Administració de dades

Remo Suppi Boldrito

P07/M2003/02287

UOC
www.uoc.edu

Índex

Introducció

Un aspecte important d'un sistema operatiu és on i com es desen les dades. Quan la disponibilitat d'aquestes dades ha de ser eficient, és necessària la utilització de bases de dades (DB).

Una base de dades és un conjunt estructurat de dades que es poden organitzar de manera simple i eficient per un controlador d'aquesta base. Les bases de dades actuals es denominen relacionals, ja que les dades es poden emmagatzemar en diferents taules que en faciliten la gestió i administració. Per això i a fi d'estandarditzar l'accés a les bases de dades s'utilitza un llenguatge denominat SQL (*structured query language*), que permet una interacció flexible, ràpida i independent de les aplicacions a les bases de dades.

La forma més utilitzada en l'actualitat consisteix a accedir a una base de dades des d'una aplicació que executa codi SQL. Per exemple, és molt comú accedir a una DB per mitjà d'una pàgina web que conté codi PHP o Perl (els més comuns). Quan un client sol·licita una pàgina, s'executa el codi PHP o Perl incrustat a la pàgina, s'accedeix a la DB i es genera la pàgina amb el seu contingut estàtic i el contingut extret de la DB que posteriorment s'envia al client. Dos dels exemples més actuals de bases de dades són els aportats per ProstgreSQL i MySQL, que serà objecte de la nostra anàlisi.

Tanmateix, quan es treballa en el desenvolupament d'un programari, hi ha altres aspectes relacionats amb les dades, que són la seva validesa i el seu àmbit (sobretot si hi ha un conjunt d'usuaris que treballen sobre les mateixes dades). Hi ha diversos paquets per al **control de versions** (revisions), però l'objectiu de tots ells és facilitar l'administració de les diferents versions de cada producte desenvolupat juntament amb les possibles especialitzacions fetes per a algun client específic.

El control de versions es duu a terme per tal de controlar les diferents versions del codi font. Tanmateix, els mateixos conceptes són aplicables a altres àmbits i no sols per a codi font sinó per a documents, imatges, etcètera. Encara que un sistema de control de versions es pot fer de manera manual, és molt aconsellable disposar d'eines que facilitin aquesta gestió (CVS, Subversion, SourceSafe, Clear Case, Darcs, Plastic SCM, RCS, etc.).

En aquest capítol veurem CVS (*control version system*) i Subversion per a controlar i administrar múltiples revisions d'arxius, en automatitzar l'emmagatzematge, lectura, identificació i barreja de diferents revisions. Aquests programes són útils quan un text es revisa freqüentment i inclou codi font, executables, biblioteques, documentació, gràfics, articles i altres arxius. [Pos03e, Mys, Ced]

La justificació de CVS i Subversion es pot trobar en el fet que CVS és un dels paquets tradicionals i més utilitzats i Subversion és un programari de sistema de control de versions dissenyat específicament per a reemplaçar el popular CVS i en soluciona diverses deficiències. El Subversion també es coneix com a svn perquè aquest és el nom de l'eina de línia d'ordres. Una característica important de Subversion és que, a diferència de CVS, els arxius amb versions no tenen cadascun d'ells un número de revisió independent. En canvi, tot el dipòsit té un únic número de versió que identifica un estat comú de tots els arxius del dipòsit en un punt del temps determinat.

1. PostgreSQL

En el llenguatge de bases de dades (DB), PostgreSQL utilitza un model client servidor [Posa]. Una sessió de PostgreSQL consisteix en una sèrie de programes que cooperen:

- Un procés servidor que utilitza els arxius de la DB accepta connexions dels clients i fa les accions que sol·liciten sobre la DB. El programa servidor s'anomena en PostgreSQL *postmaster*.

- L'aplicació del client (*frontend*) és la que sol·licita les operacions que cal fer en la DB i que poden ser molt variades; per exemple: eines en format text, gràfiques, servidors de web, etc.

Generalment, el client i el servidor es troben en diferents *hosts* i es comuniquen per mitjà d'una connexió TCP/IP. El servidor pot acceptar múltiples peticions de diferents clients, i activar per a cada nova connexió un procés que l'atendrà en exclusiva d'una manera transparent per a l'usuari. Hi ha un conjunt de tasques que pot dur a terme l'usuari o l'administrador, segons convingui, i que passem a descriure a continuació.

1.1. Com s'ha de crear una DB?

La primera acció per a verificar si es pot accedir al servidor de DB és crear una base de dades. El servidor PostgreSQL pot utilitzar moltes DB i és recomanable utilitzar-ne una de diferent per a cada projecte. Per a crear una base de dades, s'utilitza l'ordre `createdb` des de la línia d'ordres del sistema operatiu. Aquesta ordre generarà un missatge *CREATE DATABASE* si tot és correcte. És important tenir en compte que per a aquesta acció hi ha d'haver un usuari habilitat per a crear una base de dades. Es veurà en l'apartat d'instal·lació (1.4) que hi ha un usuari, el que instal·la la base de dades, que tindrà permisos per a crear bases de dades i crear nous usuaris que al seu torn puguin crear bases de dades. Generalment (i a Debian) aquest usuari és postgres per defecte. Per això, abans de fer el createdb, s'ha de fer un postgres (si s'és *root,* no és necessària cap paraula clau, però si s'és un altre usuari, necessitarem la paraula clau de postgres) i després es podrà fer el createdb. Per a crear una DB anomenada nteumdb:

```
createdb nteumdb
```

Si no trobeu l'ordre, pot ser que no estigui ben configurat el camí o que la DB estigui mal instal·lada. Es pot intentar amb tot el camí `(/usr/local/`

`pgsql/bin/createdb nteumdb`), que depèn de la instal·lació que s'hagi fet, o consulteu referències per a la solució de problemes. Altres missatges d'error serien *could not connect to server* quan el servidor no està arrencat o *CREATE DATABASE: permission denied* quan no es tenen privilegis per a crear la DB. Per tal d'eliminar la base de dades, es pot utilitzar `dropdb nteumdb`.

1.2. Com es pot accedir a una DB?

Una vegada creada la DB, s'hi pot accedir de diverses maneres:

Nota

Per a poder accedir a la DB, el servidor de base de dades haurà de ser en funcionament. Quan s'instal·la Postgresql es creen els enllaços adequats perquè el servidor s'inicii en l'arrencada de l'ordinador. Per a més detalls, consulteu l'apartat d'instal·lació (1.4).

- Executant una ordre interactiva anomenada `psql`, que permet editar i executar ordres SQL (p. ex. psql nteumdb).

- Executant una interfície gràfica com PgAccess o alguna *suite* que tingui suporti ODBC per a crear i manipular DB.

- Escrivint una aplicació utilitzant alguns dels llenguatges suportats, per exemple, PHP, Perl, Java... (vegeu PostgreSQL 7.3 Programmer's Guide).

Per simplicitat, utilitzarem psql per a accedir a la DB, per la qual cosa s'haurà d'introduir `psql nteumdb`: sortiran uns missatges amb la versió i informació i un *prompt* similar a nteumdb =>. Es poden executar algunes de les ordres SQL següents:

```
SELECT version();
```

o també

```
SELECT current date;
```

psql també tenen ordres que no són SQL i comencen per '\' per exemple \h (llista de totes les ordres disponibles) o \q per a acabar.

Exemple

Accediu a la DB nteumdb:

```
psql nteumdb [enter]
nteumdb =>
```

1.3. El llenguatge SQL

No és la finalitat d'aquest apartat fer un programa d'aprenentatge sobre SQL, però s'analitzaran uns exemples per a veure les capacitats d'aquest llenguatge. Són exemples que vénen amb la distribució de PostgreSQL en el directori DirectorioInstalacion/src/tutorial, per a accedir-hi, canvieu al directori de PostgreSQL (`cd DirectorioInstalación/src/tutorial`) i exe-

cuteu `psql -s nteumdb` i després, a dins \i basics.sql. El paràmetre \i llegeix les ordres de l'arxiu especificat (basic.sql en el nostre cas).

PostgreSQL és una base de dades relacional (*relational database management system*, RDBMS), la qual cosa significa que utilitza les dades emmagatzemades en taules. Cada taula té un nombre determinat de files i de columnes i cada columna té un tipus específic de dades. La taules s'agrupen en una DB i un únic servidor utilitza aquesta col·lecció de DB (tot el conjunt es denomina agrupació de bases de dades, *database cluster)*.

Per a crear, per exemple, una taula amb psql, executeu:

```
CREATE TABLE temps (
        ciutat          varchar(80),
        temp_min        int,
        temp_max        int,
        pluja           real,
        dia             date
);
```

Exemple

Creeu taula. Dins de *psql*:

```
CREATE TABLE NomTB (var1 tipus, var2 tipus, ...);
```

L'ordre acaba quan es posa ';' i es poden utilitzar espais en blanc i *tabs* lliurement. varchar(80) especifica una estructura de dades que pot emmagatzemar fins a 80 caràcters (en el nostre cas). El *point* és un tipus específic de PostgreSQL.

Per a esborrar la taula:

```
DROP TABLE nom_taula;
```

Per a introduir dades, es poden utilitzar dues formes, la primera és posar totes les dades de la taula i la segona, indicar les variables i els valors que es volen modificar:

Nota

Un segon exemple podria ser:

```
CREATE TABLE ciutat (
nom varchar(80),
        lloc
        point
);
```

```
INSERT INTO temps VALUES ('Barcelona', 16, 37, 0.25, '2007-03-19');

INSERT INTO temps (ciutat, temp_min, temp_max, pluja, dia)

VALUES ('Barcelona', 16, 37, 0.25, '2007-03-19');
```

Aquesta forma pot ser senzilla per a unes quantes dades, però quan cal introduir gran quantitat de dades, es poden copiar des d'un arxiu amb la sentència:

Nota

Es recomana veure el capítol 3 de PostgreSQL sobre característiques avançades (Views, Foreign Keys, Transactions, http://www.postgresql.org/docs/8.2/static/tutorial-advanced.html

[Pos03d]

`COPY temps FROM '/home/user/tiempo.txt';` (aquest arxiu ha de ser al servidor, no al client).

Per a mirar una taula, podríem fer:

```
SELECT * FROM temps;
```

En què el * significa totes les columnes.

Exemples

Introduïu dades en taula. Dins de psql:

```
INSERT INTO NomTB (valorVar1, ValorVar2...);
```

Dades des d'un arxiu. Dins de psql:

```
COPY NomTB FROM 'NomArxiu';
```

Visualitzeu dades. Dins de *psql*:

```
SELECT * FROM NomTB;
```

Exemples d'ordres més complexes serien (dins de *psql*):

- Visualitza la columna ciutat després de fer l'operació:

```
SELECT ciutat (temp_max+temp_min)/2 AS temp_media, date FROM temps;
```

- Visualitza tot on es compleix l'operació lògica:

```
SELECT * FROM temps WHERE city = 'Barcelona'
        AND pluja \verb+>+ 0.0;
```

- Unió de taules:

```
SELECT * FROM temps, ciutat WHERE ciutat = nom;
```

- Funcions, màxim en aquest cas:

```
SELECT max(temp_min) FROM temps;
```

- Funcions en etapes múltiples:

```
SELECT ciutat FROM temps WHERE temp_min = (SELECT max(temp_min)
FROM temps);
```

- Modificació selectiva:

```
UPDATE temps SET temp_max = temp_max 2, temp_min = temp_min 2
WHERE dia >'19990128';
```

- Esborrament del registre:

```
DELETE FROM temps WHERE ciutat = 'Sabadell';
```

1.4. Instal·lació PostgreSQL

Aquest pas és necessari per als administradors de la DB [Posa]. Dins de les funcions de l'administrador de DB s'inclou a la instal·lació del programari, inicialització i configuració, administració dels usuaris, DB i tasques de manteniment de la DB.

La instal·lació de la base de dades es pot fer de dues maneres, mitjançant els binaris de la distribució, la qual cosa no presenta cap dificultat, ja que els *scripts* de distribució fan tots els passos necessaris per a tenir la DB operativa, o mitjançant el codi font, que caldrà compilar i instal·lar. En el primer cas, es pot utilitzar (Debian) el *kpackage* o *apt-get*. En el segon cas, es recomana sempre anar a l'origen (o a un dipòsit mirall de la distribució original). És important tenir en compte que la instal·lació des del codi font quedarà després fora de la

DB de programari instal·lat i es perdran els beneficis d'administració de programari que presenti per exemple *apt-cache* o *apt-get*.

Instal·lació des del codi font pas a pas

- Primer, s'ha d'obtenir el programari del lloc (x.x és la versió disponible) http://www.postgresql.org/download/ i es descomprimeix (x.x.x és la versió 8.2.3 en el moment de fer aquesta revisió):

```
gunzip postgresql-x.x.x.tar.gz
tar xf postgresql-7.3.tar
```

- Canviar-se al directori postgresql i configurar-lo amb `. /configure`.

- Compilar-lo amb `gmake`, verificar la compilació amb `gmake check` i instal·lar-lo amb `gmake install` (per defecte, ho farà en /usr/local/pgsql).

1.4.1. Postinstal·lació

Inicialitzeu les variables, en *bash*, *sh*, *ksh*:

```
LD_LIBRARY_PATH = /usr/local/pgsql/lib;
PATH = /usr/local/pgsql/bin:$PATH;
export LD_LIBRARY_PATH PATH;
```

o bé, en csh:

```
setenv LD_LIBRARY_PATH /usr/local/pgsql/lib;
set path = (/usr/local/pgsql/bin $path)
```

És recomanable posar aquesta inicialització en els *scripts* de configuració de l'usuari, per exemple /etc/etc/profile o .bashrc per al *bash*. Per a tenir accés als manuals, s'ha d'inicialitzar la variable MANPATH de la mateixa manera:

```
MANPATH = /usr/local/pgsql/man:$MANPATH;
export MANPATH
```

Una vegada instal·lada la DB, s'haurà de crear un usuari que utilitzarà les bases de dades (és convenient crear un usuari diferent del *root* perquè no tingui connexió amb altres serveis de la màquina), per exemple, l'usuari postgres en utilitzar l'ordre useradd, per exemple.

A continuació, s'ha de crear una àrea d'emmagatzematge per a les bases de dades (espai únic) sobre el disc que serà un directori, per exemple /usr/local/pgsql/data. Per això, executeu l'ordre `initdb -D /usr/local/pgsql/data`, connectat com l'usuari creat en el punt anterior. Pot rebre un missatge que no pot crear el directori per falta de privilegis, per la qual cosa primer s'haurà de crear el directori i després indicar a la DB quin és; com a *root*, cal fer, per exemple:

```
mkdir /usr/local/pgsql/data
chown postgres /usr/local/pgsql/data su postgres
initdb -D /usr/local/pgsql/data
```

Inicieu el servidor (que s'anomena *postmaster*), per això, utilitzeu:

```
postmaster -D /usr/local/pgsql/data
```

per executar-la en format actiu (*foreground*), i per a executar-lo en format passiu (*background*) utilitzeu:

```
postmaster -D /usr/local/pgsql/data > logfile 2>&1 &.
```

Les redireccions es fan per a emmagatzemar els errors del servidor. El paquet també inclou un *script* (pg_ctl) per tal de no haver de conèixer tota la sintaxi de *postmaster* per a executar-lo:

```
/usr/local/pgsql/bin/pg_ctl start -l logfile \
-D /usr/local/pgsql/data
```

Per a avortar l'execució del servidor, es pot fer de diferents maneres, amb el pg-ctl, per exemple, o bé directament amb:

```
kill -INT 'head -1 /usr/local/pgsql/data/postmaster.pid'
```

1.4.2. Usuaris de DB

Els usuaris de la DB són completament diferents dels usuaris del sistema operatiu. En alguns casos podria ser interessant mantenir una correspondència, però no és necessari. Els usuaris són per a totes les DB que controla aquest servidor, no per a cada DB. Per a crear un usuari, executeu la sentència SQL:

Nota

Crear, esborrar usuaris:
createuser [opcions] nom
dropuser [opcions] nom

```
CREATE USER nom
```

Per a esborrar usuaris:

```
DROP USER nom
```

També es pot cridar els programes `createuser` i `dropuser` des de la línia d'ordres. Hi ha un usuari per defecte anomenat *postgres* (dins de la DB), que és el que permetrà crear-ne la resta (per a crear nous usuaris des de `psql -U postgres` si l'usuari de sistema operatiu amb què s'administra la DB no és *postgres*).

Un usuari de DB pot tenir un conjunt d'atributs en funció del que pot fer:

- Superusuari: aquest usuari no té cap restricció. Per exemple, pot crear nous usuaris; per això, executeu:

```
CREATE USER nom CREATEUSER
```

- Creador de DB: té permís per a crear DB. Per a crear un usuari d'aquestes característiques, utilitzeu l'ordre:

  CREATE USER nom CREATEDB

- *Password*: només és necessari si per qüestions de seguretat es vol controlar l'accés dels usuaris quan es connectin a una DB. Per a crear un usuari amb contrasenya, es pot utilitzar:

  CREATE USER nom PASSWORD 'paraula_clau'

  en què paraula_clau serà la clau per a aquest usuari.

- A un usuari se li poden canviar els atributs utilitzant l'ordre ALTER USER. També es poden fer grups d'usuaris que comparteixin els mateixos privilegis amb:

  CREATE GROUP NomGrup

  I per a inserir usuaris en aquest grup:

  ALTER GROUP NomGrup ADD USER Nom1

  O per esborrar:

  ALTER GROUP NomGrup DROP USER Nom1

Exemple

Operacions amb grup dins de psql:

CREATE GROUP *NomGrup*;
ALTER GROUP *NomGrup* ADD USER *Nom1*...; ALTER GROUP *NomGrup*
DROP USER *Nom1*...;

Quan es creï una DB, els privilegis són per a l'usuari que la creï (i per al superusuari). Per a permetre que un altre usuari utilitzi aquesta DB o una part d'ella, se li han de concedir privilegis. Hi ha diferents tipus de privilegis com SELECT, INSERT, UPDATE, DELETE, RULE, REFERENCES, TRIGGER, CREATE, TEMPORARY, EXECUTE, USAGE i ALL PRIVILEGES (consulteu les referències per a veure el seu significat). Per a assignar els privilegis, es pot utilitzar:

GRANT UPDATE ON objecte TO usuari

en què usuari haurà de ser un usuari vàlid de PostgerSQL i objecte, una taula, per exemple. Aquesta ordre l'haurà d'executar el superusuari o el propietari de la taula. L'usuari PUBLIC pot ser utilitzat com a sinònim de tots els

usuaris i ALL, com a sinònim de tots els privilegis. Per exemple, per a treure tots els privilegis a tots els usuaris d'objecte, es pot executar:

```
REVOKE ALL ON objecte FROM PUBLIC;
```

1.5. Manteniment

Hi ha un conjunt de tasques que són responsabilitat de l'administrador de DB i que s'han de dur a terme periòdicament:

1) Recuperar l'espai: s'haurà d'executar periòdicament l'ordre VACUUM, la qual recupera l'espai de disc de files esborrades o modificades, actualitza les estadístiques utilitzades pel planificador de PostgreSQL i millora les seves condicions d'accés.

2) Reindexar: PostgreSQL en alguns casos concrets pot causar alguns problemes amb la reutilització dels índexs, per aquest motiu és convenient utilitzar REINDEX periòdicament per a eliminar pàgines i files. També es pot utilitzar contrib/reindexdb per a reindexar una DB sencera (s'ha de tenir en compte que, segons la mida de les DB, aquestes ordres poden trigar un cert temps).

3) Canvi d'arxius *log*: s'ha d'evitar que els arxius de *log* siguin de mida molt gran i difícils d'utilitzar. Es pot fer fàcilment quan s'inicia el servidor amb:

```
pg_ctl start | logrotate
```

logrotate rebateja i obre un nou arxiu de *log* i es pot configurar amb l'arxiu /etc/logrotate.conf.

4) Còpia de seguretat i recuperació (*backup* i *recovery*): hi ha dues maneres de salvar les dades, per la sentència SQL Dump o salvant l'arxiu de la DB. El primer és:

```
pg_dump ArxiuDB > ArxiuBackup
```

Per a recuperar-lo, es pot utilitzar:

```
psql ArxiuDB < ArxiuDBBackup
```

Per a salvar totes les DB del servidor, es pot executar:

```
pg_dumpall > ArxiuDBBackupTotal
```

Una altra estratègia és salvar els arxius de les bases de dades en l'àmbit del sistema operatiu, per exemple amb:

```
tar -cf backup.tar /usr/local/pgsql/data
```

Hi ha dues restriccions que poden fer que aquest mètode sigui poc pràctic:

- El servidor s'ha d'aturar abans de salvar i de recuperar les dades.

- S'han de conèixer molt bé totes les implicacions a nivell arxiu en què són totes les taules, transaccions i altres, ja que si no, una DB pot quedar inútil. A més (generalment), la mida que se salvarà serà més gran que l'efectuat amb els mètodes anteriors, ja que per exemple, amb el pg_dumb no se salven els índexs, sinó l'ordre per a recrear-los.

Resum de la Instal·lació de PostgreSQL:

```
./configure
gmake
su
gmake install
adduser postgres
mkdir /usr/local/pgsql/data
chown postgres /usr/local/pgsql/data
su - postgres
/usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
/usr/local/pgsql/bin/postgres -D /usr/local/pgsql/data >logfile 2>&1 &
/usr/local/pgsql/bin/createdb test
/usr/local/pgsql/bin/psql test
```

1.6. Pgaccess

L'aplicació `pgaccess [NomDB]` (http://www.pgaccess.org/) permet accedir i administrar una base de dades amb una interfície gràfica. La manera més fàcil d'accedir-hi (per exemple, des de KDE) és des d'una terminal, l'administrador de DB haurà de fer, (si no és l'usuari *postgresql*) `xhost+` el qual permet que altres aplicacions es puguin connectar al display de l'usuari actual

```
su postgres pgaccess [NomDB]&
```

Si es configura en 'Preferències' obrirà sempre l'última DB. La figura 1 mostra la interfície de pgaccess.

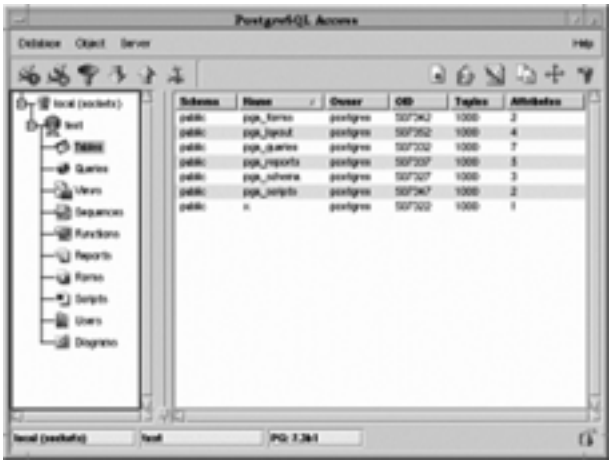

Figura 1. PgAccess

En una sessió típica l'administrador/usuari podria, en primer lloc, Obrir Base de Dades, i indicar aquí, per exemple, Port = 5432, Base de Dades = nteum (els altres paràmetres no són necessaris si la base de dades és local) i després Obrir. A partir d'aquest moment, l'usuari pot treballar amb l'espai bidimensional i seleccionar què és el que vol fer en l'eix Y (taules, consultes, vistes, etc.) i amb aquest element seleccionat, i escollir un d'aquest tipus dins de la finestra, utilitzar l'eix *X* superior per a Nou (afegir), Obrir o Disseny. Per exemple, si se selecciona a Y Usuaris i a X, Nou, l'aplicació sol·licitarà el nom de l'usuari, la seva contrasenya (amb verificació), la seva validesa en el temps i les seves característiques (per exemple, Crear DB, Crear altres usuaris). En Base de Dades també es podria seleccionar Preferències per a, per exemple, canviar el tipus de font i seleccionar la possibilitat de veure les taules del sistema.

Les configuracions personals dels usuaris queden registrades a l'arxiu ~/.pgaccessrc. La interfície permet fer/agilitar gran part del treball de l'usuari/administrador i és recomanable per a usuaris que s'acaben d'iniciar en PostgreSQL, ja que no necessiten conèixer la sintaxi de la línia d'ordre com en psql (la pròpia aplicació sol·licitarà mitjançant múltiples finestres totes les opcions d'una ordre).

Una eina més simple és mitjançant el mòdul corresponent de webmin (és necessari instal·lar els paquets webmin-core i els mòduls necessaris, per exemple en aquest cas webmin-postgresql), però la seva inclusió en moltes distribucions no hi és per defecte (consulteu més informació en http://www.webmin.com/). Durant la instal·lació, el webmin donarà un avís que l'usuari principal serà el *root* i utilitzarà la mateixa contrasenya que el *root* del sistema operatiu. Per a connectar-s'hi, es podrà fer, per exemple, des d'un navegador, https://localhost:10000, el qual sol·licitarà acceptar (o denegar) la utilització del certificat per a la comunicació SSL i, a continuació, mostrarà tots els serveis que pot administrar, entre ells PostgreSQL Data Base Server.

2. Mysql

MySQL [Mys] és (segons els seus autors) la base de dades (DB) SQL oberta, és a dir, programari lliure (Open Source) més popular, i és desenvolupada i distribuïda per MySQL AB (companyia comercial que obté els seus beneficis dels serveis que proveeix sobre la DB). MySQL és un DBMS (*database management system*). Un DBMS és el que pot afegir i processar les dades emmagatzemades dins de la DB. Igual que PostgreSQL, MySQL és una base de dades relacional, la qual cosa significa que emmagatzema les dades en taules en lloc d'una única ubicació, la qual cosa permet més velocitat i flexibilitat. En ser programari lliure, qualsevol pot obtenir el codi, estudiar-lo i modificar-lo d'acord amb les seves necessitats sense cap pagament, ja que MySQL utilitza llicència GPL. MySQL proveeix a la seva pàgina web un conjunt d'estadístiques i prestacions en comparació amb altres DB per a mostrar a l'usuari com de ràpida, fiable i fàcil és d'usar. La decisió d'elegir una DB s'ha de fer acuradament en funció de les necessitats dels usuaris i de l'entorn en què s'utilitzarà aquesta DB.

2.1. Instal·lació

- Obteniu des de http://www.mysql.com/ o des de qualsevol dels dipòsits de programari. Es poden obtenir els binaris i els arxius font per a compilar-los i instal·lar-los.

- En el cas dels binaris, utilitzeu la distribució de Debian i seleccioneu els paquets mysql-* (client, server, common són necessaris). La instal·lació, després d'unes preguntes, crearà un usuari mysql i una entrada en /etc/init.d/mysql per a arrencar/aturar el servidor en el *boot*. També es pot fer manualment fent:

  ```
  /etc/init.d/mysql start|stop
  ```

- Per a accedir a la base de dades, es pot utilitzar el monitor mysql des de la línia d'ordre. Si obté els binaris (no Debian ni RPM, amb això simplement utilitzeu les comunes -apt-get, rpm-), per exemple gz des del lloc web de MySQL, s'hauran d'executar les ordres següents per a instal·lar la DB:

  ```
  groupadd mysql
  useradd -g mysql mysql
  cd /usr/local
  gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xvf -
  ln -s full-path-to-mysql-VERSION-OS mysql
  cd mysql
  scripts/mysql_install_db --user=mysql
  chown -R root .
  chown -R mysql data
  chgrp -R mysql .
  bin/mysqld_safe --user=mysql &
  ```

Això crea l'usuari/grup/directori, descomprimeix, i instal·la la base de dades en /usr/local/mysql.

- En el cas d'obtenir el codi font, els passos són similars:

```
groupadd mysql
useradd -g mysql mysql
gunzip < mysql-VERSION.tar.gz | tar -xvf -
cd mysql-VERSION
./configure --prefix=/usr/local/mysql
make
make install
cp support-files/my-medium.cnf /etc/my.cnf
cd /usr/local/mysql
bin/mysql_install_db --user=mysql
chown -R root .
chown -R mysql var
chgrp -R mysql .
bin/mysqld_safe --user=mysql &
```

És important parar atenció quan s'executa la instrucció configure, ja que `prefix=#/usr/local/mysql` és el directori en què s'instal·la la DB i es pot canviar per a ubicar la DB en el directori que es vulgui.

2.2. Postinstal·lació i verificació

Una vegada instal·lada (ja sigui dels binaris o del codi font), s'haurà de verificar si el servidor funciona correctament. A Debian es pot fer directament:

`/etc/init.d/mysql start`	Inicia el servidor
`mysqladmin version`	Genera información de versions
`mysqladmin variables`	Mostra els valors de les variables
`mysqladmin -u root shutdown`	Finalitza l'execució del `servidor`
`mysqlshow`	Mostrarà les DB predefinides
`mysqlshow mysql`	Mostrarà les taules de la DB mysql

Si s'instal·la des del codi font, abans de fer aquestes comprovacions s'han d'executar les ordres següents per a crear les bases de dades (des del directori de la distribució):

```
./scripts/mysql_install_db
cd DirectorioInstalacionMysql
./bin/mysqld_safe --user = mysql &
```

Si s'instal·la des de binaris (RPM, Pkg...), s'ha de fer el següent:

```
cd DirectorioInstalacionMysql
./scripts/mysql_install_db
./bin/mysqld_safe user = mysql &
```

L'*script* `mysql_install_db` crea la DB `mysql` i `mysqld_safe` arrenca el servidor mysqld. A continuació, es poden provar totes les ordres donades anteriorment per a Debian, excepte la primera, que és la que arrenca el servidor. A més, si s'hi han instal·lat els tests, es podran executar amb `cd sql-bench` i després `run-all-tests`. Els resultats es trobaran en el directori `sql-bech/Results` per a comparar-los amb altres DB.

2.3. El programa monitor (client) mysql

Nota

Client (frontend) mysql:
mysql [NomDB]

Nota

Per a més informació, podeu consultar la documentació, ordres i opcions. [Mys07]
http://dev.mysql.com/doc/refman/5.0/es/

El client mysql es pot utilitzar per a crear i utilitzar DB simples, és interactiu i permet connectar-se al servidor, executar cerques i visualitzar els resultats. També funciona en format *batch* (com un *script*) en què les ordres es passen per mitjà d'un arxiu. Per tal de veure totes les opcions de l'ordre, es pot executar `mysql --help`. Podrem fer una connexió (local o remota) amb l'ordre mysql, per exemple, per a una connexió per la interfície de xarxa però des de la mateixa màquina:

```
mysql -h localhost -o mysql -p NomDB
```

Si no es posa l'últim paràmetre, no se selecciona cap DB.

Una vegada dins, el mysql posarà un *prompt* (mysql>) i esperarà que hi introduïm alguna ordre (pròpia i SQL), per exemple help. A continuació, donarem una sèrie d'ordres per a provar el servidor (recordeu posar-hi sempre el ';' per a acabar l'ordre):

```
mysql> SELECT VERSION(), CURRENT_DATE;
```

Es poden utilitzar majúscules o minúscules:

```
mysql> SELECT SIN(PI()/4), (4 +1) * 5; Calculadora.
mysql> SELECT VERSION(); SELECT NOW();
```

Múltiples ordres en la mateixa línia:

```
mysql> SELECT
-> USER()
-> ,
-> CURRENT_DATE;
```

O en múltiples línies:

```
mysql> SHOW DATABASES;
```

Mostra les DB disponibles:

```
mysql> USE test
```

Canvia la DB:

```
mysql> CREATE DATABASE nteum; USE nteum;
```

Crea i selecciona una DB anomenada nteum:

```
mysql> CREATE TABLE pet (name VARCHAR(20), owner VARCHAR(20),
species VARCHAR(20), sex CHAR(1), birth DATE, death DATE);
```

Crea una taula dins de nteum:

```
mysql> SHOW TABLES;
```

Mostra les taules:

```
mysql> DESCRIBE pet;
```

Mostra la definició de la taula:

```
mysql> LOAD DATA LOCAL INFILE “pet.txt” INTO TABLE pet;
```

Carrega dades des de pet.txt en pet. L'arxiu pet.txt ha de tenir un registre per línia separat per *tabs* de les dades d'acord amb la definició de la taula (data en format AAAA-MM-DD):

```
mysql> INSERT INTO pet
-> VALUES (‘Marcià’,‘Estela’,‘gat’,‘f’,‘1999-03-30’,NULL);
```

Carrega les dades *in-line*:

```
mysql> SELECT * FROM pet; Mostra les dades de la taula.
mysql> UPDATE pet SET birth = “1989-08-31” WHERE name = “Browser”;
```

Modifica les dades de la taula:

```
mysql> SELECT * FROM pet WHERE name = “Browser”;
```

Mostra selectiva:

```
mysql> SELECT name, birth FROM pet ORDER BY birth;
```

Mostra ordenada:

```
mysql> SELECT name, birth FROM pet WHERE MONTH(birth) = 5;
```

Mostra selectiva amb funcions:

```
mysql> GRANT ALL PRIVILEGES ON *.* TO marcià@localhost
IDENTIFIED BY ‘passwd’ WITH GRANT OPTION;
```

Crea un usuari marcià en la DB. Ho ha de fer el *root* de la DB. O també es pot fer directament amb:

```
mysql> INSERT INTO user (Host,User,Password)
VALUES(‘localhost’,‘marcià’,‘passwd’);
```

2.4. Administració

Mysql disposa d'un arxiu de configuració en /etc/mysql/my.cnf (a Debian), al qual s'hi poden canviar les opcions per defecte a la DB, com per exemple, el port de connexió, usuari, contrasenya dels usuaris remots, arxius de *log*, arxius de dades, si accepta connexions externes, etc. Respecte a la seguretat, s'han de prendre algunes precaucions:

1) No donar a ningú (excepte a l'usuari *root* de mysql) accés a la taula *user* dins la DB mysql, ja que aquí es troben les contrasenyes dels usuaris que podrien ser utilitzades amb altres finalitats.

2) Verificar `mysql -u root`. Si s'hi pot accedir, significa que l'usuari *root* no té contrasenya. Per a canviar-ho, es pot fer:

```
mysql -u root mysql
mysql> UPDATE user SET Password = PASSWORD('new_password')
WHERE user = 'root';
mysql> FLUSH PRIVILEGES;
```

Ara, per a connectar-se com a *root*:

```
mysql -u root -p mysql
```

3) Comprovar la documentació (punt 4.2) respecte a les condicions de seguretat i de l'entorn de xarxa per a evitar problemes d'atacs i/o intrusió.

4) Per a fer còpies de la base de dades, es pot utilitzar l'ordre:

```
mysqldump --tab = /DirectoriDestinació --opt NomDB
```

o també:

```
mysqlhotcopy NomDB /DirectoriDestinació
```

Així mateix, es poden copiar els arxius *.frm', *.MYD', i *.MYI amb el servidor aturat. Per a recuperar-la, executeu:

```
REPAIR TABLE o myisamchk -r
```

que funcionarà el 99% de les vegades. En cas contrari, es podrien copiar els arxius salvats i arrencar el servidor. Hi ha altres mètodes alternatius en funció del que es vulgui recuperar, com la possibilitat de salvar/recuperar part de la DB (consulteu punt 4.4 de la documentació). [Mys]

2.5. Interfícies gràfiques

Per a Mysql hi ha gran quantitat d'interfícies gràfiques, entre les quals destaquem Mysql Administrator (es pot obtenir des de l'adreça d'Internet

http://www.mysql.com/products/tools/administrator/). També poden servir com a eines Mysql-Navigator (http://sourceforge.net/projects/mysqlnavigator/), o Webmin amb el mòdul per a treballar amb Mysql (paquets webmin-core i webmin-mysql) si bé aquest últim ja no s'inclou amb algunes distribucions. De manera anàloga, a PostgreSQL, webmin permet també treballar amb Mysql (és necessari instal·lar-hi els paquets webmin-mysql a més del webmin-core). Durant la instal·lació, el webmin donarà un avís que l'usuari principal serà el *root* i utilitzarà la mateixa contrasenya que el *root* del sistema operatiu. Per a connectar-s'hi, es podrà fer, per exemple, des d'un navegador: https://localhost:10000, el qual sol·licitarà acceptar (o denegar) la utilització del certificat per a la comunicació SSL i a continuació mostrarà tots els serveis que pot administrar, entre ells Mysql Data Base Server.

MySQL Administrator és una aplicació potent per a l'administració i control de bases de dades basades en MySQL. Aquesta aplicació integra la gestió i control de la BD i el manteniment de manera simple i en un mateix entorn. Les característiques principals són: administració avançada de grans DB, reducció d'errors mitjançant una "administració visual", més productivitat i un entorn segur de gestió. La figura següent mostra un aspecte de MySQL Administrator (en http://dev.mysql.com/doc/administrator/en/index.html es pot trobar tota la documentació per a la seva instal·lació i posada en marxa).

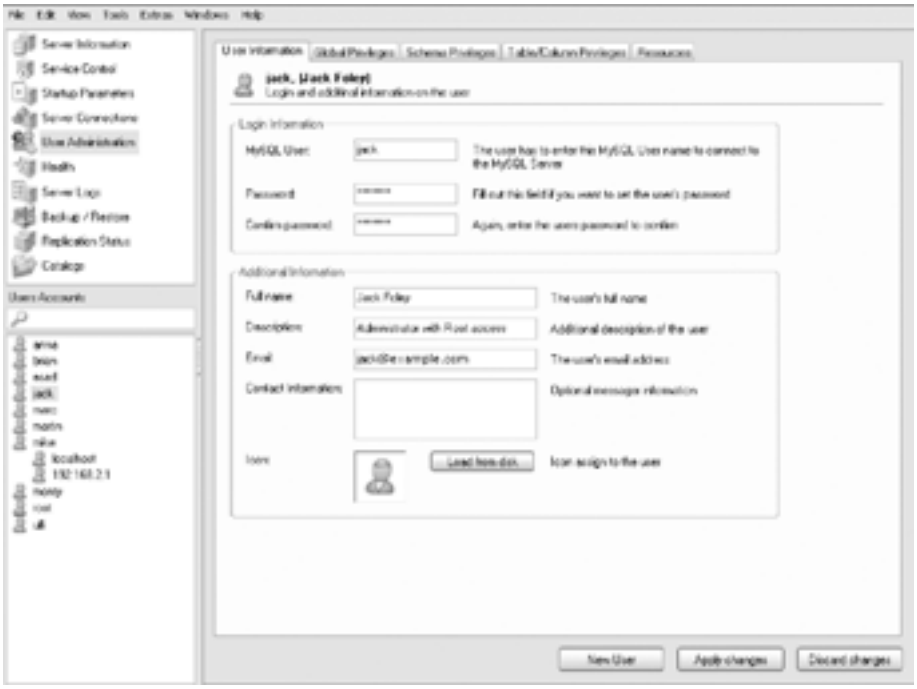

Figura 2. MySQL Administrator

3. *Source code control system* (CVS i Subversion)

El *concurrent versions system* (CVS) és un sistema de control de versions que permet mantenir versions antigues d'arxius (generalment codi font), i desar un registre (*log*) de per qui, quan i per què van ser fets els canvis. A diferència d'altres sistemes, CVS no treballa amb un arxiu/directori per vegada, sinó que actua sobre col·leccions jeràrquiques dels directoris que controla.

El CVS té per objectiu ajudar a utilitzar versions de programari i controla l'edició concurrent d'arxius font per múltiples autors. El CVS utilitza un altre paquet anomenat RCS (*revision control system*) internament com una capa de baix nivell. Si bé l'RCS es pot utilitzar independentment, no s'aconsella, ja que CVS a més de la seva pròpia funcionalitat presenta totes les prestacions d'RCS però amb millores notables quant a l'estabilitat, operació i manteniment. Entre elles podem destacar: funcionament descentralitzat (cada usuari pot tenir el seu propi arbre de codi), edició concurrent, comportament adaptable mitjançant *shell scripts*, etc. [Ced, CVS, Vasa, Kie]

Com ja s'ha explicat en la introducció Subversion (http://subversion.tigris.org/) és un programari de sistema de control de versions dissenyat específicament per a reemplaçar al popular CVS, i estendre les seves capacitats. És programari lliure sota una llicència de tipus Apache/BSD i se'l coneix també com a `svn` pel nom en línia d'ordres. Una característica important de Subversion és que, a diferència de CVS, els arxius versionats no tenen cadascun d'ells un número de revisió independent i al seu lloc tot el dipòsit té un únic número de versió que identifica un estat comú de tots els arxius del dipòsit en el temps que s'ha "versionat". Entre les característiques principals podem esmentar:

- Se segueix la història dels arxius i directoris per mitjà de còpies i reanomenaments.
- Les modificacions atòmiques i segures (inclosos canvis a diversos arxius).
- La creació de ramificacions i etiquetes és eficient i simple.
- S'envien només les diferències en ambdues direccions (en CVS sempre s'envien al servidor arxius complets).
- Es pot servir, mitjançant Apache, sobre WebDAV/DeltaV.
- Utilitza eficientment arxius binaris (a diferència de CVS que els tracta internament com si fossin de text).

Hi ha un llibre (lliure) interessant que explica tot el referent al Subversion http://svnbook.red-bean.com/index.es.html i la traducció al castellà està força avançada (http://svnbook.red-bean.com/nightly/es/index.html).

3.1. *Revision control system* (RCS)

Com que CVS es basa en RCS i en alguns sistemes encara s'utilitza, se'n faran unes breus explicacions. L'RCS és format per un conjunt de programes per a les diferents activitats de l'RCS: `rcs` (programa que controla els atributs dels arxius sota RCS), `ci` i `co` (verifiquen l'entrada i la sortida dels arxius sota el control d'RCS), `ident` (busca en l'RCS els arxius per paraules clau/atributs), rcsclean (neteja arxius no utilitzats o que no han canviat), `rcsdiff` (executa l'ordre diff per a comparar versions), `rcsmerge` (uneix dues ramificacions (arxius) en un únic arxiu), i `rlog` (imprimeix els missatges de *log*).

El format dels arxius emmagatzemats per RCS pot ser text o un altre format, com per exemple binari. Un arxiu RCS consisteix en un arxiu de revisió inicial anomenat 1.1 i una sèrie d'arxius de canvis, un per cada revisió. Cada vegada que es fa una còpia del dipòsit cap al directori de treball amb l'ordre co (obté una revisió de cada arxiu RCS i el posa a l'arxiu de treball) o s'utilitza ci (emmagatzema noves revisions en l'RCS), el número de versió s'incrementa (per exemple, 1.2, 1.3...). Els arxius (generalment) són en el directori /RCS i és necessari que el sistema operatiu tingui instal·lats les ordres `diff` i `diff3` perquè funcioni adequadament. A Debian no és necessari compilar-lo ja que s'inclou en la distribució.

Amb l'ordre rcs crearem i modificarem els atributs dels arxius (consulteu `man rcs`). La manera més fàcil de crear un dipòsit és fer en primer lloc un `mkdir rcs` en el directori d'originals i que inclou els originals al dipòsit amb: `ci nom_arxius_font`.

Es pot utilitzar el * i sempre tenir una còpia de resguard per a evitar problemes. Això crearà les versions dels arxius amb nom `./RCS/nom_arxiu` i sol·licitarà un text per a descriure l'arxiu. Després, amb `co RCS/nom_arxiu`, obtindrem una còpia de treball des del dipòsit. Es pot bloquejar o desbloquejar aquest arxiu per a evitar modificacions, respectivament, amb:

```
rcs -L nom_arxiu_de_treball
rcs -U nom_arxiu_de_treball
```

Amb `rlog nom_arxiu` podrem veure la informació sobre les diferents versions. [Kie]

3.2. *Concurrent versions system* (CVS)

En primer lloc, s'ha d'instal·lar el *concurrent versions system* (CVS) des de la distribució tenint en compte que hem de tenir instal·lat RCS i que haurem d'ins-

tal·lar també OpenSSH si es vol utilitzar juntament amb CVS per a accés remot. Les variables d'entorn EDITOR CVSROOT han d'estar inicialitzades per exemple en /etc/etc/profile (o en .bash profile):

```
export EDITOR = /bin/vi
export CVSROOT = /usr/local/cvsroot
```

Òbviament, els usuaris poden modificar aquestes definicions utilitzant /.bash profile. S'ha de crear el directori en què hi haurà el dipòsit i configurar-ne els permisos; com a *root*, cal fer, per exemple:

```
export CVSROOT = /usr/local/cvsroot
groupadd cvs
useradd -g cvs -d $CVSROOT cvs
mkdir $CVSROOT
chgrp -R cvs $CVSROOT
chmod o-rwx $CVSROOT
chmod ug+rwx $CVSROOT
```

Per a inicialitzar el dipòsit i posar-hi un arxiu de codi:

```
cvs -d /usr/local/cvsroot init
```

cvs init tindrà en compte que no s'ha de sobreescriure mai un dipòsit ja creat per a evitar pèrdues d'altres dipòsits. Després, s'hi hauran d'agregar els usuaris que treballaran amb el CVS al grup cvs; per exemple, per agregar l'usuari *nteum*:

```
usermod -G cvs,nteum
```

Ara l'usuari *nteum* haurà d'introduir els seus arxius en el directori del dipòsit (/usr/local/cvsroot en el nostre cas):

```
export EDITOR = /bin/vi
export CVSROOT = /usr/local/cvsroot
export CVSREAD = yes
cd directori_originals
cvs import NomDelDipòsit vendor_1_0 rev_1_0
```

El nom del dipòsit pot ser un identificador únic o també usuari/projecte/xxxx si és que l'usuari vol tenir organitzats els seus dipòsits. Això crearà un arbre de directoris en CVSROOT amb aquesta estructura.

Això afegeix un directori (/usr/local/cvsroot/NomDelDipòsit) al dipòsit amb els arxius que a partir d'aquest moment seran al dipòsit. Una prova per a saber si s'ha emmagatzemat tot correctament és emmagatzemar una còpia al dipòsit i després crear una còpia des d'allà i comprovar-ne les diferències. Per exem-

ple, si els originals són en el directori_usuari/dir_org i es vol crear un dipòsit com primer_cvs/proj, s'hauran d'executar les ordres següents:

`cd dir_org` Canvia al directori del codi font original.

`cvs import -m` "Fonts originals"
`primer_cvs/proj usuariX vers0`
Crea el dipòsit en primer_cvs/proj amb ususariX i vers0.

`cd..` Canvia al directori superior de dir_org.

`cvs checkout primer_cvs/proj`
Genera una còpia del dipòsit. La variable CVSROOT ha d'estar inicialitzada, si no, s'haurà d'indicar tot el path.

`diff -r dir_org primer_cvs/proj`
Mostra les diferències entre l'un i l'altre; que no n'hi ha d'haver cap excepte pel directori primer_cvs/proj/CVS que ha creat el CVS.

`rm -r dir_org` Esborra els originals (fa una còpia de resguard sempre per seguretat i per tenir una referència d'on es va iniciar el treball amb el CVS).

La figura següent mostra l'organització i com es distribueixen els arxius entre versions i ramificacions.

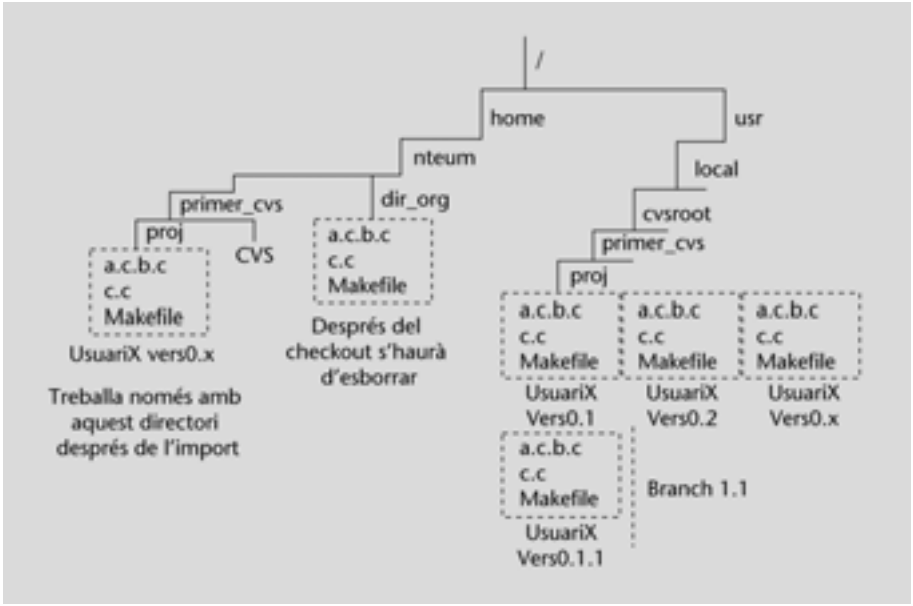

Figura 3

El fet d'esborrar els originals no és sempre una bona idea; només en aquest cas, després que s'hagi verificat que són en dipòsit, perquè per distracció no s'hi treballi a sobre i els canvis no quedin reflectits sobre el CVS. Sobre màquines en què els usuaris volen accedir (per ssh) a un servidor CVS remot, s'haurà de fer:

```
export CVSROOT = ":ext:user@CVS.server.com:/home/cvsroot"
export CVS_RSH = "ssh"
```

En què *user* és la connexió de l'usuari i cvs.server.com el nom del servidor en què es troba CVS. CVS ofereix una sèrie d'ordres (es criden amb cvs comd opcions...) per a treballar amb el sistema de revisions, entre ells: checkout, update, add, remove, commit i diff.

L'ordre inicial `cvs checkout` crea la seva còpia privada del codi font per a després treballar-hi sense interferir en el treball d'altres usuaris (com a mínim es crea un subdirectori en què hi haurà els arxius).

- `cvs update` s'ha d'executar de l'arbre privat quan cal actualitzar les seves còpies d'arxius font amb els canvis que altres programadors han fet sobre els arxius del dipòsit.

- `cvs add file`... és una ordre necessària quan cal agregar nous arxius en el seu directori de treball sobre un mòdul en què ja s'ha fet un checkout prèviament. Aquests arxius s'envien al dipòsit CVS quan s'executi l'ordre cvs commit.

- `cvs import` es pot usar per a introduir arxius nous al dipòsit.

- `cvs remove file`... aquesta ordre s'utilitzarà per a esborrar arxius del dipòsit (una vegada que s'hagin esborrat de l'arxiu privat). Aquesta ordre ha d'anar acompanyada d'un `cvs commit` perquè els canvis siguin efectius, ja que es tracta de l'ordre que transforma totes les peticions dels usuaris sobre el dipòsit.

- `cvs diff file`... es pot utilitzar sense que afecti cap dels arxius implicats si es necessita verificar les diferències entre dipòsit i directori de treball o entre dues versions.

- `cvs tag -R "versió"` es pot utilitzar per a introduir un número de versió als arxius d'un projecte i després fer un cvs commit i un `cvs checkout -r` "version" projecte per a registrar una nova versió.

Una característica interessant del CVS és poder aïllar canvis dels arxius aïllats en una línia de treball separada anomenada *ramificació* (*branch*). Quan es canvia un arxiu sobre una ramificació, aquests canvis no apareixen sobre els arxius principals o sobre altres ramificacions. Més tard, aquests canvis es poden incorporar a altres ramificacions o a l'arxiu principal (*merging*). Per a crear una nova ramificació, utilitzeu `cvs tag -b rel-1-0-patches` dins del directori de treball, la qual cosa assignarà a la ramificació el nom de rel-1-0-patches. La unió de ramificacions amb el directori de treball significa utilitzar l'ordre `cvs update -j`. Consulteu les referències per a barrejar o accedir a diferents ramificacions.

3.2.1. Exemple d'una sessió

Seguint l'exemple de la documentació donada en les referències, es mostrarà una sessió de treball (de manera general) amb CVS. Com que CVS emmagatzema tots els arxius en un dipòsit centralitzat, s'assumirà que ja ha estat inicialitzat anteriorment.

Considerem que s'està treballant amb un conjunt d'arxius en C i un *makefile*, per exemple. El compilador utilitzat és gcc i el dipòsit és inicialitzat a gccrep.

En primer lloc, s'ha d'obtenir una còpia dels arxius del dipòsit a la nostra còpia privada amb:

```
cvs checkout gccrep
```

que crearà un nou directori anomenat gccrep amb els arxius font. Si s'executa `cd gccrep` i `ls`, es veurà per exemple CVS makefile a.c b.c c.c, en què hi ha un directori CVS que es crea per al control de la còpia privada que normalment no cal tocar.

Després d'això es podria utilitzar un editor per a modificar a.c i introduir canvis substancials a l'arxiu (vegeu en la documentació sobre múltiples usuaris concurrents si es necessita treballar amb més d'un usuari en el mateix arxiu), compilar, tornar a canviar, etc.

Quan es decideix que es té una versió nova amb tots els canvis introduïts en a.c (o als arxius que sigui necessari), és moment de fer una nova versió emmagatzemant a.c (o tots els que s'han tocat) al dipòsit i fer aquesta versió disponible a la resta d'usuaris: `cvs commit a.c`.

En utilitzar l'editor definit en la variable CVSEDITOR (o EDITOR si aquesta no està inicialitzada) es pot introduir un comentari que indiqui quins canvis s'han fet perquè serveixi d'ajuda a altres usuaris o per tal de recordar què és el que va caracteritzar aquesta versió i després poder-ne fer un històric.

Si es decideix eliminar els arxius (perquè ja es va acabar amb el projecte o perquè no s'hi treballarà més), una manera de fer-ho és en l'àmbit de sistema operatiu `(rm -r gccrep)`, però és millor utilitzar el propi cvs fora del directori de treball (nivell immediat superior): `cvs release -d gccrep`. L'ordre detectarà si hi ha algun arxiu que no ha estat enviat al dipòsit, i si n'hi ha i s'esborra, significa que es perdran tots els canvis, per això preguntarà si es vol continuar o no.

Per a mirar les diferències, per exemple, s'ha modificat b.c i no es recorda quins canvis s'hi van fer, es pot utilitzar dins del directori de treball: `cvs diff b.c`. Aquest utilitzarà l'ordre del sistema operatiu diff per a comparar la versió b.c amb la versió que hi ha al dipòsit (sempre cal recordar fer un `cvs commit b.c` si es vol que aquestes diferències siguin transferides al dipòsit com una nova versió).

3.2.2. Múltiples usuaris

Quan més d'una persona treballa en un projecte programari amb diferents revisions, és summament complicat perquè hi haurà vegades en què més d'un usuari es vulgui editar el mateix fitxer simultàniament. Una possible solució és bloquejar el fitxer o utilitzar punts de verificació reservats (*reserved checkouts*), la qual cosa només permet que un usuari editi el mateix fitxer simultàniament. Per això, s'haurà d'executar l'ordre `cvs admin -l command` (vegeu man per a les opcions).

CVS utilitza un model per defecte de punts no reservats (*unreserved checkouts*), que permet que els usuaris editin simultàniament un fitxer del seu directori de treball. El primer que transfereixi els seus canvis al dipòsit ho podrà fer sense problemes, però els altres rebran un missatge d'error quan vulguin fer la mateixa tasca, per la qual cosa hauran d'utilitzar ordres de cvs per a transferir en primer lloc els canvis al directori de treball des del dipòsit i després actualitzar el dipòsit amb els seus propis canvis.

Consulteu les referències a fi de veure un exemple d'aplicació i altres formes de treball concurrent amb comunicació entre usuaris. [Vasa].

3.3. Interfícies gràfiques

Disposem d'un conjunt d'interfícies gràfiques com tkcvs (consulteu http://www.twobarleycorns.net/tkcvs.html), gcus, desenvolupada a Tcl/Tk i que suporta subversió, o una altra també molt popular, cervisia [Cerc].

En la *wiki* de CVS, (http://ximbiot.com/cvs/wiki/index.php?title=CVS_Clients) es poden trobar també un conjunt de clients, *plugins* per a CVS. A continuació, es mostren dues d'aquestes interfícies gràfiques (tkcvs i gcvs):

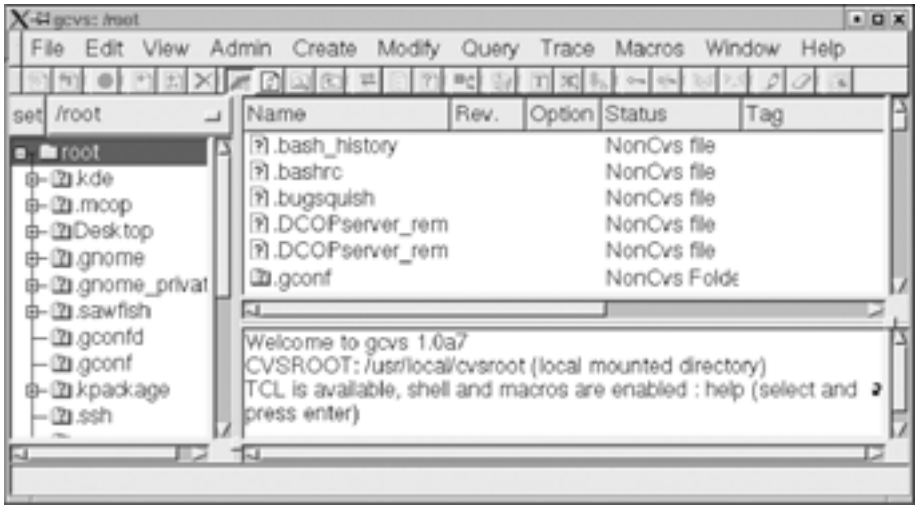

Figura 4. TkCVS (TkSVN)

4. Subversion

Com a idea inicial *subversion* serveix per a gestionar un conjunt d'arxius (dipòsit) i les seves diferents versions. És interessant remarcar que no ens importa com es desen, sinó com s'accedeix a aquests fitxers, per a la qual cosa és comú utilitzar una base de dades. La idea de dipòsit és com un directori del qual es vol recuperar un fitxer de fa una setmana o 10 mesos a partir de l'estat de la base de dades, recuperar les últimes versions i afegir-n'hi una de nova. A diferència de CVS, *subversion* fa les revisions globals del dipòsit, la qual cosa significa que un canvi al fitxer no genera un salt de versió únicament en aquest fitxer, sinó a tot el dipòsit, que en suma un a la revisió. A més del llibre esmentat (http://svnbook.red-bean.com/nightly/es/index.html), consulteu la documentació a http://subversion.tigris.org/servlets/ProjectDocumentList.

Figura 5. gCVS

A Debian haurem de fer `apt-get install subversion`, si es volen publicar els dipòsits en Apache2 `apt-get install Apache2-common` i el mòdul específic `apt-get install libApache2-subversion`.

- Primer pas: Creem el nostre dipòsit, usuari (considerem que l'usuari és svuser), grup (svgroup) com a *root*...

```
mkdir -p /usr/local/svn
addgroup svgroup
chown -R root.svgroup /usr/local/svn
chmod 2775 /usr/local/svn
```

- `addgroup svuser svggroup`
  Agrego l'usuari svuser al grup svgroup.

- Ens connectem com a svuser i verifiquem que som en el grup svgroup (amb l'ordre group).

- `svnadmin create /usr/local/svn/proves`
  Aquesta ordre crea una sèrie d'arxius i directoris per a fer el control i gestió de les versions. Si no es té permís en /usr/local/svn es pot fer en el directori local: `mkdir -p $HOME/svndir` i a continuació `svnadmin create $HOME/svndir/proves`.

- A continuació, creem un directori temporal `mkdir -p $HOME/svntmp/proves` ens passem al directori cd $HOME/svntmp/proves i creem un arxiu per exemple: `echo Primer Arxiu Svn 'date' > file1.txt`.

- El traslladem al dipòsit. Dins del directori executem la sentència `svn import file:///home/svuser/svndir/proves -m` "Ver. Inicial". Si l'hem creat en `/usr/local/svn/proves` hauríem de posar el path complet després de file://. L'`import` copia l'arbre de directoris i el -m permet indicar-li el missatge de versió. Si no posem -m s'obrirà un editor per a fer-ho (s'ha de posar un missatge per a evitar problemes). El subdirectori $HOME/svntmp/proves és una còpia del treball en dipòsit i és recomanable esborrar-la per a no tenir la temptació o l'error de treballar amb ella i no amb el dipòsit (`rm -rf $HOME/svntmp/proves`).

- Una vegada al dipòsit, es pot obtenir la còpia local en què podrem treballar i després pujar les còpies al dipòsit, per a la qual cosa fem:

  ```
  mkdir $HOME/svm-work
  cd $HOME/svn-work
  svn checkout file:///home/svuser/svndir/pruebas
  ```

  En què veurem que tenim el directori proves. Es pot copiar amb un altre nom agregant al final el nom que volem. Per a afegir-hi un nou fitxer:

  ```
  cd /home/kikov/svn-work/proves
  echo Segon Arxiu Svn 'date' > file2.txt
  svn add file2.txt
  svn commit -m "Nou arxiu"
  ```

És important remarcar que una vegada a la còpia local (svn-work) no s'ha d'indicar el *path*. svn add marca per a afegir el fitxer al dipòsit i realment s'afegeix quan fem un cvn commit. Ens donarà alguns missatges i ens indicarà que és la segona versió.

Si agreguem una altra línia la file1.txt amb echo 'date'>>file1.txt, després podem pujar els canvis amb: `svn commit -m "Nova línia"`.

És possible comparar l'arxiu local amb l'arxiu del dipòsit, per exemple, agreguem una tercera línia a file1.txt amb echo `date`>>file1.txt, però no el pugem i si volem veure'n les diferències, podem fer: `svn diff`.

Aquesta ordre ens marca quines són les diferències entre l'arxiu local i els del dipòsit. Si el carreguem amb `svn commit -m "Nova línia2"` (que en generarà una altra versió) després el `svn diff` no ens donarà diferències.

També es pot utilitzar l'ordre `svn update` dins del directori per a actualitzar la còpia local. Si hi ha dos o més usuaris treballant alhora i cadascun ha fet una còpia local del dipòsit i la modifica (fent un commit), quan el segon usuari vagi a fer el commit de la seva còpia amb les seves modificacions, li donarà un error de conflicte, ja que la còpia al dipòsit és posterior a la còpia original d'aquest usuari (és a dir, hi ha hagut canvis per enmig), amb la qual cosa, si el segon usuari fa el commit, perdríem les modificacions del primer. Per això, hem de fer un `svn update` que ens indicarà l'arxiu que crea conflicte amb un C i ens indicarà els arxius en què s'han posat les parts en conflicte. L'usuari haurà de decidir amb quina versió es queda i si podrà fer un commit.

Una ordre interessant és el `svn log file1.txt`, que ens mostra tots els canvis que hi ha hagut al fitxer i les seves versions corresponents.

Un aspecte interessant és que el *subversion* pot funcionar juntament amb Apache2 (i també sobre SSL) per a accedir des d'una altra màquina (consulteu els clients en http://svnbook.red-bean.com/) o simplement mirar el dipòsit. En Debian Administration expliquen com configurar Apache2 i SSL per a Sarge, o com ja vam indicar a la part de servidors. Per això, és necessari activar els mòduls de WebDAV (vegeu http://www.debian-administration.org/articles/285 o en el seu defecte http://www.debian-administration.org/articles/208)).

Com a *root* fem:

```
mkdir /subversion chmod www-data:www-data
```

Perquè Apache pugui accedir al directori:

```
svnadmin create /subversion
```

creem el dipòsit

```
ls -s /subversion
-rw-r--r-- 1 www-data www-data 376 May 11 20:27 README.txt
drwxr-xr-x 2 www-data www-data 4096 May 11 20:27 conf
drwxr-xr-x 2 www-data www-data 4096 May 11 20:27 dav
drwxr-xr-x 2 www-data www-data 4096 May 11 20:28 db
-rw-r--r-- 1 www-data www-data 2 May 11 20:27 format
drwxr-xr-x 2 www-data www-data 4096 May 11 20:27 hooks
drwxr-xr-x 2 www-data www-data 4096 May 11 20:27 locks
```

Per a autenticació utilitzem htpasswd (per exemple amb la sentència `htpasswd2 -c -m /subversion/.dav_svn.passwd` user) creat com a www-data. La -c només cal posar-la la primera vegada que executem l'ordre per a crear l'arxiu. Això indica que per a accedir a aquest directori es necessita passwd (que és el que hem entrat per a user).

Després s'ha de canviar l'httpd.conf perquè sigui alguna cosa semblant a:

```
<location /svn>
  DAV svn
  SVNPath /subversion
  AuthType Basic
  AuthName "Subversion Repository"
  AuthUserFile /subversion/.dav_svn.passwd
  Require valid-user
</location>
```

Reiniciem Apache i ja estem llestos per a importar alguns arxius, per exemple:

```
svn import file1.txt http://url-servidor.org/svn \
-m "Import Inicial"
```

Ens demanarà l'autenticació (user/passwd) i ens dirà que el fitxer file1.txt ha estat afegit al dipòsit.

Activitats

1. Definiu en PostgreSQL una DB que tingui almenys 3 taules amb 5 columnes (de les quals 3 han de ser numèriques) a cada taula.

Genereu una llista ordenada per cada taula/columna. Genereu una llista ordenada pel valor més gran de la columna *X* de totes les taules. Canvieu el valor numèric de la columna Y amb el valor numèric de la columna Z + valor de la columna W/2.

2. El mateix exercici anterior, però amb MySQL.

3. Configureu el CVS per a fer tres revisions d'un directori en què hi ha 4 arxius .c i un makefile. Feu una ramificació (*branch*) d'un arxiu i després barregeu-lo amb el principal.

4. Simuleu la utilització d'un arxiu concurrent amb dos terminals de Linux i indiqueu la seqüència de passos per a fer que dues modificacions alternes de cada usuari quedin reflectides sobre el dipòsit CVS.

5. El mateix exercici anterior, però un dels usuaris s'ha de connectar des d'una altra màquina al dipòsit.

6. Ídem 3, 4 i 5 a Subversion.

Altres fonts de referències i informació

[Debc, Ibi, Mou01]

PgAccess: http://www.pgaccess.org/

WebMin: http://www.webmin.com/

Mysql Administrator:
http://www.mysql.com/products/tools/administrator/

Interfícies gràfiques per a CVS:http://www.twobarleycorns.net/tkcvs.html

O en la wiki de CVS:
http://ximbiot.com/cvs/wiki/index.php?title=CVS_Clients

Subversion: http://subversion.tigris.org

Free Book sobre Subversion: http://svnbook.red-bean.com/index.es.html

Apache2 i SSL: http://www.debian-administration.org/articles/349

Apache2 i WebDav:
http://www.debian-administration.org/articles/285

Hi ha una gran quantitat de documentació sobre Apache i SSL + Subversion en Debian a més de http://www.debian-administration.org, posar a Google "Apache2 SSL and Subversion in Debian" per a obtenir alguns documents interessants.