.sql** (el fitxer el teniu en el curs). En realitat es crearà un nou usuari **geo** i una nova Base de Dades **geo**. En aquesta Base de Dades és on s'insertiran les dades. Primer haureu d'entrar al **psql** com a usuari **postgres**. Després haureu d'executar el comando (mireu que la ruta pot canviar en el vostre servidor):

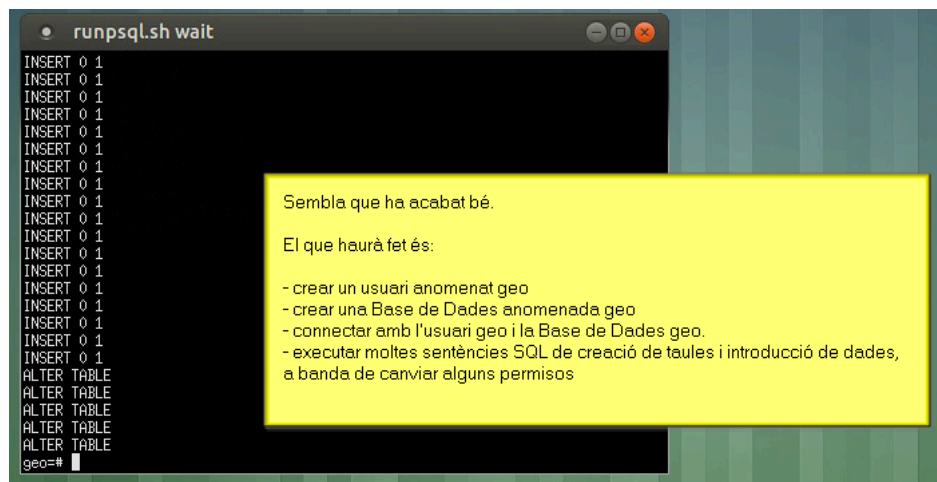
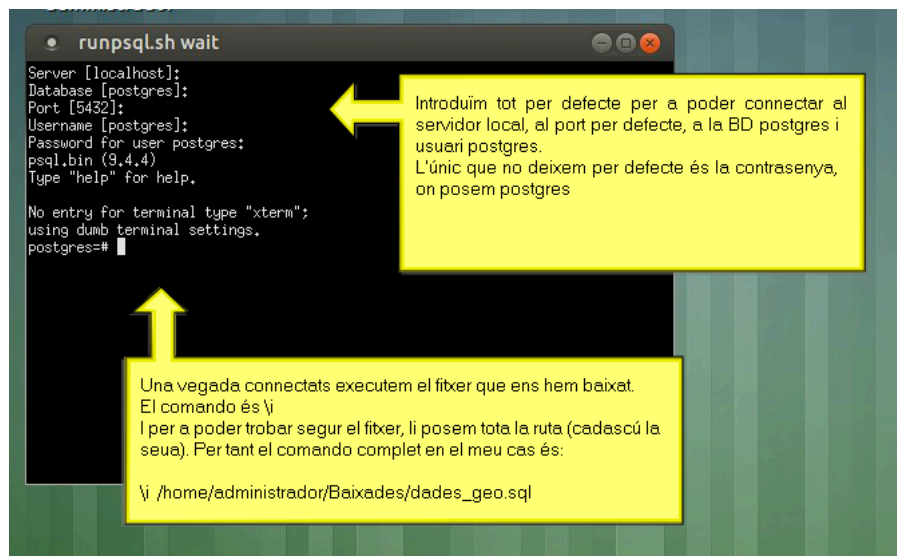
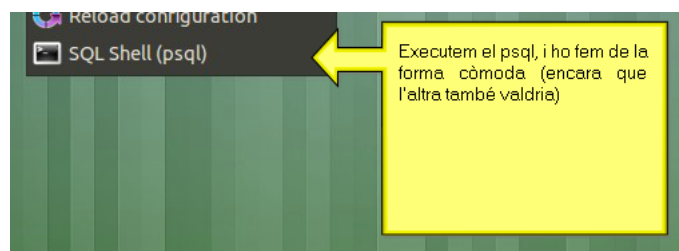
```
\i /home/administrador/Baixades/dades_geo.sql
```

1. Aquesta pràctica consistirà en importar unes dades per a poder practicar mínimament. El més còmode es treballar en una màquina virtual

Les dades estan en el nostre curs en l'apartat **-recursos-** del tema. Guardem el fitxer per a tenir-lo disponible.



2. Executem el psql o des d'un terminal



```
runpsql.sh wait
(4 rows)
geo=# \l
      List of databases
  Name | Owner  | Encoding | Collate | Ctype  | Access privileges
-----+-----+-----+-----+-----+-----
 geo   | geo    | UTF8     | ca_ES.UTF-8 | ca_ES.UTF-8 |
 postgres | postgres | UTF8     | ca_ES.UTF-8 | ca_ES.UTF-8 |
 template0 | postgres | UTF8     | ca_ES.UTF-8 | ca_ES.UTF-8 | =c/postgres +
 template1 | postgres | UTF8     | ca_ES.UTF-8 | ca_ES.UTF-8 | postgres=CTc/postgres +
        |         |         |         |         | postgres=CTc/postgres
(4 rows)

geo=# \d
      List of relations
 Schema | Name      | Type  | Owner
-----+-----+-----+-----
 public | comarques | table | geo
 public | instituts | table | geo
 public | poblacions | table | geo
(3 rows)

geo=#
```

Amb \l podem veure les Bases de Dades existents (Bases de Dades equivalents als esquemes d'Oracle o les Bases de Dades d'Access)

Amb \d podem veure les taules de l'actual Base de Dades (geo)



Exercici 1

Després d'haver fet la pràctica guiada anterior, en la qual hem introduït les dades de prova:

Quantes files hi ha en la taula **poblacions** de la Base de Dades **geo** ?.

Com que el PSQL és tan àrid, pràcticament s'imposa la utilització d'alguna eina que faci com a mínim més agradables les coses. Una de les més completes és **PGADMIN**. Ens permetrà disposar d'editors SQL i de procediments molt més agradables, a banda de la possibilitat de poder administrar la Base de Dades des d'ací.

Amb la instal·lació automàtica de PostgreSQL s'ha instal·lat també pgAdmin 4

Podem cridar el programa des d'una terminal (`/opt/PostgreSQL/9.6/pgAdmin 4/bin/pgadmin4`), però sempre serà més còmode buscar-lo amb el ratolí:

- En Ubuntu: **Aplicacions** → **PostgreSQL 9.6** → **pgAdmin 4**.
- En Windows: **Inicio** → **Todos los programas** → **Postgresql 9.6** → **pgAdmin 4**.

Tenim unes quantes maneres de poder arrancar o parar el servidor.

- **Automàtica.**

La immensa major part de les vegades l'arrancada i parada del sistema es farà de forma automàtica, quan arranquem o perem el S.O. Açò ho aconseguim per tenir el script en **/etc/init.d**, (que es posa automàticament durant la instal·lació per paquets, però no en la instal·lació a mà) i per tenir en els nivells d'execució corresponents els **start** i **kill**

- **Script en /etc/init.d** (només en Linux).

El que sempre podem fer, si estem en Linux, és aprofitar el **script** situat en **/etc/init.d** per arrancar, parar, rearrancar i veure l'estat (com qualsevol altre servei). Ho farem així (fixeu-vos que fem **sudo** per a executar-lo com a **root**):

```
$ sudo /etc/init.d/postgresql-9.6 opció
```

on l'opció pot ser:

start	arranca el servidor
stop	el para
restart	rearranca (el para, i el torna a posar en marxa)
condrestart	igual que l'anterior, però només en el cas que estiga en marxa (si estava parat, no fa res)
status	veure l'estat en què es troba actualment (en marxa, parat, ...), i altres informacions d'interès

- Hi ha més formes, però no les veurem ací per excedir la finalitat que busquem



Exercici 2

1.- Escriu exactament el que contesta la instrucció i adjunta una imatge

```
/etc/init.d/postgresql-9.6 status
```

Bàsicament són 3 els fitxers de configuració:

- **postgresql.conf** on estan la major part de paràmetres
- **pg_hba.conf** per a controlar les connexions
- **pg_ident.conf** es combina amb l'anterior per a les connexions

Si hem fet una instal·lació automàtica, tant en Linux com en Windows, els fitxers es trobaran en el directori de dades, que en Linux és **/opt/PostgreSQL/9.6/data**, i en Windows és **C:\Archivos de programa (x86)\PostgreSQL\9.6\data**.

Nota

Per a poder editar aquestos fitxers, no hi haurà problemes des de Windows, però des de Linux haureu de tenir permís. Haureu de connectar-vos com a superusuari (**sudo su**), i després, o bé utilitzar **vi**, **gedit**, etc o bé executar l'explorador d'arxius, amb el qual podreu navegar per tot el sistema de fitxers (ja que l'heu executat com a superusuari), i quan trobeu el fitxer també el podreu editar.

Per a poder protegir les dades entre les múltiples persones que es poden connectar a una Base de Dades, que les dades no puguin ser vistes per altres (a no ser que ho vulguem expressament) i molt menys que puguin ser manipulades conscient o inconscientment, els SGBD utilitzen l'autenticació per **usuaris**. En principi cada usuari només pot accedir a les seues taules, les que crea ell (és el propietari). Però també podem donar permisos. Així un usuari podrà permetre la utilització d'una taula a un altre, donant-li diferents graus d'accés: només consultar, o també afegir, o esborrar, o actualitzar, ..., o totes juntes, i fins i tot donar-li permís per a que a la seua vegada done permisos a uns altres.

Quan molts usuaris han de tenir permisos similars, és convenient la utilització de **grups d'usuaris**. Així, per exemple, si donem permís d'accés a una determinada taula a un grup d'usuaris, és com si haguérem donat el permís un a un a tots els usuaris que pertanyen al grup. Per tant facilitaràn molt la feina de l'administrador.

Fins la versió 8.0, **PostgreSQL** gestionava els usuaris i grups com a tals (**user** i **group**), però a partir de la versió 8.1 generalitza aquestos dos conceptes en un: el **rol (role)**. Abans eren dues entitats diferents. Ara és una sola, i un rol pot actuar com un usuari, un grup o ambdós. L'única diferenciació serà que alguns rols es poden connectar (login) i s'anomenen **rols d'entrada**, que seria l'equivalent d'usuari; i uns altres no (que seria l'equivalent de grup) i s'anomenen justament **rols de grup**.

Per una altra banda, PostgreSQL és molt potent en quant a la limitació de la connexió dels usuaris, podent donar més d'un sistema d'autenticació, i limitar molt l'accés des de màquines remotes, controlant tant l'usuari com la IP de la màquina des d'on es vol connectar.

Per últim, la seguretat dels fitxers queda garantida per propietari, **postgres**.

6.1 Gestió de rols: utilització com a usuaris

Un **rol** és una entitat capaç d'arreglar permisos i privilegis. Un d'aquests permisos és el de connexió (login). Aquest **rol** o **usuari** serà un nom amb una possible contrasenya (que depenent del mètode d'autenticació, servirà per controlar l'accés) que tindrà distints permisos per a crear taules i altres objectes en una determinada Base de Dades, utilitzar-los per a fer consultes o actualitzacions, ...

A banda d'açò, els permisos d'un rol poden assignar-se a un altre. Aleshores és com si el segon rol pertanyera al primer, i el primer funciona com un grup.

Els rols no estan inclosos en cap base de dades particular. Per tant són globals a tota la instal·lació de PostgreSQL (la gran Base de Dades).

Els rols, tant si poden fer login com si no, es guardaran en la taula **pg_authid**. Per comoditat (i compatibilitat amb versions anteriors) hi ha unes vistes que poden facilitar la consulta.

- **pg_roles** conté tots els rols.
- **pg_user** conté els usuaris, és a dir, els rols que poden fer un login.
- **pg_shadow** conté també les contrasenyes.
- **pg_group** conté els grups, és a dir, els rols que no poden fer login.

Per tant la primer manera de gestionar els rols seria manipular directament les taules o vistes, encara que sembla massa fort.

Anem a veure les maneres normals de crear rols.

CREATE ROLE

Haurem d'executar aquesta sentència SQL des d'un usuari amb permís per a crear rols, i connectat a qualsevol B.D.

La sintaxi és la següent:

```
CREATE ROLE nom [ [ WITH ] opció [ ... ] ]
```

on l'opció pot ser (les subratllades són les opcions per defecte):

SUPERUSER <u>NOSUPERUSER</u>	permís de superusuari (per defecte no)
CREATEUSER <u>NOCREATEUSER</u>	similar a l'anterior (obsoleta)
CREATEDB <u>NOCREATEDB</u>	permís per a crear B.D (per defecte no)
CREATEROLE <u>NOCREATEROLE</u>	permís per a crear usuaris (per defecte no)
INHERIT <u>NOINHERIT</u>	determina si el rol hereta les propietats dels grups (rols) als quals pertany
LOGIN <u>NOLOGIN</u>	permís per a connectar-se (serà un usuari)
CONNECTION LIMIT <u>connlimit</u>	Nombre màxim de connexions concurrents que pot haver (per defecte -1, que vol dir ilimitades)
[ENCRYPTED UNENCRYPTED] PASSWORD 'password'	contrasenya de l'usuari que pot anar encriptada o no
VALID UNTIL 'data'	data de caducitat de l'usuari
IN ROLE rolename [, ...]	rols (grups) als quals pertany
IN GROUP rolename [, ...]	similar a l'anterior (obsoleta)
ROLE rolename [, ...]	rols que pertanyeran a aquest rol (grup)
USER rolename [, ...]	similar a l'anterior (obsoleta)
ADMIN rolename [, ...]	similar a l'anterior però a més tindran permís per a administrar-lo (WITH ADMIN OPTION)

Per mantenir la compatibilitat amb versions anteriors tenim la sentència equivalent:

```
CREATE USER nom [ [ WITH ] opció [ ... ] ]
```

on per defecte sí que tindrà el privilegi LOGIN.

Així, per exemple, des d'una connexió per **PSQL** com a **postgres** a **geo** podem fer:

```
CREATE ROLE geo1 LOGIN;
```

pot connectar-se, no té password i no pot crear ni usuaris ni B.D.

```
CREATE ROLE geo2 LOGIN PASSWORD 'geo2';
```

pot connectar-se, té password, no pot crear ni usuaris ni B.D.

Si volem modificar algun aspecte de l'usuari, ho farem amb la sentència

```
ALTER ROLE nom [ [ WITH ] opció [ ... ] ];
```

on les opcions són les mateixes que en CREATE ROLE, per exemple:

```
ALTER ROLE geo1 PASSWORD 'geo1';
```

Si volem eliminar un rol, senzillament

```
DROP ROLE nom;
```

createuser

Una altra manera de crear rols, aquesta vegada des del sistema (sense connectar prèviament a PostgreSQL, ni amb psql ni amb cap eina gràfica). És un fitxer executable que proporciona PostgreSQL. En principi s'intentarà executar com un usuari de B.D. amb el mateix nom que l'usuari de S.O. I com de moment només tenim un superusuari (més concretament un usuari amb permís per a crear usuaris) més val executar-lo com a usuari de S.O. **postgres**, encara que ho podem esquivar amb l'opció **-U**.

La sintaxi és:

```
createuser [opcions] nom
```

I en les opcions, entre d'altres, podem posar:

- s** el nou usuari serà superusuari (si no ho posem ens ho demanarà)
- S** el nou usuari no serà superusuari
- d** el nou usuari podrà crear bases de dades (si no ho posem ens ho demanarà)
- D** el nou usuari no podrà crear bases de dades
- r** el nou usuari podrà crear usuaris (si no ho posem ens ho demanarà)
- R** el nou usuari no podrà crear usuaris
- l** el nou usuari podrà connectar-se (per defecte)
- L** el nou usuari no podrà connectar-se
- P** Demanarà la contrasenya per al nou usuari

A banda d'aquestes opcions podem posar unes altres, similars a les opcions de **psql**

- h nom** nom del servidor a què ens connectem (per defecte local)
- p port** port del servidor a través del qual ens connectem (per defecte 5432)
- U nom** nom de l'usuari que fa l'operació (no el que es crea)
- W** per a que ens demane obligatòriament la contrasenya de l'usuari *que fa l'operació*

Així, si en la consola no estem com a usuari postgres, podem fer:

```
createuser -U postgres
```

Ací tenim un exemple en què li diem expressament que l'usuari creat no siga superusuari, no puga crear Bases de Dades i no puga crear rols. Observeu que la contrasenya que demana és la de qui executa l'ordre. Per tant s'haurà de posar **postgres**.

```
$ /opt/PostgreSQL/9.4/bin/createuser -U postgres -S -D -R geo3  
Password:
```

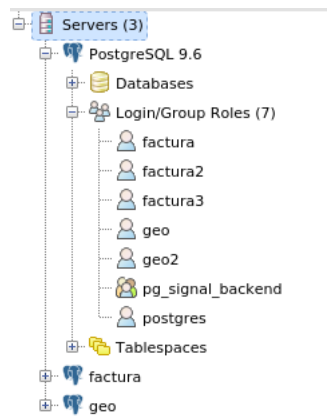
No existeix el programa **alteruser**, per a modificar un rol, però sí **dropuser** per eliminar-lo

```
dropuser [opcions] nom
```

on les opcions són algunes del createuser, amb el mateix significat: **-h -P -U -W**

Eines gràfiques: pgAdmin

Per a donar d'alta un rol utilitzat com un usuari amb PgAdmin haurem d'anar a l'opció **Rols d'entrada (Login Roles)** havent-nos autenticat com un usuari (un rol) que pot crear usuaris (rols), que en principi pot ser **postgres**.



Podem observar que els rols no estan dins de les Bases de Dades.

Si intentem crear un usuari nou, veurem que tenim totes les possibilitats que teníem en la creació per SQL, organitzades en pestanyes.

The screenshot shows a dialog box titled 'Create - Login/Group Role'. It has a tabbed interface with the following tabs: 'General', 'Definition', 'Privileges', 'Membership', 'Parameters', 'Security', and 'SQL'. The 'General' tab is selected. It contains two input fields: 'Name' and 'Comments'. At the bottom of the dialog, there are four buttons: 'i' (information), '?' (help), 'Save' (with a floppy disk icon), and 'Cancel' (with an 'X' icon). There is also a 'Reset' button (with a circular arrow icon) to the right of the 'Cancel' button.

De la mateixa manera podrem modificar un usuari (anant a les seues propietats) o eliminar-lo.

Aquestos rols d'entrada o usuaris són absolutament independents dels usuaris del S.O. (que ja s'encarrega ell de que s'autentiquen). Poden haver usuaris d'ambdós tipus amb el mateix nom, cosa que pot facilitar l'autenticació, però no té perquè ser així. Perfectament poden haver usuaris de la B.D. que no existeixen en el S.O. i a l'inrevés. Però si no fem res més, aquestos usuaris no podran connectar-se.



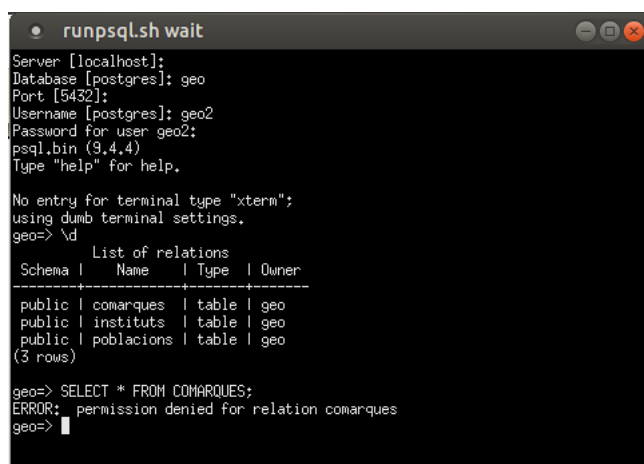
Exercici 3

Crea un usuari anomenat **geo4** que es pugui connectar , amb contrasenya **geo4** i que pugui crear Bases de Dades. Fes-lo preferiblement des de **psql**. També ho pots fer des de **pgAdmin**, però no t'oblides de consular la sentència SQL que es genera.

1. Escriu la sentència SQL que s'ha executat per a crear-lo i adjunta una imatge.

De moment, la configuració és que tots els usuaris poden connectar a totes les Bases de Dades. Però en principi només qui crea una taula (o qualsevol objecte) pot accedir a ella (llevat del superusuari). Però aquest propietari pot donar permisos als altres rols per a llegir-la, inserir, esborrar, ... També ho podrà fer el superusuari. Aquestos permisos no inclouran el d'esborrar la taula o modificar-la, que només ho podrà fer el seu propietari.

Anem a comprovar el que acabem de dir, que encara que tots els usuaris poden connectar-se a totes les Bases de Dades, no podran accedir als objectes d'ella, a no ser que li'n donem amb GRANT.



```
Server [localhost]:
Database [postgres]: geo
Port [5432]:
Username [postgres]: geo2
Password for user geo2:
psql.bin (9.4.4)
Type "help" for help.

No entry for terminal type "xterm";
using dumb terminal settings.
geo=> \d
          List of relations
Schema | Name      | Type  | Owner
-----+-----+-----+-----
public | comarques | table | geo
public | instituts | table | geo
public | poblacions | table | geo
(3 rows)

geo=> SELECT * FROM COMARQUES;
ERROR:  permission denied for relation comarques
geo=>
```

Ens hem connectat a la Base de Dades **geo** com a usuari **geo2**. No hem tingut problema en connectar. Fins i tot veiem les taules de la Base de Dades **geo** (amb `\d` podem llistar-les).

Però no podem accedir al contingut de cap taula.

GRANT

Per a donar permisos utilitzem **GRANT**. Podrem donar 3 tipus de permisos:

- Permisos sobre taules

La seua sintaxi és

```
GRANT { {SELECT|INSERT|UPDATE|DELETE|REFERENCES|TRIGGER} [,...] | ALL [ PRIVILEGES ] }
ON [ TABLE ] nom_taula [, ...]
TO { nom_usuari | GROUP nom_grup | PUBLIC } [, ...]
[ WITH GRANT OPTION ]
```

És a dir, que sobre una taula es pot donar permís només per a seleccionar, o inserir, o modificar, o esborrar, o crear una clau externa sobre aquesta taula, o crear un trigger sobre aquesta taula. O molts d'ells. O tots (**ALL**).

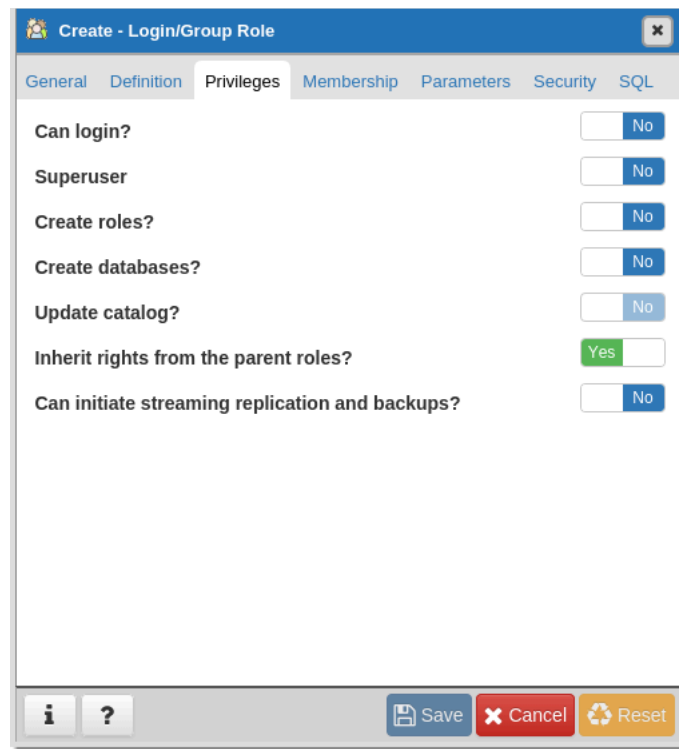
Es pot donar permís a un usuari (o més) o a un grup, o a **PUBLIC**, és a dir a tot el món. En realitat la distinció entre usuari i grup és per a mantenir compatibilitats, ja que ja sabem que ara tot són rols.

Si a més posem l'opció **WITH GRANT OPTION**, els usuaris als que hem donat permís poden donar aquestos permisos a uns altres.

Quan es dóna un permís a un rol al qual pertanyen una sèrie de membres, aquestos heretaran els permisos només si tenen el privilegi **INHERIT**.

Nota

En **PgAdmin** es veu molt gràficament en la segona pestanya de les propietats d'un usuari



Per exemple, en l'usuari i Base de dades **geo** (entrem com a usuari:geo i li donem permisos a l'usuari: geo2):

```
GRANT SELECT,UPDATE ON COMARQUES,POBLACIONS TO geo2;
```

Ara l'usuari **geo2** podrà accedir al contingut de les taules comarques i poblacions. Fins i tot poden actualitzar les files ja existents i podem comprovar com ara sí que podem accedir al seu contingut.

```
runpsql.sh wait
Server [localhost]:
Database [postgres]: geo
Port [5432]:
Username [postgres]: geo2
Password for user geo2:
psql bin (9.4.4)
Type "help" for help.

No entry for terminal type "xterm";
using dumb terminal settings.
geo-> SELECT * FROM COMARQUES;
 nom_c | provincia
-----+-----
 Safor | València
 Horta Sud | València
 Camp de Morvedre | València
 Foia de Bunyol | València
 Alacantí | Alacant
 Alt Maestrat | Castelló
 Plana Baixa | Castelló
 Horta Nord | València
 Ports | Castelló
 Racó | València
 Plana d'Utiel | València
```

REVOKE

Per a llevar els permisos utilitzarem **REVOKE**

- Per a llevar permisos sobre taules

```
REVOKE [ GRANT OPTION FOR ] { {SELECT|INSERT|UPDATE|DELETE|REFERENCES|TRIGGER} [,...] | ALL [
PRIVILEGES ] }
ON [ TABLE ] nom_taula [, ...]
FROM { nom_usu | GROUP nom_grup | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]
```

On haurem de tenir en compte que només l'usuari que ha donat un permís o el superusuari pot llevar-lo, excepte en el cas de l'opció **CASCADE**, que lleva també el permís a tots els que se l'han anat passant per tenir el permís **WITH GRANT OPTION**, si és el cas.

Una Base de Dades és un lloc on pot haver un o més d'un esquema, i en cada esquema es poden crear molts objectes.

La jerarquia serà la següent: **Servidor -> Base de Dades -> Esquema -> Taula**.

Un usuari que es connecte a una Base de Dades podrà veure (si té permís) els objectes dels diferents esquemes d'aquesta Base de Dades.

CREACIÓ

- La creació l'efectuarem amb la sentència SQL:

```
CREATE DATABASE nom
[ [ WITH ] [ OWNER [=] nom_propietari ]
[ TEMPLATE [=] template ]
[ ENCODING [=] encoding ]
[ TABLESPACE [=] tablespace ]
[ CONNECTION LIMIT [=] num_conn ] ]
```

En principi el propietari de la Base de Dades serà qui la crea (haurà de tenir permís per a crear-ne: **CREATEDB**). Si som super usuaris podrem fer que el propietari siga un altre.

Quan es crea una Base de Dades en realitat, per anar més ràpid, el que fa és copiar-la d'una altra, una espècie de plantilla. Podem triar aquesta plantilla o Base de Dades original: **template1** o **template0** (per defecte **template1**).

La clàusula **encoding** servirà per especificar un conjunt de caràcters (aconsellable UTF-8).

Podem especificar en quin **Tablespace** es guardarà la Base de Dades i els seus objectes.

També es pot dir quantes connexions concurrents es poden fer. Per defecte (-1) no hi ha límit.

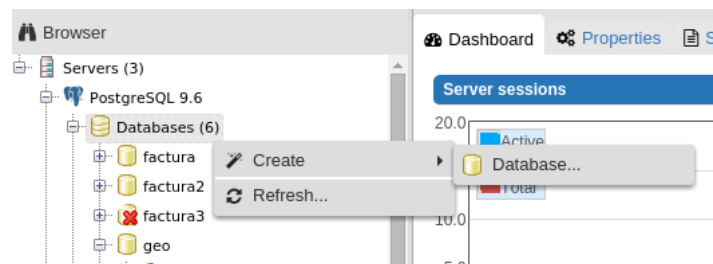
Tot açò també es podia fer amb una utilitat proporcionada per PostgreSQL, des d'una sessió de l'usuari de S.O. **postgres** (o un altre que tinga permís per crear BD)

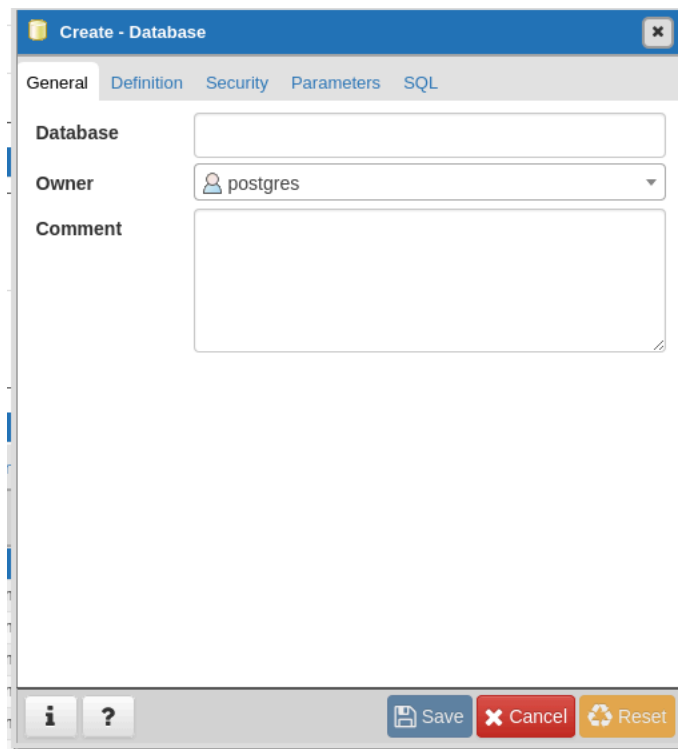
```
$ createdb [ opcions ] nom
```

on algunes opcions, a banda de les d'especificar el host, demanar contrasenya, etc., són:

-O (owner) **-T** (template) **-E** (encoding) **-D** (tablespace)

- També es pot fer des de **PgAdmin** si apremem el botó de creació d'objectes, quan estiguem situats en el servidor o en una Base de Dades (no en un nivell inferior). Tindrem les mateixes opcions que en el cas de la sentència SQL, en diferents pestanyes





MODIFICACIÓ

- La sentència **ALTER DATABASE** permet variar els paràmetres de la B.D., o renomemar-la.
- En **pgAdmin** anirem a les propietats de la Base de Dades.

ELIMINACIÓ

- Es pot utilitzar la sentència SQL **DROP DATABASE**, o la utilitat **dropdb**.
- En **pgAdmin**, la paperera.



Exercici 4

Crea un Base de Dades anomenada **geo4**, el propietari de la qual ha de ser **geo4**.

Posa tant l'usuari com estaves connectat, com la sentència SQL que t'ha permès crear-la (adjunta captura de pantalla)

Aquesta tasca tan fonamental de l'Administrador de la Base de Dades és molt senzilla de fer en PostgreSQL, i es pot planificar amb un senzill script que s'execute periòdicament.

Hi ha bàsicament dues maneres de fer el backup:

- Amb ***pg_dump***
- Copiant els fitxers

Sempre tindrem, però, la possibilitat de fer-la còmodament des de **PgAdmin**.

El mètode que utilitza **Pg_Dump** consisteix en generar un fitxer de text amb els comandos SQL necessaris per a refer la Base de Dades tal i com estava en el moment de fer el **dump**.

Utilitzarem el programa **pg_dump**, situat on tots els programes, i que té la següent sintaxi:

```
pg_dump [ opcions ] nom_bd
```

El resultat anirà per l'eixida estàndard, la pantalla. Per tant segurament sempre ho gastarem així:

```
pg_dump nom_bd > fitxer_eixida
```

Si volem que no es guarde en el directori actiu haurem de posar la ruta també. Com a qüestió d'estil, podríem acostumar-nos a posar sempre l'extensió **.sql**

Ara tindrem en aquest fitxer les sentències SQL per a deixar la Base de Dades com estava.

El pot executar qualsevol usuari, i sobre qualsevol Base de Dades, però evidentment si l'usuari no té permís d'accés sobre la B.D. fallarà.

Com que **pg_dump** és una aplicació client de PostgreSQL es podrà executar des de qualsevol lloc, en local o remot. I l'autenticació serà igual que el que hem vist fins ara.

Aquestes són algunes de les opcions:

-a	només guarda les dades
-s	només guarda l'estructura
--inserts	inclou sentències INSERT per a les dades (sinó posarà sentències COPY)
--column-inserts	el mateix però explicitant els noms de les columnes
-f fitxer	envia el resultat a un fitxer; si no s'especifica, l'envia a l'eixida estàndard.
-F format	indica el format amb què es guardarà el fitxer (p fitxer de text; t de tipus tar; c de tipus custom, que és la més flexible; amb el primer haurem de restaurar utilitzant psql ; amb els dos últims utilitzarem pg_restore)
-n nom	inclou només l'esquema especificat
-t nom	inclou només la taula especificada

A banda estaran les habituals opcions de connexió: **-h (host)**, **-p (port)**, **-U (usuari)** i **-W**

Aquesta seria la manera de fer còpia de seguretat de la Base de Dades **geo**, guardant-la en un fitxer del directori actiu anomenat **geo.sql**.

```
$ /opt/PostgreSQL/9.6/bin/pg_dump -U geo > geo.sql
```

Per a fer un **dump** no cal parar el servidor.

Una variant és el **pg_dumpall** que fa el dump sobre totes les bases de dades (segurament aquest és el candidat per a còpies de seguretat periòdiques). Evidentment l'haurem de fer com a **postgres**, per a poder tenir accés a totes les B.D. Té l'avantatge addicional que també guarda els objectes que no pertanyen a una determinada Base de Dades, com són els usuaris. L'inconvenient és que haurem de posar la contrasenya de **postgres** tantes vegades com Bases de Dades tinguem, perquè s'ha de connectar a cadascuna d'elles. Per a poder evitar açò podríem utilitzar un altre mètode d'autenticació per a l'usuari **postgres: ident**. La següent imatge mostra el procés de fer una còpia de seguretat de tot el cluster. S'ha introduït 5 vegades la contrasenya de l'usuari postgres, perquè 5 són les bases de dades existents.


```
administrador@Ubuntu:~  
Fitxer Edita Visualitza Cerca Terminal Ajuda  
administrador@Ubuntu:~$ /opt/PostgreSQL/9.4/bin/pg_dumpall -U postgres > tota.sql  
Password:  
Password:  
Password:  
Password:  
administrador@Ubuntu:~$
```

La còpia de seguretat s'haurà guardat en el fitxer **tota.sql**. Si veiem el seu contingut, haurà guardat totes les sentències per a guardar les 5 Bases de Dades existents, i també tots els rols (d'usuari i de grup) existents.

```
Ubuntu 14.04 clona (Després usuaris) [S'està executant] - Oracle VM VirtualBox  
Aplicacions Llocs Sistema  
tota.sql (~) - gedit  
Fitxer Edita Visualitza Cerca Eines Documents Ajuda  
Obre Desa Desfés  
tota.sql x  
--  
-- PostgreSQL database cluster dump  
--  
SET default_transaction_read_only = off;  
  
SET client_encoding = 'UTF8';  
SET standard_conforming_strings = on;  
  
--  
-- Roles  
--  
  
CREATE ROLE emp1;  
ALTER ROLE emp1 WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN NOREPLICATION PASSWORD  
'md5cd6ef4e86ec3e4a45fc9b9936ab098b2';  
CREATE ROLE emp2;  
ALTER ROLE emp2 WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN NOREPLICATION PASSWORD  
'md51cce225a68f03917457c9a95937147f2';  
CREATE ROLE empresa;  
ALTER ROLE empresa WITH NOSUPERUSER INHERIT CREATEROLE CREATEDB LOGIN NOREPLICATION PASSWORD  
'md595c943899e89329311489bcc90941ca2';  
CREATE ROLE g_geo;  
ALTER ROLE g_geo WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB NOLOGIN NOREPLICATION;  
CREATE ROLE g_geo2;  
ALTER ROLE g_geo2 WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB NOLOGIN NOREPLICATION;  
CREATE ROLE geo;  
ALTER ROLE geo WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN NOREPLICATION PASSWORD  
'md5ac0870ed1147c3a0ba959602774869d6';  
CREATE ROLE geo1;  
  
SQL Amplada de la tabulació: 8 Ln 1, Col. 1 INSER  
administrador@Ubu... Carpeta de l'usuari tota.sql (~) - gedit  
Control dreta
```

8.2 Restauració del Dump

Si el fitxer d'eixida del **dump** (que ara anomenarem d'entrada) no tenia cap format (no hem utilitzat l'opció -F) l'executarem en el **psql**, ja que són comandos SQL. Ho podem fer de les dues maneres següents:

```
psql nom_bd < fitxer_entrada
```

```
psql -f fitxer_entrada nom_bd
```

Només s'ha especificat la Base de Dades a la qual es connecta, però com és el comando **psql**, podrien anar qualsevol de les opcions habituals (-U per a especificar l'usuari, -h per a especificar el host, ...)

Com sempre, si el fitxer no està en el directori actiu haurem de posar la ruta. Fins i tot una vegada hem entrat en la base de dades, podem fer:

```
\i fitxer_entrada
```

Hi haurà una restricció lògica, però important: tant la Base de Dades com l'usuari han d'existir (no els crea aquesta restauració)

Per a restaurar totes les Bases de Dades, ho haurem de fer com a usuari **postgres**, i millor accedint a **template1**.

Posats a fer virgueries, fins i tot podem transvasar la informació d'una BD a una altra:

```
pg_dump bd1 | psql bd2
```

que fins i tot pot estar en un altre lloc

```
pg_dump bd1 | psql -h host bd1
```

En canvi, si el fitxer d'eixida tenia algun format, perquè hem utilitzat l'opció **-Ft** o **-Fc**, haurem d'utilitzar el programa **pg_restore**, amb el qual es poden especificar més coses:

```
pg_restore [ opcions ] nom_fitxer
```

I en les opcions, entre d'altres, podem posar:

-a	només restaura les dades
-s	només restaura l'estructura
-c	esborra els objectes abans de reomplir-los
-C	crea la B.D. abans de restaurar en ella
-d nom_bd	es connecta a la B.D. i restaura directament en ella
-F format	indica el format del fitxer d'entrada, encara que no fa falta perquè ho detecta automàticament
-t nom	restaura només la taula especificada

A banda estaran les habituals opcions de connexió: **-h** (*host*), **-p** (*port*), **-U** (*usuari*) i **-W**



Exercici 5

1. Fes una còpia de seguretat (o exportació) de la taula **comarques** de la Base de Dades **geo** (propietari **geo**). Pots utilitzar l'entorn i mètode que vulgues. El fitxer on guardar-la podria ser **comarques.sql**. Explica el mètode i entorn utilitzat.
2. Restaura (o importa) el fitxer **comarques.sql** (on està la taula comarques) en la Base de Dades **geo4**, de manera que el propietari d'aquesta taula siga **geo4**. Explica de quina manera ho has aconseguit. Hauràs de controlar especialment el propietari de la taula, ja que segurament en el fitxer d'exportació ha estat **geo**, i ara volem que siga **geo4**.
3. Fes una còpia de seguretat (o exportació) de **tota** la Base de Dades **geo4**. Pots utilitzar l'entorn i mètode que vulgues, però el format ha de ser PLA (SQL) i utilitzant INSERT (no COPY). El fitxer on guardar-la ha de ser **geo4.sql**. Explica el mètode i entorn utilitzat i **adjunta el fitxer geo4.sql**.

Has de contestar tots els exercicis en un fitxer de text, adjuntant imatges en els casos que s'especifica, i també sempre que ho consideres necessari. Posa sempre el nom de l'exercici (Exercici 1, Exercici 2, ...) i també el del subapartat.

Adjunta el fitxer geo4.sql

Puja en un fitxer comprimit els dos fitxers i dóna-li el nom: **Ex_admin_cognoms_nom.odt (o pdf)**

