Administració de dades

Remo Suppi Boldrito

PID_00174422

Universitat Oberta de Catalunya
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 necessari utilitzar bases de dades (*databases*, BD). Una base de dades és un conjunt estructurat de dades que pot organitzar de manera simple i eficient un gestor 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 l'administració. Per a 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. En aquest mòdul es veuran els gestors més importants de bases de dades en entorns GNU/Linux, una breu ressenya a SQL, i també diferents maneres de gestionar dades i repositoris dins d'un entorn distribuït i multiusuari.

Objectius

En els materials didàctics d'aquest mòdul trobareu els continguts i les eines procedimentals per a aconseguir els objectius següents:

1. Analitzar les maneres d'emmagatzemar dades de manera massiva i de ferne un ús eficient.
2. Desenvolupar els aspectes essencials de les bases de dades i la seva instal·lació i ús.
3. Treballar amb les tècniques de control de versions i analitzar-ne els avantatges.
4. Instal·lar i analitzar les diferents eines de gestió de dades i la seva integració per a entorns de desenvolupament.

1. Administració de dades

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

Tanmateix, quan es treballa en el desenvolupament d'un programari, hi ha altres aspectes relacionats amb les dades, com 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 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 fa per a controlar les diferents versions del codi font. Això no obstant, els mateixos conceptes són aplicables a altres àmbits i no solament per al codi font, sinó també per als documents, imatges, etc. Tot i que es pot fer manualment un sistema de control de versions, és molt aconsellable disposar d'eines que facilitin aquesta gestió (CVS, Subversion, GIT, SourceSafe, Clear Case, Darcs, Plastic SCM, RCS, etc.).

En aquest mòdul veurem **CVS (Control Version System)**, **Subversion** i **GIT** per a controlar i administrar múltiples revisions d'arxius, i automatitzar l'emmagatzematge, la lectura, la identificació i la 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. Finalment, també s'hi analitzarà una eina per al seguiment d'incidències i errors en l'entorn de desenvolupament o projectes anomenada *Mantis*.

La justificació de CVS i Subversion es pot trobar en el fet que CVS és un dels paquets tradicionals més utilitzats i Subversion (també se'l coneix com a *svn*, perquè aquest és el nom de l'eina d'indicador) és un programa de control de versions dissenyat específicament per a reemplaçar el popular CVS i que soluciona algunes de les deficiències d'aquest. Una característica important de Subversion és que, a diferència de CVS, cadascun dels arxius amb versions no tenen 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 determinat moment.

Com a penúltim punt, i ateses les seves prestacions i la utilització en grans projectes, s'hi descriuen les característiques i la instal·lació d'un altre projecte GPL, anomenat *GIT*. GIT és un programari de control de versions dissenyat per Linus Torvalds, basat en l'eficiència i la fiabilitat del manteniment de versions d'aplicacions quan aquestes tenen un gran nombre d'arxius de codi font. Finalment, per a completar un cercle mínim d'eines per a compartir i gestionar dades, es presenta una descripció breu de **Mantis Bug Tracker**, que és una eina de gestió d'incidències (*bug tracker*) de codi obert. Aquesta aplicació està escrita en PHP i requereix una base de dades i un servidor web. Des del projecte es recomana la utilització de MySQL i Apache, respectivament.

1.1. PostgreSQL

En el llenguatge de bases de dades, PostgreSQL utilitza un model client-servidor [19]. Una sessió de PostgreSQL consisteix en una sèrie de programes que cooperen:

1) Un procés servidor que gestiona els arxius de la BD accepta connexions dels clients i fa les accions sol·licitades per aquests sobre la BD. El programa servidor s'anomena *postmaster* en PostgreSQL.

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

Generalment, el client i el servidor estan en amfitrions (*hosts*) diferents 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 i de manera transparent per a l'usuari. Hi ha un conjunt de tasques que l'usuari o l'administrador poden portar a terme, segons convingui, i que passem a descriure a continuació.

1.1.1. Instal·lació de PostgreSQL

Aquest pas és necessari per als administradors de la BD [19]. Entre les funcions de l'administrador de BD hi ha la instal·lació del servidor, la inicialització i la configuració, l'administració dels usuaris i les tasques de manteniment de la BD. La instal·lació de la base de dades es pot fer de dues maneres: amb 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 BD operativa , o amb el codi font, que serà necessari compilar i instal·lar. En el primer cas (*pre-built binary packages*), es poden utilitzar els gestors de paquets o des de l'indicador d'ordres, per exemple, a Debian l'`apt-get`. Per al segon cas, es recomana anar sempre a l'origen (o a un dipòsit rèplica de la distribució original). És important tenir en compte que després la instal·lació des del codi font quedarà fora

de la BD de programari instal·lat i es perdran els beneficis d'administració de programari que presentin, per exemple, apt-cache o apt-get.

Instal·lació des del codi font pas a pas

Primer cal obtenir el programari del lloc web (`x.x` és la versió disponible) http://www.postgresql.org/download/ i es descomprimeix (`x.x.x` és la versió 9.0.1 en el moment de fer aquesta revisió):

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

Ara cal anar al directori `postgresql` i configurar-lo amb l'ordre `./configure`.

Es compila amb `gmake`, es verifica la compilació amb `gmake check` i s'instal·la amb `gmake install` (per defecte, ho farà a `/usr/local/pgsql`).

Postinstal·lació

Cal inicialitzar les variables en, 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/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
```

Un cop instal·lada la BD, s'haurà de crear un usuari que gestioni 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 utilitzant l'ordre `useradd`. A continuació, s'ha de crear una àrea d'emmagatzematge per a les bases de dades sobre el disc (espai únic) que serà un directori, per exemple `/usr/local/pgsql/data`. Per a això, executeu l'ordre `initdb -D /usr/local/pgsql/data`, connectat com l'usuari creat (postgres). És possible rebre un missatge que diu que no es pot crear el directori per manca de privilegis. Per a això s'ha de crear un directori i després indicar a la BD 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 a això, utilitzeu:

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

Per a executar-lo en mode actiu (foreground) i passiu (background), utilitzeu:

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

Els readreçaments es fan per a emmagatzemar els errors del servidor. El paquet també inclou un *script* (`pg_ctl`) perquè això no cal conèixer tota la sintaxi de *postmaster* per a executar-lo:

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

Es pot avortar l'execució del servidor de diferents maneres: per exemple amb el `pg-ctl` o bé directament amb

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

1.1.2. Com s'ha de crear una BD?

La primera acció per a verificar si es pot accedir al servidor de BD és crear una base de dades. El servidor PostgreSQL pot gestionar moltes BD 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 l'indicador d'ordres del sistema operatiu. Aquesta ordre generarà un missatge `CREATE DATABASE`, si tot és correcte.

És important tenir en compte que, per a dur a terme aquesta acció, cal que hi hagi un usuari habilitat per a crear una base de dades, com hem vist en el subapartat anterior, en què hi ha un usuari que instal·la la base de dades i que tindrà permisos per a crear bases de dades i nous usuaris que, al seu torn, puguin crear bases de dades. Generalment, i en Debian, aquest usuari és postgres per defecte. Per això, abans de fer el `createdb`, s'ha de fer un `su postgres` (si s'és root no cal cap paraula clau, però un altre usuari necessitarà la paraula clau de l'usuari postgres) i, a continuació, es podrà executar el `createdb`. Per a crear una BD anomenada *nteumdb*:

```
createdb nteumdb
```

Si l'execució de l'ordre dóna error, pot ser que no estigui ben configurat el camí o que la BD estigui mal instal·lada. Es pot intentar amb el camí absolut (`/usr/local/pgsql/bin/createdb nteumdb`), que dependrà de la instal·lació que s'hagi fet, o consultar referències per solucionar els proble-

mes. Altres missatges d'error són `could not connect to server`, quan el servidor no ha arrencat, o `CREATE DATABASE: permission denied`, quan no es tenen privilegis per a crear la BD. Per a eliminar la base de dades es pot utilitzar `dropdb nteumdb`.

1.1.3. Com es pot accedir a una BD?

Un cop creada la BD, s'hi pot accedir de diverses maneres:

Nota

Per poder accedir a la BD, el servidor de base de dades haurà d'estar en funcionament. Quan s'instal·la Postgresql, es creen els enllaços adequats perquè el servidor s'inicïi en l'arrencada de l'ordinador. Per a més detalls, consulteu el subapartat d'instal·lació.

1) executant una ordre interactiva anomenada `psql`, que permet editar i executar ordres SQL (per exemple, `psql nteumdb`);

2) executant una interfície gràfica com PhpPGAdmin o alguna *suite* amb suport ODBC per a crear i manipular BD;

3) escrivint una aplicació en algun dels llenguatges suportats, com PHP, Perl o Java, entre d'altres (consulteu *PostgreSQL Programmer's Guide*).

Per simplicitat, utilitzarem `psql` per a accedir a la BD, tot introduint `psql nteumdb`: hi apareixeran uns missatges amb la versió i la 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 té ordres que no són SQL i comencen amb `\`, com per exemple `\h` (enumera totes les ordres disponibles) o `\q` per a acabar.

1.1.4. Usuaris de BD

Els usuaris de la BD són completament diferents dels usuaris del sistema operatiu. En alguns casos podria ser interessant mantenir-hi una correspondència, però no és necessari. Els usuaris són per a totes les BD que controla aquest servidor, no per a cada BD.

Per a crear un usuari es pot executar la sentència SQL `CREATE USER` *`nom`* i per a esborrar-lo, `DROP USER` *`nom`*.

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

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

- **Superusuari**: aquest usuari no té cap restricció. Per exemple, podrà crear nous usuaris: `CREATE USER nom CREATEUSER;`

- **Creador de BD**: té permís per a crear BD. Per a crear un usuari d'aquestes característiques, utilitzeu l'ordre: `CREATE USER nom CREATEDB;`

- **Contrasenya**: només és necessari si per qüestions de seguretat es vol controlar l'accés dels usuaris quan es connectin a una BD. Per a crear un usuari amb contrasenya (`paraula_clau` serà la clau d'aquest usuari): `CREATE USER nom WITH PASSWORD 'paraula_clau';`

Es poden canviar els atributs d'un usuari amb l'ordre `ALTER USER`.

També es poden fer grups d'usuaris que comparteixin els mateixos privilegis amb:

```
CREATE GROUP NomGrup;
```

Per a afegir usuaris en aquest grup:

```
ALTER GROUP NomGrup ADD USER Nom1;
```

Per a esborrar-los:

```
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 crea una BD, els privilegis són per a l'usuari que la crea (i per al *superusuari*). Per a permetre que un altre usuari utilitzi aquesta BD o una part, cal que li concedeixin privilegis. N'hi ha de diferents tipus, com `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `RULE`, `REFERENCES`, `TRIGGER`, `CREATE`, `TEMPORARY`, `USAGE`, `EXECUTE` i `ALL PRIVILEGES` (s'han de consultar les referències per a saber què signifiquen).

Per a assignar els privilegis, es pot utilitzar `GRANT UPDATE ON objecte TO usuari`, en què `usuari` ha de ser un usuari vàlid de PostgerSQL i `objecte`, una taula, per exemple.

El superusuari o l'amo de la taula ha d'executar aquesta ordre. L'usuari PUBLIC es pot utilitzar 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'objcte, es pot executar:

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

1.1.5. Manteniment

Hi ha un conjunt de tasques que són responsabilitat de l'administrador de BD i que s'han de fer periòdicament:

1) Recuperar l'espai: cal executar periòdicament l'ordre `VACUUM`, que recuperarà l'espai de disc de files esborrades o modificades, actualizarà les estadístiques utilitzades pel planificador de PostgreSQL i millorarà les condicions d'accés.

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

3) Canvi d'arxius de registre (*logs*): s'ha d'evitar que els arxius de registre siguin massa grans i difícils de manipular. Això es pot fer fàcilment quan s'inicia el servidor amb `pg_ctl start | logrotate`. `logrotate` canvia el nom i obre un nou arxiu de registre i es pot configurar amb `/etc/logrotate.conf`.

4) Còpia de seguretat i recuperació (*backup* i *recovery*): les dades es poden desar amb la sentència SQL `dump` o bé desant l'arxiu de la BD. En el primer cas, `pg_dump ArxiuDB > ArxiuBackup`. Per a recuperar-la, utilitzeu `psql ArxiuDB < ArxiuBackup`. Per a desar totes las BD del servidor, es pot executar `pg_dumpall > ArxiuBackupTotal`. Una altra estratègia és desar els arxius de les bases de dades en el sistema operatiu, per exemple amb `tar -cf backup.tar /usr/local/pgsql/data`. Hi ha dues restriccions que fan que aquest mètode sigui poc pràctic:

a) el servidor s'ha d'aturar abans de desar i recuperar les dades i

b) cal conèixer molt bé totes les implicacions en relació amb l'arxiu, on són totes les taules, transaccions, etc. Altrament, la BD pot quedar inútil. A més, en general, la mida que es desarà serà més gran que l'obtinguda amb els mètodes anteriors, ja que, per exemple, amb el `pg_dumb` no es desen els índexs, sinó l'ordre per a recrear-los.

1.1.6. 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
```

1.2. El llenguatge SQL

La finalitat d'aquest subapartat no és fer un programa d'aprenentatge sobre SQL, però se n'analitzaran alguns exemples per a veure les capacitats d'aquest llenguatge. Aquests exemples els trobareu a PostgreSQL, en el directori `DirectorioInstalacion/src/tutorial`. Per a accedir-hi, canvieu al directori de PostgreSQL (`cd DirectorioInstalación/src/tutorial`) i executeu `psql -s nteumdb` i després, un cop dins, `ıbasics.sql`. El paràmetre `ı` 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 treballa amb 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. Les taules s'agrupen en una BD i un únic servidor utilitza aquesta col·lecció de BD (tot el conjunt es denomina *agrupació* o *clúster de bases de dades* (*database cluster*). Per crear, per exemple, una taula amb psql, executeu:

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

L'ordre s'acaba amb ';' i s'hi poden usar espais en blanc i tabulacions 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;`
Les dades s'hi poden introduir de dues maneres: s'hi posen totes les dades de la taula o s'hi indiquen les variables i els valors que es volen modificar:
`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 manera pot ser senzilla per a unes quantes dades, però quan se n'ha d'introduir una gran quantitat, es poden copiar des d'un arxiu amb la sentència:
`COPY temps FROM '/home/user/temps.txt';` Aquest arxiu ha d'estar en el servidor, no en el client). Per mirar una taula, podríem fer:
`SELECT * FROM temps;`
on l'`*` significa "totes les columnes".
Exemple: introduir dades a la taula. Dins de psql:
`INSERT INTO NomTB (valorVar1, ValorVar2,...);`
Dades des d'un arxiu. Dins de psql:
`COPY NomTB FROM 'NomArxiu';`
Visualitzar dades. Dins de psql:
`SELECT * FROM NomTB;`
Alguns exemples d'ordres més complexes serien (dins de psql), visualitza la columna ciutat després de fer l'operació:
`SELECT ciutat, (temp_màx+temp_mín)/2 AS temp_mitjana, date FROM temps;`
Visualitza tot allò que compleix l'operació lògica:
`SELECT * FROM temps WHERE ciutat = 'Barcelona' AND pluja > 0.0;`
Unió de taules:
`SELECT * FROM temps, ciutat WHERE ciutat = nom;`
Funcions, màxim en aquest cas:
`SELECT max(temp_mín) FROM temps;`
Funcions imbricades:

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

Modificació selectiva:

```
  UPDATE temps SET temp_màx = temp_màx 2, temp_mín = temp_mín 2 WHERE
dia > '19990128';
```

Esborrament del registre:

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

Altres ordres útils:

Login com a usuari postgres (SuperUser) per a treballar amb la base de dades: `# su - postgres`

Crear bases de dades: `$ createdb nom_BD`

Eliminar base de dades: `$ dropdb nom_BD`

Accedir a la base de dades: `$ psql nom_BD`

Ajuda (dins de la base de dades; per exemple, mynteum: `mynteum=# \h`

Sortir del psql: `mynteum=# \q`

Desar la base de dades: `$ pg_dump mynteum > db.out` Carregar la base de dades desada anteriorment: `$ psql -d mynteum -f db.out`

Per a desar totes les bases de dades: primer, `# su - postgres` i després `$ pg_dumpall > /var/lib/pgsql/backups/dumpall.sql`

Per a restaurar una base de dades: primer, `# su - postgres` i a continuació `$ psql -f /var/lib/pgsql/backups/dumpall.sql mynteum`

Per a mostrar les bases de dades: `psql -l` o sinó, des de dins `mynteum=# \l;`

Mostrar els usuaris: des de dins de l'ordre `psql` executeu `mymydb=# SELECT * FROM "pg_user";`

Mostrar les taules: des de dins de l'ordre `psql` executeu `mynteum=# SELECT * FROM "pg_tables";`

Canviar la contrasenya: des de dins de l'ordre `psql mynteum=# UPDATE pg_shadow SET passwd = 'new_password' where usename = 'username';` o també `ALTER USER nom WITH PASSWORD 'password';`

Netejar totes les bases de dades: `$ vacuumdb - -quiet - -all`

1.2.1. Entorns d'administració gràfics

Hi ha diversos entorns d'administració gràfics, com **Phppgadmin*** i **Pgadmin****. Una altra aplicació que funciona però que ha deixat de ser mantinguda (i, per tant, no és recomanable) és **pgaccess*****. Aquests programes permeten accedir i administrar una base de dades amb una interfície gràfica. Per a la majoria, la manera més fàcil d'accedir-hi és des d'una terminal. L'administrador de la BD (si no és l'usuari postgresql) haurà de fer `xhost +`, la qual cosa permet que altres aplicacions es puguin connectar al *display* de l'usuari actual i, a continuació, executar el nom de l'aplicació. Entre tots aquests entorns, **Phppgadmin** presenta molt bones característiques (s'inclou en la majoria de distribucions) i un cop instal·lat, s'hi accedeix mitjançant `http://localhost/phppgadmin`. Després de configurar el llenguatge podrem seleccionar el servidor, l'usuari i la contrasenya per a la base de dades. És recomanable crear un usuari amb contrasenya per a accedir a la base de dades des de l'usuari postgres del sistema operatiu (per exemple, `su - postgres`) amb l'ordre `psql` i fer:

*http://phppgadmin.sourceforge.net
**http://www.pgadmin.org
***http://sourceforge.net/projects/pgaccess

```
CREATE USER admin WITH Passwd 'poseu_passwd';
```

Per a veure que tot és correcte, es pot fer `SELECT * FROM "pg_user";` i es veurà l'usuari creat amb * en la contrasenya. En Phppgadmin podrem seleccionar a l'esquerra el servidor i configurar els paràmetres, tot indicant que el servidor és *localhost* i l'usuari i la contrasenya, els creats anteriorment. A partir d'aquí veurem les bases de dades, i també veurem tots els paràmetres i les configuracions.

Una altra eina interessant és **Webmin**. És una interfície basada en la web per a administrar sistemes per a Unix que permet configurar usuaris, Apache, DNS, compartir arxius, servidors de bases de dades, etc. L'avantatge de Webmin, sobretot per a usuaris que s'inicien en l'administració, és que elimina la necessitat d'editar manualment els fitxers de configuració i permet administrar un sistema de manera local o remota.

Enllaços d'interès

Webmin no està disponible en algunes distribucions i s'ha de baixar i instal·lar directament del web: http://www.webmin.com. Tota la documentació i la informació dels mòduls de Webmin disponibles es poden trobar a: http://doxfer.webmin.com/Webmin.

1.3. MySQL

MySQL [7] és, segons els seus autors, la base de dades (BD) SQL oberta, és a dir, de programari lliure (*Open Source*) més popular. En l'actualitat és propietat d'Oracle, que fa les inversions i els manteniments i que, a partir de la versió (encara en fase Beta) de MySQL 5.5 (primera versió després de la compra de la companyia Sun), ha insistit que continuarà invertint en aquesta base de dades. En aquesta versió s'ha integrat el motor de persistència InnoDB i s'han afegit millores internes per a incrementar el rendiment fins a un 200%, a més d'aconseguir temps de recuperació deu vegades menors que en la versió anterior. MySQL és un DBMS (*database management system*) i, com a tal, pot afegir i processar les dades emmagatzemades a la BD.

Enllaç d'interès

Tota la documentació sobre MySQL es pot obtenir des de la pàgina web: http://dev.mysql.com/usingmysql/get_started.html.

Igual que PostgreSQL, MySQL és una base de dades relacional, és a dir, que emmagatzema les dades en taules, en lloc d'una única ubicació, la qual cosa permet augmentar la velocitat i la flexibilitat.

Com que és programari lliure, qualsevol pot obtenir-ne el codi, estudiar-lo i modificar-lo d'acord amb les seves necessitats, sense cap pagament, ja que MySQL utilitza llicència GPL. En comparació d'altres BD, MySQL proporciona un conjunt d'estadístiques i prestacions a la seva pàgina web per mostrar a l'usuari que és ràpid, fiable i fàcil d'usar. La decisió de triar una BD s'ha de prendre acuradament en funció de les necessitats dels usuaris i de l'entorn en què s'utilitzarà aquesta BD.

1.3.1. Instal·lació de MySQL

MySQL es pot obtenir des de la pàgina web* del projecte 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` i `common` són necessaris). La instal·lació, després d'unes preguntes, crearà un usuari mysql i una entrada a `/etc/init.d/mysql` per arrencar o aturar el servidor en l'arrencada. També es pot fer manualment:

*http://www.mysql.com

/etc/init.d/mysql start|stop

Per a accedir a la base de dades, es pot utilitzar el monitor mysql des de l'indicador d'ordres. Si obté els binaris (no Debian ni RPM, amb això simplement utilitzeu les utilitats comunes apt-get i rpm), per exemple un arxiu comprimit des del lloc web de MySQL, haurà d'executar les ordres següents per instal·lar la BD:

```
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 BD en el directori `/usr/local/mysql`. En cas que se n'obtingui 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 prestar atenció quan se'n fa la configuració, ja que el directori on s'instal·larà la BD és `prefix=/usr/local/mysql` i es pot canviar per a ubicar la BD en el directori que es vulgui.

1.3.2. Postinstal·lació i verificació de MySQL

Un cop acabada la instal·lació, sigui dels binaris o del codi font, s'haurà de verificar si el servidor funciona correctament. En Debian es pot fer directament (l'usuari és el que s'ha definit durant la instal·lació, igual que la seva contrasenya, indicats en ordres com `-u` i `-p`, respectivament):

```
/etc/init.d/mysql start      Inicia el servidor
mysqladmin -u user -p version      Genera informació de versions
mysqladmin -u user -p variables      Mostra els valors de les variables
mysqladmin -u root -p shutdown      Finalitza l'execució del servidor
mysqlshow -u user -p      Mostra les BD predefinides
mysqlshow mysql -u user -p      Mostra les taules de la BD 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 binarios (RPM, Pkg...), cal fer el següent:

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

L'*script* `mysql_install_db` crea la BD `mysql` i `mysqld_safe` arrenca el servidor `mysqld`. Després 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'han instal·lat els *tests*, es podran executar amb `cd sql-bench` i després amb `run-all-tests`. Els resultats es desaran en el directori `sql-bech/Results` per comparar-los amb altres BD.

1.3.3. El programa monitor (client) mysql

El client `mysql` es pot usar per a crear i utilitzar BD simples, és interactiu i permet connectar-se al servidor, executar cerques i visualitzar els resultats. També funciona en mode *batch* (com un *script*), amb les ordres que li arriben per mitjà d'un arxiu. Per a 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 -u usuari -p [NomDB]
```

Si no es posa l'últim paràmetre, no se selecciona cap BD. Un cop dins, el `mysql` posarà un *indicador* (**mysql**>) i esperarà que li introduïm alguna ordre (pròpia i SQL), per exemple, *help*. A continuació, donarem una sèrie d'ordres per provar el servidor (recordeu posar sempre el ';' per 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 línees.
```
mysql> SHOW DATABASES;
```
Mostra les BD disponibles.
```
mysql> USE test
```
Canvia la BD.
```
mysql> CREATE DATABASE nteum; USE nteum;
```
Crea i selecciona una BD 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` a pet. L'arxiu `pet.txt` ha de tenir un registre per línia separat per tabulacions de les dades, d'acord amb la definició de la taula (data en format AAAA-MM-DD)

```
mysql> INSERT INTO pet
-> VALUES ('Marciano','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 marciano@localhost -> IDENTIFIED
BY 'passwd' WITH GRANT OPTION;
```

Crea un usuari marciano a la BD. Ho ha de fer el root de la BD. També es pot fer directament amb:

```
mysql> INSERT INTO user (Host,User,Password) ->
VALUES('localhost','marciano','passwd');
```

1.3.4. Cas d'ús: processament de dades amb MySQL

Considerarem les dades del Banc Mundial de dades i, concretament, les d'Internet: amplada de banda internacional (bits per persona). Des de l'adreça http://datos.bancomundial.org/indicador podem descarregar un full de dades que tindrà una informació com (la informació mostra el parcial d'un total de 196 països i des de l'any 1960):

País ... 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
...
Espanya 297,14 623,61 1126,83 1938,19 2821,65 2775,71 6058,60 11008,05...

Es crearà una base de dades en MySQL per a importar aquestes dades i generar les llistes que calculin la mitjana per país i la mitjana anual, i ordenin de major a menor el volum per a l'any 2008; i una llista dels deu països on més s'utilitza Internet per any. A partir d'aquestes últimes dades, s'ha de fer una classificació dels cinc països on més s'utilitza Internet des que hi ha dades (s'hi mostren les ordres més rellevants i no per a totes les dades).

```
mysql -u root -p
Welcome to the MySQL monitor. Commands end with ; or \g.
Server version: 5.0.51a24+ lenny3 (Debian)
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> create database bancmundial;

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
```

```
| information_schema |
| bancmundial       |
| mysql              |
+--------------------+
3 rows in set (0.00 sec)

mysql> use bancmundial;
Database changed
mysql> create table paisbai(pais varchar(100), 1960
varchar(100), 1961 varchar(100), 1962 varchar(100), 1963
varchar(100), 1964 varchar(100), 1965 varchar(100), 1966
varchar(100), 1967 varchar(100), 1968 varchar(100), 1969
varchar(100), 1970 varchar(100), 1971 varchar(100), 1972
varchar(100), 1973 varchar(100), 1974 varchar(100), 1975
varchar(100), 1976 varchar(100), 1977 varchar(100), 1978
varchar(100), 1979 varchar(100), 1980 varchar(100), 1981
varchar(100), 1982 varchar(100), 1983 varchar(100), 1984
varchar(100), 1985 varchar(100), 1986 varchar(100), 1987
varchar(100), 1988 varchar(100), 1989 varchar(100), 1990
varchar(100), 1991 varchar(100), 1992 varchar(100), 1993
varchar(100), 1994 varchar(100), 1995 varchar(100), 1996
varchar(100), 1997 varchar(100), 1998 varchar(100), 1999
varchar(100), 2000 varchar(100), 2001 varchar(100), 2002
varchar(100), 2003 varchar(100), 2004 varchar(100), 2005
varchar(100), 2006 varchar(100), 2007 varchar(100), 2008
varchar(100), 2009 varchar(100), 2010 varchar(100) );

mysql>
CREATE TABLE ‘bancmundial‘.‘paisbai‘ (‘pais‘ VARCHAR(100) NOT
NULL, ‘1960‘ VARCHAR(100) NOT NULL, ‘1961‘ VARCHAR(100) NOT
NULL, ‘1962‘ VARCHAR(100) NOT NULL, ‘1963‘ VARCHAR(100) NOT
NULL, ‘1964‘ VARCHAR(100) NOT NULL, ‘1965‘ VARCHAR(100) NOT
NULL, ‘1966‘ VARCHAR(100) NOT NULL, ‘1967‘ VARCHAR(100) NOT
NULL, ‘1968‘ VARCHAR(100) NOT NULL, ‘1969‘ VARCHAR(100) NOT
NULL, ‘1970‘ VARCHAR(100) NOT NULL, ‘1971‘ VARCHAR(100) NOT
NULL, ‘1972‘ VARCHAR(100) NOT NULL, ‘1973‘ VARCHAR(100) NOT
NULL, ‘1974‘ VARCHAR(100) NOT NULL, ‘1975‘ VARCHAR(100) NOT
NULL, ‘1976‘ VARCHAR(100) NOT NULL, ‘1977‘ VARCHAR(100) NOT
NULL, ‘1978‘ VARCHAR(100) NOT NULL, ‘1979‘ VARCHAR(100) NOT
NULL, ‘1980‘ VARCHAR(100) NOT NULL, ‘1981‘ VARCHAR(100) NOT
NULL, ‘1982‘ VARCHAR(100) NOT NULL, ‘1983‘ VARCHAR(100) NOT
NULL, ‘1984‘ VARCHAR(100) NOT NULL, ‘1985‘ VARCHAR(100) NOT
NULL, ‘1986‘ VARCHAR(100) NOT NULL, ‘1987‘ VARCHAR(100) NOT
NULL, ‘1988‘ VARCHAR(100) NOT NULL, ‘1989‘ VARCHAR(100) NOT
NULL, ‘1990‘ VARCHAR(100) NOT NULL, ‘[...]
mysql> describe paisbai;
+-------------------------------------+
|Field | Type         |Null |K| Def. |E|
| país | varchar(100) | YES | | NULL | |
| 1960 | varchar(100) | YES | | NULL | |
| 1961 | varchar(100) | YES | | NULL | |
| 1962 | varchar(100) | YES | | NULL | |
| 1963 | varchar(100) | YES | | NULL | |
| 1964 | varchar(100) | YES | | NULL | |
| 1965 | varchar(100) | YES | | NULL | |
...
| 2009 | varchar(100) | YES | | NULL | |
| 2010 | varchar(100) | YES | | NULL | |
+-------------------------------------+

1. Per a carregar les dades, es pot entrar a http://localhost/phpmyadmin i
importar les dades des d’un fitxer a la base de dades bancmundial o des de la línia del mysql.
LOAD DATA LOCAL INFILE ’/tmp/phpO5Dhpl’ REPLACE INTO TABLE ‘paisbai‘
FIELDS TERMINATED BY ’;’
ENCLOSED BY ’"’
ESCAPED BY ’\\’
LINES TERMINATED BY ’\n’;

2. Llista que permet calcular la mitjana per país.
consulta SQL: SELECT ‘pais‘,
(‘1960‘+‘1961‘+‘1962‘+‘1963‘+‘1964‘+‘1965‘+‘1966‘+‘1967‘+‘1968‘+
‘1969‘+‘1970‘+‘1971‘+‘1972‘+‘1973‘+‘1974‘+‘1975‘+‘1976‘+‘1977‘+‘
```

```
1978‘+‘1979‘+‘1980‘+‘1981‘+‘1982‘+‘1983‘+‘1984‘+‘1985‘+‘1986‘+‘1
987‘+‘1988‘+‘1989‘+‘1990‘+‘1991‘+‘1992‘+‘1993‘+‘1994‘+‘1995‘+‘19
96‘+‘1997‘+‘1998‘+‘1999‘+‘2000‘+‘2001‘+‘2002‘+‘2003‘+‘2004‘+‘200
5‘+‘2006‘+‘2007‘+‘2008‘+‘2009‘+‘2010‘)/51 as mitjanapais FROM
‘paisbai‘ LIMIT 0, 30 ;
Files: 30
país        mitjanapais
Afganistan  0.0203921568627
Albània     7.3696078431373
Argèlia     0.4425490196078
```

3. Generar una llista que visualitzi la mitjana anual.

```
consulta SQL: SELECT AVG (‘1989‘) AS ‘1989‘, AVG(‘1990‘) as
‘1990‘, AVG(‘1991‘) as ‘1991‘, AVG(‘1992‘) as ‘1992‘,
AVG(‘1993‘) as ‘1993‘, AVG(‘1994‘) as ‘1994‘, AVG(‘1995‘) as
‘1995‘, AVG(‘1996‘) as ‘1996‘, AVG(‘1997‘) as ‘1997‘,
AVG(‘1998‘) as ‘1998‘, AVG(‘1999‘) as ‘1999‘, AVG(‘2000‘) as
‘2000‘, AVG(‘2001‘) as ‘2001‘, AVG(‘2002‘) as ‘2002‘,
AVG(‘2003‘) as ‘2003‘, AVG(‘2004‘) as ‘2004‘, AVG(‘2005‘) as
‘2005‘, AVG(‘2006‘) as ‘2006‘, AVG(‘2007‘) as ‘2007‘,AVG(‘2008‘)
as ‘2008‘, AVG(‘2009‘) as ‘2009‘, AVG(‘2010‘) as ‘2010‘ FROM
‘paisbai‘;
Files: 1
1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000
0.0001 0.000 0.000 0.001 0.0019 0.005 0.044 0.158 0.6547 1.3870 27.483 126.9760 387.70
```

3. Generar una llista que visualitzi l’ordre de més a menys volum per a l’any 2008.

```
SELECT ‘pais‘, ‘2008‘ FROM ‘paisbai‘ ORDER BY ‘2008‘ DESC LIMIT 0 , 30;

+--------------------+
| país     | 2008    |
| Lituània | 9751.01 |
| Mongòlia | 946.53  |
| Romania  | 9110.51 |
| Zimbabwe | 9.71    |
...
| Bhutan   | 65.52   |
| Hongria  | 5977.17 |
| Vietnam  | 580.72  |
+--------------------+
```

4. Generar una llista dels deu països on més s’utilitza Internet per any.

Alternativa 1.

```
SELECT ‘pais‘ , SUM( ‘1989‘ ) AS ‘1989‘ , SUM( ‘1990‘ ) AS
‘1990‘ , SUM( ‘1991‘ ) AS ‘1991‘ , SUM( ‘1992‘ ) AS ‘1992‘ ,
SUM( ‘1993‘ ) AS ‘1993‘ , SUM( ‘1994‘ ) AS ‘1994‘ ,
SUM( ‘1995‘ ) AS ‘1995‘ , SUM( ‘1996‘ ) AS ‘1996‘ ,
SUM( ‘1997‘ ) AS ‘1997‘ , SUM( ‘1998‘ ) AS ‘1998‘ ,
SUM( ‘1999‘ ) AS ‘1999‘ , SUM( ‘2000‘ ) AS ‘2000‘ ,
SUM( ‘2001‘ ) AS ‘2001‘ , SUM( ‘2002‘ ) AS ‘2002‘ ,
SUM( ‘2003‘ ) AS ‘2003‘ , SUM( ‘2004‘ ) AS ‘2004‘ ,
SUM( ‘2005‘ ) AS ‘2005‘ , SUM( ‘2006‘ ) AS ‘2006‘ ,
SUM( ‘2007‘ ) AS ‘2007‘ , SUM( ‘2008‘ ) AS ‘2008‘ ,
SUM( ‘2009‘ ) AS ‘2009‘ , SUM( ‘2010‘ ) AS ‘2010‘
FROM ‘paisbai‘
ORDER BY ‘pais‘ DESC
LIMIT 0 , 10;
```

Alternativa 2:

```
SELECT ‘pais‘ , MAX( ‘1989‘ ) AS ‘1989‘ , MAX( ‘1990‘ ) AS
‘1990‘ , MAX( ‘1991‘ ) AS ‘1991‘ , MAX( ‘1992‘ ) AS ‘1992‘ ,
MAX( ‘1993‘ ) AS ‘1993‘ , MAX( ‘1994‘ ) AS ‘1994‘ ,
MAX( ‘1995‘ ) AS ‘1995‘ , MAX( ‘1996‘ ) AS ‘1996‘ ,
MAX( ‘1997‘ ) AS ‘1997‘ , MAX( ‘1998‘ ) AS ‘1998‘ ,
MAX( ‘1999‘ ) AS ‘1999‘ , MAX( ‘2000‘ ) AS ‘2000‘ ,
MAX( ‘2001‘ ) AS ‘2001‘ , MAX( ‘2002‘ ) AS ‘2002‘ ,
MAX( ‘2003‘ ) AS ‘2003‘ , MAX( ‘2004‘ ) AS ‘2004‘ ,
MAX( ‘2005‘ ) AS ‘2005‘ , MAX( ‘2006‘ ) AS ‘2006‘ ,
MAX( ‘2007‘ ) AS ‘2007‘ , MAX( ‘2008‘ ) AS ‘2008‘ ,
MAX( ‘2009‘ ) AS ‘2009‘ , MAX( ‘2010‘ ) AS ‘2010‘
FROM ‘paisbai‘
GROUP BY ‘pais‘
```

```
LIMIT 0 , 10;
+----------------+------------------+
| país           | mitjana          |
+----------------+------------------+
| Luxemburg      |    284474.679628 |
| Hong Kong      | 21468.6420499333 |
| San Marino     | 5464.22342423529 |
| Països Baixos  | 3949.18559792941 |
| Dinamarca      |     3743.7899214 |
| Estònia        | 3118.98744675686 |
| Suècia         | 2967.78367829608 |
| Suïssa         | 1897.13803142745 |
| Bèlgica        | 1781.95881669216 |
+----------------+------------------+
```

1.3.5. Administració de MySQL

MySQL disposa d'un arxiu de configuració en `/etc/mysql/my.cnf` (en Debian), al qual es poden canviar les opcions per defecte de la BD, com per exemple el port de connexió, l'usuari, la contrasenya dels usuaris remots, els arxius de registre (*logs*), els arxius de dades, si accepta connexions externes, etc. Pel que fa a la seguretat, heu de prendre algunes precaucions:

1) No doneu a ningú (tret de l'usuari root de Mysql) accés a la taula `user` dins de la BD mysql, ja que aquí hi ha les contrasenyes dels usuaris, les quals es podrien utilitzar amb altres finalitats.

2) Verifiqueu `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) Comproveu la documentació* respecte a les condicions de seguretat i de l'entorn de xarxa per a evitar problemes d'atacs o intrusions.

***http://dev.mysql.com /usingmysql/get_started.html**

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 amb extensió FRM, MYD i MYI amb el servidor aturat. Per a recuperar, es pot executar l'ordre `REPAIR TABLE` o `myisamchk -r`, la qual cosa funcionarà la majoria de les vegades. En cas contrari, es podrien copiar els arxius desats i arrencar el servidor. Hi ha altres mètodes alternatius en funció del que es vulgui recuperar, com la possibilitat de desar o recuperar part de la BD (consulteu la documentació).

1.3.6. Interfícies gràfiques

Per a Mysql hi ha gran quantitat d'interfícies gràfiques, fins i tot pròpies del paquet MySQL. Una d'aquestes és **Mysql Administrator**, que és una aplicació potent per a l'administració i el control de bases de dades basades en MySQL. Aquesta aplicació integra la gestió, el control i el manteniment de manera simple i en un mateix entorn de la BD. Les característiques principals són una administració avançada de BD grans, la reducció d'errors per mitjà d'una "administració visual", més productivitat i un entorn segur de gestió. En el lloc original*, aquestes aplicacions ara formen part del paquet MySQL Workbench, que proveeix eines per a modelitzar i dissenyar la base de dades, desenvolupar SQL (equivalent al MySQL Browser) i administrar la base de dades (equivalent a MySQL Administrator).

Enllaç d'interès

Es pot trobar tota la documentació per a la instal·lació i posada en marxa de Mysql Administrator a http://dev.mysql.com/doc/administrator/en/index.html.

*http://www.mysql.com/downloads/workbench

Com a eina d'administració, també recomanem **phpMyAdmin** (inclosa en la majoria de les distribucions), que és una eina de programari lliure escrit en PHP per a gestionar l'administració de MySQL per mitjà de la xarxa. PhpMyAdmin és compatible amb un ampli conjunt d'operacions en MySQL, com per exemple la gestió de bases de dades, taules, camps, relacions, índexs, usuaris, permisos, etc., i també té la capacitat d'executar directament qualsevol sentència SQL. La seva instal·lació es pot fer des de la gestió de programes de distribució de treball (apt/rpm, yum/Synaptic, etc.) que durant el procés d'installació demanarà amb quins servidors de WWW es vol fer la integració. Un cop instal·lada es pot arrencar amb `http://localhost/phpmyadmin`, que demanarà l'usuari i la contrasenya del servidor (indicats en els passos anteriors).

Hi ha altres eines que permeten fer tasques similars, com **Webmin**, que permet gestionar i administrar bases de dades MySQL (incloent-hi el mòdul corresponent). Si bé aquest paquet ja no s'inclou amb algunes distribucions, es pot descarregar des d'http://www.webmin.com. Durant la instal·lació, el Webmin avisarà que l'usuari principal serà el root i utilitzarà la mateixa contrasenya que el root del sistema operatiu. Serà possible connectar-se, per exemple, des d'un navegador `https://localhost:10000`, el qual demanarà acceptar (o denegar) la utilització del certificat per a la comunicació SSL i, a continuació, mostrarà tots els serveis que pot administrar, entre aquests Mysql Data Base Server.

1.4. 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 qui, quan i perquè es van fer els canvis.

A diferència d'altres sistemes, CVS no treballa amb un fitxer o directori cada vegada, sinó que actua sobre col·leccions jeràrquiques dels directoris que controla. El CVS té per objectiu ajudar a gestionar versions de programari i controla l'edició concurrent d'arxius font feta per múltiples autors. El CVS utilitza internament un altre paquet, anomenat **RCS** (Revision Control System), com una capa de baix nivell. Si bé l'RCS es pot utilitzar independentment, això no s'aconsella, ja que el CVS, a més de la seva pròpia funcionalitat, presenta totes les prestacions de l'RCS, però amb millores notables pel que fa a l'estabilitat, el funcionament i el manteniment. Entre aquestes millores cal destacar el funcionament descentralitzat (cada usuari pot tenir el seu arbre de codi), l'edició concurrent, el comportament adaptable mitjançant *shell scripts*, etc. [2, 22, 12].

Com ja s'ha explicat en la introducció, **Subversion*** és un programari de sistema de control de versions dissenyat específicament per a reemplaçar el popular CVS i ampliar les seves capacitats. És programari lliure amb una llicència de tipus Apache/BSD i se'l coneix també com **svn** pel nom de l'indicador d'ordres.

*http://subversion.apache.org

Una característica important de Subversion és que, a diferència de CVS, cadascun dels arxius versionats no té un número de revisió independent, sinó que 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:

1) se segueix la història dels arxius i directoris per mitjà de còpies i reanomenats;

2) les modificacions atòmiques i segures (inclosos en canvis en diversos arxius);

3) la creació de branques i etiquetes és eficient i simple;

4) s'envien només les diferències en ambdues direccions (en CVS sempre s'envien fitxers complets al servidor);

5) pot ser servit, mitjançant Apache, sobre WebDAV/DeltaV;

6) gestiona eficientment arxius binaris (a diferència de CVS, que els tracta internament com si fossin de text).

Lectura recomanada

Hi ha un llibre interessant (de lliure distribució) que explica tot el que fa referència a Subversion. Està disponible a http://svnbook.red-bean.com/index.es.html i la seva traducció està força avançada (http://svnbook.red-bean.com/nightly/es/index.html).

1.4.1. Revision Control System (RCS)

Com que CVS es basa en RCS i en alguns sistemes encara s'utilitza, en donarem algunes explicacions. L'RCS està 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 de RCS), `ident` (busca en l'RCS els arxius per paraules clau o atributs), `rcsclean` (neteja arxius no utilitzats o que no han canviat), `rcsdiff` (executa l'ordre `diff` per comparar versions), `rcsmerge` (uneix dues branques [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 vers el directori de treball amb l'ordre `co` (obté una revisió de cada arxiu RCS i la posa en 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, etc.). Generalment, els arxius són en el directori `./RCS` i cal que el sistema operatiu tingui instal·lades les ordres `diff` i `diff3` perquè funcioni adequadament. Amb Debian no cal 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 incloure els originals en el dipòsit amb: `ci nom_arxius_fonts`. Es pot utilitzar l'`*` i es recomana tenir sempre una còpia de seguretat per a evitar problemes. Això crearà les versions dels arxius amb nom `./RCS/nom_arxiu` i demanarà un text per descriure l'arxiu. A continuació, amb `co RCS/nom_arxiu`, obtindrem una còpia de treball des del dipòsit. Per a evitar modificacions, aquest arxiu es pot bloquejar o desbloquejar amb: `rcs -L nom_arxiu_de_treball` i `rcs -U nom_arxiu_de_treball`, respectivament. Finalment, amb `rlog nom_del_fitxer` podrem veure la informació de les diferents versions [12].

1.4.2. 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 l'RCS i que haurem d'instal·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/profile` (o en `.bashrc` o `.profile`):

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

Òbviament, els usuaris poden modificar aquestes definicions utilitzant /.bashrc i s'ha de crear el directori on hi haurà el dipòsit i configurar 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 arxiu de codi:

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

`cvs init` tindrà en compte no sobreescriure mai un dipòsit ja creat per a evitar la pèrdua d'altres dipòsits. Després s'hauran afegir al grup cvs els usuaris que treballaran amb el CVS. Per exemple, per a afegir l'usuari nteum:

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

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

```
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, encara queo també pot ser `usuari/projecte/xxxx` si és que l'usuari vol tenir organitzats els seus dipòsits. Això crearà un arbre de directoris a `CVSROOT` amb aquesta estructura i afegeix un directori (`/usr/local/cvsroot/NomDelDipòsit`) al dipòsit amb els arxius que a partir d'aquest moment seran en el dipòsit. Una prova per a saber si tot s'ha emmagatzemat correctament és emmagatzemar una còpia en el dipòsit, crear després una còpia des d'allà i comprovar les diferències. Per exemple, 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` *Canviar al directori del codi font original.*
`cvs import -m "Fonts originals" primer_cvs/proj usuarioX vers0`
Crea el repositorio en `primer_cvs/proj` con `usuarioX` y `vers0`.
`cd..` *Canviar al directori superior de dir_org.*
`cvs checkout primer_cvs/proj`
Genera una còpia del dipòsit. La variable `CVSROOT` ha d'estar inicialitzada; altrament, caldrà indicar tot el camí.
`diff -r dir_org primer_cvs/proj`
Mostra les diferències entre l'un i l'altre. No n'hi ha d'haver cap, excepte pel que fa al directori `primer_cvs/proj/CVS` que ha creat el CVS.
`rm -r dir_org`
Esborra els originals (feu sempre una còpia de seguretat i així tindreu una referència d'on es va començar el treball amb el CVS).

```
Estructura dels directoris, arxius i branques:
/home/nteum/primer_cvs/proj: a.c b.c c.c Makefile
            usuarioX, vers0.x  -> Treballar només amb aquest directori després del'\emph{import}
/home/nteum/dir_org/: a.c b.c c.c Makefile              Després del \emph{checkout}, s'haurà d'esborrar

/usr/local/cvsroot/primer_cvs/proj/: a.c b.c c.c Makefile usuarioX, vers0.1
/usr/local/cvsroot/primer_cvs/proj/: a.c b.c c.c Makefile usuarioX, vers0.1.1  -> Branch 1.1

/usr/local/cvsroot/primer_cvs/proj/: a.c b.c c.c Makefile usuarioX, vers0.2
/usr/local/cvsroot/primer_cvs/proj/: a.c b.c c.c Makefile usuarioX, vers0.x
```

El fet d'esborrar els originals no sempre és una bona idea, excepte en aquest cas, després que s'hagi verificat que són en el dipòsit, perquè no s'hi treballi a sobre per distracció i perquè els canvis no es reflecteixin sobre el CVS. En

màquines en les quals 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",
```

on `user` és el *login* de l'usuari i `cvs.server.com` el nom del servidor en el qual hi ha CVS. CVS ofereix diverses ordres (es criden amb `cvs comd opcions`...) per treballar amb el sistema de revisions, entre les quals `checkout`, `update`, `add`, `remove`, `commit` i `diff`.

Ordres de CVS

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

`cvs update` s'ha d'executar de l'arbre privat quan cal actualitzar-ne les 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 afegir nous fitxers en el seu directori de treball sobre un mòdul on ja s'ha fet prèviament un *checkout*. Aquests arxius s'enviaran al dipòsit CVS quan s'executi l'ordre `cvs commit`.

`cvs import` es pot usar per a introduir arxius nous en el dipòsit.

`cvs remove file` 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 el dipòsit i el directori de treball o entre dues versions.

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

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

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 aquest ja s'ha 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 en la nostra còpia privada amb:

```
cvs checkout gccrep
```

Això crearà un nou directori anomenat `gccrep` amb els arxius font. Si s'executa `cd gccrep; ls`, es veurà per exemple `CVS makefile a.c.b.c.c.c`, on 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 en l'arxiu (consulteu 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 a `a.c` (o en els arxius que sigui necessari), és el moment de fer una nova versió i emmagatzemar `a.c` (o tots els que s'han tocat) en el dipòsit i fer que aquesta versió estigui disponible per a la resta dels usuaris:

```
cvs commit a.c
```

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

Si es decideix eliminar els arxius (perquè ja es va acabar el projecte o perquè no s'hi treballarà més), una manera de fer-ho és en el sistema operatiu (`rm -r gccrep`), però és millor utilitzar el mateix cvs fora del directori de treball (nivell immediatament superior): `cvs release -d gccrep`. L'ordre detectarà si hi ha algun arxiu que no s'ha enviat al dipòsit i, si n'hi ha i s'esborra, preguntarà si es vol continuar o no, per a evitar que es perdin tots els canvis. Per a mirar les diferències, per exemple, si s'ha modificat `b.c` i no es recorda quins canvis es van fer, es pot utilitzar `cvs diff b.c` dins el directori de treball. Aquest utilitzarà l'ordre del sistema operatiu `diff` per a comparar la versió `b.c` amb la versió que hi ha en el dipòsit (cal recordar-se de fer un `cvs commit b.c` si es vol que aquestes diferències es transfereixin al dipòsit com una nova versió).

Múltiples usuaris

Quan més d'una persona treballa en un projecte de programari amb diferents revisions, es tracta d'una situació summament complicada perquè de vegades més d'un usuari voldrà editar el mateix fitxer simultàniament. Una possible solució és bloquejar el fitxer o utilitzar punts de verificació reservats (*reserved checkouts*), amb la qual cosa només un usuari podrà editar un fitxer en un moment determinat. Per a 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 als usuaris editar simultàniament un fitxer del seu directori de treball. El primer que transfereixi els seus canvis al dipòsit ho podrà fer sense problemes, però la resta 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 aquest darrer amb els seus propis canvis. Consulteu les referències per veure un exemple d'aplicació i altres maneres de treball concurrent amb comunicació entre usuaris [22].

Interfícies gràfiques

Disposem d'un conjunt d'interfícies gràfiques, com tkcvs* [21], desenvolupada en Tcl/Tk i que suporta Subversion o Cervisia, també molt popular [3]. En la wiki de CVS** també hi ha un conjunt de clients, *plugins* per a CVS.

***http://www.twobarleycorns.net/tkcvs.html**
****http://ximbiot.com/cvs/wiki/CVS%20Clients**

1.4.3. 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 i que és comú utilitzar una base de dades. El dipòsit és com un directori del qual es vol recuperar un fitxer de fa una setmana o deu mesos a partir de l'estat de la base de dades, recuperar les últimes versions i afegir-hi una de nova. A diferència de CVS, Subversion fa les revisions globals del dipòsit, la qual cosa significa que un canvi en el fitxer no genera un salt de versió únicament en aquest fitxer, sinó en tot el dipòsit, que en suma un a la revisió. En Debian haurem de fer `apt-get install subversion`, si volem publicar els dipòsits en `apache2 apt-get install apache2-common` i el mòdul específic `apt-get install libapache2-subversion`.

Enllaç d'interès

A més del llibre disponible a http://svnbook.red-bean.com/nightly/es/index.html, consulteu la documentació a http://subversion.tigris.org/servlets/ProjectDocumentList.

Primer pas: cal crear el nostre dipòsit, usuari (considerem que l'usuari és *svuser*) i grup (*svgroup*). Com a root, feu:

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

Afegeixo 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 crearà diversos arxius i directoris per a controlar i gestionar les versions. Si no es té permís a /usr/local/svn, es pot fer en el directori local:

```
mkdir -p $HOME/svndir
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, tot fent dins del directori:

```
svn import file:///home/svuser/svndir/proves -m "Vegeu. Inicial"
```

Si l'hem creat a `/usr/local/svn/proves`, hauríem d'indicar el camí complet després de l'arxiu. L'`import` copia l'arbre de directoris i `-m` permet indicar-li el missatge de versió. Si no hi posem `-m`, s'obrirà un editor per a fer-ho (cal posar-hi un missatge per a evitar

problemes). El subdirectori `$HOME/svntmp/proves` és una còpia del treball en el dipòsit i és recomanable esborrar-la per a no tenir la temptació o cometre l'error de treballar-hi en comptes de fer-ho amb el dipòsit (`rm -rf $HOME/svntmp/proves`).
Una vegada en el dipòsit, es pot obtenir la còpia local en la qual podrem treballar i després pujar les còpies al dipòsit. Per a això fem:

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

on veurem que tenim el directori proves. Es pot copiar amb un altre nom i agregar 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 en la còpia local (`svn-work`), no cal indicar el camí (*path*). `svn add` marca per a afegir el fitxer al dipòsit i realment s'hi afegeix quan fem un `cvn commit`.

Ens donarà alguns missatges i ens indicarà que és la segona versió. Si hi afegim 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 el del dipòsit. Per a fer-ho, podem afegir-hi una tercera línia a `file1.txt` amb `echo date >> file1.txt` sense pujar-lo i, si volem veure les diferències, podem fer: `svn diff`. Aquesta ordre ens marcarà quines són les diferències entre l'arxiu local i els del dipòsit. Si el carreguem amb `svn commit -m "Nova línia2"` (que generarà una altra versió), després l'`svn diff` no ens donarà diferències.

També es pot utilitzar l'ordre `svn update` en el directori per a actualitzar la còpia local. Si hi ha dos o més usuaris treballant al mateix temps i cadascun ha fet una còpia local del dipòsit i la modificació (amb un `commit`), quan el segon usuari faci el `commit` de la seva còpia amb les seves modificacions, es produirà un error de conflicte, ja que la còpia en el dipòsit és posterior a la còpia original d'aquest usuari (és a dir, hi ha hagut canvis entremig), de manera que si el segon usuari fa el `commit` perdríem les modificacions del primer. Per això haurem de fer un `svn update` que ens indicarà l'arxiu en conflicte i en aquest ens indicarà quin és l'arxiu i les parts que estan en conflicte. L'usuari haurà de decidir amb quina versió es queda i després podrà fer un `commit`. Una ordre interessant és `svn log file1.txt`, que ens mostrarà tots els canvis fets en el fitxer i les versions corresponents.

Enllaç d'interès

Consulteu els clients per accedir a Subversion a: http://svnbook.red-bean.com.

Un aspecte interessant és que Subversion pot funcionar juntament amb Apache2 (i també sobre SSL) per a accedir des d'una altra màquina o simplement mirar el dipòsit. La configuració d'Apache2 i SSL es va indicar a la part de servidors, però també s'explica a Debian Administration. Per a configurar-los, cal activar els mòduls de WebDAV.

Enllaços d'interès

Sobre l'activació dels mòduls de WebDAV podeu consultar l'adreça 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 Creo el dipòsit
  ls -s /subversion
-rw-r-r-   1 www-data www-data   376 Sep 11 20:27 README.txt
drwxr-xr-x   2 www-data www-data   4096 Sep 11 20:27 conf
drwxr-xr-x   2 www-data www-data   4096 Sep 11 20:27 dav
drwxr-xr-x   2 www-data www-data   4096 Sep 11 20:28 db
-rw-r-r-   1 www-data www-data   2 Sep 11 20:27 format
drwxr-xr-x   2 www-data www-data   4096 Sep 11 20:27 hooks
drwxr-xr-x   2 www-data www-data   4096 Sep 11 20:27 locks
```

Per a l'autentificació s'utilitza `htpasswd` (per exemple, amb l'ordre `htpasswd2 -c -m /subversion/.dav_svn.passwd user`) que al nostre exemple és `www-data`. La *-c* només és la primera vegada que executem l'ordre per crear l'arxiu. Això indica que per a accedir a aquest directori es necessita la contrasenya (que és la que hem entrat per a user). A continuació s'ha de canviar l'*httpd.conf* per alguna cosa com:

```
<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 preparats per a importar alguns arxius, per exemple:

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

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

1.4.4. Git

Git és un programa de control de versions dissenyat per Linus Torvalds, basat en l'eficiència i la fiabilitat del manteniment de versions d'aplicacions amb un nombre alt de fitxers de codi font que, si bé es va dissenyar com un motor de baix nivell sobre el qual es poguessin escriure aplicacions d'usuari (*front ends*), s'ha convertit des de llavors en un sistema de control de versions amb funcionalitat plena a partir de la seva adopció com a eina de revisió de versions per al grup de programació del nucli Linux. Entre les característiques més rellevants hi ha:

1) Suporta el desenvolupament no lineal i permet la gestió eficient de ramificacions i barreja de diferents versions. Git inclou eines específiques per a navegar i visualitzar un historial de desenvolupament no lineal.

2) Gestió distribuïda: permet a cada programador tenir una còpia local de l'historial del desenvolupament sencer i que els canvis es propaguin entre els dipòsits locals.

3) Els dipòsits es poden publicar amb HTTP, FTP, rsync o mitjançant un protocol nadiu, sigui per mitjà d'una connexió TCP/IP simple o mitjançant el xifrat SSH, i permet emular servidors CVS per a interactuar amb clients CVS preexistents.

4) Els dipòsits Subversion i svk es poden usar directament amb git-svn.

Instal·lació d'un dipòsit públic de Git sobre Debian en dotze passos

1) Instal·lem GIT: amb l'apt-get (o també amb Synaptic), instal·lem els paquets `git-arch`, `git-core` i `gitweb` (interfície web per a git). Després, creem el directori `/var/cache/git` (el dipòsit git) i també `/var/www/git` per a la interfície web: `mkdir /var/www/git` i `mkdir /var/cache/git` (verifiqueu que no existeix). Hi ha un altre paquet anomenat `git`, però no és necessari [10].

2) Configurem el nom i el correu:

```
git config -global user.name "Remo"
git config -global user.email remo@debian.nteum.org
```

Ho podem verificar amb `git config -l`

```
user.name=Remo
user.email=remo@debian.nteum.org
```

3) Preparem un projecte, creem el directori en el nostre $HOME amb l'ordre `mkdir -p projects/hello` i hi entrem `cd projects/hello`. Aquí creem un arxiu, per exemple, `gedit hello.c` amb el contingut següent (aquest serà el nostre dipòsit inicial):

```
#include <stdio.h>
int main (void)
{printf ("Hola UOC!\n");}
```

4) Creem el nostre primer dipòsit GIT:

```
remo@debian:~$ cd projects/hello
remo@debian:~/projects/hello$ git init
  Initialized empty Git repository in /home/remo/projects/hello/.git/
remo@debian:~/projects/hello$ ls .git/
  branches config description HEAD hooks info objects refs
```

Ara hauríem de descriure el nostre projecte. Així doncs, fem `cd .git` i modifiquem l'arxiu `description` perquè contingui una línia com:

```
My first GIT project - Hello World.
```

i esborrem l'existent.

5) Afegim l'arxiu al nostre projecte:

```
remo@debian:~/projects/hello$ git add hello.c
remo@debian:~/projects/hello$ git status
  # On branch master
  # Initial commit
  # Changes to be committed:
  # (use "git rm -cached <file>..." to unstage)
  # new file: hello.c
```

6) Fem el *commit* inicial (no oblideu que la `-m` descriu el nostre projecte i serà la informació d'allò que s'ha fet):

```
remo@debian:~/projects/hello$ git commit -a -m "Initial Commit"
  Created initial commit e833201: Initial Commit
   1 files changed, 5 insertions(+), 0 deletions(-)
   create mode 100644 hello.c
       remo@debian:~/projects/hello$ git status
  # On branch master
  nothing to commit (working directory clean)
```

Podem mirar el registre amb: **git log**

```
commit e8332018c780c396ee442674e420046ccb397f3b
Author: remo <remo@debian.(none)>
Date: Sun Nov 28 04:38:01 2010 -0500
 Initial Commit
```

7) Afegim més arxius al dipòsit. Per a això, editem un arxiu `gedit library.h` amb el contingut següent:

```
#ifndef DEFINITIONS_H
#define DEFINITIONS_H 1
/* Implement a number using a linked list. */
struct LinkedListNumber
{ struct LinkedListNumber*
one_less_;
};
#endif /* DEFINITIONS_H */
```

També editem el `hello.c` de tal manera que incloem al principi `library.h` de la següent manera: `#include "library.h"`. Si mirem l'estat amb `git status`, veurem:

```
  # On branch master
  # Changed but not updated: (use "git add <file>..."
  to update what will be committed)
```

```
       # modified: hello.c
       # Untracked files: # (use "git add <file>..."
       to include in what will be committed)
       # library.h
       no changes added to commit (use "git add"and/or "git commit -a")
```

Ara hi hem d'afegir la biblioteca i fer un nou *commit*:

```
remo@debian:~/projects/hello$ git add library.h
remo@debian:~/projects/hello$ git commit -a -m "library.h file added"
  Created commit b1042af: library.h file added
   2 files changed, 11 insertions(+), 0 deletions(-)
   create mode 100644 library.h
```

Si mirem l'estat amb `git status`:

```
       # On branch master nothing to commit (working directory clean)
```

I el registre amb `git log`:

```
       commit b1042afd6085db51eeb80586876d87c9ebd9ad36
       Author: remo <remo@debian.(none)>
       Date: Sun Nov 28 04:43:13 2010 -0500
        library.h file added
       commit e8332018c780c396ee442674e420046ccb397f3b
       Author: remo <remo@debian.(none)>
       Date: Sun Nov 28 04:38:01 2010 -0500
        Initial Commit
```

Amb aquest pas ja tenim el primer dipòsit Git i hem fet dos *commits*. No us preocupeu si en aquest moment us sentiu perduts; és ben normal i és qüestió de treballar i conèixer les ordres per a descobrir la potència de Git.

8) Configurem Apache i Gitweb (s'ha de fer com a root). Per a això cal copiar els arxius `gitweb.cgi, logo` i `css` al directori `/var/www/git`: `cp /usr/share/gitweb/* /var/www/git`; `cp /usr/lib/cgi-bin/gitweb.cgi /var/www/git`. Tenint present que el *DocumentRoot* d'Apache és a `/var/www`, es modifica la configuració d'Apache amb l'arxiu `/etc/apache2/conf.d/git` i al seu interior es posa:

```
  <Directory /var/www/git>
      Allow from all
      AllowOverride all
      Order allow,deny
      Options ExecCGI
      <Files gitweb.cgi>
       SetHandler cgi-script
      </Files>
  </Directory>
DirectoryIndex gitweb.cgi
SetEnv GITWEB_CONFIG /etc/gitweb.conf
```

9) Modifiquem l'arxiu `/etc/gitweb.conf` per incloure (mireu les diferències amb l'original, sobretot les / dels directoris):

```
    # path to git projects (<project>.git)
    $projectroot = "/var/cache/git";
    # directory to use for temp files
    $git_temp = "/tmp";
    # target of the home link on top of all pages
    #home_link =my_uri || "/";
    $home_link = "/git/";
    # html text to include at home page
    $home_text = "indextext.html";
    # file with project list; by default, simply scan the projectroot dir.
    $projects_list = $projectroot;
    # stylesheet to use
    $stylesheet = "gitweb.css";
    # logo to use
    $logo = "git-logo.png";
    # the 'favicon'
    $favicon = "git-favicon.png";
```

Això indica que totes les nostres còpies i dipòsits Git seran a `/var/cache/git`, que és el lloc per defecte en Debian. Si el voleu canviar, haureu modificar totes aquestes variables i la d'Apache. Finalment, s'ha de recarregar Apache amb `/etc/init.d/apache2 reload`.

10) Modifiquem el dipòsit de treball. Per fer-ho, ens canviem al directori de treball `project` original com a root:

```
    debian:/home/remo/projects# git clone -bare hello hello.git
Initialized empty Git repository in /home/remo/projects/hello.git/
    debian:/home/remo/projects#  ls -al
```

```
...
drwxr-xr-x 3 remo remo 4096 2010-11-28 04:40 hello
drwxr-xr-x 7 root root 4096 2010-11-28 05:10 hello.git
```

Verifiquem que existeix el servei `cat /etc/services | grep ĝit`, que ens haurà de donar `git 9418/tcp # Git Version Control System` i fem que el directori hello.git sigui exportable: `touch hello.git/git-daemon-export-ok`.

11) Ara movem el directori al seu lloc definitiu (al lloc indicat per la variable de l'arxiu `/etc/gitweb.conf`):

debian:/home/remo/projects# `mv hello.git /var/cache/git/`.

I finalment ja podeu veure el vostre projecte a `http://localhost/git/`, que donarà una (sortida) navegable com:

```
projects/hello.git/
summary | shortlog | log | commit | commitdiff | tree
  description My first Git Repository: hello.c + library.h
  owner root
  last change Sun, 28 Nov 2010 09:43:13 +0000
Shortlog
2 hours ago remo library.h file added master commit | commitdiff | tree | snapshot
2 hours ago remo Initial Commit commit | commitdiff | tree | snapshot
heads
2 hours ago master shortlog | log | tree
My first Git Repository: hello.c + library.h
RSS Atom
```

12) Finalment, com que tot estarà a la web, és necessari esborrar el dipòsit original per a continuar treballant en un clon del dipòsit web:

```
remo@debian:~/projects$ rm -rf hello
remo@debian:~/projects$ git clone /var/cache/git/hello.git/ hello
  Initialized empty Git repository in /home/remo/projects/hello/.git/
remo@debian:~/projects$ ls hello
  hello.c library.h
```

Com es pot veure, aquests arxius tenen totes les modificacions. Finalment, consulteu la documentació per a clonar i fer canvis des d'una altra màquina.

1.4.5. Mantis Bug Tracker

Mantis Bug Tracker és un sistema (GPL2) de seguiment d'errors a través de la web. MantisBT s'usa habitualment per a fer un seguiment d'errors o incidències, però també serveix com a eina de seguiment de gestió i administració de projectes. Des de 2008, el projecte ha canviat el programa de seguiment de projectes de Subversion a GIT, i actualment (novembre de 2010) es pot integrar amb Gitweb, GitHub, WebSVN, SourceForge (*open-source software hosting facility*; únicament per a integracions de Subversion). A més del paquet de Mantis, cal tenir instal·lat MySQL, Apache Web Server i el mòdul PHP per a Apache. Per a verificar que MysSQL funciona correctament es pot utilitzar phpMyAdmin (`http://localhost/phpmyadmin/`) i per a comprovar que també s'han instal·lat els mòduls PHP a Apache es pot crear un arxiu `/var/www/prova.php` amb el contingut següent:

```
<html>
<body>
<h1>Prova de PHP i Apache. </h1>
<?php phpinfo() ;?>
</body>
</html>
```

En accedir a `http://localhost/prueba.php` es pot veure tota la informació (un gran conjunt de taules acolorides amb una gamma de blaus) amb

la informació de php. A partir d'aquí, es pot instal·lar Mantis amb `apt-get install mantis` o des de Synaptic. Durant la instal·lació ens indicarà que la contrasenya per a l'usuari administrador és root (ens avisarà que és recomanable canviar-la) i ens demanarà si volem fer la integració amb MySQL mitjançant el paquet `dbconfig-common`. És recomanable fer-ho d'aquesta manera i ens demanarà l'usuari i la contrasenya per accedir a MySQL. A partir d'aquest moment ens podrem connectar a `http://localhost/mantis/` introduint l'usari administrador i la contrasenya root (es recomana que la primera acció sigui canviar la contrasenya en la secció `MyAccount`). En accedir des de la interfície web es podran crear nous usuaris amb permisos d'usuari per rols (administrador, viewer, reporter, updater, etc.); definir els projectes i les categories dins de cada projecte; gestionar els anuncis, informes i registres; tenir una visió general dels projectes i el seu estat; gestionar els documents associats, etc.

Existeix un considerable conjunt d'opcions que es poden modificar en el fitxer `/usr/share/mantis/www/config_inc.php` [14]. Així, per a canviar la llengua de l'entorn, s'edita i s'afegeix `$g_language_choices_arr = array( 'english', 'spanish'); $g_default_language = 'spanish';` i perquè la pàgina principal de Mantis Bug Tracker sigui automàticament el mateix Bug Tracker i es permeti un accés anònim als projectes públics:

1) Crear un compte d'usuari, per exemple anonymous o guest, tot deixant en blanc el `Real Name`, `Email=anonymous@localhost` (o deixar-ho en blanc si es posa `$g_allow_blank_email = ON`), `Access Level = viewer` o també `reporter` (depenent dels que es vulgui) `Enabled = true` i `Protected = true`.

2) Després s'ha de modificar en l'arxiu anterior (`config_inc.php`), afegint-hi les variables següents:

```
$g_allow_anonymous_login = ON;
$g_anonymous_account = 'anonymous'
```

i, opcionalment, per a deixar en blanc les adreces de correu:

```
$g_allow_blank_email = ON.
```

Per a més configuracions o integracions amb altres paquets, consulteu el manual [8] o accediu a la pàgina de la *wiki* del projecte* [14]. Es pot executar una demostració en els mateixos servidors del projecte**.

***http://www.mantisbt.org /wiki/doku.php /mantisbt:mantis_recipies**
****http://demo.mantisbt.org /my_view_page.php**

Activitats

1. Definiu en PostgreSQL una base de dades que tingui almenys tres taules amb cinc columnes (de les quals, tres han de ser numèriques)a cada taula. Genereu una llista ordenada per a cada taula o columna. Genereu una llista amb els valors de la columna X de totes les taules, ordenats de manera ascendent. Canvieu el valor numèric de la columna Y pel valor numèric de la columna Z més el valor de la columna W/2.

2. Feu l'exercici anterior, aquest cop amb MySQL.

3. Configureu el CVS per fer tres revisions d'un directori en què hi ha quatre 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 dues terminals de Linux i indiqueu la seqüència de passos que cal fer perquè dues modificacions alternes de cada usuari quedin reflectides en el dipòsit CVS.

5. Feu l'exercici anterior, però en aquest cas un dels usuaris es connecta al dipòsit des d'una altra màquina.

6. Feu novament els tres exercicis anteriors, però en Subversion.

7. Creeu un dipòsit sobre Git i feu-lo visible amb la web, de manera que dos usuaris de la mateixa màquina puguin modificar els arxius i actualitzar el dipòsit.

8. Instal·leu Mantis i genereu un usuari anònim que pugui accedir a un projecte públic com a reporter, però no a un de privat.

Bibliografia

[1] *Apache2 i SSL*. <http://www.debian-administration.org/articles/349>

[2] **Cederqvist**. *Version Management with CVS*.
<http://www.cvshome.org>

[3] **Cervisia**. *Interfície Cervisia per a CVS*.
<http://cervisia.kde.org/>

[4] **Debian.org**. *Debian*.
<http://www.debian.org>

[5] **Ibiblio.org** (2010). *Linux Documentation Center*.
<http://www.ibiblio.org/pub/Linux/docs/HOWTO/>

[6] **Mourani, G.** (2001). *Securing and Optimizing Linux: The Ultimate Solution*. Open Network Architecture, Inc.

[7] *Documentació de MySQL*.
<http://dev.mysql.com/usingmysql/get_started.html>

[8] *Documentació del projecte Mantis*.
<http://www.mantisbt.org/documentation.php>

[9] *Git. Fast Control Version system*.
<http://git-scm.com/>

[10] *Instal·lació de Git sobre Debian*.
<http://linux.koolsolutions.com/2009/08/07/learn-git-series-part-1-installing-git-on-debian/>

[11] *Integració WebDAv amb Subversion*.
<http://www.debian-administration.org/articles/285>
<http://www.debian-administration.org/articles/208>

[12] **Kiesling, R.** *The RCS (Revision Control System)*. The Linux Documentation Project.

[13] *Llibre sobre Subversion*.
<http://svnbook.red-bean.com/nightly/es/index.html>
(versió en castellà <http://subversion.tigris.org/servlets/ProjectDocumentList>)

[14] *Mantis Bug Tracker Wiki*.
<http://www.mantisbt.org/wiki/doku.php>

[15] *Mòduls de Webmin*.
<http://doxfer.webmin.com/Webmin>

[16] *Mysql Administrator*.
<http://www.mysql.com/products/tools/administrator/>

[17] *PgAdmin*.
<http://www.pgadmin.org>

[18] *PgpPGAdmin*.
<http://phppgadmin.sourceforge.net/>

[19] *PostgreSQL*.
<http://www.postgresql.org>

[20] *Subversion*.
<http://subversion.apache.org/>

[21] **Tkcvs**. *Interfície Tkcvs per a CVS*.
<http://www.twobarleycorns.net/tkcvs.html>

[22] **Vasudevan, A.** *CVS-RCS (Source Code Control System)*.

[23] *WebMin*.
<http://www.webmin.com/>