Administració de dades

Remo Suppi Boldrito

PID_00239614

Universitat Oberta de Catalunya

Índex

Introducció

Segons afirmen alguns experts, entre el 2013-2018 es generaran tantes dades com en els últims 5000 anys! Aquest fenomen, conegut com a *big data*, és utilitzat per a referir-se a situacions en què el conjunt de dades que cal tractar supera les mesures habituals de dades gestionades avui dia pels sistemes d'informació i hauran de ser obtingudes, emmagatzemades i processades en temps raonables. Com es considera *Big Data*? És un criteri dinàmic, però normalment es pot considerar com un conjunt de dades únic d'entre desenes de *terabytes* i desenes de *petabytes* (10 PBytes = 10.000 TBytes = 10 milions de gigabytes) i en què les dificultats que es trobaran estaran en la seva captura, emmagatzematge, cerca, compartició, lectura, anàlisi i visualització. Vinculades a aquesta nova tendència, han recuperat força tecnologies conegudes com a *data minining* (mineria de dades) en la versió més enfocada al processament de dades socials (*social data mining*) o el *datawarehouse* (repositori de dades), que seran aspectes predominants per a tenir en compte en el futur proper.[BD]

Per aquest motiu, la gestió de dades és un aspecte molt important en la utilització de la tecnologia actual, i els sistemes operatius són la base en què s'executaran les aplicacions que capturaran, emmagatzemaran i gestionaran aquestes dades. Un maneig eficient en l'emmagatzematge, gestió i processament de les dades no pot veure's, avui dia, separat d'una **base de dades** (*databases*, DB), i serà el punt crític per a les noves tecnologies que es desenvolupin a partir de les *Big Data*. Una base de dades és un conjunt estructurat de dades que poden organitzar-se de manera simple i eficient per part d'un gestor d'aquesta base. Les bases de dades actuals es denominen *relacionals*, ja que les dades poden emmagatzemar-se en diferents taules que faciliten la seva gestió i administració (exemples d'aquestes són MySQL/MariaDB, PostgreSQL). Per a això, i amb la finalitat 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 amb les bases de dades.

També es necessari destacar que comencen, amb el tractament derivat de les *Big Data*, a desenvolupar-se tecnologies per a gestionar dades no estructurades, en allò que es denominen *bases de dades no estructurades* (o *NoSQL* o també *no solament SQL*) i que defineixen sistemes de gestió de bases de dades que difereixen de les tradicionals BD relacionals (RDBMS), en les quals, per exemple, no fan servir SQL com el principal llenguatge de consultes, ja que no es fan cerques exactes. L'operació més típica és la cerca per similitud, les dades emmagatzemades no requereixen estructures fixes com ara taules i sí utilitzen categories com per exemple clau-valor. Les bases de dades NoSQL han crescut amb l'auge de grans companyies d'Internet (Google, Amazon, Twitter i

Facebook, entre d'altres), ja que necessiten trobar formes de tractament de les dades produïdes que amb les RDBMS no poden o és molt complex/ineficient i en què el rendiment/eficiència en el seu processament i les seves propietats de temps real són més importants que la coherència. Exemple d'aquestes bases de dades són Hadoop-Hbase (utilitzada per gairebé totes les grans companyies d'Internet), MongoDB (NoSQL orientat a documents), Elasticsearch (*multitenancy*, *full-text search engine* amb interfície web RESTful i suport per a *schema-free JSON documents*) o Redis (motor de base de dades en memòria, basat en l'emmagatzematge en taules d'*hashes* –clau/valor).

En aquest mòdul es veuran els gestors més importants de bases de dades en entorns GNU/Linux, una breu ressenya a SQL, així com diferents formes 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 en forma massiva i ús eficient.
2. Desenvolupar els aspectes essencials de bases de dades i la seva instal·lació i ús.
3. Treballar amb les tècniques de control de versions i analitzar els seus 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 forma més utilitzada per a accedir a una base de dades és mitjançant una aplicació que executa codi SQL. Per exemple, és molt comú accedir a una DB a través 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 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 **PostgreSQL** i **MySQL** (o el seu *fork* **MariaDB**), que seran objecte de la nostra anàlisi.

També veurem alguns aspectes introductoris sobre SQLite, que és un sistema de gestió de bases de dades relacional (i compatible amb ACID) contingut en una biblioteca (escrita en C) i en què, a diferència dels sistemes anteriors (client-servidor), el motor de SQLite no és un procés independent que es comunica amb el programa sinó que SQLite s'enllaça amb el programa i passa a ser-ne part. El programa utilitza la funcionalitat de SQLite mitjançant crides simples a funcions, reduint l'accés a la base de dades i essent molt més eficients. El conjunt de la base de dades (definicions, taules, índexs i les mateixes dades) és guardat en un sol arxiu estàndard a la màquina *host* i la coherència de la BD s'aconsegueix bloquejant tot el fitxer de base de dades al principi de cada transacció. Finalment en aquest apartat de bases de dades farem una breu introducció a les de NoSQL instal·lant i analitzant uns petits exemples en **MongoDB**, que és una de les bases de dades d'aquest tipus de més acceptació en els últims temps.

D'altra banda, quan es treballa en el desenvolupament d'un programari, existeixen altres aspectes relacionats amb les dades, com la validesa i el seu àmbit (sobretot si existeix 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 al costat de les possibles especialitzacions fetes per a algun client específic. El control de versions es fa per a controlar les diferents versions del codi font. No obstant això, els mateixos conceptes són aplicables a altres àmbits i no només per al codi font sinó també per als documents, imatges, etc. 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, GIT, Bazaar, Darcs, Mercurial, Monotone, Codeville, RCS, etc.).

En aquest mòdul, veurem **CVS** (***Control Version System***), **Subversion**, **GIT** i **Mercurial** per a controlar i administrar múltiples revisions d'arxius, automatitzant 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'analitzarà una eina per al seguiment d'incidències i errors en entorn de desenvolupament o de projectes anomenat **Mantis**.

La justificació de CVS i Subversion es pot trobar en què CVS és un dels paquets tradicionals més utilitzats i Subversion (també és conegut com a `svn` per ser el nom de l'eina de línia d'ordres) és un programa de control de versions dissenyat de manera específica per a reemplaçar el popular CVS i que soluciona algunes de les seves deficiències. Una característica important de Subversion és que, a diferència de CVS, els arxius amb versions no tenen cadascun un número de revisió independent. Per contra, tot el repositori té un únic número de versió que identifica un estat comú de tots els arxius del repositori en un determinat moment.

A continuació, veurem dos grans entorns en la gestió de repositoris donades les seves prestacions i utilització en grans projectes: GIT i Mercurial. **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. **Mercurial** és un sistema de control de versions (bàsicament desenvolupat en Python), i dissenyat per a obtenir el millor rendiment i escalabilitat, amb un desenvolupament completament distribuït (sense necessitat d'un servidor) i que permet una gestió robusta d'arxius (text o binaris) i amb capacitats avançades de ramificació i integració. Finalment, per a completar un mínim cercle d'eines per a compartir i gestionar dades, es presenta una breu descripció 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 (generalment MySQL i Apache).

1.1. PostgreSQL

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

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

2) L'aplicació del client (*frontend*) és la que sol·licita les operacions que cal portar a terme en la DB i que poden ser d'allò més variades; per exemple: eines en mode text, gràfiques, servidors de web, etc.

Generalment, el client i el servidor es troben en diferents *hosts* i es comuniquen mitjançant 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 poden ser dutes a terme per l'usuari o per l'administrador, 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 DB, ja que dins de les funcions de l'administrador de DB s'inclou la instal·lació del servidor, la inicialització i configuració, l'administració dels usuaris 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 a través del 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 línia d'ordres, per exemple, en Debian el `apt-get`. Per al segon cas, es recomana anar sempre a l'origen (o a un repositori mirall de la distribució original). És important tenir en compte que després la instal·lació des del codi font quedarà fora de la DB de programari instal·lat i es perdran els beneficis d'administració de programari que presentin per exemple `apt-cache` o `apt-get`.[Posa]

Instal·lació pas a pas

En aquest apartat, s'optarà per la instal·lació i configuració dels paquets de la distribució (tant de la versió distribuïda com la de l'última versió de desenvolupador), però la instal·lació des de les fonts no genera cap dificultat i és l'adequada si es desitja tenir l'última versió de la BD disponible. Les fonts es poden obtenir des de http://www.postgresql.org/ftp/source/ i seguint les indicacions des de https://www.postgresql.org/docs/9.5/static/installation.html.

La instal·lació és simple, executant `apt-get install postgresql` instal·larà la versió* 9.x (a més d'altres paquets addicionals com `postgresql-client postgresql-client-common postgresql-common`). En el cas que es desitgi instal·lar l'última versió 9.x (9.4, en el moment d'escriure aquest mòdul) des dels paquets binaris, PostgreSQL facilita els paquets per a Debian**, on es trobaran les instruccions per a instal·lar el binari de la darrera versió compilada per a aquesta distribució. En el cas de que es desitgi una distribució més actual per a Debian, és possible consultar en Debian Backports*** si existeix una nova distribució. En aquest cas es pot instal·lar des del repositori.

***A Debian Jessie és 9.4 però va canviant amb les diferents versions**

****http://www.postgresql.org/download/linux/debian/**

*****https://backports.debian.org/**

Si es desitja instal·lar el paquet que inclou diferents contribucions de la comunitat*, llavors haurem de fer `apt-get install postgresql-contrib` (aquestes contribucions inclouen *fuzzystrmatch, unaccent o citext* per a cerques intensives o difuses, per exemple). Després de la instal·lació tindrem:

1) client postgresql (`psql`),

2) usuari del sistema per defecte: `postgres` (no s'ha de canviar el *passwd* d'aquest usuari)

3) usuari per defecte de postgresql: és el *superuser* `postgres` (el *passwd* s'haurà de canviar des de dins de la BD),

4) usuari de la BD: postgres,

5) clúster per defecte: `main`,

6) BD per defecte: `template1`,

7) schema per defecte: `public`.

Per a canviar el *passwd* de l'usuari postgres de la BD fem: `su -l root`, a continuació `su - postgres` i després `psql`, per entrar a la consola (client o *frontend*). Després dels missatges i davant del *prompt postgres=#* introduim `\password postgres` i el *passwd* seleccionat dues vegades. Per sortir fem `\q`. A continuació, s'hauran de fer alguns ajustos com per exemple el de l'arxiu */etc/postgresql/x.y/main/postgresql.conf* (on *x.y* serà el número de versió que hi ha instal·lada, per exemple 9.4) i en què s'haurà de fer el canvi de la línia *# listen_addresses = 'localhost'* per *listen_addresses = 'localhost, 192.168.1.200'* en què la IP és la de servidor. També caldrà definir el mètode d'autenticació en */etc/postgresql/9.1/ main/pg_hba.conf* i agregar, per exemple

```
host all all 192.168.1.200/24 md5
host all all 192.168.1.100/24 md5
```

On 192.168.1.200 és l'adreça de servidor i 192.168.1.100 la IP de gestió de la DB i haurem d'activar, si ho tenim instal·lat, `iptables` per a acceptar comunicacions en el port 5432. Finalment, haurem de reiniciar el servidor amb `service postgresql restart` perquè els canvis siguin efectius. Els arxius essencials de PostgreSQL es troben a: configuració */etc/postgresql/x.y/main/*, binaris */usr/lib/postgresql/x.y/bin*, arxius de dades */var/lib/postgresql/x.y/main* i *logs* */var/log/postgresql/postgresql-x.y-main.log*. Per a veure els clústers que hi ha disponibles (PostgreSQL es basa en CLUSTERS que és un grup de BD manejades per un servei comú i on cadascuna d'aquestes BD conté un o més SCHEMATA), podem executar `pg_lsclusters` que en el nostre cas és:

```
Ver  Cluster Port Status Owner    Data directory              Log file
9.4  main    5432 online postgres /var/lib/postgresql/9.4/main var/log/postgresql/postgresql-9.4-main.log
```

Vegeu la documentació de PostgreSQL en la següent pàgina del seu lloc web: https://www.postgresql.org/docs/9.4/static/index.html.

1.1.2. Com s'ha de crear una base de dades?

La primera acció per a verificar si es pot accedir al servidor de DB és crear una base de dades. El servidor PostgreSQL pot gestionar 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. Si dins de l'usuari postgres executem `psql` i dins del client fem `\l` veurem totes les BD definides. També es pot fer `\d` i mostrarà les relacions a la base de dades (taules, seqüències, etc.). A més a més, `\d [nom_taula]` donarà una descripció d'una taula (nom de columnes, tipus de dades, etc.).

És important tenir en compte que per a dur a terme aquesta acció, hi ha d'haver un usuari habilitat per a crear una base de dades, com hem vist en l'apartat anterior, en què existeix un usuari que instal·la la base de dades i que tindrà permisos per a crear bases de dades i crear 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` (o prèviament fer `su -l root`) i a continuació es podrà portar a terme el `createdb`. Per a crear una DB denominada *nteumdb*:

```
createdb nteumdb
```

Si l'execució de l'ordre dóna error, és possible que no estigui ben configurat el camí o que la DB estigui mal instal·lada. Es pot intentar amb el camí absolut (`/usr/lib/postgresql/x.y/bin/createdb nteumdb`), on la ruta dependrà de la instal·lació que s'hagi fet (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 a eliminar la base de dades, es pot utilitzar `dropdb nteumdb`.

1.1.3. Com es pot accedir a una base de dades?

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

Nota

Per a poder accedir a la DB, 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'iniciï en l'arrencada de l'ordinador. Per a més detalls, consulteu l'apartat 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* que tingui suport ODBC per a crear i manipular DB;

3) escrivint una aplicació amb alguns 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 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 =#` o `nteumdb =>` (el primer si és el superusuari

i el segon si és un usuari normal). 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 per \, per exemple \h (enumera totes les ordres disponibles) o \q per a acabar. Podem consultar en https://www.postgresql.org/docs/9.4/static/sql-commands.html una llista de les ordres SQL que permet.

1.1.4. Usuaris de la base de dades

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, es pot executar la sentència SQL

`CREATE USER` *nom*

i per a esborrar usuaris, `DROP USER` *nom*.

També es poden 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 els restants (per a crear nous usuaris, si l'usuari de sistema operatiu amb el qual s'administra la DB no és postgres `psql -O usuari`).

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

- **Superusuari**: aquest usuari no té cap restricció. Per exemple, podrà crear nous usuaris: `CREATE USER nom SUPERUSER;`
- **Creador de DB**: té permís per a crear DB. 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 desitja controlar l'accés dels usuaris quan es connectin a una DB. Per a crear un usuari amb contrasenya (`paraula_clau` serà la clau per a aquest usuari):

```
CREATE USER nom WITH PASSWORD 'paraula_clau';
```

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;
```

Per a inserir usuaris en aquest grup:

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

Per a 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 crea una DB, els privilegis són per a l'usuari que la crea (i per al *superusuari*). Per a permetre que un altre usuari utilitzi aquesta DB o una part, 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` (s'han de consultar les referències per a veure el seu significat).

Per a assignar els privilegis, es pot utilitzar `GRANT UPDATE ON objecte TO usuari` on `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;
```

1.1.5. Manteniment

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

1) Recuperar l'espai: per a això s'haurà d'executar periòdicament la instrucció `VACUUM` (si és des de dins del `psql` o l'ordre `vacuum` en minúscules des del sistema operatiu), que recuperarà l'espai de disc de files esborrades o modificades, actualitzarà 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 `reindexdb` per a

reindexar una DB sencera (s'ha de tenir en compte que depenent de la grandària de les DB, aquestes ordres poden trigar un cert temps).

3) Canvi d'arxius de registre (*logs*): s'ha d'evitar que els arxius de registre siguin molt grans i difícils de manejar. Es pot fer fàcilment quan s'inicia el servidor amb `pg_ctl start | logrotate`, en què aquesta última ordre renombra i obre un nou arxiu de registre i es pot configurar amb */etc/logrotate.conf*.

4) Còpia de seguretat i recuperació (*backup* i *recovery*): hi ha dues formes de guardar les dades, amb la sentència SQL `dump` o guardant l'arxiu de la DB. El primer és `pg_dump ArxiuDB > ArxiuBackup`. Per a recuperar, es pot utilitzar `psql ArxiuDB < ArxiuBackup`. Per a guardar totes les DB del servidor, es pot executar `pg_dumpall > ArxiuBackupTotal`. Una altra estratègia és guardar els arxius de les bases de dades en un nivell del sistema operatiu, per exemple amb `tar -cf backup.tar /var/lib/postgresql/x.y/main`. Hi ha dues restriccions que poden fer que aquest mètode sigui poc pràctic:

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

b) s'han de conèixer molt bé totes les implicacions en un àmbit d'arxiu, on hi ha totes les taules, transaccions i altres, ja que en cas contrari, una DB pot quedar inútil. A més, en general, la grandària que es guardarà serà major que allò fet amb els mètodes anteriors, ja que, per exemple, amb el `pg_dumb` no es guarden els índexs, sinó l'ordre per a recrear-los.

1.2. El llenguatge SQL

No és la finalitat d'aquest subapartat fer un tutorial sobre SQL, però s'analitzaran uns exemples per a veure les capacitats d'aquest llenguatge. Són exemples que vénen amb la pròpia distribució de les fonts de PostgreSQL en el directori `DirectoriInstal·lació/src/tutorial`. Per a accedir-hi, canvieu al directori de PostgreSQL (`cd DirectoriInstal·lació/src/tutorial`) i executeu `psql -s nteumdb` i dins `\i basics.sql`, després. El paràmetre `\i` llegeix les ordres de l'arxiu* especificat (`basic.sql` en el nostre cas). PostgreSQL, com ja hem esmentat, és una base de dades relacional (*Relational Database Management System, RDBMS*), la qual cosa significa que maneja 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 maneja aquesta col·lecció de DB (tot el conjunt es denomina *agrupació* o *clúster de bases de dades*, *database cluster*). Per a crear, per exemple, una taula amb psql, executeu:

***Sinó s'han descarregat les fonts de PostgreSQL, aquest arxiu es pot obtenir des de GitHub https://github.com/postgres/postgres/tree/master/src/tutorial.**

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

L'ordre acaba quan es posa ';' i es poden utilitzar 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;
```
Per a introduir dades, es poden utilitzar dues formes: posar totes les dades de la taula o indicar les variables i els valors que es desitgen modificar:
```
INSERT INTO temps VALUES ('Barcelona', 16, 37, 0.25, '2007-03-19');
INSERT INTO temp (ciutat, temp_min, temp_max, pluja, dia) VALUES
('Barcelona', 16, 37, 0.25, '2007-03-19');
```
Aquesta manera pot ser senzilla per a unes poques dades, però quan cal introduir gran quantitat de dades, 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 a mirar una taula, podríem fer:
```
SELECT * FROM temps;
```
on el * significa "totes les columnes".
Exemple: introduir dades en taula. Dins de psql:
```
INSERT INTO NomTB (valorVar1, ValorVar2,...);
```
Dades des d'un arxiu. Dins de psql:
```
COPY NomTB FROM 'NomArxiu';
```
Visualitzar dades. Dintre de psql:
```
SELECT * FROM NomTB;
```
Alguns exemples d'ordres més complexes serien (dins de psql). Visualitza la columna ciutat després de portar a terme 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 > 0.0;
```
Unió de taules:
```
SELECT * FROM temps, ciutat WHERE ciutat = nom;
```
Funcions, màxim en aquest cas:
```
SELECT max(temp_min) FROM temps;
```
Funcions imbricades:
```
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';
```
Altres ordres d'utilitat:
Login com a usuari postgres (SuperUser) per a treballar amb la base de dades:
`# su - postgres`
Crear base 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`
Guardar la base de dades: `$ pg_dump mynteum > db.out`
Carregar la base de dades guardada anteriorment: `$ psql -d mynteum -f db.out`
Per a guardar totes les bases de dades: `# su - postgres` i després `$ pg_dumpall > /var/lib/postgresql/backups/dumpall.sql`
Per a restaurar una base de dades: `# su - postgres` i després
`$ psql -f /var/lib/postgresql/backups/dumpall.sql mynteum`
Per a mostrar les bases de dades: `psql -l` o si no, des de dintre `mynteum=# \l;`
Mostrar els usuaris: des de dins de l'ordre `psql` executar `mymydb=# SELECT * FROM "pg_user";`
Mostrar les taules: des de dins de l'ordre `psql` executar `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`

Per a un usuari que s'inicia en PostgreSQL és interessant obtenir una base de dades ja construïda i practicar amb aquesta. Per exemple, en l'adreça http://pgfoundry.org/projects/dbsamples/ podem obtenir la base World 1.0,

que ens dóna una llista de 4.079 ciutats del món de 239 països i la seva població. Una vegada descarregada (format tar.gz) la descomprimim amb l'ordre `tar xzvf world-1.0.tar.gz` i entrem en `psql`, des del prompt fem

```
\i /tmp/world.sql
```

Veurem com es creen les taules i ja estarem en condicions de fer consultes com:

```
SELECT * FROM "pg_tables" WHERE schemaname = 'public';         --Mirem les taules
 schemaname |    tablename    | tableowner | tablespace | hasindexes | hasrules | hastriggers
------------+-----------------+------------+------------+------------+----------+-------------
 public     | city            | postgres   |            | t          | f        | t
 public     | country         | postgres   |            | t          | f        | t
 public     | countrylanguage | postgres   |            | t          | f        | t

SELECT * FROM "city" WHERE countrycode = 'ESP';               --Mirem les ciutats d'un país
 id  |              name               | countrycode |      district      | population
-----+---------------------------------+-------------+--------------------+------------
 653 | Madrid                          | ESP         | Madrid             |    2879052
 654 | Barcelona                       | ESP         | Katalonia          |    1503451
 655 | València                        | ESP         | València           |     739412
...

SELECT count(*) FROM "city" WHERE countrycode = 'ESP';          --Comptem les ciutats d'un país
 count
-------
    59

SELECT * FROM "city" WHERE name = 'Barcelona';                  --Mirem les ciutats amb un nom
  id  |   name    | countrycode |  district  | population
------+-----------+-------------+------------+------------
  654 | Barcelona | ESP         | Katalonia  |    1503451
 3546 | Barcelona | VEN         | Anzoátegui |     322267

SELECT * FROM "city" WHERE name = 'Barcelona' and countrycode = 'ESP'; --I només les d'un país
 id  |   name    | countrycode | district  | population
-----+-----------+-------------+-----------+------------
 654 | Barcelona | ESP         | Katalonia |    1503451

SELECT * FROM "city" WHERE population > '9500000';       --Mirem les ciutats amb una població major
  id  |      name       | countrycode |   district   | population
------+-----------------+-------------+--------------+------------
  206 | Sao Paulo       | BRA         | Sao Paulo    |    9968485
  939 | Jakarta         | IDN         | Jakarta Raya |    9604900
 1024 | Mumbai (Bombay) | IND         | Maharashtra  |   10500000
 1890 | Shanghai        | CHN         | Shanghai     |    9696300
 2331 | Seoul           | KOR         | Seoul        |    9981619

SELECT TRUNC(AVG(population),1) AS Total  FROM "city" WHERE population > '9500000';
                                --Obtenim la mitjana de les ciutats més poblades
   total
------------
 9950260.8
```

Per a més detalls sobre PostgreSQL i la seva utilització, recomanem [Mar][SQL].

1.2.1. Entorns d'administració gràfics

Hi ha diversos entorns d'administració gràfics, com **Phppgadmin** i **Pgadmin** (disponible en la majoria de distribucions, Debian inclosa). Aquests programes

permeten accedir i administrar una base de dades amb una interfície gràfica ja sigui mitjançant web o per interfície pròpia. En el cas d'interfície pròpia, la manera més fàcil d'accedir és des d'una terminal. L'administrador de la DB (si no és l'usuari postgresql) haurà de fer `xhost +`, la qual cosa permet que altres aplicacions puguin connectar-se al *display* de l'usuari actual i, a continuació, executar el nom de l'aplicació.

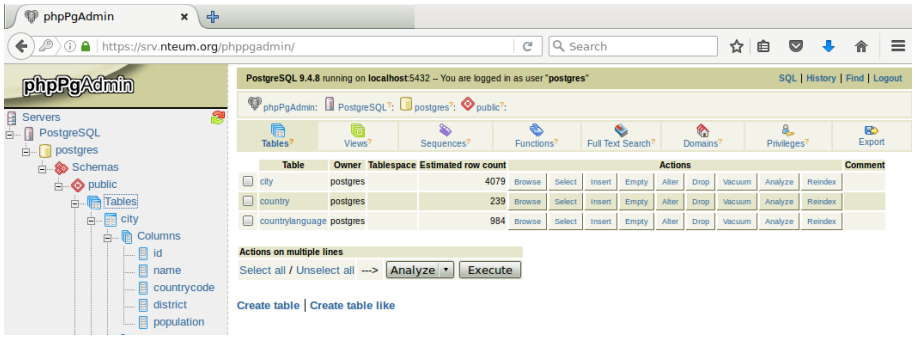
Per a instal·lar PhpPgAdmin executarem `apt-get install phppgadmin` i estarà disponible en l'URL *http://localhost/phppgadmin*. Quan desitgem entrar en la base de dades en el menú de l'esquerra, ens demanarà un usuari/*passwd* que si introduïm "postgres" i el *passwd* del superusuari ens donarà un error dient *Login disallowed for security reasons*. Per a habilitar el seu accés, haurem d'anar a l'arxiu */etc/phppgadmin/config.inc.php* i llavors haurem de canviar la línia `$conf['extra_login_security'] = true;` per aquesta altra línia:

```
$conf['extra_login_security'] = false;
```

A continuació, haurem de recarregar la pàgina i reiterar el procediment i ja podrem accedir a la gestió de la base de dades.

En la figura 1 es pot veure la interfície de `phppgadmin` amb la base de dades *world* i les taules amb un format amigable.

Figura 1



Si es desitja evitar treballar amb l'usuari "postgres" (és recomanable), es pot crear un usuari amb contrasenya per a accedir a la base de dades, i per a això cal fer des de dins de

```
psql: CREATE USER admin WITH PASSWD 'posar_passwd';
```

Per 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 indicant que el servidor és *localhost* i l'usuari i la contrasenya creats anteriorment. A partir d'aquí, veurem les bases de dades, així com tots els paràmetres i configuracions.

Una altra eina interessant és **Webmin**. És una interfície basada en el 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 arxius de configuració i permet administrar un sistema de forma local o remota. La instal·lació per a Debian es detalla en http://www.webmin.com/deb.html (Webmin va ser retirat del repositori Debian/Ubuntu el 2006 per retards en la solució d'errors i problemes en la política de manteniment del paquet https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=343897).

Enllaços d'interès

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

1.3. MySQL

Enllaç d'interès

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

MySQL [Mys] és, segons els seus autors, la base de dades (DB) SQL oberta, és a dir, de programari lliure (*Open Source*) més popular i utilitzada per grans companyies d'Internet (Facebook, Google, Adobe, Twitter, YouTube, Zappos, entre d'altres). La companyia que desenvolupa i manté la BD i les seves utilitats es denomina MySQL AB (el 2008, aquesta va ser adquirida per Sun Microsystems i al seu torn aquesta ho va ser el 2010 per Oracle Corporation, per la qual cosa MySQL és subsidiària d'Oracle Co.) i desenvolupa MySQL com a programari lliure en un esquema de llicència dual. D'una banda, s'ofereix sota la GNU GPL per a qualsevol ús compatible amb aquesta llicència (versió anomenada MySQL *Community Edition*), però per a aquelles empreses que vulguin incorporar-ho en productes privats, han de comprar una llicència específica que els permeti aquest ús (versions MySQL *Enterprise Edition* i MySQL Cluster CGE bàsicament). És interessant analitzar les diferents situacions (sobretot en la nostra orientació cap a l'*Open Source*) de llicències de MySQL. En l'opció GPL, podem utilitzar lliurement MySQL, és a dir, sense cost quan l'aplicació es desenvolupa en un nivell local i no s'utilitza comercialment. Per exemple, MySQL es pot fer servir lliurement dins d'un lloc web (si es desenvolupa una aplicació PHP i s'instal·la en un proveïdor de serveis d'Internet, tampoc s'ha de fer que el codi PHP estigui disponible lliurement en el sentit de la GPL). Igualment, un proveïdor de serveis d'Internet pot fer que MySQL estigui disponible per als seus clients sense haver de pagar llicència de MySQL (sempre que MySQL s'executi de manera exclusiva en l'equip de l'ISP) o MySQL es pot utilitzar de manera gratuïta per a tots els projectes GPL o llicència lliure comparable (per exemple, si hem desenvolupat un client de correu electrònic *Open Source* per a GNU/Linux, i es desitja emmagatzemar dades dels correus en una base de dades MySQL) i tots els desenvolupaments/extensions que es facin sobre els productes MySQL han de ser oberts/publicats i no comercials. L'ús de MySQL amb una llicència comercial s'aplica quan desenvolupem productes comercials i no estan oberts/publicats; és a dir, no es pot desenvolupar un producte comercial (p. ex.,. un programa de comptabilitat amb MySQL com a base de dades) sense que el codi estigui disponible com a codi obert. Si les limitacions de la GPL no són acceptables per a qui desenvolupa l'aplicació, llavors es pot vendre el producte (programa), juntament amb una llicència de MySQL

comercial (més informació en http://www.mysql.com/about/legal/). Considerant la llicència GPL, es pot obtenir el codi font, estudiar-lo i modificar-lo d'acord amb les seves necessitats sense cap pagament, però estarem subjectes als deures que imposa GPL i haurem de publicar com a codi obert el que hem desenvolupat també. MySQL proveeix la seva pàgina web d'un conjunt d'estadístiques i prestacions en comparació a altres DB per a mostrar a l'usuari com de ràpida, fiable i fàcil és d'usar. La decisió de triar una DB s'ha de fer acuradament en funció de les necessitats dels usuaris i de l'entorn on s'utilitzarà aquesta DB.

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

1.3.1. Instalació de MySQL

MySQL es pot obtenir des de la pàgina web* del projecte o des de qualsevol dels repositoris de programari. Es poden obtenir els binaris i els arxius font per a compilar-los i instal·lar-los. En el cas dels binaris, s'ha d'utilitzar la distribució de Debian i seleccionar els paquets `mysql-*` (`client-x.y`, `server-x.y` i `common` són necessaris) on *x.y* és la versió, que a Debian 8.5 pot ser 5.5. o 5.6. La instal·lació, després d'unes preguntes, crearà un usuari que serà el superusuari de la base de dades (p. ex., admin) i una entrada en `/etc/init.d/mysql` per a arrencar o aturar el servidor en el *boot*. 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 la línia d'ordres. Si obté els binaris (que no siguin Debian ni RPM, per a aquests simplement utilitzeu les utilitats comunes `apt-get` i `rpm`), per exemple un arxiu comprimit des del lloc web de MySQL, haurà d'executar les següents ordres 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 DB en el directori `/usr/local/mysql`. En 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 esment quan es fa la configuració, ja que

```
prefix=/usr/local/mysql
```

és el directori on s'instal·larà la DB i es pot canviar per a situar la DB en el directori que es desitgi.

1.3.2. Postinstal·lació i verificació de MySQL

Una vegada feta la instal·lació (ja 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 les ordres per `-u` i `-p`, de manera respectiva):

```
/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 DB predefinides
mysqlshow mysql -u user -p      Mostra les taules de la DB mysql
```

1.3.3. El programa monitor (client) mysql

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 mode *batch* (com un *script*), on les ordres se li passen a través d'un arxiu. Per a veure totes les opcions de l'ordre, es pot executar `mysql -help`. Podrem portar a terme 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 DB. Una vegada dintre, el `mysql` posarà un *prompt* (**mysql**>) i esperarà que introduïm alguna ordre (pròpia i SQL), per exemple, *help*. A continuació, donarem una sèrie d'ordres per a provar el servidor (recordeu posar sempre el ';' per a acabar l'ordre):

```
  mysql> SELECT VERSION(), CURRENT_DATE;
```
Es poden fer servir 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 denominada nteum.
```
  mysql> CREATE TABLE pet (name VARCHAR(20), owner VARCHAR(20),
  -> species VARCHAR(20), sex CHAR(1), birth DATE, death DATE);
```
Crea una taula dintre 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 tabulacions 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 marcia@localhost -> IDENTIFIED
BY 'passwd'WITH GRANT OPTION;
```
Crea un usuari marcia en la DB. Ho ha de fer el *root* de la DB. També es pot fer directament amb:
```
  mysql> INSERT INTO user (Host,User,Password) ->
  VALUES('localhost','marcià','passwd');
  mysql> select user,host,password from mysql.user;
```
Mostra els usuaris, *host* de connexió i *passwd* (també es poden posar les ordres en minúscules).
```
  mysql> delete from mysql.user where user='';
```
Esborra els usuaris anònims (sense nom en la informació anterior).

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 web 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 10 països amb major utilització d'Internet per any. Sobre aquestes últimes dades, s'ha de fer una classificació dels cinc països amb major utilització d'Internet des que existeixen dades (es 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: 5.5.35-0+wheezy1 (Debian)
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> create database bancomundial;

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| bancomundial       |
| mysql              |
+--------------------+
3 rows in set (0.00 sec)

mysql> use bancomundial;
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 'bancomundial'.'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> descriu paisbai;
+--------------------------------------+
|Field | Type         |Null |K| Def. |E|
| pais | 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 en http://localhost/phpmyadmin i
importar les dades des d'un fitxer a la base de dades bancomundial 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 ;
Fileres: 30
país        mitjanapaís
Afganistan  0.0203921568627
Albània     7.3696078431373
Algèria     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';
Fileres: 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 major a menor volum per a l'any 2008.
SELECT 'pais', '2008' FROM 'paisbai' ORDER BY '2008' DESC LIMIT 0 , 30;

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

```
4. Generar una llista dels 10 països amb major utilització d'Internet per any.
Alternativa 1.
SELECT 'país' , 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           | promig           |
+----------------+------------------+
| Luxemburg      |    284474.679628 |
| Hong Kong      | 21468.6420499333 |
| San Marino     | 5464.22342423529 |
| Països Baixos  | 3949.18559792941 |
| Dinamarca      |     3743.7899214 |
| Estònia        | 3118.98744675686 |
| Suècia         | 2967.78367829608 |
| Regne Unit     | 1902.25120777059 |
| Suïssa         | 1897.13803142745 |
| Bèlgica        | 1781.95881669216 |
+----------------+------------------+
```

1.3.5. Administración 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 DB, 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, s'han de prendre algunes precaucions:

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

2) Verificar `mysql -o 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 -o root -p mysql
```

3) Comprovar 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, és possible copiar els arxius amb extensió FRM, MYD i MYI amb el servidor parat. Per a recuperar, es pot executar mitjançant l'ordre `REPAIR TABLE` o `myisamchk -r`, la qual cosa funcionarà la majoria de vegades. En cas contrari, es podrien copiar els arxius guardats i arrencar el servidor. Hi ha altres mètodes alternatius en funció d'allò que es vulgui recuperar, com la possibilitat de guardar/recuperar part de la DB (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 Workbench***, que és una aplicació potent per al disseny, administració i control de bases de dades basades en MySQL (inclou el *Database Administration* que reemplaça l'anterior MySQL Administrator). Aquesta aplicació és un conjunt d'eines per als desenvolupadors i administradors de BD que integra el desenvolupament, la gestió, el control i el manteniment de manera simple i en un mateix entorn de la BD. Les parts principals són: disseny i modelatge de BD, desenvolupament en SQL (reemplaça el MySQL Query Browser), administració de la BD (reemplaça MySQL Administrator) i migració de BD.

Enllaç d'interès

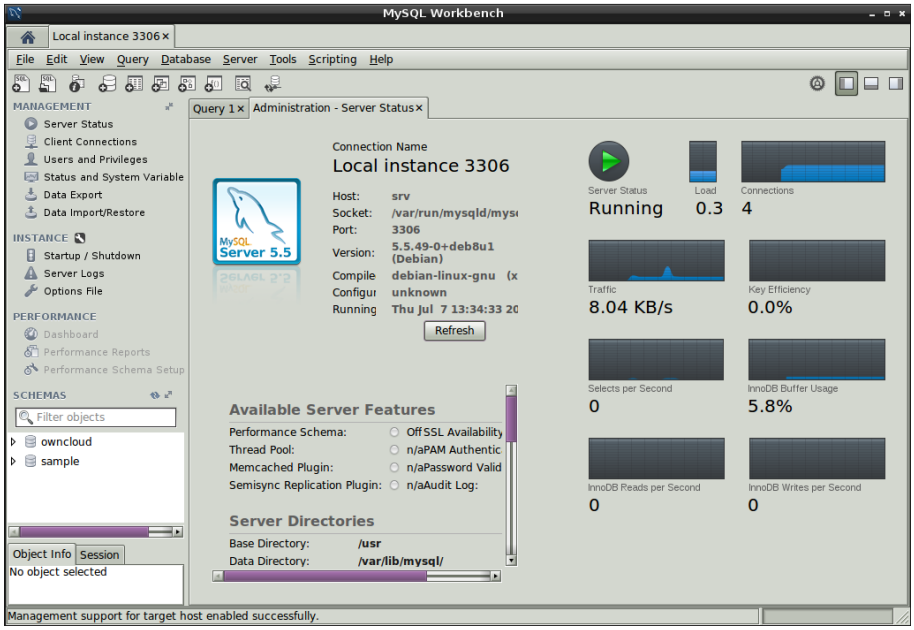
Es pot trobar tota la documentació per a la instal·lació i posada en marxa de Mysql Workbench en http://dev.mysql.com/downloads/workbench/.

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

Per a la seva instal·lació es pot fer `apt-get install mysql-workbench` i després executar-lo amb `mysql-workbench`. Després d'autentificar-nos ens mostrarà una interfície gràfica com es mostra a la figura 2, amb l'estat i possibilitats d'aplicació.

Per a usuaris recentment iniciats (o experts) recomanem, per la seva facilitat en l'ús i simplicitat, **PhpMyAdmin** (inclosa en la majoria de distribucions, també Debian), que és una eina de programari lliure escrit en PHP per a gestionar l'administració de MySQL mitjançant el web. PhpMyAdmin és compatible amb un ampli conjunt d'operacions en MySQL com, per exemple, gestió de bases de dades, taules, camps, relacions, índexs, usuaris, permisos, etc., i també té la capacitat d'executar directament qualsevol sentència

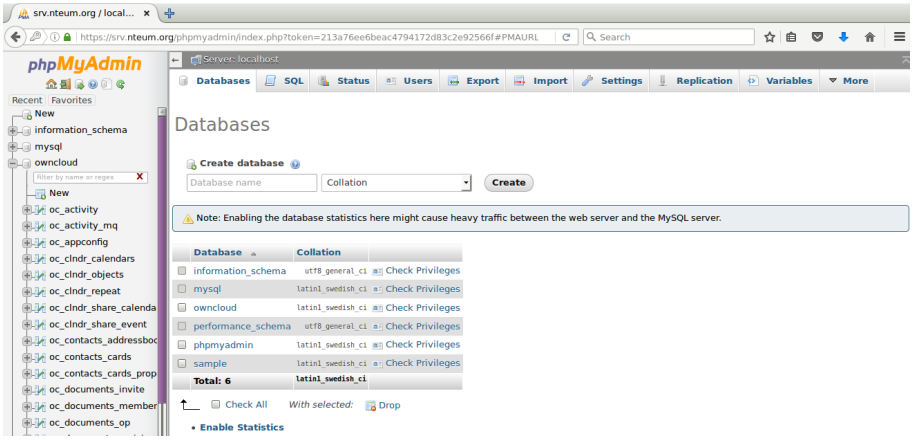
Figura 2



SQL. La seva instal·lació es pot fer des de la gestió de programes de distribució de treball (`apt-get install phpmyadmin`) que durant el procés d'instal·lació demanarà amb quins servidors de web es vol fer la integració i l'usuari de la base de dades. Una vegada instal·lada, es pot iniciar mitjançant `http://localhost/phpmyadmin`, que demanarà l'usuari i la contrasenya del servidor (indicats en els passos anteriors).

La figura 3 mostra la interfície de PhpMyAdmin on es poden veure totes les opinions i facilitats que presenta l'aplicació.

Figura 3



Hi ha altres eines que permeten fer tasques similars, com la que ja hem comentat en la secció de PostgreSQL com **Webmin**, que permet gestionar i administrar bases de dades MySQL (incloent el mòdul corresponent). Si bé aquest paquet ja no s'inclou amb algunes distribucions (p. ex., Debian/Ubuntu), es pot descarregar des de 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 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 els mateixos Mysql Data Base Server.

1.4. MariaDB

MariaDB[Mari] és un sistema de gestió de bases de dades derivat de MySQL amb llicència GPL que incorpora dos mecanismes d'emmagatzematge nous: Aria (que reemplaça amb amplis avantatges a MyISAM) i XtraDB (que substitueix InnoDB, l'actual del Mysql). Com a *fork* de MySQL manté una alta compatibilitat ja que posseeix les mateixes ordres, interfícies, API i biblioteques amb l'objectiu que es pugui canviar un servidor per un altre. La motivació dels desenvolupadors de fer un *fork* de MySQL va ser la compra de Sun Microsystems per part d'Oracle el 2010 (Sun prèviament havia comprat MySQL AB que és la companyia que desenvolupa i manté MySQL) amb l'objectiu de mantenir aquesta sobre GPL, ja que estaven/estan convençuts que l'únic interès d'Oracle en MySQL era reduir la competència que MySQL donava al seu producte comercial (Oracle DB).

MariaDB és equivalent a les mateixes versions de MySQL (MySQL 5.5 = MariaDB 5.5, no obstant això, la versió actual és MariaDB10.1) i té una sèrie d'avantatges en relació amb MySQL que resumim a continuació*:

***https://mariadb.com/kb/en/mariadb/mariadb-vs-mysql-features/**

1) Mecanismes d'emmagatzematge: a més dels estàndard (MyISAM, Blackhole, CSV, Memory i Arxive) s'inclouen Aria (alternativa a MyISAM resistent a caigudes), XtraDB (reemplaçament directe d'InnoDB), FederatedX (reemplaçament directe de Federated), OQGRAPH, SphinxSE, Cassandra (NoSQL i en MariaDB 10.0) TokuDB (des de MariaDB 5.5), i s'estan treballant en altres NoSQL, i les últimes CONNECT, SEQUENCE i Spider (només a partir de MariaDB 10.0).

2) Facilitat d'ús i velocitat: inclou noves taules i opcions per a generar estadístiques d'índexs i taules, processos, gestió del temps en microsegons, característiques NoSQL (p. ex.,. HandlerSocket), columnes dinàmiques i subqueries. En relació amb la velocitat de processament, es poden veure les comparacions en relació amb MySQL en https://mariadb.com/blog/mariadb-53-optimizer-benchmark.

3) Tests, errors i alertes: inclou un extens conjunt de tests en la distribució que permeten analitzar diferents combinacions de configuració i sistema operatiu. Aquests tests han permès reduir notablement els errors i alertes per la qual cosa es tradueix en una BD altament estable i segura en relació amb l'execució i funcionament.

4) Llicència: el codi en MariaDB està sota GPL, LPGL o BSD i no disposa de mòduls tancats com MySQL *Enterprise Ed* (de fet, tot el codi tancat en MySQL

5.5 E.Ed. està en MariaDB com a *open source version*). MariaDB inclou tests per a tots els errors solucionats, mentre que Oracle no fa el mateix amb MySQL 5.5 i tots els errors i plans de desenvolupaments són públics i els desenvolupadors pertanyen a la comunitat.

És important tenir en compte que MariaDB és desenvolupada per la comunitat, i la responsabilitat de desenvolupament i manteniment són a càrrec de SkySQL Corporation Ab, empresa fundada el 2010 per les mateixes persones de MySQL AB (David Axmark, Michael 'Monty' Widenius i Kaj Arnö). Aquesta empresa es va fusionar amb el Monty Program el 2013 (programa fundat per Monty Widenius -desenvolupador de MySQL juntament amb D. Axmark- després que va vendre la companyia a Sun MS i va decidir escriure MariaBD). El 2014, SkySQL crea una altra marca anomenada MariaDB AB i ofereix MariaDB *Enterprise*, MariaDB *Enterprise Cluster* i *MaxScale* sota un model diferent de negoci que la llicència de programari com a MySQL, per a dirigir-se a un mercat empresarial garantint fiabilitat i confiabilitat en aplicacions de missió crítica i alt rendiment. *https://mariadb.com/about*

1.4.1. Instal·lació MariaDB sobre Debian

Debian 8.5 ja disposa de la versió 10.0 i es pot instal·lar amb la instrucció `apt-get install mariadb-server`, però si es desitja una altra versió es pot accedir al repositori* i seguir les indicacions. Se selecciona la distribució, versió del SO i repositori, i s'executa:

***https://downloads.mariadb.org/mariadb/repositories/#mirror=tedeco&distro=Debian**

```
    Si tenim BD en MySQL, fer una còpia de seguretat parant els serveis primer i
    després abocant la BD:
    service apache2 stop; service mysql stop
    mysqldump -u root -p --all-databases > mysqlbackup.sql
Configurem el repositori:
    sudo apt-get install software-properties-common
    sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 0xcbcb082a1bb943db
    sudo add-apt-repository 'deb arch=amd64
    http://tedeco.fi.upm.es/mirror/mariadb/repo/10.1/debian jessie main'
Tingueu en compte que la comanda anterior ha de ser tot en un línia sinó donarà un error.
Després d'importar la clau, actualitzem el repositori i instal·lem:
    apt-get update
    apt-get install mariadb-server-10.1
Podrem mirar la versió amb: mysql  --version
    mysql  Ver 15.1 Distrib 10.1.0-MariaDB, for debian-linux-gnu (x86_64) using readline 5.2
O les bases de dades existents amb:
    mysql -u root -p -Be 'show databases'
    I si accedim a phpmyadmin veurem en "Home"
    Welcome to the MariaDB monitor.  Commands end with ; or \g.
    Your MariaDB connection id is 45
    Server version: 10.1.0-MariaDB-0+deb8u1 (Debian)
    Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.
    Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Veurem que si tenim instal·lat `mysql-server`, el desinstal·la i el reemplaça per `mariadb`, fent les mateixes preguntes de les bases de dades creades (usuari i *passwd*).

Es pot veure l'estat dels processos amb `systemctl status` i veure el servei corresponent:

```
||-mysql.service
  |||-654 /bin/bash /usr/bin/mysqld_safe
  |||-655 logger -p daemon.err -t /etc/init.d/mysql -i
  |||-797 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib/
  |||-798 logger -t mysqld -p daemon.error
```

Una de les característiques més interessant que presenta MariaDB és que permet una sèrie de ordres i interfícies que està més a prop de sistemes NoSQL que d'SQL. Per exemple, una d'elles són les columnes dinàmiques que permeten emmagatzemar diferents conjunts de columnes per a cada fila d'una taula. La seva funció consisteix en emmagatzemar un conjunt de columnes en unes columnes en una *blob* i disposa d'un conjunt de funcions per a manipular-la que poden ser utilitzades quan no és possible utilitzar columnes regulars. Un cas d'ús típic és quan es necessita emmagatzemar articles que poden tenir molts atributs diferents (com mida, color, pes, etc.), i el conjunt d'atributs possibles és molt gran i/o es coneix amb antelació. En aquest cas, els atributs es poden posar en columnes dinàmiques.

Per això creem una base de dades dins de mysql:

```
createdb test;
use test;
```

La taula ha de tenir una columna *blob* que serà utilitzada com a emmagatzematge per a les columnes dinàmiques:

```
create table assets (
item_name varchar(32) primary key, -- A common attribute for all items
dynamic_cols blob -- Dynamic columns will be stored here
);
```

Inserim una fila amb dues columnes dinàmiques: *color=blue, size=XL*:

```
INSERT INTO assets VALUES ('MariaDB T-shirt', COLUMN_CREATE('color', 'blue',
'size', 'XL'));
```

Inserim una altra fila amb dues columnes dinàmiques (una d'elles repeteix amb l'anterior), *color=black, price=500*:

```
INSERT INTO assets VALUES ('Thinkpad Laptop', COLUMN_CREATE('color', 'black',
'price', 500));
```

I obtenim la columna dinàmica *color* per a totes les files:

```
SELECT item_name, COLUMN_GET(dynamic_cols, 'color' as char) AS color FROM assets;
```

```
+-------------------+---------------+
| item_name         color     |
+-------------------+---------------+
| MariaDB T-shirt          blue      |
| Thinkpad Laptop          black     |
+-------------------+---------------+
```

És possible afegir i eliminar columnes dinàmiques d'una fila:

1) Eliminar:

```
UPDATE assets SET dynamic_cols=COLUMN_DELETE(dynamic_cols, "price")
WHERE COLUMN_GET(dynamic_cols, 'color' as char)='black';
```

2) Afegir:

```
UPDATE assets SET dynamic_cols=COLUMN_ADD(dynamic_cols, 'warranty', '3 years')
WHERE item_name='Thinkpad Laptop';
```

3) Llistar (i també a JSON format):

```
SELECT item_name, column_list(dynamic_cols) FROM assets;
```

```
+--------------------------+----------------------------------------+
| item_name                |   column_list(dynamic_cols)            |
+--------------------------+----------------------------------------+
| MariaDB T-shirt          |   `size`,`color`                       |
| Thinkpad Laptop          |   `color`,`warranty`                   |
+--------------------------+----------------------------------------+
```

```
SELECT item_name, COLUMN_JSON(dynamic_cols) FROM assets;
```

```
+--------------------------+----------------------------------------+
| item_name                | COLUMN_JSON(dynamic_cols)              |
+--------------------------+----------------------------------------+
| MariaDB T-shirt          | {"size":"XL","color":"blue"}           |
| Thinkpad Laptop          | {"color":"black","warranty":"3 years"} |
+--------------------------+----------------------------------------+
```

1.5. SQLite

Com ja vam dir en la introducció, SQLite és un sistema de gestió de bases de dades relacional compatible amb ACID, continguda en una biblioteca (escrita en C) sota llicència *Public Domain* (http://www.sqlite.org/copyright.html). Com a gran diferència dels sistemes de gestió de bases de dades client-servidor, el motor de SQLite no és un procés independent del programa que desitja portar a terme un accés a la base de dades, sinó una biblioteca que s'enllaça juntament amb el programa i passa a ser-ne part com qualsevol altra biblioteca (p. ex., igual que la */lib/x86_64-linux-gnu/libc.so.6* en Debian per a un programa

en C). El programa que necessita la funcionalitat de SQLite simplement fa la crida a subrutines i funcions tenint com a avantatge substancial la reducció de la latència d'accés a la base de dades ja que les crides a funcions són més eficients que la comunicació entre processos. El conjunt de la base de dades (definicions, taules, índexs i les pròpies dades) és guardat en un arxiu fitxer sobre la màquina *host* i la coherència en les transaccions s'obté bloquejant tot el fitxer al principi de cada transacció. La biblioteca implementa la major part de l'estàndard SQL-92, manté les característiques necessàries perquè una sèrie d'instruccions puguin ser considerades com una transacció (*ACID compliant*) i el seu disseny permet que diversos processos o *threads* puguin accedir a la mateixa base de dades sense problemes on els accessos de lectura poden ser servits en paral·lel i un accés d'escriptura només pot ser servit si no s'està fent cap altre accés de manera concurrent (o donarà un error o es podrà reintentar durant un cert *timeout* programable).[SQLL]

En Debian, s'inclou un paquet que implementa una interfície en línia d'ordres per a SQlite2/3 anomenat `sqlite` (versió 2) o `sqlite3` (que s'instal·la amb `apt-get install sqlite3`) per a accedir i modificar BD SQLite a més d'un conjunt de biblioteques que permeten la interacció entre el llenguatge i l'aplicació, per exemple, python-sqlite, ruby-sqlite, php5-sqlite com a interfícies per a SQLite des de phyton, ruby i php, respectivament. A més a més, hi ha altres eines per a dissenyar, crear o editar una base de dades compatible amb SQLite, com per exemple SQLite Database Browser (GPL) (que s'instal·la amb `apt-get install sqlitebrowser`), que permet de manera simple i visual accedir als fitxers que conté la base de dades [SQLDB]. Entre els principals usuaris de SQLite, trobem* Adobe (Photoshop Lightroom, AIR), Airbus (*flight software* A350), Apple (OSX, iPhone, iPod), Dropbox, Firefox, Thunderbird, Google (Android, Chrome), PHP, Python, Skype i Tcl/Tk, entre d'altres.

***http://www.sqlite.org/famous.html**

Per a instal·lar la biblioteca SQLite en Debian, fem

```
apt-get install libsqlite0-dev
```

(que inclou la biblioteca i els arxius *.h* per desenvolupar aplicacions en C, si només necessitem la biblioteca amb *libsqlite0* n'hi ha prou). Per a visualitzar per exemple una de les bases de dades de Firefox fem:

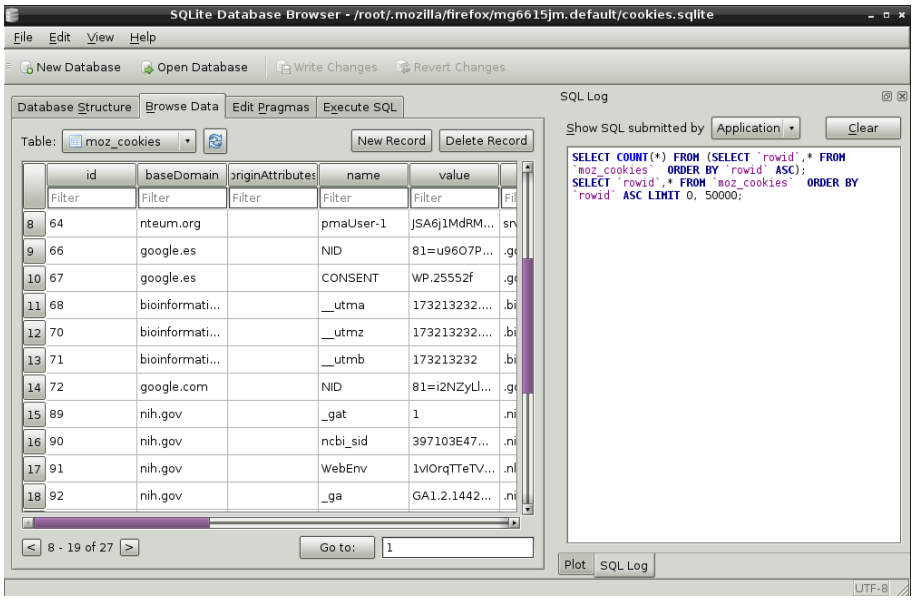
```
Busquem el nostre profile de firefox/iceweasel en l'usuari de login: cd .mozilla/firefox/*.default
Obrim, per exemple, la base de dades:: sqlite3 places.sqlite
       SQLite version 3.7.13 2012-06-11 02:05:22
       Enter ".help" for instructions. Enter SQL statements terminated with a ";"
Mirem la taules amb   .tables
       moz_anno_attributes  moz_favicons         moz_items_annos
       moz_annos            moz_historyvisits    moz_keywords
       moz_bookmarks        moz_hosts            moz_places
       moz_bookmarks_roots  moz_inputhistory
Mirem la taula moz_hosts:    SELECT * FROM moz_hosts;
       1|mozilla.org|140|0|
       2|sysdw.nteum.org|11180|1|
       3|127.0.0.1|2000|1|
       4|10.0.0.58|0|0|
Mirem la taula moz_bookmarks (abans fem un bookmark, per exemple a SQLite.org:
```

```
SELECT * FROM moz_bookmarks;
...
24|1|164|2|4|SQLite Home Page|||1404403846763242|1404403846777354|yfpoHwYrL7Ys
```

En l'adreça web http://www.sqlite.org/quickstart.html tindrem una guia ràpida de com accedir mitjançant les diferents interfícies de programació a la base de dades i la documentació per a treballar amb la línia d'ordres en l'adreça web http://www.sqlite.org/cli.html. La documentació la podem trobar en http://www.sqlite.org/docs.html.

La figura 4 mostra la visualització de l'ordre `sqlitebrowser` sobre la base de dades de les *cookies* del navegador Firefox.

Figura 4



1.6. MongoDB

MongoDB deriva d'*humongous*, que significa 'gigantesc'.

MongoDB [Mon] és un sistema de base de dades NoSQL (*open source*) orientat a documents. És un actor molt important dins de la família de sistemes de base de dades NoSQL. En aquests sistemes, en lloc de desar les dades en taules, tal com es fa en les bases de dades relacionals (tradicionals), les desa en documents amb un format similar a JSON, però amb un esquema dinàmic que es diu BSON. D'aquesta manera, permet que la integració de les dades en certes aplicacions sigui més fàcil i ràpida. Des dels seus inicis al 2007, ha evolucionat i avui dia es troba en producció en els principals actors d'Internet (Ebay, MetLife, ADP, BuzzFeed, Forbes, Bosch, etc.)

MongoDB és *open source* (AGPL) i existeixen *drivers* per als llenguatges de programació habituals (amb llicència Apache). També existeix una llicència comercial amb característiques avançades (integració amb SASL, LDAP, Kerbe-

ros10, SNMP, Rosette Linguistics PBT, o eines de gestió, monitorització i suport, així com suport en diversos plans).

Entre les principals característiques, destaquem que permet la cerca en camps, les consultes de rangs i les expressions regulars que poden retornar un camp específic del document o quelcom abstracte, com per exemple, una funció JavaScript definida per l'usuari. Qualsevol camp pot ser indexat (fins i tot es poden fer índexs secundaris), suporta replicació (de tipus primari-secundari) i pot escalar utilitzant un concepte de *sharding* (una mena de bloc definit per l'usuari); també es poden replicar i executar en diversos servidors. També permet ser utilitzat com un sistema d'arxius amb capacitat per fer un balanç de càrrega i una replicació en diversos servidors. A banda de les habituals funcions d'agregació (GROUP BY) i que està implementat com un *pipeline* eficient, MongoDB dóna suport a operacions de MapReduce per al processament per lots de dades i operacions d'agregació, i permet executar consultes utilitzant Javascript en la BD.

Els seus desenvolupaors critiquen certs aspectes de MongoDB, entre els quals es troben:

- no implementa les propietats ACID,
- poden existir problemes de consistència,
- bloqueig a nivell de document davant de cada operació d'escriptura (operacions d'escriptura concurrents entre diferents documents) i
- s'han detectat certs problemes d'escalabilitat, entre d'altres.

Per a la seva instal·lació, fem simplement `apt-get install mongodb-server` i després modifiquem l'arxiu */etc/mongodb.conf* agregant *smallfiles = true* per reduir, durant les proves, les necessitats d'espai, limitant-lo a 512 MBytes, i la grandària d'arxiu *Journal* d'1 GB a 128 MB. Amb això podrem veure el procés */usr/bin/mongod –config /etc/mongodb.conf* en execució. Per als usuaris experts en SQL existeix una certa equivalència, bàsicament entre les operacions d'agregació amb SQL, que es poden trobar a [MonAgre].

Existeixen molts entorns gràfics amb diferents objectius i prestacions, com es pot veure a [MonGUI]. Vegem-ne alguns d'ells:

1) UMongo*. És dels més simples (sense grans complexitats per a la seva instal·lació), sobretot per a principiants. Simplement s'ha de descarregar de la pàgina web, descomprimir-lo, després, executar (requereix tenir java instal·lat) l'ordre `launch-umongo.sh` i, des del menú, connectar el servidor MongoDB. A la figura 5 podeu veure una imatge de l'entorn de UMongo.

*http://edgytech.com/umongo/

2) RoboMongo**. Tan sols és necessari descomprimir i executar la instrucció `robomongo` des del directori *bin* de la instal·lació. Aquest programa és molt indicat per conèixer la potència del Shell de Mongo, ja que permet anar veient quines ordres es van executant i quina és la seva sintaxi. A la figura 6 podeu veure una imatge de l'entorn de RoboMongo.

**https://robomongo.org/download

Figura 5

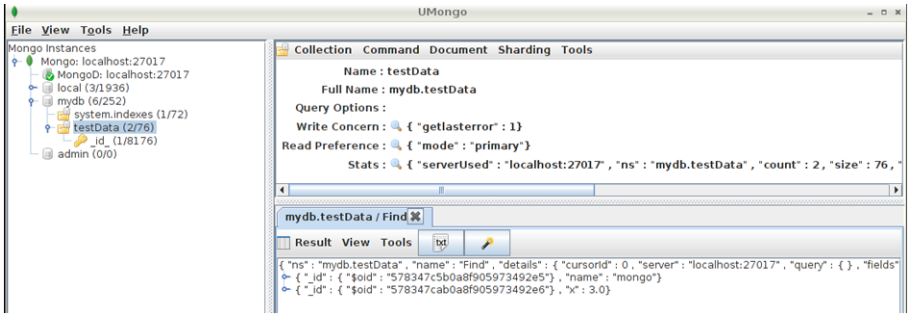
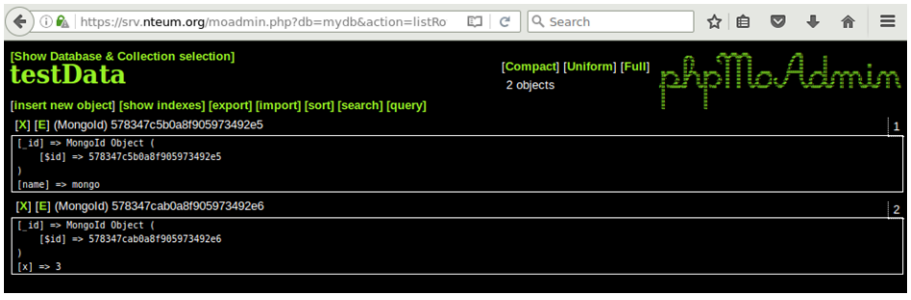


Figura 6

3) phpMoAdmin*. Té una interfície senzilla però molt interessant. Després de descomprimir, de moure l'arxiu al directori web (fent servir la instrucció `mv moadmin.php /var/www/html`) i de carregar el driver de Mongo per php (amb l'ordre `apt-get install php5-mongo`), podrem posar l'adreça URL al navegador i connectar-nos a la base de dades i interactuar amb ella (http://srv.nteum.org/moadmin.php, en el nostre cas). A la figura 7 podeu veure una imatge de l'entorn de phpMoAdmin.

*http://www.phpmoadmin.com/

Figura 7



4) MongoBird**. Una altra eina interessant és MongoBird, que permet el monitoratge de MongoDB generant estadístiques i un panell de control amb tota l'activitat de la base de dades, i una administració eficient i escalable de la base de dades que analitza els principals paràmetres d'interès.

**http://mongobird.citsoft.ne

A continuació, es realitzarà un petit exemple sobre el client de mongo (`mongo`) per veure alguns dels elements principals de MongoDB:

```
root@srv# mongo

MongoDB shell version: 2.4.10
connecting to: test


> show dbs          miro les DB

admin (empty)
local 0.03125GB
mydb 0.0625GB


> use mydb          genero/utilitzo la DB

switched to db mydb


> db.workers.insert({Nom: 'Joan', Cognom: 'SenseNom', edat: 40});
```

inserit

```
> show collections          miro les col·leccions de la DB

system.indexes
workers


> db.workers.find()          miro les dades de la col·lecció de la DB

{"_id" : ObjectId("578365a66da65899d9603db9"), "Nom" : "Joan",
"Cognom" : "SenseNom", "edat" : 40}


> db.workers.insert({Nom: 'Pere', Cognom: 'SenseNom', edat: 41});


> db.workers.insert({Nom: 'Horaci', Cognom: 'ElmeuNom', edat: 42});


> db.workers.insert({Nom: 'Àlvar', Cognom: 'ElteuNom', edat: 43});


> db.workers.find()

{ "_id" : ObjectId("578365a66da65899d9603db9"), "Nom" : "Joan",
"Cognom" : "SenseNom", "edat" : 40 }
{ "_id" : ObjectId("57836656f4a8ec103eed440a"), "Nom" : "Pere",
"Cognom" : "AmbNom", "edat" : 41 }
{ "_id" : ObjectId("5783666af4a8ec103eed440b"), "Nom" : "Horaci",
"Cognom" : "ElmeuNom", "edat" : 42 }
{ "_id" : ObjectId("57836681f4a8ec103eed440c"), "Nom" : "Àlvar",
"Cognom" : "TuNombre", "edat" : 43 }


> db.workers.find({Nom: 'Carles'}); busco en les col·leccions de la DB


> db.workers.find({Nom: 'Horaci'});

{ "_id" : ObjectId("5783666af4a8ec103eed440b"), "Nom" : "Horaci",
"Cognom" : "ElmeuNom", "edat" : 42 }


> db.workers.insert({Nom: 'Julio', Cognom: 'ElNom', edat: 42,
creat: new Date(2016, 7, 11)});
```

```
> db.workers.find({edat: {$gt: 41} } );
```
busco amb restriccions en les col·leccions de la DB

```
{ "_id" : ObjectId("5783666af4a8ec103eed440b"), "Nom" : "Horaci",
"Cognom" : "ElmeuNom", "edat" : 42 }
{ "_id" : ObjectId("57836681f4a8ec103eed440c"), "Nom" : "Àlvar",
"Cognom" : "ElteuNom", "edat" : 43 }
{ "_id" : ObjectId("5783672df4a8ec103eed440d"), "Nom" : "Julio",
"Cognom" : "ElNom", "edat" : 42, "creat" : ISODate("2016-08-10T22:00:00Z") }
```

```
> db.workers.find({$or: [{edad: 42}, {edad: 43}]} );
```

```
{ "_id" : ObjectId("5783666af4a8ec103eed440b"), "Nom" : "Horaci",
"Cognom" : "ElmeuNom", "edat" : 42 }
{ "_id" : ObjectId("57836681f4a8ec103eed440c"), "Nom" : "Àlvar",
"Cognom" : "ElteuNom", "edat" : 43 }
{ "_id" : ObjectId("5783672df4a8ec103eed440d"), "Nom" : "Julio",
"Cognom" : "ElNom", "edat" : 42, "creat" : ISODate("2016-08-10T22:00:00Z") }
```

```
> db.workers.find({creado: {$exists: false}})
```

```
{ "_id" : ObjectId("578365a66da65899d9603db9"), "Nom" : "Joan",
"Cognom" : "SenseNom", "edat" : 40 }
{ "_id" : ObjectId("57836656f4a8ec103eed440a"), "Nom" : "Pere",
"Cognom" : "AmbNom", "edat" : 41 }
{ "_id" : ObjectId("5783666af4a8ec103eed440b"), "Nom" : "Horaci",
"Cognom" : "ElmeuNom", "edat" : 42 }
{ "_id" : ObjectId("57836681f4a8ec103eed440c"), "Nom" : "Àlvar",
"Cognom" : "ElteuNom", "edat" : 43 }
```

```
> db.workers.find({Nom: 'Julio'}).pretty();
```
format agradable

```
{
        "_id" : ObjectId("5783672df4a8ec103eed440d"),
        "Nom" : "Julio",
        "Cognom" : "ElNom",
        "edat" : 42,
        "creat" : ISODate("2016-08-10T22:00:00Z")
}
```

```
> db.workers.count()
```
compto en les col·leccions de la DB

```
  5
```

```
> db.workers.update( {Nom: "Joan"}, {$set: {Experiència: 10}})
```
modifico un document

```
> db.workers.find()
```

```
...
{ "_id" : ObjectId("578365a66da65899d9603db9"), "edat" : 40, "Experiència" : 10,
"Nom" : "Joan", "Cognom" : "SenseNom" }
```

```
> db.workers.update({Nom: 'Joan'}, {$inc: {Experiència: 1}}, false, true)
```

```
> db.workers.find()
```

```
...
```

```
{ "_id" : ObjectId("578365a66da65899d9603db9"), "edat" : 40, "Experiència" : 11,
"Nom" : "Joan", "Cognom" : "SensNom" }
```

`> db.workers.stats()` *Genero estadístiques*

```
{
        "ns" : "mydb.workers",
        "count" : 5,
        "size" : 480,
        "avgObjSize" : 96,
        "storageSize" : 4096,
        "numExtents" : 1,
        "nindexes" : 1,
        "lastExtentSize" : 4096,
        "paddingFactor" : 1,
        "systemFlags" : 1,
        "userFlags" : 0,
        "totalIndexSize" : 8176,
        "indexSizes" : {
                "_id_" : 8176
        },
        "ok" : 1
}
```

`mongodump -db mydb --collection workers --out backup` *Genero un suport de la DB*

```
connected to: 127.0.0.1
Mon Jul 11 11:55:21.717 DATABASE: mydb    to      backup/mydb
Mon Jul 11 11:55:21.718         mydb.workers to backup/mydb/workers.bson
Mon Jul 11 11:55:21.718                 5 objects
Mon Jul 11 11:55:21.718         Metadata for mydb.workers to backup/mydb/workers.metadata.json
```

`mongoexport --db mydb -collection workers --csv -fields \`

`Nom,Cognom,Experiència` *exporto dades en csv*

```
connected to: 127.0.0.1
Nom,Cognom,Experiència
"Joan","SenseNom",11.0
"Pere","AmbNom",
"Horaci","ElmeuNom",
"Àlvar","ElteuNom",
"Julio","ElNom",
exported 5 records
```

`mongoexport --db mydb -collection workers -fields \`

`Nom,Cognom,Experiència` *exporto dades en JSON*

```
connected to: 127.0.0.1
{ "_id" : { "$oid" : "578365a66da65899d9603db9" }, "Experiència" : 11,
"Nom" : "Joan", "Cognom" : "SensNom" }
{ "_id" : { "$oid" : "57836656f4a8ec103eed440a" }, "Nom" : "Pere",
"Cognom" : "AmbNom" }
{ "_id" : { "$oid" : "5783666af4a8ec103eed440b" }, "Nom" : "Horaci",
"Cognom" : "ElmeuNom" }
{ "_id" : { "$oid" : "57836681f4a8ec103eed440c" }, "Nom" : "Àlvar",
"Cognom" : "ElteuNom" }
{ "_id" : { "$oid" : "5783672df4a8ec103eed440d" }, "Nom" : "Julio",
"Cognom" : "ElNom" }
exported 5 records
```

Per continuar veient la potencialitat de MongoDB és aconsellable seguir l'exemple del tutorial de la documentació*, on es comença important una llista de més de 25.000 documents en format JSON i, després, es pot buscar i treballar (inserir, buscar, actualitzar, etc.), o veure les estadístiques amb diferents ordres a través del client `mongo`, per exemple:

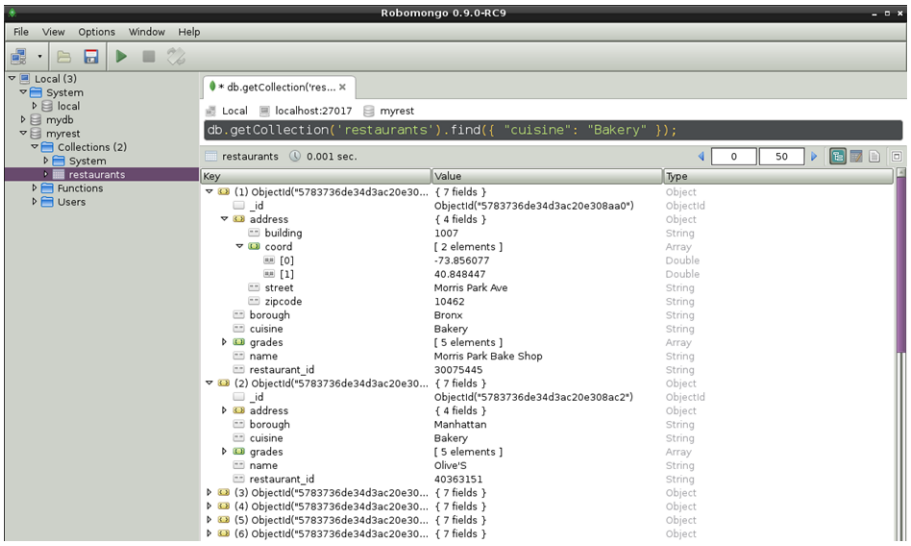
***https://docs.mongodb.com getting-started/shell /import-data/**

```
> db.restaurants.stats()

{
        "ns" : "myrest.restaurants",
        "count" : 25359,
        "size" : 11123920,
        "avgObjSize" : 438.65767577585865,
        "storageSize" : 22499328,
        "numExtents" : 6,
        "nindexes" : 1,
        "lastExtentSize" : 11325440,
        "paddingFactor" : 1,
        "systemFlags" : 1,
        "userFlags" : 0,
        "totalIndexSize" : 833952,
        "indexSizes" : {
                "_id_" : 833952
        },
        "ok" : 1
}
```

La figura 8 mostra una sortida de RoboMongo amb una cerca personalitzada per un camp, els paràmetres utilitzats i el temps emprat en la consulta dels 25.359 documents amb una coincidència de 691 documents.

Figura 8



1.7. Source Code Control System

Els sistemes de control de revisions (o també coneguts com a *control de versions/codi font*) es refereixen a la gestió dels canvis en els arxius, programes,

pàgines o qualsevol informació que és modificada i actualitzada, ja sigui per un únic usuari per a portar un control del que ha fet i quan o per múltiples usuaris que treballen de manera simultània en el mateix projecte. Els canvis s'identifiquen generalment per un nombre o codi, denominat *nombre de revisió, nivell de revisió, revisió* i el sistema de control permet tornar enrere, acceptar un canvi, publicar-lo o saber qui l'ha fet i quin ha estat. El sistema treballa amb marques de temps, repositoris (per a guardar les diferents versions) i usuaris i permetrà, en funció d'aquests paràmetres, comparar els repositoris/arxius, restaurar-los i/o fusionar-los. En l'actualitat, és una tècnica aplicada en grans aplicacions informàtiques (com, per exemple, en Google Docs o Wikipedia -o en qualsevol wiki-), però hi ha diferents aplicacions que poden treballar com a servidors o aplicacions internes o en mode client servidor en funció de les necessitats del projecte/usuaris com per exemple CVS (uns dels inicials juntament amb RCS), Subversion (molt estès), Git (molt utilitzat en projectes *Open Source*) o Mercurial (molt utilitzat pel seu aspecte de distribuït –no és necessari un servidor).

1.7.1. Concurrent Versions System (CVS)

El ***Concurrent Versions System*** (CVS) és un sistema de control de versions que permet mantenir versions antigues d'arxius (generalment codi font), guardant un registre (*log*) de qui, quan i per què van ser efectuats 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 gestionar versions de programari i controla l'edició concurrent d'arxius font per múltiples autors. El CVS utilitza internament un altre paquet anomenat **RCS** (*Revision Control System*) com una capa de baix nivell. Si bé el RCS pot ser utilitzat de manera independent, això no s'aconsella, ja que CVS, a més de la pròpia funcionalitat, presenta totes les prestacions de RCS, però amb notables millores quant a l'estabilitat, el funcionament i el manteniment. Entre aquestes, cal destacar: funcionament descentralitzat (cada usuari pot tenir el seu propi arbre de codi), edició concurrent, comportament adaptable mitjançant *shell scripts*, etc.

En primer lloc, s'ha d'instal·lar el Concurrent Versions System (CVS) des de la distribució i haurem d'instal·lar també OpenSSH, si es vol utilitzar conjuntament 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=/usr/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 repositori i configurar els permisos; com *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 repositori i posar-hi arxiu de codi:

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

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

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

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

```
export EDITOR=/usr/bin/vi
export CVSROOT=/usr/local/cvsroot
export CVSREAD=yes
cd directori_d'originals
cvs import NomDelRepositori vendor_1_0 rev_1_0
```

El nom del repositori pot ser un identificador únic o també hi ha la possibilitat que sigui `usuari/projecte/xxxx` si és que l'usuari desitja tenir organitzats els seus repositoris.

Això crearà un arbre de directoris en `CVSROOT` amb aquesta estructura i afegeix un directori (`/usr/local/cvsroot/NomDelRepositori`) en el repositori amb els arxius que a partir d'aquest moment estaran en el repositori. Una prova per a saber si s'ha emmagatzemat tot correctament és emmagatzemar una còpia en el repositori, crear després una còpia des d'allà i comprovar les diferències. Per exemple, si els originals estan en el `directori_de l'usuari/dir_org` i es desitja crear un repositori com `primer_cvs/proj`, s'hauran d'executar les següents ordres:

```
cd dir_org    Canviar al directori del codi font original.
cvs import -m "Fonts originals" primer_cvs/proj usuariX vers0
```

Crea el repositori en `primer_cvs/proj` amb `usuariX` i `vers0`.

```
cd..    Canviar al directori superior de dir_org.
cvs checkout primer_cvs/proj
```

Generar una còpia del repositori. La variable `CVSROOT` ha d'estar inicialitzada, en cas contrari, s'haurà d'indicar tot el camí.

```
diff -r dir_org primer_cvs/proj
```

Mostra les diferències entre l'un i l'altre. No n'hi haurà d'haver cap excepte pel directori `primer_cvs/proj/CVS` que ha creat el CVS.

```
rm -r dir_org
```

Esborra els originals (feu sempre una còpia de seguretat per motius de seguretat i per tenir una referència d'on es va iniciar el treball amb el CVS).

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

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

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

El fet d'esborrar els originals no sempre és una bona idea, excepte en aquest cas, després que s'hagi verificat que estan en el repositori, perquè no s'hi treballi per distracció i perquè els canvis no quedin reflectits sobre el CVS. Sobre màquines on 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 està CVS. CVS ofereix una sèrie d'ordres (s'obtenen amb `cvs comd opcions`...) per a treballar amb el sistema de revisions, entre aquestes: `add`, `checkout`, `update`, `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 estaran 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 repositori.

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

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

`cvs remove file` s'utilitzarà per a esborrar arxius del repositori (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 repositori.

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

`cvs tag -R "versión"` es pot usar per a introduir un nombre 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 en una línia de treball separada denominada *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 del directori de treball, la qual cosa assignarà a la branca el nom de `rel-1-0-patches`. La unió de branques amb el directori de treball significa usar `cvs update -j`. Consulteu les referències per a barrejar diferents branques o accedir-hi.

Exemple d'una sessió

Seguint l'exemple de la documentació donada en les referències, es mostrarà una sessió de treball (en forma general) amb CVS. Com que CVS emmagatzema tots els arxius en un repositori 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 repositori és inicialitzat a `gccrep`. En primer lloc, s'ha d'obtenir una còpia dels arxius del repositori a 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 és necessari 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 en `a.c` (o en els arxius que sigui necessari), és el moment de fer una nova versió emmagatzemant `a.c` (o tots els que s'han tocat) en el repositori i fer aquesta versió disponible per a la resta d'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 a aquesta versió i després poder fer un històric.

Si es decideix eliminar els arxius (perquè ja es va acabar el projecte o perquè no s'hi treballarà), una forma de fer-ho és en un nivell 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 repositori i, si n'hi ha i s'esborra, preguntarà si es desitja 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 del directori de treball. S'utilitzarà l'ordre del sistema operatiu `diff` per a comparar la versió `b.c` amb la versió que es té en el repositori (sempre cal recordar fer un `cvs commit b.c` si es desitja que aquestes diferències es transfereixin al repositori 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 hi haurà ocasions en les quals més d'un usuari vulgui editar el mateix fitxer de manera simultània. Una possible solució és bloquejar el fitxer o utilitzar punts de verificació reservats (*reserved checkouts*), la qual cosa només permetrà a un usuari editar el mateix fitxer simultàniament. Per a això, s'haurà d'executar l'ordre `cvs admin -l command` (consulteu `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 d'ells que transfereixi els seus canvis al repositori ho podrà fer sense problemes, però els restants rebran un missatge d'error quan desitgin 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 repositori i després actualitzar el repositori amb els seus propis canvis. Consulteu les referències per a veure un exemple d'aplicació i altres formes de treball concurrent amb comunicació entre usuaris. Com a interfícies gràfiques es poden consultar tkcvs* [Tkc] desenvolupada en Tcl/Tk i que suporta Subversion o la també molt popular Cervisia [Cerc] (les dues en Debian).

***http://www.twobarleycorns.net/tkcvs.html**

1.7.2. Subversion

Com a idea inicial, **Subversion** serveix per a gestionar un conjunt d'arxius (repositori) i les seves diferents versions. És interessant remarcar que no ens importa com es guarden, sinó com s'accedeix a aquests fitxers i que és comú utilitzar una base de dades. El repositori 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 agregar-ne una de nova. A diferència de CVS, Subversion fa les revisions globals del repositori, la qual cosa significa que un canvi en el fitxer no genera un salt de versió únicament en aquest fitxer, sinó en tot el repositori, el qual en suma un a la revisió. En Debian, haurem de fer `apt-get install subversion subversion-tools`, i si desitgem publicar els repositoris en Apache2 haurem de tenir instal·lat aquest, a més del mòdul específic `apt-get install libapache2-svn`. A més del paquet Subversion, hi ha en els repositoris (de la majoria de distribucions) paquets complementaris, com per exemple, `subversion-tools` (eines complementàries), `svn-load` (versió millorada per a la importació de repositoris),

Enllaç d'interès

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

`svn-workbench` (*Workbench* per a Subversion) i svnmailer (eina que estén les possibilitats de les notificacions del *commit*).

Primer pas: crear el nostre repositori, usuari (considerem que l'usuari és *svuser*), grup (*svgroup*) com a *root* fer:

```
mkdir -p /usr/local/svn
adduser svuser
addgroup svgroup
chown -R root.svgroup /usr/local/svn
chmod 2775 /usr/local/svn
addgroup svuser svggroup Afegim l'usuari svuser al grup svgroup
```

Ens connectem com *svuser* i verifiquem que estem en el grup *svgroup* (amb l'ordre `id`).

```
svnadmin create /usr/local/svn/proves
```

Aquesta ordre crearà un 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
svnadmin create $HOME/svndir/proves
```

Considerant que el repositori el tenim en */usr/local/svn/proves/*, a continuació crearem un directori temporal en el nostre directori:

```
mkdir -p $HOME/svntmp/proves
```

Ens passem al directori `cd $HOME/svntmp/proves` i crearem un arxiu, per exemple:

```
echo Primer Arxiu Svn ` date ` > file1.txt
```

El traslladem al repositori fent dins del directori:

```
svn import file:///usr/local/svn/proves/ -m "Ver. Inicial"
```

Si l'hem creat en `$HOME/svndir/proves`, l'hauríem de reemplaçar per *file:///$HOME/svndir/proves*. 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 repositori i és recomanable esborrar-la per a no tenir la temptació o cometre l'error de treballar amb aquesta en comptes de fer-ho amb el repositori (per fer-ho fem servir `rm -rf $HOME/svntmp/proves`).

Una vegada en el repositori, es pot obtenir la còpia local on podrem treballar i després pujar les còpies al repositori. Per a això fem:

```
mkdir $HOME/svn-work; cd $HOME/svn-work
svn checkout file:///usr/local/svndir/proves
```

on 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/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 s'ha d'indicar el camí (*path*). `svn add` marca per a afegir el fitxer al repositori i realment s'afegeix quan fem un `svn commit`.

Ens donarà alguns missatges que ens indicaran 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 el del repositori. Per a fer-ho, podem agregar 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 repositori. Si ho carreguem amb `svn commit -m "Nova línia2"` (que 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 al mateix temps i cadascun

ha fet una còpia local del repositori 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 en el repositori és posterior a la còpia original d'aquest usuari (és a dir, hi ha hagut canvis entremig), amb la qual cosa si el segon usuari fa el `commit`, perdem les modificacions del primer. Per a això haurem de fer un `svn update`, que ens indicarà l'arxiu en conflicte i en l'arxiu en conflicte ens indicarà quin és l'arxiu i les seves 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 seves corresponents versions.

Enllaç d'interès

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

Un aspecte interessant és que Subversion pot funcionar conjuntament amb Apache2 (i també sobre SSL) per a accedir des d'una altra màquina o simplement mirar el repositori. La configuració d'Apache2 i SSL es va indicar en la part de servidors. Per a configurar-los, és necessari activar els mòduls de WebDAV (`a2enmod dav dav_fs`) i instal·lar la biblioteca com s'ha especificat abans amb `apt-get install libapache2-svn` verificant que el mòdul dav_svn estigui habilitat també (ho farà la instal·lació de la biblioteca). A continuació, crearem un arxiu en */etc/apache2/conf.d/svn.conf* amb el següent contingut:

```
<location "/svn">
       DAV svn
       SVNParentPath /usr/local/svn
       SVNListParentPath On
   AuthType Basic
       AuthName "Subversion"
       AuthUserFile /etc/apache2/htpasswd
       Require valid-user
</location>
```

On hem d'apuntar cap a on es troba el nostre repositori amb validació d'usuari (ha de ser un usuari vàlid creat amb `htpasswd /etc/apache2/htpasswd user`), reiniciem Apache2 i ja tindrem aquest habilitat per a accedir mitjançant l'URL http://sysdw.nteum.org/svn/. Una altra forma d'accedir és a través d'un client en forma local o en forma remota, que és una opció útil per a repositoris petits i mitjans. Debian n'inclou una sèrie (alguns compatibles amb CVS) com per exemple **rapidsvn**, **subcommander**, **svnkit** (en Java), **tkcvs**, **viewvc** (aquests dos últims suporten també repositoris CVS) i **websvn** (en PHP). Si es desitja treballar en remot, s'haurà d'engegar el servidor de svn amb `svnserve -d` i canviar l'arxiu *svnserve.conf* per a permetre l'accés i el mode (p. ex., traient el comentari en la línia `anon-access = read`). Aquest arxiu es troba en el directori */Dir_Repo/conf*, que és on hem inicialitzat el repositori (*/usr/local/svn/proves/conf/svnserve.conf* en el nostre cas).

Si desitgem treballar amb un repositori gestionat via URL, el més pràctic és crear-ne un i gestionar-lo des de la consola per a inserir els arxius i via web per a posar-los a la disposició del públic. En aquest cas, haurem de fer algunes modificacions amb els permisos perquè puguem treballar amb URL. Crearem un

segon repositori per a aquest exemple, però farem que l'usuari **svuser** generi les seves còpies dins del repositori:

Com a *root*, fem:

```
  mkdir /var/svn El nostre nou repositori
  svnadmin create /var/svn Creem el repositori
  chown -R www-data:www-data Perquè Apache pugui accedir al directori
  ls -s /var/svn
-rw-r-r-   1 www-data www-data   229 Jul 4 20:27 README.txt
drwxr-xr-x   2 www-data www-data   4096 Jul 4 20:27 conf
drwxr-xr-x   2 www-data www-data   4096 Jul 4 20:27 dav
drwxr-xr-x   2 www-data www-data   4096 Jul 4 20:28 db
-rw-r-r-   1 www-data www-data   2 Jul 4 20:27 format
drwxr-xr-x   2 www-data www-data   4096 Jul 4 20:27 hooks
drwxr-xr-x   2 www-data www-data   4096 Jul 4 20:27 locks
```

Per a l'autenticació, s'usa `htpasswd` (per exemple, amb la instrucció

```
htpasswd -c -m /etc/apatxe2/htpasswd user
```

on el paràmetre *-c* només s'ha de posar la primera vegada quan s'ha de crear l'arxiu). Després, crearem un arxiu en */etc/apache2/conf.d/svn2.conf* per a alguna cosa com:

```
<location /svn2>
   DAV svn
   SVNPath /var/svn
   AuthType Basic
   AuthName "Subversion Repository"
   AuthUserFile /etc/apache2/htpasswd
   Require valid-user
</location>
```

Reiniciem Apache i ja estem llestos per a importar alguns arxius, per exemple des de la consola com svuser:

```
svn import file1.txt http://sysdw.nteum.org/svn/file1.txt -m " Import Inicial"
```

Ens demanarà l'autenticació i ens dirà que el fitxer `file1.txt` s'ha afegit al repositori. Des de l'URL http://sysdw.nteum.org/svn2/ veurem el file1.txt.

1.7.3. 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 elevat nombre arxius 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 trobem:

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 que cada programador tingui una còpia local de l'historial del desenvolupament sencer i que els canvis es propaguin entre els repositoris locals.

3) Els repositoris poden publicar-se per HTTP, FTP, rsync o mitjançant un protocol natiu, mitjançant una connexió TCP/IP simple o a través de xifratge SSH, i permet emular servidors CVS per a interactuar amb clients CVS preexistents.

4) Els repositoris Subversion i svk es poden fer servir directament amb git-svn.

Instal·lació d'un repositori de Git, Gitolite i Gitweb

Instal·larem un servidor de control de versions Git sobre Debian perquè pugui gestionar els arxius de codi font d'un equip de desenvolupament. Per a això, utilitzarem Gitolite (com a mètode de control d'accés dels nostres programadors als repositoris de programari), que és una capa d'autenticació a Git basat en ssh i que permet especificar permisos no només per repositori, sinó també per branca o etiquetes dins de cada repositori. Finalment, instal·larem Gitweb per a l'accés als repositoris i adaptarem una interfície basada en http://kogakure.github.io/gitweb-theme/. [git, len]

Sobre el servidor, instal·lem els paquets: `apt-get install git-core git-doc gitolite3 git-daemon-run gitweb highlight`
Crearem un usuari per al servei git: `adduser git`
Adoptem la seva identitat i configurem:

```
su - git
git config --global user.name "git"
git config --global user.email git@sysdw.nteum.org
```

Es pot verificar amb `git config -l`
S'han de generar les claus públiques dels usuaris i enviar al servidor, per la qual cosa podem fer com a usuari adminp:

```
ssh-keygen
scp .ssh/id_rsa.pub git@sysdw.nteum.org:/tmp/adminp.pub
```

Si l'usuari adminp està en el mateix servidor utilitza `cp` en lloc de `scp`, però se suposa que és una altra màquina.
En el servidor, activem la clau d'adminp (i la de tots els usuaris), i com a usuari git fem

```
gitolite setup -pk /tmp/adminp.pub
```

A continuació, canviarem els permisos:

```
chmod g+r /home/git/projects.list
chmod -R 770 /home/git/repositories
```

Editem */etc/group* i afegim l'usuari www-data al grup git modificant la línia a git:x:1001:www-data.
En la màquina local i com a usuari adminp (també es pot fer en la mateixa màquina del servidor en el compte de l'usuari adminp), carregarem la configuració de gitolite i agregarem dos repositoris:

```
git clone git@sysdw.nteum.org:gitolite-admin.git
cd gitolite-admin
```

Editem l'arxiu *conf/gitolite.conf* amb els repositoris:

```
repo    gitolite-admin
RW+     =   adminp

repo    testing
    RW+     =   @all

repo wiki
    RW+ = @all
    R = daemon
    R = gitweb

repo data
    RW+ = adminp remix
    R = daemon
    R = gitweb
```

Després agreguem en el repositori: `git add *`
I fem el commit: `git commit -m "Repositoris wiki i data amb permís d'escriptura per a admin i remix"`
El pugem: `git push origin master`

Per a la configuració de gitweb, editem */etc/gitweb.conf* modificant les línies indicades:

```
$projectroot = "/home/git/repositories";
$projects_list = "/home/git/projects.list";
```

Modifiquem /etc/sv/git-daemon/run (abans hem de fer una còpia com run.org, per exemple)

```
#!/bin/sh
exec 2>&1
echo 'git-daemon starting.'
exec chpst -ugitdaemon \
"$(git --exec-path)"/git-daemon --verbose --reuseaddr \
    --base-path=/home/git/repositories /home/git
```

Després reiniciem el *daemon*: `sv restart git-daemon`

També reiniciem Apache2 i en l'URL http://sysdw.nteum.org/gitweb/ tindrem els projectes.

Per a canviar l'aspecte de la pàgina web amb http://kogakure.github.io/gitweb-theme/. Podem descarregar l'arxiu tgz d'aquesta pàgina, el descomprimim i des de dins del directori copiem el contingut

`cp * /usr/share/gitweb/static/`

Si recarreguem el directori esborrant la *cache* (Crtl+F5) en el navegador, hauríem de veure el nou aspecte. Un aspecte interessant és veure ara com des de l'usuari admin inserim arxius en un dels repositoris i els publiquem:

Creem un directori hello amb dos arxius, un arxiu anomenat *hello.c* i un altre, *library.h*

```
 /* Hello.c */
#include <stdio.h>
#include "library.h"
    int main (void) {printf ("Hola UOC!\n");}
/* library.h */
#ifndef DEFINITIONS_H
#define DEFINITIONS_H 1
#endif /* DEFINITIONS_H */
```

Dins del directori, executem:

```
git init
git add .
git commit -m "initial commit"
git remote add origin git@sysdw.nteum.org:wiki.git
git push origin master
```

Si actualitzem la pàgina web, veurem que ja tenim els arxius en el repositori. Si volguéssim clonar el repositori, només hauríem de fer:

```
git clone git@sysdw.nteum.org:wiki.git
```

I on estem, crearà un directori wiki amb els arxius *hello.c library.h* (a més d'un altre .git que és on es troba tota la informació de Git).

Integració entre Git i GitHub

És interessant tenir, quan treballem amb repositoris, una còpia en GitHub (https://github.com/). Per fer això, primer generarem un compte i crearem un repositori (públic) per exemple, amb el nom *hello*. Després creem el repositori local fent:

```
mkdir myhello
cd myhello
vi README
git init

Initialized empty Git repository in /home/adminp/myhello/.git/


git add README
git commit -m "first commit"

[master (adminp-commit) 666b682] first commit
Committer: adminp <adminp@srv.nteum.org>
1 file changed, 4 insertions(+)
create mode 100644 README


git remote add origin https://github.com/usuario/hello.git
```

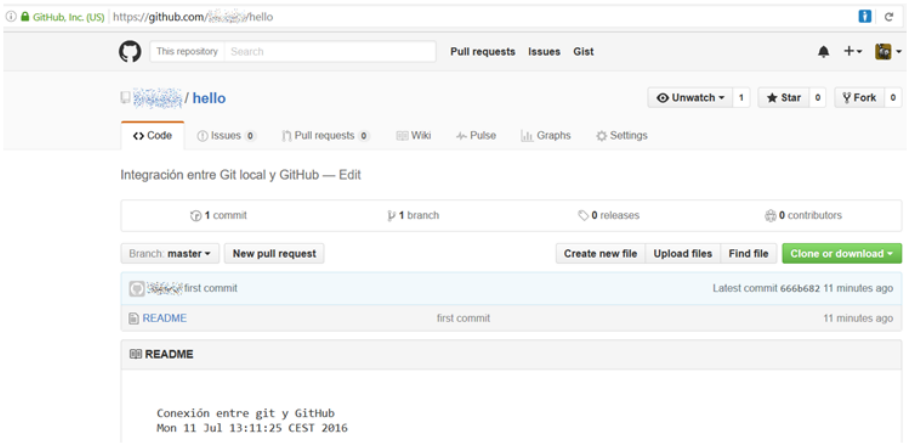
reemplaçar USUARI amb l'usuari de GitHub

```
git push -u origin master

Username for 'https://github.com': USUARI
Password for 'https://USUARI@github.com':
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 266 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/USUARI/hello.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

La figura 9 mostra el resultat de la creació del repositori remot i les possibilitats que s'obren treballant amb ell, entre diferents usuaris, clonant el repositori inicial i actualitzant-lo o creant *Branchs* per continuar amb el desenvolupament. Més informació a [GIT-SRV].

Figura 9



1.7.4. Mercurial

Mercurial és un sistema de control de versions (escrit en la seva major part en Python i algunes parts en C -diff-) dissenyat per a facilitar la gestió d'arxius/directoris als desenvolupadors de programari. L'ús bàsic de Mercurial és en línia d'ordres i existeix una ordre que és la que gestiona tot el paquet mitjançant opcions anomenada `hg` (en referència al símbol químic del mercuri). Les premisses de disseny de mercurial són el rendiment i l'escalabilitat mitjançant un desenvolupament distribuït i sense necessitat d'un servidor. A més, presenta una gestió robusta d'arxius tant de text com binaris i posseeix capacitats avançades de ramificació i integració i una interfície web integrades.[mer4]

La seva instal·lació és molt simple amb `apt-get mercurial` i és interessant instal·lar un altre paquet anomenat Tortoisehg [mer2] (`apt-get install tortoisehg`), que és una extensió gràfica a Mercurial i actua com *Workbench*, suportant múltiples repositoris, i permet resoldre conflictes en una forma simple i gràfica. Després de la instal·lació, podrem executar l'ordre `hg version` i `hg debuginstall` on aquesta última ens dirà que hem d'identificar l'usuari i incloure una adreça de correu. Per a això, crearem l'arxiu more $HOME/.hgrc amb el següent contingut (la segona part és per a Tortoise):

```
[ui]
username = AdminP <adminp@sysdw.nteum.org>
[extensions]
graphlog =
```

Amb això, ja estem en condicions de començar a treballar en el nostre repositori. Per a usuaris recentment iniciats, és molt recomanable seguir la guia

de [mer1], que mostra el procediment per a gestionar un repositori entre dos usuaris. Una seqüència d'ordres podria ser:

Creem un repositori: `hg init hello` veurem un directori amb un subdirectori *.hg*
Dins de hello creem un arxiu: `echo "Hello World" > hello.txt`
Mirem l'estat però veurem que no forma part de repositori: `hg status`
L'afegim i executem l'ordre anterior. Veurem que ja està dins: `hg add hello.txt`
Per a publicar-ho en el repositori: `hg commit -m "El meu primer hg-hello"`
La història la podem veure amb: `hg log`
Modifiquem i mirem les diferències: `echo "Bye..." > hello.txt; hg diff`
Podem revertir els canvis: `hg revert hello.txt`
Afegim informació: `echo "by Admin." >> hello.txt`
Publiquem els canvis: `hg commit -m "Added author."`
Preguntem a HG on estan aquests: `hg annotate hello.txt`
Les anotacions són de molta ajuda per a fixar els errors, ja que ens permeten veure qui i què ha canviat: `hg log -r 1` o millor en forma gràfica utilitzant Tortoise `hg glog`
Que ens mostrarà que el canvi és fill de la mateixa branca-autor.
Mirem els parents i tornem enrere (a una versió anterior): `hg parents`, després `hg update 0` si fem `hg parents` i mirem l'arxiu, veurem que tenim la versió original. Però no ho hem perdut, si volem mirar l'anterior: `hg cat -r 1 hello.txt`
I recuperar-ho: `hg update`
Si volem treballar entre dos usuaris que comparteixen sistema d'arxius (podria ser per SSH o HTTP també), l'usuari interessat podrà fer una còpia del repositori:
`hg clone ../adminp/hello`
Fins i tot es podria fer per correu/USB utilitzant `hg bundle` per a generar un binari i portar-lo a una altra màquina. Aquest nou usuari serà independent d'allò que faci l'altre usuari (adminp).
Aquest usuari (remix) podrà fer:
`cd hello; echo "My hola" > hello2.txt; echo "by Remix." >> hello2.txt`
Ho agrega al seu repositori i publica: `hg add` i `hg commit -m "My hello."`
Després modifiquem el d'Adminp: `echo "not by Remix" >> hello.txt`
Publiquem: `hg commit -m "Not my file."`
I podem preguntar si hi ha hagut canvis pel que fa a adminp clone: `hg incoming`.
I preguntar quins canvis no estan en adminp: `hg outgoing`.
Remix no té accés al directori d'Adminp, per la qual cosa no pot introduir els canvis però Adminp sí pot llegir-los i agregar-los. Ha de posar en el seu arxiu *.hg/hgrc*

```
[paths]
default = /home/remix/hola
```

I adminp pot observar els canvis amb `hg incoming`
i agregar-los amb: `hg pull`. És interessant veure'ls amb Tortoise, que els mostrarà gràficament.
És interessant que d'una manera simple aquest repositori el podem publicar via web fent en el nostre repositori: `hg serve`
A partir d'això, es podrà accedir a l'URL http://sysdw.nteum.org:8000 (o en *localhost*).
Aquesta solució és simple (però eficient) i temporal. Repositoris més estables hauran de ser publicats utilitzant Apache amb l'*script* hgweb.cgi/hgwebdir.cgi indicat en el manual d'instal·lació. [mer4][mer5] També és interessant l'ordre `hg-ssh` quan els repositoris s'ofereixen mitjançant SSH.

Per a executar TortoiseHG [Tortoise] des de l'usuari que disposa el repositori* utilitzem l'ordre `thg` que obrirà una interfície gràfica. A continuació, haurem de seleccionar des de *File Open repository* el directori del nostre repositori i podrem treballar en la interfície gràfica am botes les ordres i gestionar tot el repositori**.

***Sinó és el propietari de l'escriptori perquè haguem fet un `su` haurem d'executar abans del canvi d'usuari `xhost +`.**

****Vegeu la documentació per a informació detallada.**

És interessant completar la informació amb un tutorial [mer3] i veure les extensions que admet HG [mer7], i no menys interessants són els paquets (inclosos en Debian) **hgview** (*interactive history viewer*), **mercurial-git** (git *plugin* per a mercurial), **mercurial-server** (*shared Mercurial repository service*) o **eclipsi-mercurialeclipse** (integració amb Eclipse).

1.8. Mantis Bug Tracker

Mantis Bug Tracker és un sistema (GPL) de seguiment d'errors a través del web. L'ús més comú de MantisBT és fer un seguiment d'errors o incidències, però també serveix com a eina de seguiment de gestió i administració de projectes. A més del paquet de Mantis, és necessari tenir instal·lat MySQL, Apache Web Server i el mòdul PHP per a Apache.

A partir de la versió 7 de Debian, Mantis no forma part del repositori Debian* pel que haurem d'instal·lar manualment:

***Per problemes amb el temps en resoldre errors d'acord a la política Debian.**

1) Verifiquem que està instal·lat Apache, MySQL (o MariaDB), PHP5 i les extensions php5-curl, php5-mysql, php5-gd.

2) Descarreguem la darrera versió estable des de la pàgina web del projecte https://sourceforge.net/projects/mantisbt/files/mantis-stable/, descomprimim (`tar xzvf mantisbt-x.y.tar.gz`, on `x.y` és la versió que s'hagi baixat) i la movem al directori (`mv mantisbt-x.y /var/www/html/mantibt`).

3) Carreguem la URL http://srv.nteum.org/mantisbt/admin/install.php i introduïm les dades sol·licitades (base de dades, usuari administrador DB, *passwd*, *host=localhost*, usuari privilegiat-DB, etc.) i fem clic a *Install/Upgrade Database*.

4) Farà un error en escriure l'arxiu */var/www/html/mantisbt/config/config_inc.php* el que és normal pels permisos. Editem aquest arxiu i incloem el contingut indicat (veure més opcions en l'arxiu d'exemple en el mateix directori):

```
<?php
$g_hostname               = 'localhost';
$g_db_type                = 'mysqli';
$g_database_name          = 'bugtracker';
$g_db_username            = 'root';
$g_db_password            = 'passwd_de_root_en_MySQL';
$g_default_timezone       = 'Europe/Madrid';
$g_crypto_master_salt     = 'Vg/t0ANaJypnV92PpZnJtwMuV5NCAWkmdUKa1L3K9I8=';

 // Personalització de Mantis
 $g_window_title='Mantis Nteum';
?>
```

A partir d'ara, ens podrem connectar a `http://localhost/mantis/` introduint l'usuari *administrator* i la contrasenya *root* (es recomana que la primera acció sigui canviar la contrasenya en la secció `MyAccount`). Accedint des de la interfície web (*ítem manage*), es podran crear nous usuaris amb permisos d'usuari per rols (*administrator, viewer, reporter, updater*, etc.), definir els projectes i les categories dins de cada projecte, gestionar els anuncis, informes, registres, tenir una visió general dels projectes i el seu estat, gestionar els documents associats, etc. Es recomana per a crear nous usuaris, noves categories i després projectes assignats a aquestes categories. A partir d'aquest moment, es poden inserir incidències, assignant-les i gestionant-les mitjançant la interfície. Si tenim instal·lada MariaDB, ens podrà donar un error similar a *mysql_connect():*

Headers and client library minor version mismatch i haurem d'actualitzar els *headers i drivers* de PHP `apt-get install php5-mysqlnd`.

Existeix un ampli conjunt d'opcions que es poden canviar modificant l'arxiu */var/www/mantisbt/config/config_inc.php* [mant2], així, per a canviar la llengua de l'entorn, s'edita i afegeix

```
$g_language_choices_arr = array( 'english', 'spanish');
$g_default_language = 'spanish';
```

i perquè la pàgina principal del Mantis Bug Tracker sigui automàticament el propi Bug Tracker i es permeti un accés anònim als projectes públics:

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

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

```
$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
```

http://www.mantisbt.org /wiki/doku.php /mantisbt:mantis_recipies Per a més configuracions o integracions amb altres paquets, consulteu el manual [mant] o accediu a la pàgina de la *wiki* del projecte* [mant2].

Es pot veure el propi *mantis* funcionant a la pàgina web del projecte*, que s'utilitza per portar els propis errors de *mantis*.

***http://www.mantisbt.org/bugs /my_view_page.php**

Activitats

1. Definiu en PostgreSQL una base de dades que tingui almenys 3 taules amb 5 columnes (de les quals 3 han de ser numèriques) en cada taula. Genereu una llista ordenada per cada taula/columna. Genereu una llista ordenada pel major valor de la columna X de totes les taules. Canvieu el valor numèric de la columna i amb el valor numèric de la columna Z + el valor de la columna W/2.

2. Feu el mateix que amb l'exercici anterior, però amb MySQL o amb MariaDB.

3. Utilitzant les taules de World http://pgfoundry.org/projects/dbsamples/ obtingueu la població mitjana de les poblacions de les ciutats entre 10.000 i 30.000 habitants, i un percentatge de quina és la llengua més parlada i la menys parlada en relació amb els habitants.

4. Utilitzant MongoDB i una de les bases de dades d'exemple que tingui més de 5.000 documents feu una cerca simple i analitzeu les prestacions. Repetiu la recerca però que sigui de valors combinats. Feu la visualització utilitzant algunes de les eines proposades (per exemple, RoboMongo).

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

6. 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 repositori CVS.

7. Feu el mateix exercici anterior, però un dels usuaris ha de connectar-se al repositori des d'una altra màquina.

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

9. Creeu un repositori sobre Git i feu-lo visible a través del web, de tal manera que dos usuaris de la mateixa màquina puguin modificar els arxius i actualitzar el repositori.

10. Repetiu el pas anterior amb Mercurial.

11. Instal·leu Mantis, genereu 3 usuaris, 2 categories, 2 projectes i 3 incidències per a cada usuari, i gestioneu-les a través de la interfície. Genereu un usuari anònim que pugui accedir a un projecte públic com a *reporter*, però no a un de privat.

Bibliografia

Enllaços visitats al juliol de 2016.

[BD] *Big Data - Infografía*.
<http://i2.wp.com/unadocenade.com/wp-content/uploads/2013/05/Infograf%C3%ADa-Big-Data.jpg>

[Cerc] *Cervisia - CVS Frontend*. <http://www.kde.org/applications/development/cervisia/>

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

[git3] **D. Mühlbachler**. *How-To: Install a private Debian Git server using gitolite and GitLabHQ*. <http://blog.muehlbachler.org/2012/01/how-to-install-a-private-debian-git-server-using-gitolite-and-gitlabhq/>

[GIT-SRV] *Git on the Server - Setting Up the Server*. <https://git-scm.com/book/it/v2/Git-on-the-Server-Setting-Up-the-Server>

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

[len] **L. Hernández**. *Servidor Git + Gitolite + Gitweb sobre Debian 7.0 Wheezy (obsoleto)*.
<http://leninmhs.wordpress.com/2014/01/19/git-gitolite-gitweb/>

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

[mant2] *Mantis Bug Tracker Wiki*. <http://www.mantisbt.org/wiki/doku.php>

[Mar] **M. Gibert, O. Pérez**. *Bases de datos en PostgreSQL*.
<http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02152.pdf>

[Mari] *MariaDB*. <https://mariadb.org/>

[Mari1] *Setting up MariaDB Repositories*.
<https://downloads.mariadb.org/mariadb/repositories/>

[mer4] *Mercurial*. <https://www.mercurial-scm.org/>

[mer1] *Basic Mercurial*. <http://mercurial.aragost.com/kick-start/en/basic/>

[mer2] *TortoiseHg Docs*. <http://tortoisehg.bitbucket.org/docs.html>

[mer3] *HgInit: a Mercurial tutorial*. <http://hginit.com/>

[mer4] *Publishing Repositories with hgwebdir.cgi*.
<https://www.mercurial-scm.org/wiki/HgWebDirStepByStep>

[mer5] *Publishing Mercurial Repositories*.
<https://www.mercurial-scm.org/wiki/PublishingRepositories>

[mer6] *Remote Repositories*. <http://mercurial.aragost.com/kick-start/en/remote/>

[mer7] *Using Mercurial Extensions*. <https://www.mercurial-scm.org/wiki/UsingExtensions>

[Mon] *MongoDB*. <https://docs.mongodb.com/>

[MonAgre] *Mongo & SQL Aggregation Comparison*.
<https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/>

[MonGUI] *Mongo DB Adminsitration Interfaces*.
<https://docs.mongodb.com/ecosystem/tools/administration-interfaces/>

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

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

[mysqlA] *Mysql Administrator*. <http://www.mysql.com/products/workbench/>

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

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

[Posa] *PostgresSQL*. <http://www.postgresql.org>

[SQL] **PostgreSQL**. *The SQL Language*.
<http://www.postgresql.org/docs/9.1/interactive/tutorial-sql.html>

[SQLL] *SQLite*. <http://www.sqlite.org/>

[SQLDB] *SQLite Database Browser*. <https://github.com/sqlitebrowser/sqlitebrowser>

[subB] *Llibre sobre Subversion*. <http://svnbook.red-bean.com/nightly/es/index.html> (versión en español)

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

[subB2] *Subversion Documentation*.
<http://subversion.tigris.org/servlets/ProjectDocumentList>

[subwebdav] *Multiples repositoris amb Subversion*.
<http://www.debian-administration.org/articles/208>

[Tkc] *TkCVS: Tcl/Tk-based graphical interface to the CVS*.
<http://www.twobarleycorns.net/tkcvs.html>

[Tortoise] *TortoiseHg*.
<http://tortoisehg.bitbucket.org/manual/1.0/intro.html#what-is-tortoisehg>

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