Gestió d'usuaris

Joan Anton Pérez Braña

Índex

Introducció

Si volem utilitzar un SGBD per a accedir a la informació continguda en una base de dades, el primer que caldrà comprovar és quines autoritzacions tenim sobre aquelles dades; d'això s'encarrega el component de seguretat de l'SGBD.

Aquest component cada dia esdevé més important, atès que ara tots els ordinadors estan interconnectats i, per tant, qualsevol persona podria esdevenir usuari d'una base de dades. En moltes organitzacions la informació és un actiu intangible i de naturalesa sensible, i per això cal saber quins són les obligacions legals que tenim.

En aquesta unitat formativa plantejarem els components lògics de control sobre les estructures de dades prèviament definides. La definició de cadascun d'aquests components es fa mitjançant sentències SQL, que ja hem estudiat quasi completament.

Les vistes havien estat durant molt de temps un simple mecanisme de simplificació de consultes, però actualment tenen una importància cabdal en diferents àrees: el disseny extern, el gestor de dades (Data Warehouse), la informàtica distribuïda. Veurem els mecanismes que incorpora el SGBD PostgreSQL que permeten de fer actualitzable qualsevol vista mitjançant la definició de regles.

També caldrà tenir en compte que en la definició d'aquests components hi poden haver divergències entre el que diu la darrera versió de l'SQL estàndard. En el nostre cas, estudiarem aquestes característiques amb el SGBD PostgreSQL, de manera que en acabar l'estudi d'aquesta unitat es pugui definir correctament en el sistema mencionat cadascun dels diferents components lògics de dades i de control.

Com a última consideració cal tenir en compte que, per aprendre els conceptes que apareixen en la unitat i aplicar amb agilitat les tècniques esmentades, serà imprescindible implementar els exemples ilustratius, efectuar totes les activitats proposades i els exercicis d'autoavaluació.

Resultats d'aprenentatge

En finalitzar aquesta unitat l'alumne/a:

1. Implanta mètodes de control d'accés utlitzant assistents, eines gràfiques i comandes del llenguatge del sistema gestor de bases de dades corporatiu.

- Coneix la normativa vigent sobre la protecció de dades.
- Identifica els diferents tipus d'usuaris d'una organització, per tal identificar els privilegis.
- Crea, modifica i elimina comptes d'usuaris; assignant privilegis sobre la base de dades i els seus objectes, garantint el compliment dels requisits de seguretat.
- Agrupa i desagrupa privilegis, per tal d'assignar i eliminar privilegis a usuaris, garantint el compliment dels requisits de seguretat.
- Agrupa i desagrupa grups de privilegis a usuaris, garantint el compliment dels requisits de seguretat.
- Assigna i desassigna rols a usuaris.
- Crea vistes personalitzades per a cada tipus d'usuari de la base de dades.

1. Gestió d'usuaris i privilegis

En un sistema informàtic les dades constitueixen un recurs valuós que ha d'estar controlat i gestionat estrictament.

Entenem per *seguretat d'un sistema* el conjunt de mecanismes de protecció enfront d'accessos no autoritzats, ja siguin intencionats o accidentals.

A més, si la informació fa referència a persones i s'emmagatzemen dades de naturalesa sensible ens caldrà saber quines són les obligacions legals que tenim.

1.1 Introducció als problemes de seguretat en les bases de dades

Quan utilitzem un sistema gestor de bases de dades (SGBD) per accedir a la informació emmagatzemada en una base de dades, primerament cal comprovar quines autoritzacions tenim sobre aquelles dades; d'això s'encarrega el component de seguretat de l'SGBD. Aquest component cada dia esdevé més important, ja que avui dia tots els ordinadors estan interconnectats i, per tant, qualsevol persona podria esdevenir un usuari potencial d'una base de dades.

En un sistema d'informació, les diferents aplicacions i usuaris de l'organització fan servir un únic conjunt de dades, anomenat *base de dades corporativa*, amb l'SGDB. D'una banda, això resol problemes de redundància, inconsistència i independència entre les dades i els programes i, de l'altra, fa que la seguretat esdevingui un dels problemes més importants en aquests entorns.

En moltes organitzacions la informació és un actiu intangible i de naturalesa sensible, i per això cal saber quines són les obligacions legals que tenim.

1.1.1 Conceptes associats a la seguretat

La paraula *seguretat* incorpora diferents conceptes. Els més importants són aquests:

1. **Confidencialitat:** cal protegir l'ús de la informació per part de persones no autoritzades. Això implica que un usuari només ha de poder accedir a la informació per a la qual té autorització i que a partir d'aquesta informació no podrà inferir altra informació que es consideri secreta.
2. **Integritat:** la informació s'ha de protegir de modificacions no autoritzades; això també inclou tant la inserció de dades falses com la destrucció de dades.

3. **Disponibilitat:** la informació ha d'estar disponible en el moment que li faci falta a l'usuari.

1.1.2 Amenaces i violacions del sistema

Per aconseguir seguretat en un entorn de base de dades és necessari identificar les amenaces a la quals pot estar subjecta i triar les polítiques i els mecanismes per evitar-les.

Definirem el concepte **amenaça** com tot aquell agent hostil que, de manera casual o intencionada i utilitzant una tècnica especialitzada, pot revelar o modificar la informació gestionada pel sistema.

Com s'ha esmentat anteriorment, les violacions sobre una base de dades consisteixen en lectures, modificacions o esborraments incorrectes de les dades. Les conseqüències d'aquestes violacions es poden agrupar en tres categories:

1. **Lectura inadequada d'informació**. Causat per la lectura de dades per part d'usuaris no autoritzats mitjançant un accés intencionat o accidental. S'inclouen les violacions del secret derivades de les deduccions d'informació que es considera secreta.
2. **Modificació impròpia de les dades**. Correspon a totes les violacions de la integritat de les dades per tractaments o modificacions fraudulentes d'aquestes. Les modificacions impròpies no involucren necessàriament lectures no autoritzades, ja que les dades es poden falsificar sense ser llegides.
3. **Denegació de serveis**. Correspon a accions que puguin impedir que els usuaris accedeixin a les dades o utilitzin els recursos que tenen assignats.

Les amenaces a la seguretat es poden classificar d'acord amb la manera en què poden ocórrer.

1. **Amenaces no fraudulentes.** Les amenaces no fraudulentes són accidents casuals, entre els quals es poden distingir els següents:
    - *Desastres naturals o accidentals*: normalment són accidents que danyen el maquinari del sistema, com per exemple aquells produïts per terratrèmols, inundacions o foc.
    - *Errors del sistema*: corresponen a tots aquells errors accidentals en el maquinari o en el programari que poden conduir a accessos no autoritzats.
    - *Errors humans*: corresponen a aquelles errades involuntàries derivades de l'acció dels usuaris en introduir dades o utilitzar aplicacions que treballen sobre aquestes.

2. **Amenaces fraudulentes.** Aquestes amenaces generen violacions intencionades i són causades per dos tipus d'usuaris diferents:

   - *Usuaris autoritzats* que abusen dels seus privilegis.
   - *Agents hostils o usuaris impropis* que executen accions de vandalisme sobre el programari o el maquinari del sistema o també lectures o escriptures de dades.

1.1.3 Nivells de seguretat

Com hem esmentat, la seguretat de les bases de dades es refereix a la protecció enfront d'accessos malintencionats. No és possible una protecció absoluta de la base de dades contra aquest mal ús, però es pot incrementar suficientment el cost per a qui el comet per dissuadir-lo en la major part, si no en la totalitat, de tenir accés a la base de dades sense l'autorització adequada. Per protegir la base de dades s'han d'adoptar mesures a diferents nivells:

En diferent documentació veureu que s'utilitza l'expressió anglesa *database management system* (DBMS) per fer referència als sistemes gestors de bases de dades.

- **Sistema gestor de base de dades:** pot ser que alguns usuaris de la base de dades solament tinguin accés a una part limitada de la base de dades. Pot ser que altres usuaris tant sols tinguin autorització per fer consultes però que no puguin modificar les dades. És responsabilitat de l'administrador de l'SGBD que no es violin aquestes restriccions d'autorització.
- **Sistema operatiu:** independentment del nivell de seguretat assolit en l'SGBD la debilitat de la seguretat del sistema operatiu pot servir com a mitjà per a accessos no autoritzats a la base de dades.
- **Xarxa:** atès que gairebé tots els sistemes de bases de dades permeten l'accés remot mitjançant terminals o xarxes, la seguretat en el nivell de programari de la xarxa és tan important com la seguretat física, tant a Internet com en les xarxes privades de les empreses.
- **Físic:** els llocs on estan ubicats els sistemes d'informació cal que estiguin adequadament protegits contra l'entrada d'intrusos.
- **Humà:** els usuaris han d'estar degudament autoritzats per reduir la possibilitat que algun doni accés a intrusos a canvi de suborns o d'altres favors.

1.1.4 Mecanismes bàsics de seguretat emprats en l'SGBD

Els sistemes d'informació i les dades que s'emmagatzemen i es processen són recursos molt valuosos que cal protegir. Els mecanismes emprats per protegir les dades enfront d'amenaces intencionades o accidentals van des dels controls físics fins a procediments administratius.

Identificació i autenticació

La primera acció que cal fer per assolir la seguretat d'un sistema d'informació és la capacitat de verificar la identitat dels usuaris. Aquest procés està format per dues parts:

- **Identificació**: implica la manera en què l'usuari proporciona la seva identitat al sistema (veure qui és). Segons els requisits operacionals, una identitat pot descriure un individu, més d'un individu, o un o més individus només durant un període de temps.

- **Autenticació**: és la manera en què un individu estableix la validesa de la seva identitat (verificar que l'usuari és qui diu que és).

Mentre que les identitats poden ser públiques, la informació d'autenticació es desa en secret, i això proporciona el recurs pel qual es prova que és realment qui diu que és.

Les contrasenyes són el mecanisme clàssic d'autenticació. La seguretat d'aquest mecanisme depèn de la capacitat de mantenir-les en secret. L'administrador de l'SGBD s'encarregarà d'emprar l'algorisme de xifratge més adequat per a cada cas.

Les targetes, ja siguin amb banda magnètica o amb microxip incorporat, donen un sistema de seguretat més gran. En aquests casos la contrasenya proporcionada ha de coincidir amb la que hi ha emmagatzemada a la targeta i a més alguna informació de la targeta ha de coincidir amb alguna informació emmagatzemada a l'ordinador.

Val la pena esmentar que avui dia es tendeixen a utilitzar sistemes biomètrics com poden ser empremtes dactilars, veu, iris o d'altres patrons que es poden considerar únics.

Control d'accés

Són mecanismes que asseguren que els usuaris accedeixen només als llocs als quals estan autoritzats amb l'objectiu de poder fer exclusivament allò per al qual tenen permís.

> Definim *control d'accés* com el conjunt de funcions de l'SGBD per assegurar que els accessos al sistema estan d'acord amb les regles establertes per la política de protecció fixada pel model de negoci.

Així doncs, direm que el control d'accés controla la interacció (lectura, escriptura, modificació i esborrament) entre els subjectes (usuaris i processos) i els objectes als quals accedeixen (taules, esquemes, funcions, altres usuaris, etc.).

El control d'accés es pot considerar format per dos components:

1. *Polítiques d'accés*: defineixen els principis pels quals s'autoritza un usuari o es denega l'accés específic a un objecte de la base de dades.

2. *Mecanismes de seguretat*: formats per tots aquells procediments que s'apliquen a les consultes amb l'objectiu que els usuaris compleixin els principis anteriors.

Les diferents polítiques d'accés es poden classificar en control d'accés obligatori i control d'accés discrecional.

Control d'accés discrecional basat en privilegis

El control d'accés discrecional (DAC) es basa en la identitat dels usuaris o grups d'usuaris per autoritzar o restringir l'accés als diferents objectes de la base de dades. El control discrecional és el mecanisme més comú en els sistemes d'informació actuals.

DAC

DAC correspon a les sigles de *discretionary access controls*, i és el mecanisme més emprat en els SGBD actuals.

Papers

Un paper fa referència al conjunt d'autoritzacions o restriccions que té un usuari o grups d'usuaris en la base de dades.

Podem representar l'estructura de control d'accés discrecional amb una taula (taula 1.1) en què veiem que les interseccions entre files i columnes indiquen els drets de cada usuari o grup d'usuaris sobre cada objecte.

Taula 1.1. Estructura de control d'accés discrecional

	Objecte 1	**Objecte 2**	**...**	**Objecte *n***
Paper 1				Autoritzacions o restriccions dels usuaris del **paper 1** sobre l'**objecte *n***
Paper 2	Autoritzacions o restriccions dels usuaris del **paper 2** sobre l'**objecte 1**			
...				
Paper *n*		Autoritzacions o restriccions dels usuaris del **paper *n*** sobre l'**objecte 2**		

Els objectes als quals fa referència aquesta taula corresponen a objectes de la base de dades, com poden ser taules, esquemes, funcions, altres usuaris, etc.

Control d'accés obligatori per la seguretat multinivell

El control d'accés obligatori (MAC) s'acostuma a fer servir en aquelles bases de dades en les quals les dades tenen una estructura de classificació molt rígida i estàtica, com per exemple, les bases de dades militars i governamentals. Sovint aquest control d'accés es pot combinar amb el descrit anteriorment.

A continuació farem una pinzellada sobre com es fa aquest tipus de control d'accés.

Les polítiques de control d'accés obligatori es basen en la idea que cada dada té un nivell de classificació pel que fa a la seva seguretat. Les classes de seguretat usuals són:

- secret màxim (TS: *top secret*)
- secret (S)
- confidencial (C)
- no classificat (U: *unclassified*)

en què TS correspon al nivell més alt i U al més baix.

El model que se sol emprar s'anomena *Bell-LaPadula* i assigna a cada subjecte (usuari, grup d'usuaris o programa) i a cada objecte (taula, registres, atribut, etc.) una de les classificacions de seguretat descrites anteriorment. Ens referirem a la classificació del subjecte com a *classif(S)* i a la classificació de l'objecte com a *classif(O)*. Així doncs, les restriccions d'accés es basen en el següent:

- Un subjecte pot veure un objecte si i solament si $classif(S) \geq classif(O)$
- Un subjecte pot modificar un objecte si el seu nivell d'acreditació és igual que el nivell de classificació de l'objecte, és a dir, si $classif(S) = classif(O)$

Integritat i consistència

Són mecanismes perquè la base de dades resti sempre en un estat que compleixi totes les regles de negoci del model de dades, encara que es produeixin canvis.

Per assolir aquest objectiu el dissenyador de la base de dades ha hagut d'establir les regles d'integritat referencial i altres restriccions perquè en qualsevol cas els canvis indeguts tinguin el menor efecte. Cal tenir en compte l'estat dels atributs derivats que sovint s'utilitzen en una base de dades per assolir millores en el rendiment.

Auditoria

L'auditoria correspon a un conjunt de mecanismes per saber qui ha fet què, és a dir, portar un registre de qui fa tots els canvis i consultes a la base de dades. Més que un mecanisme de seguretat és un mecanisme per detectar el culpable.

S'utilitza per als casos següents:

- La investigació d'una activitat sospitosa.
- El monitoratge d'activitats específiques de la base de dades.

El sistema d'auditoria ha de permetre diferents formes d'utilització:

- Auditar sentències. L'auditoria indicarà quan i qui ha utilitzat un tipus de sentència correcta. Per exemple, auditar totes les insercions o esborraments.
- Auditar objectes. El sistema auditarà cada vegada que es faci una operació sobre un objecte determinat.

- Auditar sentències sobre objectes, una versió combinada de les dues anteriors.
- Auditar usuaris o grups.

La informació que s'acostuma a emmagatzemar quan es fa una tasca d'auditoria és el nom de l'usuari, l'identificador de la sessió, l'identificador del terminal, el nom de l'objecte al qual s'ha accedit, l'operació executada o intentada, el codi complet de l'operació, la data i l'hora.

1.1.5 El paper de l'administrador de l'SGBD en la seguretat de les bases de dades

Una de les principals raons d'emprar un SGBD és tenir un control centralitzat tant de les dades com dels accessos que fan els usuaris. La persona que fa el control central sobre el sistema s'anomena *administrador de la base de dades*. Pel que fa a la seguretat, les funcions de l'administrador de la base de dades inclouen:

1. **Definició de l'esquema.** L'administrador crea l'esquema original de la base de dades escrivint un conjunt d'instruccions de definició de dades.
2. **Definició de l'estructura i del mètode d'accés.** Referent al programari client emprat i les diferents activitats relacionades amb l'emmagatzematge i recuperació utilitzant diferents estàndards.
3. **Modificació de l'esquema i l'organització física.** Els administradors de la base de dades fan canvis en l'esquema i l'organització física per reflectir les necessitats canviants dins de l'organització, o per fer alteracions en l'organització física per millorar-ne el rendiment.
4. **Concessió d'autorització per a l'accés a les dades.** La concessió de diferents tipus d'autorització permet a l'administrador de la base de dades determinar a quines parts de la base de dades pot accedir cada usuari: la informació d'autorització es manté en una estructura de l'esquema especial que el sistema de base de dades consulta quan s'intenta fer l'accés a les dades.
5. **Manteniment rutinari.** Alguns exemples d'activitats rutinàries de manteniment de l'administrador són:
    - Còpia de seguretat periòdica de la base de dades, sobre cinta o sobre servidors remots per prevenir la pèrdua de dades a causa de desastres naturals.
    - Assegurar-se que hi ha prou espai lliure al disc per a les operacions habituals i incrementar-lo en cas que sigui necessari.
    - Supervisar les tasques que s'executen a la base de dades i assegurar-se que el rendiment no es degrada per tasques molt costoses iniciades per alguns usuaris.

1.2 L'SGBD PostgreSQL

PostgreSQL és un gestor de bases de dades relacional orientat a objectes molt conegut i usat en entorns de programari lliure perquè compleix els estàndards SQL92 i SQL99, i també pel conjunt de funcionalitats avançades que suporta, cosa que el situa al mateix nivell o a un de millor que molts SGBD comercials.

L'origen del PostgreSQL se situa en el gestor de bases de dades POSTGRES, desenvolupat a la Universitat de Berkeley, i que es va abandonar en favor del PostgreSQL a partir de 1994. Aleshores ja tenia prestacions que el feien únic en el mercat i que altres gestors de bases de dades comercials han anat afegint durant aquest temps.

El PostgreSQL es distribueix sota llicència BSD, la qual cosa en permet l'ús, la redistribució i la modificació amb l'única restricció de mantenir el copyright del programari dels seus autors, en concret el PostgreSQL Global Development Group i la Universitat de Califòrnia.

El PostgreSQL pot funcionar en múltiples plataformes: Linux, FreeBSD, Solaris, Mac OS X i Windows.

1.2.1 Procés d'instal·lació del PostgreSQL

El PostgreSQL està disponible per a la majoria de distribucions de GNU/Linux. La instal·lació és tan senzilla com executar l'instal·lador de paquets corresponent.

En Debian, el procediment següent instal·la el servidor i el client, respectivament:

```
# apt-get install postgresql

# apt-get install postgresql-client
```

En distribucions basades en RPM, els noms dels paquets són una mica diferents:

```
# rpm -Uvh postgresql-server

# rpm -Uvh postgresql
```

Una vegada instal·lat, s'escriurà un *script* d'inici que permet llençar i aturar el servei PostgreSQL; d'aquesta manera, per iniciar el servei, haurem d'executar l'ordre següent:

```
# /etc/init.d/postgresql start
```

Anàlogament per aturar el servei cal fer:

```
# /etc/init.d/postgresql stop
```

1.2.2 L'usuari postgres

En acabar la instal·lació, en el sistema operatiu s'haurà creat l'usuari *postgres*, i en PostgreSQL s'haurà creat un usuari amb el mateix nom. En aquests moments, aquest és l'únic usuari existent en la base de dades i ara, doncs, serà l'únic que podrà crear noves bases de dades i nous usuaris.

Normalment, a l'usuari *postgres* del sistema operatiu no se li permetrà l'accés des de l'intèrpret d'ordres ni tindrà contrasenya assignada, excepte en el cas que en el procés d'instal·lació ens hagi sol·licitat la seva paraula de pas, per la qual cosa ens haurem de convertir en l'usuari *root*, per després convertir-nos en l'usuari *postgres* i fer tasques en nom seu:

```
ioc@localhost:~$ su


Password:

# su – postgres

postgres@localhost:~$
```

L'usuari *postgres* pot crear noves bases de dades des de l'intèrpret d'ordres utilitzant l'ordre **createdb**. En aquest cas, li indiquem que l'usuari propietari de la base de dades serà l'usuari *postgres*:

```
postgres@localhost:~$ createdb demo -–owner=postgres

create database
```

De manera anàloga podem emprar l'ordre **dropdb** per eliminar una base de dades.

1.2.3 El client psql

Per connectar-se amb un servidor, es requereix, òbviament, un programa client. Amb la distribució de PostgreSQL s'inclou un client, *psql*, fàcil d'utilitzar, que permet la introducció interactiva d'ordres en mode text. Abans d'intentar connectar-nos amb el servidor, ens hem d'assegurar que està funcionant i que admet connexions, locals (l'SGBD s'està executant a la mateixa màquina que intenta la connexió) o remotes.

El pas següent és conèixer el nom d'una base de dades resident en el servidor.

L'ordre següent permet conèixer les bases de dades residents en el servidor:

```
ioc@localhost:~$ psql –l

List of databases

Name | Owner | Encoding

```

```
-----------+----------+-----------

demo | postgres | SQL_ASCII

template0 | postgres | SQL_ASCII

template1 | postgres | SQL_ASCII

(3 rows)

~$
```

Per fer una connexió, es requereixen les dades següents:

- Servidor. Si no s'especifica, s'utilitza `localhost`.
- Usuari. Si no s'especifica, s'utilitza el nom d'usuari Unix que executa el *psql*.
- Base de dades.

Exemples de l'ús del *psql* per connectar-se amb un servidor de bases de dades:

```
ioc@localhost:~$ psql -d demo

ioc@localhost:~$ psql demo
```

Les dues formes anteriors executen `psql` amb la base de dades `demo`:

```
~$ psql -d demo -U nom_usuari

~$ psql demo nom_usuari

~$ psql -h nom_servidor.org -U nom_usuari -d nom_basedades
```

A partir del fragment anterior, el client *psql* mostrarà una cosa similar al següent:

```
Welcome to psql, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms

\h for help with SQL commands

\? for help on internal slash commands

\g or terminate with semicolon to execute query

\q to quit

demo=#
```

El símbol **#** significa que el *psql* està llest per llegir l'entrada de l'usuari. Les sentències SQL s'envien directament al servidor per interpretar-les, les ordres internes tenen la forma **\ordre** i ofereixen opcions que no estan incloses en SQL i són interpretades internament pel *psql*.

Per acabar la sessió de *psql*, utilitzem l'ordre **\q** o podem polsar **Ctrl-D**.

Podeu veure els indicadors d'estat del *psql* a la taula 1.2:

Taula 1.2. Indicadors d'estat del //psql//

Indicador	Significat
=#	Espera una nova sentència
-#	La sentència encara no s'ha acabat amb ; o \g
"#	Hi ha una cadena en cometes dobles que no s'ha tancat
'#	Hi ha una cadena en cometes simples que no s'ha tancat
(#	Hi ha un parèntesi que no s'ha tancat

Introducció de sentències

Les sentències SQL que escriguem en el client hauran d'acabar amb ; o bé amb \g:

```
demo=# select user;

current_user

──────────────

postgres

(1 row)

demo=#
```

Quan una ordre ocupa més d'una línia, l'indicador canvia de forma i va assenyalant l'element que encara no s'ha completat:

```
demo=# select

demo-# user\g

current_user

──────────────

postgres

(1 row)

demo=#
```

La memòria intermèdia en el psql

El client *psql* emmagatzema la sentència fins que se li dóna l'ordre d'enviar-la a l'SGBD. Per visualitzar el contingut de la memòria intermèdia (*buffer*) on ha emmagatzemat la sentència, disposem de l'ordre **\p**:

```
demo=# SELECT

demo-# 2 * 10 + 2

demo-# \p
```

```
SELECT

2 * 10 + 2

demo-# \g

?column?

----------

22

(1 row)

demo=#
```

El client també disposa d'una ordre **\r** que permet esborrar completament la memòria intermèdia per començar de nou amb la sentència:

```
demo=# select 'Hola Mon'\r

Query buffer reset (cleared).

demo=#
```

Ordres de consulta d'informació

El client *psql* ofereix diverses alternatives per obtenir informació sobre l'estructura de la nostra base de dades. En la taula 1.3 es mostren algunes ordres de molta utilitat:

Taula 1.3. Ordres de consulta d'informació

Ordre	Descripció
\l	Fa una llista de les bases de dades
\d	Descriu les taules de la base de dades en ús
\ds	Fa una llista de les seqüències
\di	Fa una llista dels índexs
\dv	Fa una llista de les vistes
\dp \z	Fa una llista dels privilegis sobre les taules
\da	Fa una llista de les funcions d'agregats
\df	Fa una llista de les funcions
\g arxiu	Executa les ordres d'arxiu
\H	Canvia el mode de sortida HTML
\! ordre	Executa una ordre del sistema operatiu

1.2.4 El client gràfic pgAdmin III

El màxim exponent de client gràfic de PostgreSQL és el programari *pgAdmin3*, que té llicència "*Artist License*", aprovada per l'FSF.

En el *pgAdmin3* podem treballar amb gairebé tots els objectes de la base de dades, examinar-ne les propietats i fer tasques administratives.

Una característica interessant del pgAdmin3 és que, cada vegada que fem alguna modificació en un objecte, escriu la sentència o sentències SQL corresponents, cosa que fa que, a més d'una eina molt útil, sigui alhora didàctica.

El *pgAdmin3* també incorpora funcionalitats per fer consultes, examinar-ne l'execució (com l'ordre *explain*) i treballar amb les dades.

Totes aquestes característiques fan del *pgAdmin3* l'única eina gràfica que realment necessitarem per treballar amb PostgreSQL, tant des del punt de vista de l'usuari com de l'administrador.

Hi ha altres eines gràfiques que tenen prestacions semblants al PgAdmin3, com l'SQuirreL.

Evidentment, les accions que podem fer en cada moment dependran dels permisos de l'usuari amb què ens connectem a la base de dades.

1.3 Gestió d'usuaris

Conceptualment, els usuaris de la base de dades estan totalment separats dels usuaris del sistema d'explotació.

Per crear un usuari des d'un client de PostgreSQL s'utilitza l'ordre SQL `CREATE USER`:

Cal diferenciar entre `CREATE USER`, que és una instrucció SQL, i `createuser`, que és una sentència que es pot executar des de l'intèrpret de comades un cop s'ha instal·lat el Postgres.

```
1 CREATE USER nom_usuari [ [ WITH ] opcions [ ... ] ];
```

També es poden crear usuaris des de l'intèrpret de comandes, o shell del sistema, amb la instrucció `createuser`.

Les opcions poden ser:

```
1  SYSID ID_usuari
2
3  CREATEDB | NOCREATEDB
4
5  CREATEUSER | NOCREATEUSER
6
7  IN GROUP nom_grup [, ...]
8
9  [ ENCRYPTED | UNENCRYPTED ]
10
11 PASSWORD 'password'
12
13 VALID UNTIL 'abstime'
```

abstime vol dir 'vàlid fins a'. La clàusula posa un temps absolut després del qual la contrasenya de l'usuari ja no és vàlida. Si aquesta clàusula s'omet la contrasenya serà vàlida per sempre.

I per donar de baixa un usuari s'utilitza `DROP USER`:

```
DROP USER nom_usuari;
```

Cal diferenciar entre `DROP USER`, que és una instrucció SQL i `dropuser`, que és una sentència que es pot executar des de l'intèrpret de comades un cop s'ha instal·lat el Postgres.

També es poden eliminar usuaris des de l'intèrpret d'ordres amb la instrucció `dropuser`.

Un usuari d'una base de dades pot tenir una sèrie d'atributs que defineixen els seus privilegis i la interacció amb el sistema d'autenticació del client. Són els atributs `CREATEUSER` o `CREATEDB`, que li confereixen el permís de crear nous usuaris o crear bases de dades, respectivament.

Els atributs dels usuaris es poden modificar amb l'ordre `ALTER USER`:

```
ALTER USER nom_usuari [ [ WITH ] opcions [ ... ] ]
```

I les opcions poden ser:

```
CREATEDB | NOCREATEDB

| CREATEUSER | NOCREATEUSER

| [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'

| VALID UNTIL 'abstime''
```

alguns exemples són:

```
ALTER USER nom RENAME TO nou_nom
```

Amb `SET` canvia la configuració de sessió per defecte d'un usuari per una configuració determinada.

```
ALTER USER nom SET paràmetre { TO | = } { valor | DEFAULT }
```

Amb `RESET` es restaura la configuració per defecte.

```
ALTER USER nom RESET paràmetre
```

1.4 Autoritzacions: grups i papers

A més de poder donar permisos d'utilització dels diferents recursos del sistema de manera individual a cada usuari, el nostre SGBD disposa de diverses eines que permeten:

- Donar privilegis a un determinat paper al qual s'assignaran els usuaris (tots els usuaris que exerceixin aquest paper heretaran els privilegis i permisos d'aquest).
- Gestionar diversos grups preestablerts de l'SGBD.

Trobareu SGBD que només implementen una de les dues eines i en troba-reu que les implementen totes dues. Ara aprendrem les diferències entre totes dues eines i com utilitzar-les:

1) Rols . Un paper és una forma d'agrupar diferents permisos. Es tracta de pensar en les tasques que han de fer una sèrie d'usuaris i agrupar les que són comunes dins d'un paper. Una vegada establert i definit aquest paper, pot ser assignat als usuaris que facin aquestes tasques. Ens ajudarà a simplificar la gestió de la seguretat dins del nostre sistema.

Exemple de creació de paper

```
1 CREATE role ADMINISTRADOR ;
2 GRANT select,insert,update, delete ON empleats TO ADMINISTRADOR ;
3 GRANT select,insert,update, delete ON projectes TO ADMINISTRADOR ;
4 GRANT ADMINISTRADOR TO usuari_amb_permisos ;
```

Aquí creem un paper ''ADMINISTRADOR'', al qual donem certs permisos en dues taules diferents "empleats" i "projectes". Finalment, assignem a l'usuari "usuari_amb_permisos" el paper que acabem de crear.

2) Grups . Els grups tenen una filosofia molt similar als papers. Ara els grups ja vénen predefinits pel sistema, l'únic que podem fer és assignar usuaris als grups que ja tenen uns certs permisos establerts i no modificables. Els grups més comuns que podem trobar als SGBD més comercialitzats són:

- **Administrador de sistema**. És l'usuari que té accés a totes les bases de dades i disposa de tots els recursos. És el nivell més alt i més poderós de tot el sistema.

- **Administrador de les bases de dades**. És l'usuari que té accés a tots els recursos d'una base de dades específica. Pot fer modificacions en tots els objectes de la base de dades específica.

- **Administrador de seguretat**. Té el poder de donar o restringir l'accés a qualsevol usuari dintre de l'SGBD.

- **Operacions de control**. És el grup d'usuaris que té permès fer les còpies de seguretat o les restauracions del sistema.

1.4.1 Grups de PostgreSQL

En el PostgreSQL també tenim la possibilitat de crear grups amb l'ordre `CREATE GROUP`, modificar-los amb `ALTER GROUP` i esborrar-los amb `DROP GROUP`.

No obstant això, cal esmentar que actualment PostgreSQL difereix de l'SQL estàndard, ja que aquest utilitza `CREATE ROLE`, per crear grups d'usuaris.

1.4.2 Els papers

El PostgreSQL 9.0 administra els permisos d'accés a la base de dades d'accés utilitzant el concepte dels papers.

Un paper pot ser entès com un usuari de base de dades, o un grup d'usuaris, depenent de com s'estableixi aquest paper. Un paper pot ser propietari dels objectes de la base de dades (per exemple, taules) i pot assignar privilegis dels objectes dels quals és propietari a d'altres papers per controlar qui té accés a aquests objectes. A més, és possible la concessió de la pertinença d'un paper a un altre paper, la qual cosa permet a un membre del paper utilitzar els privilegis assignats a un altre paper. Podem veure, doncs, que permet implementar el concepte d'herència de privilegis.

El concepte dels papers aglutina els conceptes d'*usuaris* i *grups*. En les versions de PostgreSQL anteriors a la 8.1, els usuaris i grups corresponien a diferents tipus d'entitats, però en aquesta versió només hi ha papers. Així doncs, qualsevol paper pot actuar com un usuari, grup, o totes dues coses.

Creació i eliminació de papers

Els papers d'una base de dades estan conceptualment completament separats dels usuaris del sistema operatiu. En la pràctica, podria ser convenient mantenir-hi una correspondència, però això no és necessari. Els papers d'una base de dades són globals quan fem una instal·lació en clúster d'una bases de dades; per tant, no són exclusivament locals per a cada instància de la base de dades individual dins del clúster.

Per crear un paper cal utilitzar l'ordre SQL `CREATE ROLE`:

```
1 CREATE ROLE nom_del_paper;
```

en què *nom_del_paper* segueix les regles dels identificadors de SQL.

Per eliminar un paper existent, utilitzeu l'ordre anàloga `DROP ROLE`:

```
1 DROP ROLE nom_del_paper;
```

Per a més comoditat, els programes incorporats en el sistema **createuser** i **dropuser** ens proporcionen un substitutiu d'aquestes ordres SQL que ens permeten fer la crida corresponent des de l'intèrpret d'ordres:

```
1 createuser nom_del_paper
2
3 dropuser nom_del_paper
```

Per determinar el conjunt de papers existents, cal examinar el catàleg del sistema; en concret, la taula *pg_roles*; per exemple:

```sql
SELECT rolname FROM pg_roles;
```

La metainstrucció del programa *psql* **\du** també és útil per veure la llista de papers existents.

Per tal d'arrencar el sistema de base de dades, el nou sistema inicialitzat sempre conté una funció d'identificació predefinida. Aquest paper és sempre un *root*, i per defecte (si no és alterat quan s'executa `initdb`) tindrà el mateix nom que l'usuari del sistema operatiu que inicialitza el clúster de base de dades. Habitualment, aquest paper serà anomenat *postgres*. Per tal de crear més papers primerament cal connectar-se com aquest paper inicial.

Cada connexió amb el servidor de base de dades es fa mitjançant el nom d'algun paper en particular, i aquest paper determina els privilegis d'accés inicial.

El nom del paper que s'utilitza per a una connexió de base de dades en particular s'indica al programari client que inicia la sol·licitud de connexió d'una manera específica. Per exemple, el programa *psql* utilitza l'ordre `-U` per indicar el paper amb què ens connectarem a l'inici de la sessió.

Moltes aplicacions assumeixen el nom de l'usuari actual del sistema operatiu per defecte (aquí inclourem **createuser** i **psql**). Per tant, en aquest casos sol ser convenient mantenir una correspondència entre els noms dels papers i els usuaris del sistema operatiu.

Atès que un paper és una funció d'identitat i determina el conjunt de privilegis a disposició d'un client connectat, és important configurar acuradament privilegis quan treballem en un entorn multiusuari.

Atributs dels papers

Un *paper de base de dades* pot tenir una sèrie d'atributs que defineixen els seus privilegis i li permeten interactuar amb el sistema d'autenticació del client.

- *Privilegi `LOGIN`*: solament els papers que tenen atribut d'inici de sessió, `LOGIN`, poden ser utilitzats com a nom de paper inicial d'una connexió de base de dades. Un paper amb l'atribut `LOGIN` pot ser considerat com un "usuari de la base de dades". Per crear un paper amb el privilegi d'inici de sessió, podeu utilitzar indistintament :

```sql
CREATE ROLE nom_del_paper LOGIN;
-- o
CREATE USER nom_del_paper;
```

- *Estatus de superusuari*: un superusuari de base de dades supera qualsevol comprovació de permisos. Aquest és un privilegi perillós i no ha de ser utilitzat amb descuit; el millor és fer un usuari que faci la major part del seu treball amb un paper que no sigui de superusuari. Per crear un nou superusuari, utilitzeu:

```
CREATE ROLE nom_del_paper SUPERUSER
```

- *Creació de noves bases de dades*: a un paper se li pot atorgar explícitament el permís per crear bases de dades (a excepció de superusuaris, ja que passen per alt tots els controls de permís, i per tant ja adquireixen aquest privilegi.

```
CREATE ROLE nom_del_paper CREATEDB
```

- *Creació de nous papers*: a un paper li podem donar explícitament permisos per crear nous papers. Un paper amb el privilegi `CREATEROLE` pot modificar i eliminar altres papers, i també atorgar o revocar la pertinença d'un usuari o paper a un altre paper. No obstant això, per crear, modificar, treure o canviar la pertinença a un paper de superusuari es requereix que qui faci aquest canvi també sigui un superusuari.

```
CREATE ROLE nom_del_paper CREATEROLE
```

- *Contrasenya:* les contrasenyes són només útils si el mètode d'autenticació del client requereix que l'usuari proporcioni una contrasenya quan es connecta a la base de dades. El mètodes d'autenticació MD5 permeten fer un bon ús de les contrasenyes. Les contrasenyes de bases de dades són independents de les contrasenyes del sistema operatiu.

```
CREATE ROLE nom_del_paper PASSWORD 'la_contrasenya'
```

Superusuari

És una bona pràctica crear un paper que tingui els privilegis `CREATEDB` i `CREATEROLE`, però que aquest no sigui un superusuari, i després utilitzar aquest paper per a totes les tasques de bases de dades i d'altres papers. Aquest enfocament evita els perills d'operar com un superusuari per a les tasques que realment no ho requereixen.

Membres d'un paper

Els membres d'un paper poden utilitzar els privilegis de la funció de dues maneres.

En primer lloc, qualsevol membre d'un grup pot fer de manera explícita `SET ROLE` per convertir-se de manera temporal a aquell nou grup. En aquest estat, la sessió de base de dades té accés als privilegis de la funció de grup en lloc del paper assignat originalment a l'inici de sessió, i qualsevol objecte de base de dades creat es considerarà propietat del grup no és el paper d'inici de sessió.

En segon lloc, els membres d'un paper que poden heretar (`INHERIT`) poden fer ús dels privilegis dels papers dels quals són membres de manera automàtica, incloent-hi els privilegis heretats pels papers.

A tall d'exemple, suposem que hem fet:

```
CREATE ROLE joan LOGIN INHERIT;
CREATE ROLE admin NOINHERIT;
CREATE ROLE gestor NOINHERIT;
GRANT admin TO joan;
GRANT gestor TO admin;
```

Immediatament després de connectar-nos a la base de dades amb el paper de *joan* podrem fer ús dels privilegis concedits directament a *joan*, a més dels privilegis concedits a *admin*, perquè *joan* "hereta" els privilegis d'*admin*. En canvi *admin* no hereta els privilegis de *gestor*, i en conseqüència tampoc *joan*.

Si després fem:

```
SET ROLE admin;
```

La nostra sessió tindria ús exclusiu dels privilegis concedits a *admin*, i però no dels que hem concedit a *joan*.

Si ara fem:

```
SET ROLE gestor;
```

La sessió tindrà ús exclusiu dels privilegis concedits a *gestor*, però cap dels que es concedeixen a *joan* ni a *admin*.

El conjunt de privilegis originals es pot restaurar amb qualsevol acció d'aquestes:

```
SET ROLE joan;
SET ROLE NONE;
RESET ROLE;
```

1.5 Privilegis i permisos

Quan es crea un objecte aquest és assignat a un propietari. El propietari normalment és el mateix usuari que ha executat la comanda de creació.

Per a la majoria dels objectes, l'estat inicial és aquell en què el propietari (o un superusuari) pot fer alguna cosa amb aquest objecte. Per tal de deixar a altres usuaris utilitzar l'objecte, cal atorgar-li privilegis.

Existeixen diferents privilegis: `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `RULE`, `REFERENCES`, `TRIGGER`, `CREATE`, `TEMPORARY`, `EXECUTE` i `USAGE`.

Per exemple si el privilegi és `RULE` o `TRIGGER` vol dir que l'usuari pot crear regles o triggers en la taula especificada.

Per tal d'assignar privilegis s'utilitza la comanda `GRANT`.

On `PUBLIC` significa que els drets són donats a tots els usuaris. Inclòs aquells que es puguin crear posteriorment.

`WITH GRANT OPTION` significa que el que té aquest privilegi el pot transferir a altres usuaris.

`ALL PRIVILEGES` significa que li dóna tots els drets disponibles de l'objecte de cop.

Per eliminar els drets d'un usuari o grups d'usuaris s'utilitza `REVOKE`.

Un objecte pot ser assignat a un nou usuari amb la comanda `ALTER`.

1.5.1 Tipus de privilegis

Els privilegis es poden donar sobre diversos recursos a diferents usuaris del sistema gestor de bases de dades. Els recursos més comuns són:

- Connexió a la base de dades
- Taules: qui hi pot accedir i les modificar.
- Objectes de la base de dades: qui pot crear/esborrar els objectes que formen part de la base de dades.
- Sistema: qui pot efectuar accions de sistema en l'SGBD.
- Programa: qui pot crear, modificar i usar programes de la base de dades.
- Programes emmagatzemats: qui pot executar funcions i procediments específics.

Privilegis sobre bases de dades

Quan es crea una base de dades el propietari té tots els privilegis sobre aquesta. La base de dades serà inaccessible a altres usuaris, excepte l'usuari *postgres*, fins que el propietari els autoritzi privilegis.

En el PostgreSQL hi ha tres tipus de privilegis sobre bases de dades.

`CONNECT`: permet a l'usuari connectar-se a la base de dades especificada. Aquest privilegi es comprova en l'inici de connexió (a més de comprovar que no s'infringeix cap de les restriccions imposades en l'arxiu de configuració *pg_hba.conf*).

`CREATE`: permet crear nous esquemes a la base de dades.

`TEMPORAL`: permet crear taules temporals durant l'ús de la base de dades especificada.

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

Per altra banda, cal recordar que una base de dades pot estar formada per diferents esquemes.

Teniu més informació sobre la gestió d'esquemes en la secció "Annexos" del web del mòdul.

Així doncs, es poden atorgar privilegis sobre aquests. Aquests són:

`USAGE`: pot fer servir els elements d'un determinat esquema d'una base de dades a la qual tingui accés.

`CREATE`: pot crear objectes dins de l'esquema de la base de dades a la qual té accés.

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

Privilegis de taules

Els privilegis que es poden donar sobre les taules estan relacionats amb totes les accions que es poden fer sobre les taules i vistes, que són:

- `SELECT`: permet seleccionar dades d'una vista i taula donades.
- `INSERT`: permet inserir dades a una vista/taula.
- `UPDATE`: permet actualitzar dades d'una taula o vista.
- `DELETE`: permet esborrar dades d'una taula donada.
- `ALL`: permet fer les accions anteriors sobre una taula/vista en concret.
- `REFERENCES`: permet referenciar mitjançant restriccions de clau forana a una taula de la qual l'usuari no és propietari.

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

Exemple de gestió de privilegis de taules:

```
GRANT UPDATE ON NOTES TO SECRETARIA1
```

Permet a l'usuari "secretaria1" modificar el contingut dels registres de la taula "notes". No podrà crear un registre nou, però sí que podrà modificar les dades enregistrades.

```
GRANT DELETE ON MATRICULA TO ADMINISTRATIU4
```

Permet a l'usuari "administratiu4" esborrar els registres de la taula "matrícula".

Privilegis d'objectes de bases de dades

Els objectes d'una base de dades estan formats per totes les estructures que es poden crear, que són:

- bases de dades
- espai per a taules (*tablespace*)
- les taules
- índexs

- *triggers* (disparadors)

Qui tingui permís sobre els objectes de la base de dades podrà crear estructures de la base de dades.

```
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 ]
```

Especificar privilegis sobre espais de taules:

```
GRANT { CREATE | ALL [ PRIVILEGES ] }
    ON TABLESPACE tablespace_name [, ...]
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

Especificar privilegis sobre seqüències:

```
GRANT { { USAGE | SELECT | UPDATE }
    [,...] | ALL [ PRIVILEGES ] }
    ON { SEQUENCE sequence_name [, ...]
        | ALL SEQUENCES IN SCHEMA schema_name [, ...] }
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

Generalment només el DBA tindrà aquest privilegi, ja que, si l'estén a més usuaris, serà difícil controlar el creixement de la base de dades.

Exemple de gestió de privilegis d'objectes de bases de dades:

```
GRANT CREATE table,
 CREATE index
TO usuari456,
 Usuari_excepcional;
```

En aquest exemple podem veure com es dóna permís als usuaris *usuari456* i *usuari_excepcional* per poder crear taules i índexs.

Privilegis de sistema

Els privilegis de sistema estan relacionats amb totes les gestions que es poden portar a terme respecte al sistema gestor, que són:

- arxivar arxius `LOG`,
- reiniciar o apagar el servidor de bases de dades,
- tasques de monitorització,
- etc.

Privilegis sobre programes i procediments

Els privilegis sobre programes i procediments donen el privilegi `EXECUTE` als usuaris que hagin d'executar algun programa o procediment emmagatzemat en l'SGBD.

```
GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { FUNCTION function_name ( [ [ argmode ] [ arg_name ] arg_type [, ...] ]
         ) [, ...]
        | ALL FUNCTIONS IN SCHEMA schema_name [, ...] }
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

Exemple de gestió de privilegis sobre programes i procediments:

```
GRANT EXECUTE ON procediment
TO user456;
```

Aquí donem permís a l'usuari "user456" perquè pugui executar el programa "procediment".

Privilegis per a tothom

Els privilegis per a tothom és un tipus de privilegi que utilitzarem quan haguem de donar permís sobre un cert recurs a tots els usuaris de l'SGBD. Val a dir que aquest tipus d'assignació de permisos és molt còmode però també és molt perillós. Heu d'anar molt en compte quan el feu servir, ja que una vegada fet públic pot ser molt complicat tornar a tenir un control absolut del recurs.

Intentarem sempre evitar clàusules en què apareixen les dues instruccions següents: `PUBLIC` i `WITH GRANT OPTION`.

Exemple de gestió de privilegis per a tothom:

```
GRANT DELETE ON Llibres TO PUBLIC;
```

Aquí acabem de fer que tot usuari de l'SGBD pugui en qualsevol moment esborrar registres de la taula llibres.

1.5.2 Retirar privilegis

Per retirar els privilegis concedits anteriorment, tenim la sentència `REVOKE`. Es tracta de formar les mateixes ordres que quan donem un permís, però ara canviarem la paraula `GRANT` per `REVOKE`. Si un objecte és eliminat de la base de dades, automàticament també es perden els privilegis sobre l'objecte.

Exemple de retirada de privilegis:

```
REVOKE UPDATE on Llibres (isbn) FROM usuari;
```

En aquest cas traiem el permís a l'usuari "usuari" perquè pugui modificar l'atribut "isbn" de la taula "Llibres".

Heu d'estar molt atents quan retireu privilegis si aquests han estat atorgats amb `WITH GRANT OPTION`, ja que la retirada d'un privilegi a un usuari que hagi donat privilegis a altres usuaris implica que tots ells perdin el permís per utilitzar el recurs. Es coneix com a **retirada de permís en cascada**. Així doncs, eviteu donar privilegis amb l'opció `WITH GRANT OPTION`.

S'ha de fer una última consideració respecte al fet de crear exclusions en grups

d'usuaris a l'hora de donar o treure permisos. Imagineu que voleu donar privilegis a tots els usuaris de l'SGBD excepte a un (o uns quants). Una manera de fer-ho seria:

```
1 GRANT DELETE on Llibres to PUBLIC;
2 REVOKE DELETE on Llibres from usuari;
```

Heu de tenir present que aquest tipus d'accions no són permeses en tots els SGBD. Així doncs, haureu d'esbrinar consultant el manual del sistema gestor si és una forma viable de fer exclusions de grups d'usuaris o bé haureu de buscar formes alternatives de fer aquest tipus d'accions.

1.6 La Llei de protecció de dades de caràcter personal

La legislació espanyola recull una sèrie de drets i deures relatius a la utilització de les dades personals en l'àmbit dels sistemes d'informació.

En concret la **Llei orgànica 15/1999 de protecció de dades de caràcter personal (LOPD)** té com a objectiu garantir i protegir les dades personals, les llibertats públiques i els drets fonamentals de les persones físiques, i especialment la seva intimitat i privadesa personal i familiar. Fou aprovada a les Corts espanyoles el 13 de desembre del 1999. Aquesta llei es desenvolupa fonamentada en la **Constitució espanyola de 1978**, sobre el dret a la intimitat familiar i personal i el secret de les comunicacions. En concret sobre els articles següents:

- Article 18.4. La llei ha de limitar l'ús de la informàtica per tal de garantir l'honor i la intimitat personal i familiar dels ciutadans i el ple exercici dels seus drets.

- Article 20.1. Dret a comunicar o rebre lliurement informació.

- Article 105.b. La llei regularà l'accés dels ciutadans als arxius i registres.

El desenvolupament de la LOPD la fa el **Reial decret 1720/2007 de 21 de desembre de desenvolupament de la Llei orgànica de protecció de dades (RDLOPD)**. Desenvolupa tant els principis de la Llei com les mesures de seguretat que cal aplicar en els sistemes d'informació. S'aplica tant a fitxers en suport automatitzat com a qualsevol altre tipus de suports.

1.6.1 Conceptes bàsics

A continuació esmentem tot un conjunt de conceptes bàsics per entendre l'objectiu de la LOPD.

- **Dada personal:** qualsevol informació del tipus que sigui que permeti identificar o faci identificable la persona.
- **Afectat:** persona identificada o identificable a qui corresponen les dades personals.
- **Fitxer:** tot conjunt organitzat de dades de caràcter personal, que permeti accés a les dades amb criteris determinats, qualsevol que sigui la forma o modalitat de creació, enregistrament, organització, tractament i accés.
- **Responsable del fitxer:** persona física o jurídica de naturalesa pública o privada a qui pertany el fitxer, amb independència que executi o no materialment el tractament.
- **Sistema de tractament:** qualsevol forma o modalitat que permeti l'ús i gestió de les dades, des que es registren fins que s'eliminen.
- **Encarregat de tractament:** pot ser el responsable del fitxer o qualsevol altra persona física o jurídica de naturalesa privada o pública que tracti per encàrrec del responsable les dades de caràcter personal dels fitxers. Alhora, aquest encarregat de tractament pot fer-ho sol o conjuntament i en cada cas tots els agents implicats estan obligats a la normativa de confidencialitat desenvolupada.
- **Usuari:** qualsevol persona que tingui accés a les dades personals que componen els fitxers del responsable.
- **Responsable de seguretat:** persona o persones físiques o jurídiques que tenen la funció de vetllar pel compliment, aplicació i manteniment del document de seguretat.
- **Document de seguretat:** recull de normativa i processos per a l'aplicació dels aspectes regulats en matèria de protecció de dades que tot responsable de fitxer ha de tenir obligatòriament.
- **Comunicació de dades:** qualsevol cessió de les dades personals del responsable del fitxer a tercers. Tota comunicació o cessió de dades entre parts s'ha de donar en un marc regulat entre les parts de confidencialitat estricta i s'han de garantir l'aplicació de les mesures de seguretat corresponents i també que les dades seran tractades per a la finalitat amb què van ser registrades, amb l'excepció dels requisits de determinades administracions públiques relacionades amb les funcions policíaques, de justícia i sanitat.

Fitxer de dades personals

És el punt de partida de la normativa. Un fitxer serà qualsevol conjunt organitzat de dades personals, de propietat pública o privada, qualsevol que sigui la seva forma d'enregistrament i tractament amb una finalitat determinada. No ha estat fins a l'RDLOPD que s'ha concretat que el sistema de tractament en paper constitueix també un fitxer. Per tant, tota entitat de dret públic o privat té fitxers que contenen dades personals registrades per a diferents finalitats. Cal identificar-los i classificar-los per funcionalitat i naturalesa i registrar-los públicament.

Totes les mesures de seguretat que es desenvolupen s'han d'aplicar sobre els fitxers en funció del seu sistema de tractament. Són així l'element central en matèria de protecció de dades personals, ja que són el conjunt de dades registrades i tractades per a una o diverses finalitats concretes. En qualsevol suport. I és propietat del responsable del fitxer, que és qui ha d'assumir l'aplicació de la normativa corresponent sobre els fitxers.

Registre de fitxers

Tot responsable de fitxer haurà de registrar en l'Agència Espanyola de Protecció de Dades tants fitxers com tingui identificats. No tenir els fitxers registrats és una primera infracció. Això té nombroses implicacions i obligacions concretes per als responsables de fitxer i alhora implica una garantia mínima de qualitat per a la persona respecte de les seves dades personals. El registre de fitxers és d'accés públic i el que es comunica a l'Agència és la identificació del fitxer, el contingut de les dades (la seva qualitat) i també la finalitat i cessions que se'n facin i la identificació de tercers implicats en la gestió o tractament de les dades. No es comuniquen les dades concretes, sinó la seva composició per a cada un dels fitxers. I es fa per mitjà de l'aplicació informàtica única que l'Agència disposa públicament.

1.6.2 Principis generals de protecció de dades

A continuació es passen a descriure els principis generals de la protecció de les dades:

Qualitat de dades: les dades de caràcter personal només es podran recollir per tractar-les, i també es podran sotmetre a aquest tractament quan sigui adequat, escaient i no excessiu en relació amb l'àmbit i les finalitats per a les quals s'hagin obtingut.

Dret d'informació en la recollida de dades: s'haurà d'informar amb antelació els interessats als quals se sol·licitin les dades personals dels punts següents:

- De l'existència d'un fitxer o tractament de dades de caràcter personal.
- Del caràcter obligatori o facultatiu de la resposta donada a les preguntes que els siguin plantejades.
- De les conseqüències de l'obtenció de les dades o la negativa a subministrar-les.
- De la possibilitat d'exercitar els drets d'accés, rectificació, cancel·lació i oposició.
- De la identitat i direcció del responsable del tractament o, segons del cas, del seu representant.

Consentiment de l'afectat: llevat que la llei disposi una altra cosa, per al tractament de les dades de caràcter personal es requerirà el consentiment inequívoc dels afectats.

Dades especialment protegides: les dades de caràcter personal que revelin ideologia, afiliació sindical, religió i creences només podran ser objecte de tractament amb el consentiment, exprés i per escrit, de l'afectat.

Dades relatives a la salut: les institucions i els centres sanitaris públics i privats i els professionals corresponents podran procedir al tractament de les dades de caràcter personal relatives a la salut de les persones sempre que siguin facilitades pel titular per motiu d'assistència sanitària.

Seguretat de les dades i deure de secret: tant el responsable del fitxer com l'encarregat del tractament, en el seu cas, hauran d'adoptar les mesures tècniques i organitzatives necessàries que garanteixin la seguretat de les dades de caràcter personal i n'evitin l'alteració, la pèrdua, el tractament o l'accés no autoritzat.

Comunicació de dades: les dades de caràcter personal objecte de tractament només podran ser comunicades a un tercer, amb l'anterior consentiment de l'interessat, per al compliment de finalitats directament relacionades amb les funcions legítimes del cedent i del cessionari.

Accés a les dades per part de terceres persones: els tractaments que facin terceres persones hauran d'estar regulats en un contracte per escrit o en alguna altra forma que permeti d'acreditar-ne la subscripció i el contingut. L'accés d'un tercer a aquestes dades no es considerarà comunicació de dades quan sigui necessari per a la prestació d'un servei al responsable del tractament.

1.6.3 Nivells de seguretat esmentats a la Llei

L'article 4.1 de la LOPD estableix que les dades de caràcter personal només es podran recollir per tractar-les, i també sotmetre-les a aquest tractament, quan siguin adequades, pertinents i no excessives en relació amb l'àmbit i les finalitats determinades, explícites i legítimes per a les quals s'hagin obtingut.

La LOPD defineix quina és la qualitat de la dada, en base a la qual s'estableixen els nivells de seguretat que cal aplicar sobre el tractament, i que desenvolupa l'RDLOPD. Això tindrà implicacions respecte del sistema de tractament i també respecte dels usuaris (control d'accessos, registre, identificació, autenticació, compromís de confidencialitat del personal, etc.). En tot cas, els nivells es defineixen en tres nivells:

- **Nivell bàsic:** qualsevol fitxer que contingui dades personals de qualsevol tipus que facin identificable la persona: nom, imatge, adreça, etc.
- **Nivell mitjà:** fitxers que continguin, a més de les anteriors, dades relatives a la comissió d'infraccions administratives o penals, hisenda pública, serveis

financers i aquells corresponents a la prestació de serveis de solvència i crèdit. Generalment, però, s'entén que estarem davant de dades de nivell mitjà quan aquestes permetin fer una valoració de l'entorn social i psicològic de la persona, més enllà de la seva simple identificació.

- **Nivell alt:** fitxers que continguin dades personals sobre ideologia, religió, creences, origen racial, salut o vida sexual o els registrats per a finalitats policíaques sense consentiment. Aquestes són les dades que han de ser especialment protegides.

En funció del nivell de dades registrades en el fitxer amb un sistema de tractament o l'altre, caldrà aplicar unes mesures de seguretat concretes que es desenvolupen en l'RDLOPD per a cada un dels nivells. Igualment, el règim sancionador és diferent en funció de la qualitat de la dada implicada en la infracció. De manera que la normativa protegeix especialment les dades qualificades com a nivell alt, que tenen una especial relació amb els drets fonamentals de les persones.

Nivell bàsic de seguretat

Tots els fitxers que continguin dades de caràcter personal han de seguir les mesures de seguretat del nivell bàsic.

Les condicions del nivell bàsic de seguretat es definiran en el document de seguretat que haurà d'elaborar i implementar el responsable del fitxer. Aquest document haurà d'incloure els punts següents:

1. L'àmbit d'aplicació del document amb especificació detallada dels recursos protegits.
2. Les mesures, normes, procediments, regles i estàndards encaminats a garantir el nivell de seguretat exigit per l'RDLOPD.
3. Les funcions i obligacions del personal.
4. L'estructura dels fitxers que continguin dades de caràcter personal i la descripció dels sistemes d'informació que les tracten.
5. El procediment de notificació, gestió i resposta de les incidències (tant tecnològiques com funcionals i d'accés).
6. Els procediments de realització de còpies de suport i de recuperació de dades.
7. La periodicitat amb la qual s'han de substituir les contrasenyes dels usuaris.
8. El personal autoritzat per concedir, modificar o alterar els accessos autoritzats a les dades o recursos, i també els criteris per fer aquestes accions.
9. El personal autoritzat per accedir al lloc on s'emmagatzemen dades de caràcter personal.

Nivell mitjà de seguretat

Els fitxers que continguin dades relatives a la comissió d'infracció penals o administratives, la Hisenda Pública i serveis financers (solvència patrimonial i crèdit, compliment o incompliment de les obligacions monetàries) hauran de seguir les mesures del nivell mitjà.

Les mesures de seguretat de nivell mitjà inclouen les del nivell bàsic i, a més, exigeixen el compliment dels punts següents:

1. Identificació del responsable o responsables de seguretat.
2. Controls periòdics (auditoria) per verificar el compliment del que està establert en el document de seguretat mateix.
3. Mesures que calgui adoptar quan un suport que contingui dades de caràcter personal estigui a punt de ser destruït o reutilitzat.
4. Personal autoritzat per accedir als locals on els sistemes d'informació amb dades de caràcter personal es trobin físicament ubicats.

Nivell alt de seguretat

Els fitxers amb continguts de dades relacionades amb la ideologia, religió, creences, origen racial, salut o vida sexual seguiran unes mesures del nivell alt.

Les mesures de seguretat de nivell alt inclouen les del nivell mitjà i, a més, exigeixen el compliment dels punts següents:

1. Les dades de caràcter personal que s'hagin de distribuir en qualsevol tipus de suport es xifraran.
2. Per a cada accés es desarà com a mínim la identificació de l'usuari, la data i l'hora en què es va fer l'accés, el nom del fitxer al qual s'ha accedit i el tipus d'accés.
3. S'haurà de desar una còpia de suport i dels procediments de recuperació de les dades en un lloc diferent d'on es trobin físicament els sistemes d'informació.
4. Les dades de caràcter personal que s'hagin de transmetre per la xarxa de telecomunicacions es xifraran.

1.6.4 L'agència de protecció de dades

L'organisme de control del compliment de la normativa de protecció de dades dins del territori de l'Estat espanyol, amb caràcter general, és l'Agència Espanyola de Protecció de Dades (AGPD). No obstant això, hi ha altres agències de protecció de

dades de caràcter autonòmic, a les comunitats autònomes de Madrid, Catalunya i el País Basc.

2. Vistes i regles

2.1 Concepte de vista

Sovint per obtenir dades de diferents taules cal construir una sentència `SELECT` complexa i si en un altre moment hem de fer la mateixa consulta, cal construir de nou la sentència `SELECT`.

També cal considerar que seria molt còmode obtenir les dades d'una consulta complexa a partir d'una senzilla consulta `SELECT`.

Taula lògica o virtual

De fet, quan emprem els termes *taula lògica* o *taula virtual* ens referim al concepte de relació inherent a l'àlgebra relacional

Una vista és una *taula lògica* que permet accedir a la informació d'una o diverses taules per mitjà d'una consulta predefinida. No conté informació per si mateixa sinó que aquesta està basada en informació d'altres taules.

Com ja sabem, una vista és una *taula virtual* per mitjà de la qual es pot veure, i en alguns casos canviar, informació d'una o més taules. Una vista té una estructura semblant a una taula: files i columnes. Mai no conté dades, sinó una sentència `SELECT` que permet accedir a les dades que es volen presentar per mitjà de la vista. Podem dir que la gestió de vistes és semblant a la gestió de taules, ja que en tots dos casos en el fons parlem de relacions.

Fer un ús adient de les vistes és un aspecte clau per assolir un bon disseny d'una base de dades, ja que ens permet ocultar els detalls de les taules i mantenir la visió de l'usuari independent de l'evolució que pugui anar tenint l'estructura de taules.

La sentència `CREATE VIEW` és la instrucció proporcionada pel llenguatge SQL per a la creació de vistes.

Les vistes en PostgreSQL s'implementen utilitzant el sistema de regles (*rules*).

Aquest sistema és molt poderós i es pot emprar per a procediments, vistes i disparadors.

El sistema de regles en PostgreSQL consisteix a modificar les consultes d'acord amb regles emmagatzemades com a part de la base de dades. Aquestes consultes modificades són passades, en ordre, a l'optimitzador, posteriorment al planificador i finalment a l'executor. Això el diferencia d'altres SGBD, en què s'implementen els sistemes de regles com a procediments i disparadors emmagatzemats.

2.1.1 Creació de vistes

Per crear una vista s'utilitza `CREATE VIEW` amb la sintaxi següent:

```
1 CREATE [ OR REPLACE ] [ TEMP | TEMPORARY ] VIEW name [ ( column_name [, ...] )
      ]
2     AS query
```

Però com que les vistes s'implementen utilitzant el sistema de regles (*rules*), no hi ha diferència entre les sentències anteriors i la següent:

```
CREATE TABLE nom_vista (llista d'atributs de nom_taula);


CREATE RULE nom_rule AS ON SELECT TO nom_vista DO INSTEAD SELECT * FROM
    nom_taula;
```

Ja que això és el que fa internament la instrucció `CREATE VIEW`. Veiem que crea una relació anomenada `nom_vista` amb la definició dels atributs corresponents i posteriorment estableix una regla que permet visualitzar les ocurrències de `nom_taula` per mitjà de `nom_vista`, ja que totes dues relacions, en tenir els mateixos atributs, són compatibles. Cal tenir en compte que en aquest cas com a "nom_rule" caldria emprar el nom `_RETURN`.

Això té alguns efectes, i un és que la informació sobre una vista en el sistema de catàlegs de PostgreSQL és exactament la mateixa que per a una taula.

Les opcions opcionals en l'SQL estàndard de `CREATE VIEW` són les següents:

```
CREATE VIEW nom_vista [(columna [, ...])]

AS query

[WITH [CASCADE | LOCAL ] CHECK OPTION ]
```

- `CHECK OPTION`. Cal utilitzar aquesta opció amb les vistes actualitzables. Totes les ordres `INSERT` i `UPDATE` sobre la vista es controlen per assegurar que les dades són conformes a la condició definida en la vista. Si no, l'actualització o inserció és rebutjada.

- `LOCAL`. Per controlar la integritat de la vista.

- `CASCADE`. Per controlar la integritat sobre aquesta vista i totes les vistes dependents.

Si no s'especifica ni `LOCAL` ni `CASCADE` s'assumeix `CASCADE` per defecte.

Quan es fa servir el *psql*, es pot veure una llista de les vistes de la base de dades utilitzant la metainstrucció **\dv**:

També es pot visualitzar la definició d'una vista emprant la metainstrucció **\d nom_vista** :

2.1.2 Modificació de vistes

`ALTER VIEW` permet fer diversos canvis auxiliars referents a les propietats d'una vista. Si voleu modificar la definició de consulta de la vista, caldrà utilitzar `CREATE`

`OR REPLACE ALTER VIEW`. Com a mínim cal ser propietari de la vista per emprar una sentència `ALTER VIEW`. El superusuari pot fer els canvis sobre qualsevol vista.

Anem a veure'n uns quants exemples d'ús.

Establir un valor per defecte en una columna

El sistema de regles de PostgreSQL el veurem més endavant en aquesta mateixa unitat formativa.

`SET DEFAULT` és la manera d'establir el valor per defecte per a una columna. Un valor per defecte associat amb una columna de la vista s'insereix en les instruccions `INSERT` sobre la vista abans d'aplicar una regla (*rule*) `ON INSERT`, sempre que l'operació `INSERT` no especifiqui un valor per a la columna.

```
1 ALTER VIEW nom_vista ALTER [ COLUMN ] column SET DEFAULT expression
```

Eliminar un valor per defecte en una columna

`DROP DEFAULT` és la manera d'eliminar el valor per defecte per a una columna de la vista.

```
1 ALTER VIEW nom_vista ALTER [ COLUMN ] column DROP DEFAULT
```

Canvi de propietari d'una vista

Per modificar el propietari d'una vista, també s'ha de ser membre directe o indirecte de la regla nova que s'estableix i aquest nou paper ha de tenir permís `CREATE` en l'esquema de la vista.

```
1 ALTER VIEW nom_vista OWNER TO nou_propietari
```

Reanomenar una vista

```
1 ALTER VIEW nom_vista RENAME TO nou_nom_vista
```

Canvi de l'esquema al qual correspon la vista

Per canviar una vista a un altre esquema el propietari de la vista cal que tingui el privilegi `CREATE` en el nou esquema.

```
1 ALTER VIEW nom_vista SET SCHEMA nou_esquema
```

2.1.3 Eliminació de vistes

Per eliminar una vista podem utilitzar la mateixa ordre que l'SQL estàndard, `DROP VIEW`. Cal ser el propietari de la vista per eliminar-la.

La sintaxi de `DROP VIEW` és la següent:

```
DROP VIEW nom_vista [,...] [CASCADE | RESTRICT ]
```

Amb l'opció `CASCADE` s'esborren automàticament els objectes dependents de la vista, com poden ser altres vistes.

Amb `RESTRICT` no s'esborra la vista si hi ha objectes que en depenen. És l'opció per defecte.

2.2 Vistes del sistema

PostgreSQL disposa d'algunes vistes ja confeccionades. Algunes vistes del sistema tenen accés a les consultes més utilitzades en els catàlegs del sistema. Altres donen accés a l'estat del servidor intern.

Algunes de les principals vistes que hi ha disponibles són les següents:

- **pg_indexes** índexs
- **pg_locks** bloquejos
- **pg_rules** regles
- **pg_settings** paràmetres
- **pg_stats** estadístiques
- **pg_tables** taules
- **pg_user** usuaris
- **pg_views** vistes

Qualsevol de les vistes anteriors utilitza altres vistes també ja definides.

2.3 Avantatges i desavantatges en l'ús de les vistes

Ja sabem, llavors, quina és la definició de vista, i podeu imaginar també que aquest model de representació de les dades té els seus avantatges i desavantatges; a continuació veurem quins són els beneficis i problemes d'utilitzar vistes en un model de base de dades relacional.

2.3.1 Avantatges en l'ús de les vistes

- **Seguretat**: les vistes poden proporcionar un nivell addicional de seguretat. Per exemple, en la taula d'empleats, cada responsable de departament només tindrà accés a la informació dels seus empleats.

- **Simplicitat**: les vistes permeten ocultar la complexitat de les dades. Una base de dades es compon de moltes taules. La informació de dues o més taules es pot recuperar utilitzant una combinació de dues o més taules (relacional), i aquestes combinacions poden arribar a ser molt confuses. Creant una vista com a resultat de la combinació es pot ocultar la complexitat a l'usuari.

- **Organització**: les vistes ajuden a mantenir un noms de la base de dades per accedir a consultes complexes.

- **Exactitud de les dades demanades**: permeten accedir a un subconjunt de dades específiques, i ometen dades i informació innecessària i irrellevant per a l'usuari.

- **Amplia les perspectives de la base de dades**: proporciona diversos models d'informació basats en les mateixes dades, i els enfoca envers diferents usuaris amb necessitats específiques. Mostrar la informació des de diferents angles ens ajuda a crear ambients de treball i operació d'acord amb els objectius de l'empresa. S'ha d'avaluar el perfil i els requisits d'informació dels usuaris destinataris de la vista.

- **Transparència en les modificacions**: l'usuari final no es veurà afectat pel disseny o alteracions que es fan en l'esquema conceptual de la base de dades. Si el sistema requereix una modificació en el seu funcionament intern, es podran afectar diverses estructures que proveeixen l'acompliment d'aquest, i es pretén que els usuaris finals ho no adverteixin com a alteracions.

2.3.2 Possibles desavantatges en l'ús de les vistes

Encara que l'ús de vistes implica molts avantatges, i molt profitosos tots, també comporta una sèrie de desavantatges que cal considerar a l'hora de dissenyar una base de dades relacional. Aquests desavantatge tenen a veure amb les limitacions del motor de base de dades que s'utilitzarà. Per això en cada implementació d'un SGBD relacional veurem diferents restriccions en aquest aspecte.

En el SGBD que ens ocupa, PostgreSQL, les vistes no són actualitzables; és a dir, si bé és cert que són tractades com a taules, no és possible fer `INSERT`, `DELETE` ni `UPDATE` sobre les vistes. Aquest desavantatge és una característica particular del PostgreSQL, atès que aquesta qualitat sí que està disponible en altres motors de bases de dades com ORACLE, Informix i SQL Server, però cal notar que el PostgreSQL cobreix aquesta mancança en les vistes amb la creació de regles (`CREATE RULE`) que permeten omplir el buit deixat per la vista i ens permeten controlar quin tipus de modificacions podem fer per mitjà de la vista seguint les regles del negoci per al qual fa servei la base de dades.

2.4 Vistes actualitzables

2.4.1 Restriccions de les vistes actualitzables

En la majoria d'SGBD hi ha una sèrie de restriccions que cal considerar en l'esborrament, l'actualització i la inserció de vistes en una taula per mitjà d'una vista:

Esborrament de files per mitjà d'una vista: per esborrar files d'una taula per mitjà d'una vista, aquesta s'ha de crear:

- amb files d'una sola taula
- sense utilitzar la clàusula `GROUP BY` ni `DISTINCT`
- sense usar funcions de grup ni referències a pseudocolumnes

Actualització de files per mitjà d'una vista: per actualitzar files en una taula per mitjà d'una vista, aquesta ha d'estar definida segons les restriccions anteriors i, a més, cap de les columnes que es volen actualitzar no s'ha d'haver definit com una expressió.

Inserció de files per mitjà d'una vista. per inserir files en una taula per mitjà d'una vista s'han de tenir en compte totes les restriccions anteriors, i a més totes les columnes obligatòries de la taula associada han d'estar presents en la vista.

2.4.2 El sistema de regles emprat en el PostgreSQL

Situació

Tenim una taula de clients i una taula de línies telefòniques. Volem tenir una vista des de la qual es vegin totes les línies i els camps dels clients a qui pertanyen.

També volem crear nous clients i línies, i a més poder modificar les dades tant de la línia com del client, emprant aquesta vista.

```
CREATE SEQUENCE linies_id_seq
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 9223372036854775807
  START 1
  CACHE 1;

CREATE TABLE clients
(
  client_id serial NOT NULL,
  nom character varying(100),
  CONSTRAINT "PK_clients" PRIMARY KEY (client_id)
)
WITH (
  OIDS=FALSE
);


CREATE TABLE linies
(
  client_id integer,
  numero character(9),
  linia_id integer NOT NULL DEFAULT nextval('linies_id_seq'::regclass),
  CONSTRAINT "PK_linia" PRIMARY KEY (linia_id),
  CONSTRAINT "FK_linies_clients" FOREIGN KEY (client_id)
      REFERENCES clients (client_id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
```

La vista

```
CREATE VIEW clients_linies AS

SELECT c.client_id, linia_id, nom, numero

FROM clients c, linies l

WHERE c.client_id = l.client_id;
```

Ara verifiquem el funcionament de la vista:

```
SELECT * FROM clients_linies;
```

Les regles d'inserció

La regla següent ens permetrà inserir un client nou i la seva línia telefònica per mitjà de la vista. Observem que la condició d'inserció indica que l'identificador del client sigui *null*, ja que inserirem aquest valor emprant el nou valor de la seqüència definit per defecte en la definició de la taula corresponent. Com a segona acció inserim a la taula *línies* el mateix valor de la seqüència més el número de telèfon. Per tant, com a conseqüència d'aquesta regla quan inserim per mitjà de la vista un nou client amb el seu número de línia de telèfon tan sols cal emprar aquests dos valors: el nom i el telèfon.

```
CREATE RULE ins_clients_linies_nou AS

 ON INSERT TO clients_linies

WHERE NEW.client_id IS NULL

DO INSTEAD

(

INSERT INTO clients (nom)

VALUES (NEW.nom)

;

INSERT INTO linies (client_id, numero)

VALUES (currval('clients_client_id_seq'),NEW.numero);

 );
```

Definim ara la regla corresponent a la inserció d'un número de telèfon nou per a un client existent. En tot cas suposem que el valor de l'identificador del client que utilitzem existeix a la taula *clients* per mantenir la integritat referencial. Així doncs, tan sols necessitarem emprar dos valors en la inserció: l'identificador del client i el nou número de telèfon.

```
CREATE RULE ins_client_linia_existent AS

ON INSERT TO clients_linies

WHERE NEW.client_id IS NOT NULL

DO INSTEAD
INSERT INTO linies (client_id, numero)

VALUES (NEW.client_id, NEW.numero);
```

Ara definim una última regla incondicional per als casos d'inserció i així ens assegurem que en altres casos no definits no faci res.

```
CREATE RULE ins_client_linia_nothing AS

ON INSERT TO clients_linies

DO INSTEAD NOTHING;
```

Amb posterioritat a les insercions següents podrem, mitjançant un `SELECT` tant de la vista com de les taules implicades, hem de verificar la correctesa de les regles.

Primerament inserim un client nou amb la seva línia i així verifiquem el funcionament de la regla **ins_clients_linies_nou**

```
INSERT INTO clients_linies (nom, numero) VALUES ('Pau Pi', '234–4567');
```

A continuació inserim una línia nova per a un client existent i així verifiquem el funcionament de la regla **ins_client_linia_existent**

```
INSERT INTO clients_linies (client_id, numero) VALUES (3, '987–1233');
```

Les regles d'actualització

Regla que ens permet actualitzar el nom del client per mitjà de la vista

```
CREATE RULE upd_clients_linies_client AS

ON UPDATE TO clients_linies

 WHERE NEW.client_id IS NOT NULL

DO INSTEAD

UPDATE clients

SET nom= NEW.nom

WHERE client_id = NEW.client_id;
```

Regla que ens permet actualitzar el número de la línia de telèfon per mitjà de la vista

```
CREATE RULE upd_clients_linies_linia AS

ON UPDATE TO clients_linies

WHERE NEW.linia_id IS NOT NULL

DO INSTEAD

UPDATE linies

SET numero = NEW.numero

WHERE linia_id = NEW.linia_id;
```

Regla incondicional

```
CREATE RULE upd_clients_linies_nothing AS

ON UPDATE TO clients_linies

DO INSTEAD NOTHING;
```

Fem l'actualització seguint la definició de la regla **upd_clients_linies_client**

```
UPDATE clients_linies SET nom = 'Josep Pons' WHERE client_id = 3;
```

Fem l'actualització seguint la definició de la regla **upd_clients_linies_linia**

```
UPDATE clients_linies SET numero = '1-800-8888' WHERE linia_id = 4;
```

També podem fer actualitzacions emprant les dues regles **upd_clients_linies_client** i **upd_clients_linies_linia** alhora

```
UPDATE clients_linies
SET numero = '1-800-7777', nom = 'Maria Bassas'
WHERE client_id = 3 AND linia_id = 6;
```

2.4.3 Traducció de consultes sobre vistes

La informació sobre una vista en el sistema de catàlegs de PostgreSQL és la mateixa que per a una taula. D'aquesta manera, per als traductors de *queries*, no hi ha diferència entre una taula i una vista, ja que són el mateix: relacions.

El sistema de regles incorpora les definicions de les vistes en l'arbre de traducció original (*querytree*). La implementació del sistema de regles és una tècnica anomenada *reescriptura de la consulta*. El sistema de reescriptura és un mòdul que hi ha entre l'etapa del traductor i el planificador/optimitzador.

El sistema de reescriptura de la consulta processa l'arbre tornat per l'etapa de traducció, que representa una consulta de l'usuari, i si existeix una regla que calgui aplicar a la consulta, reescriu l'arbre d'una manera alternativa.

El *querytree* és la representació interna d'una consulta en la qual se separen i agrupen els components en forma d'arbre.

Components d'un *querytree*:

- La instrucció: `SELECT`, `UPDATE`, `INSERT` o `DELETE`.
- La *range table* (abast de la taula): inclou les relacions que utilitza.
- La *result relation* (relació resultant): un índex a la *range table* on hi haurà els resultats. Generalment `SELECT` no l'inclou.
- La *target list* (llista d'etiquetes): és la llista d'elements entre el `SELECT` i el `FROM` en una instrucció de `SELECT`, la llista de files afectades en `INSERT` i `UPDATE`. No s'utilitza en `DELETE`.
- La qualificació correspon al `WHERE` i indica si cal actualitzar o no una fila.
- El *join tree* (arbre del `JOIN`), combina parts del `FROM` i el `WHERE` per descriure l'estructura del `JOIN`.
- *others* (la resta), altres clàusules com `ORDER BY`.

Els beneficis d'implementar les vistes amb el sistema de regles són que l'optimització té tota la informació sobre quines taules han de ser revisades, les relacions entre aquestes taules, les qualificacions restrictives a partir de la definició de les vistes i les qualificacions de la *query* original, tot en un únic arbre de traducció. I aquesta és també la situació quan la *query* original és una JOIN entre vistes.

L'optimitzador cal que decideixi quina és la millor ruta per executar la *query*. Com més informació tingui l'optimitzador, millor serà la decisió. I la manera com s'implementa el sistema de regles de PostgreSQL assegura que tota la informació sobre la *query* és utilitzable.

Per comprendre com treballa el sistema de regles, és necessari conèixer quan s'invoca i quines són les seves entrades i els seus resultats.

El sistema de regles se situa entre el traductor de la *query* i l'optimitzador. Agafa la sortida del traductor, un *querytree*, i les regles de reescriptura del catàleg `pg_rewrite`, que són també *querytree*, amb alguna informació extra, i crea cap o molts *querytree* com a resultat. D'aquesta manera, l'entrada i la sortida són sempre tal com el traductor mateix les podria haver produït i tot apareix representable com una instrucció SQL.

Aquests *querytree* són visibles quan arrenquem el motor de PostgreSQL amb nivell de depuració 4 i teclegem *queries* en l'interfície d'usuari interactiva. Les accions de les regles emmagatzemades en el catàleg de sistema `pg_rewrite` estan emmagatzemades també com a *querytree*. No estan formades com la sortida del *debug*, però contenen exactament la mateixa informació.

Les representacions d'SQL de *querytree* són suficients per entendre el sistema de regles.

A continuació mostrarem un exemple de com es poden implementar vistes utilitzant el sistema de regles.

El sistema de reescriptura fa els passos següents:

- Pren la consulta donada per la part d'acció de la regla.
- Adapta la llista objectiu per recollir el nombre i ordre dels atributs donats en la consulta d'usuari.
- Afegeix la qualificació donada en la clàusula `WHERE` de la consulta de l'usuari a la qualificació de la consulta donada en la part de l'acció de la regla.

D'acord amb això la consulta de l'usuari serà reescrita de la manera següent:

La reescriptura es fa en la representació interna de la consulta de l'usuari tornada per l'etapa de traducció però la nova estructura de dades representarà la consulta anterior.