Bases de dades en MySQL

Luis Alberto Casillas Santillán
Marc Gibert Ginestà
Óscar Pérez Mora

P06/M2009/02151

www.uoc.edu

Índex

Introducció

MySQL és un sistema gestor de bases de dades (SGBD, DBMS per les seves sigles en anglès) molt conegut i utilitzat per la seva simplicitat i notable rendiment. Encara que manca d'algunes característiques avançades disponibles en altres SGBD del mercat, és una opció atractiva tant per a aplicacions comercials, com d'entreteniment precisament per la seva facilitat d'ús i temps reduït de posada en marxa. Això i la seva lliure distribució a Internet sota llicència GPL li atorguen com a beneficis addicionals (no menys importants) tenir un alt grau d'estabilitat i un ràpid desenvolupament.

Nota

Podrem utilitzar la llicència GPL de MySQL sempre que el programa que l'usi també ho sigui, en cas contrari s'ha d'adquirir la "llicència comercial", que costa entre 250 i 500 €, en el moment d'escriure aquest material.

MySQL està disponible per a múltiples plataformes, la seleccionada per als exemples d'aquest llibre és GNU/Linux. Tanmateix, les diferències amb qualsevol altra plataforma són pràcticament nul·les, ja que l'eina utilitzada en aquest cas és el client *mysql-client,* que permet interactuar amb un servidor MySQL (local o remot) en mode text. D'aquesta manera és possible realitzar tots els exercicis sobre un servidor instal·lat localment o, per Internet, sobre un servidor remot.

Per a la realització de totes les activitats, és imprescindible que disposem de les dades d'accés de l'usuari administrador de la base de dades. Encara que en alguns d'ells els privilegis necessaris seran menors, per als capítols que tracten l'administració de l'SGBD serà imprescindible disposar de les credencials d'administrador.

Nota

Les sentències o ordres escrites per l'usuari seran en `font monoespaiada`, i les paraules que tenen un significat especial en MySQL estaran en **negreta**. És important fer notar que aquestes últimes no sempre són paraules reservades, sinó ordres o sentències de *mysql-client*.

La versió de MySQL que s'ha utilitzat durant la redacció d'aquest material, i en els exemples, és la 4.1, l'última versió estable en aquell moment, encara que no hi haurà cap problema en executar-los en versions anteriors, fins a la 3.23.

Objectius

L'objectiu principal d'aquesta unitat és el d'adquirir les habilitats i coneixements de MySQL necessaris per a utilitzar i administrar aquest SGBD (sistema gestor de bases de dades).

1. Característiques de MySQL

En aquest apartat enumerarem les prestacions que caracteritzen aquest SGBD, així com les deficiències de disseny, limitacions o parts de l'estàndard encara no implementades.

1.1. Prestacions

MySQL és un SGBD que ha guanyat popularitat per una sèrie d'atractives característiques:

- Està desenvolupat en C/C++.
- Es distribueixen executables per a prop de dinou plataformes diferents.
- L'API es troba disponible en C, C++, Eiffel , Java, Perl, PHP, Python, Ruby i TCL.
- Està optimitzat per a equips de múltiples processadors.
- És molt destacable la seva velocitat de resposta.
- Es pot utilitzar com a client-servidor o incrustat en aplicacions.
- Compte amb un ric conjunt de tipus de dades.
- Suporta múltiples mètodes d'emmagatzemament de les taules, amb prestacions i rendiment diferents per a poder optimitzar l'SGBD a cada cas concret.
- La seva administració es basa en usuaris i privilegis.
- Es té constància de casos en els quals gestiona cinquanta milions de registres, seixanta mil taules i cinc milions de columnes.
- Les seves opcions de connectivitat inclouen TCP/IP, *sockets* Unix i *sockets* NT, a més de suportar completament ODBC.
- Els missatges d'error poden estar en espanyol i fer ordenacions correctes amb paraules accentuades o amb la lletra *ñ*.
- És altament confiable quant a estabilitat.

Per a tots aquells que són adeptes a la filosofia de Unix i del llenguatge C/C++, l'ús de MySQL els serà molt familiar, ja que el seu disseny i les seves interfícies són concordes amb aquesta filosofia: "crear eines que facin una sola cosa i que la facin bé". MySQL té com a principal objectiu ser una base de dades fiable i eficient. Cap característica no és implementada en MySQL si abans no es té la certesa que funcionarà amb la millor velocitat de resposta i, sens dubte, sense causar problemes d'estabilitat.

La influència de C/C++ i Unix es pot observar de la mateixa en la seva sintaxi. Per exemple, la utilització d'expressions regulars, la diferenciació de funcions pels parèntesis, els valors lògics com 0 i 1, la utilització del tabulador per a completar sentències, per esmentar-ne alguns.

1.2. Limitacions

En comprendre els seus principis de disseny, es poden explicar millor les raons d'algunes de les seves carències. Per exemple, el suport de transaccions o la integritat referencial (la gestió de claus foranes) en MySQL està condicionat a un esquema d'emmagatzemament de taula concret, de manera que si l'usuari no usa transaccions, pot usar l'esquema d'emmagatzemament "tradicional" (MyISAM) i obtindrà més rendiment, mentre que si la seva aplicació requereix transaccions, haurà d'usar l'esquema que ho permet (InnoDB), sense cap altra restricció o implicació.

Nota

L'esquema de taula que cal usar es decideix per a cada una en el moment de la seva creació, encara que es pot canviar posteriorment. Actualment, MySQL suporta diversos esquemes i permet la incorporació d'esquemes definits per l'usuari.

Altres limitacions són les següents:

- No suporta procediments emmagatzemats (s'inclouran en la pròxima versió 5.0).
- No inclou disparadors (s'inclouran en la pròxima versió 5.0).
- No inclou vistes (s'inclouran en la pròxima versió 5.0).
- No inclou característiques d'objectes com tipus de dades estructurades definides per l'usuari, herència, etc.

2. Accés a un servidor MySQL

En aquest apartat veurem les diferents formes d'accés a un servidor MySQL existent que ens proporciona el mateix SGBD. L'accés des de llenguatges de programació o eines en mode gràfic es tractarà en altres apartats.

2.1. Connectant-se amb el servidor

Per connectar-se amb el servidor ens haurem d'assegurar que està funcionant i que admet connexions, tant si són locals (l'SGBD s'està executant a la mateixa màquina que intenta la connexió) com si són remotes.

Addicionalment, haurem de disposar de les credencials necessàries per a la connexió. Diferents tipus de credencials ens permetran diferents nivells d'accés. Per simplificar, suposarem que disposem de les credencials (usuari i contrasenya) de l'administrador de la base de dades (normalment, usuari *root* i la seva contrasenya). En l'apartat dedicat a l'administració de MySQL, es comenten detalladament els aspectes referents al sistema d'usuaris, contrasenyes i privilegis de l'SGBD.

2.1.1. Servidors i clients

El servidor MySQL és el servei *mysqld*, que pot rebre sol·licituds de clients locals o remots per mitjà de TCP/IP, *sockets* o *pipes* en forma de fitxers locals a la màquina en la qual s'està executant. En la distribució s'inclou un client anomenat *mysql-client*, al qual d'ara endavant ens referirem simplement com a *mysql* (així és com s'anomena el programa executable). Si s'invoca sense paràmetres, *mysql* realitza una connexió al servidor local utilitzant el nom de l'usuari Unix que l'ha invocat, i suposa que aquest usuari no requereix contrasenya. La connexió a un servidor remot i un nom d'usuari específics requereix almenys dos arguments:

Nota

El servidor MySQL és *mysqld*. S'hi poden connectar múltiples clients. *mysql* és el client en mode text que proporciona el mateix SGBD.

- -h per a especificar el nom del servidor.
- -u per al nom de l'usuari.

Perquè el programa client pregunti la contrasenya de connexió a l'usuari, haurem de proporcionar addicionalment el paràmetre -p.

```
$ mysql -h servidor.elmeulloc.org -u <usuari> -p
```

2.1.2. Connectar-se i desconnectar-se

Si es té cap problema per a realitzar la connexió, cal consultar amb l'administrador del sistema, que ens proporcionarà un nom d'usuari, una contrasenya i el nom del servidor, segons sigui necessari, i ens informarà de les restriccions que té el nostre compte.

Nota

Els usuaris del sistema operatiu i els de MySQL no són els mateixos, encara que l'administrador de MySQL (amb finalitats pràctiques) pugui utilitzar els mateixos noms per als comptes dels usuaris MySQL.

L'administració i seguretat de MySQL està dissenyada sobre un esquema d'usuaris i privilegis. Els usuaris han de ser creats per l'administrador amb els seus respectius privilegis i restriccions. És l'administrador qui decideix si els noms dels usuaris de MySQL es corresponen o no als del sistema operatiu.

Aparença de *mysql* en ingressar en el mode interactiu:

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 5 to server version: 3.23.49-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Amb l'ordre *help* obtenim una sèrie d'opcions (veurem les més utilitzades).

Per a sortir del client podem escriure '*\q*' o '*quit*':

```
mysql> quit;
```

Tant per a l'ordre *quit* com per a l'ordre *help*, el punt i coma al final és opcional.

2.2. Introducció de sentències

El client de MySQL en mode interactiu ens permet tant la introducció de sentències SQL per a treballar amb la base de dades (crear taules, fer consultes i veure'n els resultats, etc.) com l'execució d'ordres pròpies de l'SGBD per a obtenir informació sobre les taules, índexs, etc. o executar operacions d'administració.

Sentències

Les sentències en *mysql* poden incloure múltiples línies i acaben amb punt i coma.

2.2.1. Sentències

A continuació presentem una execució de la sentència *select* amb quatre columnes de dades:

```
mysql> select user(), connection_id(), version(), database();
+--------------+-----------------+-------------+------------+
|   user()     | CONNECTION_ID() | version()   | database()  |
+--------------+-----------------+-------------+------------+
| yo@localhost |    4            | 3.23.49-log |            |
+--------------+-----------------+-------------+------------+
1 row in set (0.00 sec)
mysql>
```

En aquesta consulta se sol·licita, per mitjà de funcions incorporades en l'SGBD, el nom de l'usuari actual de MySQL, el número de connexió al servidor, la versió del servidor i la base de dades en ús. Les funcions es reconeixen pels parèntesis al final. *mysql* lliura els seus resultats en taules, la primera línia de les respostes conté els encapçalaments de les columnes. És important no deixar espai entre el nom d'una funció i els parèntesis, d'una altra manera, *mysql* marcarà un missatge d'error.

L'última línia lliurada per *mysql* informa sobre el nombre de files trobat com a resultat de la consulta i el temps estimat que va portar la seva realització. Aquesta mesura de temps no s'ha de considerar gaire precisa per a mesurar el rendiment del servidor, es tracta simplement d'un valor aproximat que es pot veure alterat per múltiples factors.

Observem que la columna amb el nom de la base de dades actual aquesta buida. Això és natural, ja que no hem creat encara cap base de dades ni li hem indicat al gestor sobre quina volem treballar.

2.2.2. Ordres en múltiples línies

Les ordres es poden expandir en diverses línies per comoditat, sobretot en escriure llargues sentències SQL. El client no enviarà la sentència SQL al servidor fins a trobar el punt i coma, d'aquesta manera, l'ordre anterior es pot escriure així:

```
mysql> select user(),
-> connection_id(),
-> version(),
-> database();
+--------------+-----------------+-------------+------------+
|   user()     | CONNECTION_ID() | version()   | database()  |
+--------------+-----------------+-------------+------------+
| yo@localhost |    4            | 3.23.49-log |            |
+--------------+-----------------+-------------+------------+
1 row in set (0.00 sec)
mysql>
```

Observeu l'indicador de *mysql* que es transforma en ->, signe que significa que l'ordre encara no està completa. També es poden escriure diverses ordres en una sola línia, cada una ha de portar el seu respectiu punt i coma:

```
mysql> select now(); select user();
+----------------------+
| CONNECTION_ID()      |
+----------------------+
|    4                 |
+----------------------+
1 row in set (0.00 sec)
+----------------------+
|   user()             |
+----------------------+
|  yo@localhost        |
+----------------------+
1 row in set (0.01 sec)
mysql>
```

S'executaran en l'ordre que estan escrits. Les ordres es poden cancel·lar amb la combinació \c, amb la qual cosa el client ens tornarà a mostrar l'indicador perquè escriguem de nou la sentència:

```
mysql> select now(),
-> uso
-> ver \c
mysql>
```

Indicadors de *mysql*

Indicador	Significat
mysql>	Espera una nova sentència
->	La sentència encara no s'ha acabat amb ;
">	Una cadena en cometes dobles no s'ha tancat
'>	Una cadena en cometes simples no s'ha tancat

2.2.3. Cadenes de caràcters

Les cadenes de caràcters es poden delimitar mitjançant cometes dobles o simples. Evidentment, s'han de tancar amb el mateix delimitador amb què s'han obert.

```
mysql> select "Hola món",'Felicitats';
```

Les cadenes de caràcters es poden escriure en diverses línies:

```
mysql> select "Aquest és un text
      "> en dues línies";
```

Al principi, és comú oblidar el punt i coma en introduir una ordre, i també oblidar tancar les cometes. Si aquest és el cas, cal recordar que *mysql* no interpreta el que està entre cometes, de manera que per a utilitzar l'ordre de cancel·lació '\c' cal tancar abans les cometes obertes:

```
mysql> select "Aquest és un text
"> \c
"> " \c
mysql>
```

2.2.4. Expressions i variables

MySQL disposa de variables de sessió, visibles únicament durant la connexió actual. Aquestes poden emmagatzemar valors de tipus enters, flotants o cadenes, però no taules. Es defineixen com en l'exemple següent:

```
mysql> select @x := 1;
```

La variable local @x té ara el valor 1 i es pot utilitzar en expressions:

```
mysql> select @x, sqrt(@x), sin(@x), @x + 10, @x > 10;
+------+----------+----------+---------+---------+
| @x   | sqrt(@x) | sin(@x)  | @x + 10 | @x > 10 |
+------+----------+----------+---------+---------+
| 1    | 1.000000 | 0.841471 |  11     |  0      |
+------+----------+----------+---------+---------+
1 row in set (0.01 sec)
```

Les variables locals permeten emmagatzemar dades entre consultes i, a la pràctica, és recomanable utilitzar-les exclusivament amb aquesta finalitat, per exemple:

```
mysql> select @hora_ingres := now();
mysql> mysql> select now() - @ingres;
+-------------------+
| now() - @ingreso  |
+-------------------+
|  20040124138051   |
+-------------------+
1 row in set (0.00 sec)
```

2.2.5. Expressions

Cal anar amb compte amb l'ús de les variables locals pels motius següents:

- S'avaluen en el servidor en ser enviades pel client.
- Es realitzen conversions de tipus implícites.

```
mysql> do @ingres := now();
```

Nota

L'ordre **do** avalua expressions sense mostrar els resultats en pantalla. Es pot avaluar qualsevol expressió que admet l'ordre **select**.

Les variables no requereixen declaració i, per omissió, contenen el valor NULL que significa 'absència de valor', observeu en la consulta següent els resultats d'utilitzar valors nuls:

```
mysql> select @y,
-> sqrt( @y ),
-> @y + 10,
-> @y < 1 ;
+------+------------+---------+--------+
| @y   | sqrt( @y ) | @y + 10 | @y < 1 |
+------+------------+---------+--------+
| NULL |  NULL      |  NULL   | NULL   |
+------+------------+---------+--------+
1 row in set (0.00 sec)
```

La raó d'aquest comportament és que no és possible realitzar cap operació quan es desconeix algun valor. L'entrada de valors NULL sempre significarà sortida de valors NULL.

Nota

En una expressió on qualsevol dels seus elements sigui NULL, automàticament lliurarà com a resultat el valor NULL.

2.3. Procés per lots

MySQL pot processar per lots les sentències contingudes en un arxiu de text. Cada sentència haurà d'acabar en ';' igual que si l'escriguéssim en el client. La sintaxi és la següent:

```
$ mysql -u juan -h servidor.elmeulloc.org -p < demo.sql
```

En aquest cas, es realitzarà una connexió amb el servidor, ens demanarà la contrasenya de l'usuari 'joan' i, si és correcta, executarà les ordres incloses en l'arxiu *demo.sql*, una a una i pel mateix ordre. Imprimirà els resultats (o errors) en la sortida estàndard (o d'error) i acabarà. D'aquesta manera evitarem la molèstia que les processi un per un de manera interactiva.

Una altra manera de processar un arxiu és mitjançant l'ordre `source` des de l'indicador interactiu de MySQL:

```
mysql> source demo.sql
```

L'arxiu *demo.sql* crea una nova base de dades.

L'usuari ha de tenir permisos per a crear bases de dades si vol que sigui processat l'arxiu *demo.sql*. Si l'administrador crea la base de dades per nosaltres, serà necessari editar-ho, comentant la línia on es crea la base de dades amb el símbol '#' a l'inici:

```
# create database demo;
```

És necessari processar el contingut del fitxer *demo.sql* tal com el transcrivim aquí, a fi de poder realitzar els exemples de la resta de l'apartat. Si s'observa el seu contingut, possiblement moltes coses s'expliquen per si mateixes; de tota manera, seran explicades en aquest apartat. També es poden executar les seves ordres en el client directament.

Contingut del fitxer demo.sql

```
#drop database demo;
create database demo;
use demo;
---
--- estructura de la taula productes
---

create table productes (
  part      varchar(20),
  tipus     varchar(20) ,
  especificacio varchar(20) ,
  psuggerit float(6,2),
  clau int(3)  zerofill not null auto_increment,
  primary key (clau)
);
insert into productes (part,tipus,especificacio,psuggerit) values
    ('Processador','2 GHz','32 bits',null),
    ('Processador','2.4 GHz','32 bits',35),
    ('Processador','1.7 GHz','64 bits',205),
    ('Processador','3 GHz','64 bits',560),
    ('RAM','128MB','333 MHz',10),
    ('RAM','256MB','400 MHz',35),
    ('Disc Dur','80 GB','7200 rpm',60),
```

```
    ('Disc Dur','120 GB','7200 rpm',78),
    ('Disc Dur','200 GB','7200 rpm',110),
    ('Disc Dur','40 GB','4200 rpm',null),
    ('Monitor','1024x876','75 Hz',80),
    ('Monitor','1024x876','60 Hz',67)
    ;
--
-- Estructura de la taula 'proveïdor'
--
create table proveidors (
 empresa   varchar(20) not null,
 pagament set('crèdit','efectiu'),
 primary key (empresa)
);
--
-- Valors de la taula 'proveïdor'
--
insert into proveidors (empresa,pagament) values
   ('Tecno-k','crèdit'),
   ('Patito','efectiu'),
   ('Nacional','crèdit,efectiu')
;
create table guany(
  venda  enum('A l'engròs','Al detall'),
   factor  decimal(2,2)
);
insert into guany values
  ('A l'engròs',1.05),
    ('Al detall',1.12)
;
create table preus (
   empresa  varchar(20) not null,
   clau  int(3) zerofill not null,
   preu float(6,2),

   foreign key (empresa)  references proveidors,
   foreign key (clau)   references productes
);

insert into preus values
   ('Nacional',001,30.82),
   ('Nacional',002,32.73),
   ('Nacional',003,202.25),
   ('Nacional',005,9.76),
   ('Nacional',006,31.52),
   ('Nacional',007,58.41),
   ('Nacional',010,64.38),
   ('Patito',001,30.40),
   ('Patito',002,33.63),
   ('Patito',003,195.59),
   ('Patito',005,9.78),
   ('Patito',006,32.44),
```

```
   ('Patito',007,59.99),
   ('Patito',010,62.02),
   ('Tecno-k',003,198.34),
   ('Tecno-k',005,9.27),
   ('Tecno-k',006,34.85),
   ('Tecno-k',007,59.95),
   ('Tecno-k',010,61.22),
   ('Tecno-k',012,62.29)
;
```

Si es desitja portar un registre de totes les operacions d'una sessió, es pot utilitzar l'expressió següent; d'aquesta manera es guardaran totes les ordres i els seus resultats a *arxiu_registre.txt*:

```
mysql> tee arxiu_registre.txt
```

Per a cancel·lar la captura, n'hi ha prou d'escriure el següent:

```
mysql> notee
```

2.4. Usar bases de dades

La consulta següent informa sobre la base de dades actualment en ús:

```
mysql> select database();
+------------+
| database() |
+------------+
|            |
+------------+
1 row in set (0.13 sec)
```

El camp està buit perquè no estem fent ús de cap base de dades. Per a veure les bases de dades existents en el sistema, s'ha d'efectuar la consulta següent:

```
mysql> show databases;
+-------------+
|  Database   |
+-------------+
|   demo      |
|  mysql      |
|   test      |
+-------------+
3 rows in set (0.01 sec)
```

MySQL ens mostra el llistat de les bases de dades definides al servidor. Ha d'aparèixer la base de dades *demo* que creem amb l'arxiu *demo.sql*. Per a poder treballar amb aquesta, l'hem d'obrir:

```
mysql> use demo;
```

use

L'ordre `use base_de_dades` permet obrir una base de dades per al seu ús.

Nota

Es poden fer consultes en una base de dades sense utilitzar l'ordre **use**; en aquest cas, tots els noms de les taules han de portar el nom de la base de dades a la qual pertanyen de la manera: *demo.productes*.

Una altra possibilitat consisteix a proporcionar el nom de la base de dades en iniciar una sessió interactiva amb *mysql:*

```
$ mysql demo -u joan -p
```

La consulta de les taules que conté la base de dades `demo` es realitza amb la sentència `show` de la manera següent:

```
mysql> show tables;
+----------------+
| Tables_in_demo |
+----------------+
|   parts        |
|  proveidors    |
+----------------+
2 rows in set (0.00 sec)
```

Nota

L'ordre `show` és útil per a mostrar informació sobre les bases de dades, taules, variables i d'altra informació sobre l'SGBD. Podem utilitzar `help show` en l'intèrpret d'ordres per a obtenir totes les variants d'aquesta sentència.

Així mateix, podem consultar les columnes de cada una de les taules:

```
mysql> descriu productes;
+---------------+----------------------+------+------+----------+------------------+
| Field         |       Type           | Null | Key  | Default  | Extra            |
+---------------+----------------------+------+------+----------+------------------+
| part          | varchar(20)          | YES  |      | NULL     |                  |
| tipus         | varchar(20)          | YES  |      | NULL     |                  |
| especificicacio | varchar(20)        | YES  |      | NULL     |                  |
| suggerit      | float(6,2)           | YES  |      | NULL     |                  |
| clau          | int(3) unsigned zerofill | YES | PRI | NULL     | auto_increment   |
+---------------+----------------------+------+------+----------+------------------+
5 rows in set (0.00 sec)
```

Per a crear una nova base de dades usarem la sentència `create database`:

```
mysql> create database prova;
```

Per a eliminar una base de dades, usarem la sentència `drop database`:

```
mysql> drop database prova;
```

MySQL és sensible a l'ús de majúscules i minúscules, tant en la definició de bases de dades, com de taules o columnes.

3. Creació i manipulació de taules

3.1. Crear taules

Una vegada realitzada la connexió amb el servidor MySQL i després d'obrir una base de dades, podem crear-hi taules de la manera següent:

```
mysql> create table persones (
-> nom char(30),
-> adreça char(40),
-> telefon char(15)
-> );
Query OK, 0 rows affected (0.02 sec)
```

En aquest cas, la sentència `create table` construeix una nova taula en la base de dades en ús. La taula conté tres columnes, *nom*, *adreça* i *telèfon*, totes de tipus caràcter i de longituds 30, 40 i 15 respectivament. Si s'intenta guardar-hi valors que sobrepassin aquests límits, seran truncats per a poder emmagatzemar-los. Per aquest motiu, és important reservar prou espai per a cada columna. Si es preveu que molts registres ocuparan només una fracció de l'espai reservat, es pot utilitzar el tipus **varchar**, similar a **char**, amb la diferència que el valor ocuparà un espai menor a l'especificat si la cadena és més curta que el màxim indicat, estalviant així espai d'emmagatzemament.

Els noms de les columnes admeten caràcters accentuats.

Les taules es poden eliminar amb **drop table**:

```
mysql> drop table persones;
Query OK, 0 rows affected (0.01 sec)
```

Alternativament, es pot utilitzar la sintaxi següent:

```
mysql> drop table if exists personas;
```

Atributs de columna

Atribut	Significat
null	Es permeten valors nuls, atribut per omissió si no s'especifica el contrari.
not null	No es permeten valors nuls.
default valor	Valor per omissió que s'assigna a la columna.
auto_increment	S'incrementa en un el valor màxim registrat fins al moment i el nou valor s'assigna automàticament. S'aplica només a les columnes marcades com a clau primària.
primary key	Assenyala el camp com a clau primària, implícitament també el declara com a not null.

Vegem-ho amb un exemple:

```
mysql> create table persones (
-> nom varchar(40) not null,
-> adreça varchar(50) null,
-> edo_civil char(13) default 'Solter',
-> num_registre int primary key auto_increment,
-> );
Query OK, 0 rows affected (0.01 sec)
```

En aquest cas, la taula conté quatre columnes, de les quals *nom* i *estat_civil* permeten valors nuls, en *estat_civil* està implícit ja que no es declara el contrari. La columna *num_registre* no accepta valors nuls perquè està definida com a clau primària.

Nota

La definició de columnes té el format següent:

nom_columna **tipus atributs**.

Encara que la creació d'una clau primària es pot declarar com a atribut de columna, és convenient definir-la com a restricció de taula, com es veurà de seguida.

També és possible indicar restriccions sobre la taula i no sobre columnes específiques:

```
mysql> create table persones (
-> nom varchar(40) not null,
-> naixement date not null,
-> parella varchar(40),
-> proveedor int not null,
->
-> primary key (nom,naixement),
-> unique (parella),
-> foreign key (proveïdor) references proveidors
-> );
Query OK, 0 rows affected (0.01 sec)
```

Restriccions de taula

Restricció	Significat
primary key	Defineix la columna o les columnes que serviran com a clau primària. Les columnes que formen part de la clau primària han de ser **not null**.
unique	Defineix les columnes en les quals no es poden duplicar valors. Seran les claus candidates del model relacional.
foreign key (*columna*) **references** ***tabla*** (*columna2*)	Defineix que els valors de *columna* es permetran només si existeixen en *taula(columna2)*. És a dir, *columna* fa referència als registres de *taula*, això assegura que no es realitzin referències a registres que no existeixen.

Es defineixen tres restriccions sobre la taula després de la definició de quatre columnes:

- La primera restricció es refereix a la clau primària, composta per les columnes *nom* i *naixement*: no hi pot haver dues persones que es diguin igual i que hagin nascut en la mateixa data. La clau primària permet identificar de manera unívoca cada registre de la taula.

- La segona restricció defineix que la parella d'una persona ha de ser única: dues persones no poden tenir la mateixa parella. Tot intent d'inserir un nou registre on el nom de la parella ja existeixi, serà rebutjat. Quan es restringeix una columna amb **unique**, els valors **null** reben un tracte especial, ja que es permeten múltiples valors nuls.

- La tercera restricció afecta la columna *proveïdor*, només pot prendre valors que hi hagi en la clau primària de la taula *proveïdors*.

Claus foranes

Les restriccions de taula **foreign key** no tenen cap efecte en MySQL 4.0 i anteriors, ja que aquesta característica no està implementada. S'admet en la sintaxi per compatibilitat, ja que serà implementada en una versió posterior. En la versió 4.1, està suportada si s'utilitza el tipus de taula InnoDB.

Les restriccions de taula es poden definir amb un identificador útil per a fer referències posteriors a la restricció:

```
mysql> create table persones (
-> nom varchar(40) not null,
-> naixement date not null,
-> parella varchar(40),
-> proveidor int not null,
->
-> constraint clau primary key (nom,naixement),
-> constraint monogam unique (parella),
-> constraint traballa_en foreign key (proveidor) references
proveïdors
-> );
```

key / index

La definició d'índexs es pot fer també en el moment de creació de la taula, mitjançant la paraula clau key (o index), a la qual haurem de proporcionar el nom que assignarem a aquesta clau i les columnes que la formen, entre parèntesis. Hi ha modificadors opcionals sobre l'índex que ens permeten especificar si es tracta d'un índex únic o múltiple (segons si poden existir o no diversos valors iguals de l'índex en la taula).

En versions recents de MySQL hi ha altres tipus d'índexs (espacials, de text complet, etc.) per a tipus de dades concretes i que ofereixen prestacions addicionals.

3.2. Tipus de dades

MySQL té un ric conjunt de tipus de dades per a les columnes, que és necessari conèixer per a triar millor com definir les taules. Els tipus de dades es poden classificar en tres grups:

- Numèriques
- Cadenes de caràcters
- Dates i hores

El valor **null** és un cas especial de dada, ja que en significar *absència de valor* s'aplica a tots els tipus de columna. Els següents símbols s'utilitzen en la definició i descripció dels tipus de dades en MySQL:

- **M:** l'amplada de la columna en nombre de caràcters.
- **D:** nombre de decimals que cal mostrar.
- **L:** longitud o mida real d'una cadena.
- **[ ]:** el que s'escrigui entre aquests és opcional.

3.2.1. Tipus de dades numèriques

Els tipus de dades numèriques comprenen dues categories, els enters i els nombres amb punt flotant.

Nombres enters

La principal diferència entre cada un dels tipus d'enters és la seva mida, que va des d'1 byte d'emmagatzemament fins als 8 bytes. Les columnes de tipus enter poden rebre dos atributs addicionals, que s'han d'especificar immediatament després del nom del tipus:

- **unsigned**. Indica que l'enter no podrà emmagatzemar valors negatius. És responsabilitat de l'usuari verificar, en aquest cas, que els resultats de les restes no siguin negatius, perquè MySQL els converteix en positius.

- **zerofill**. Indica que la columna, en ser mostrada, omplirà amb zeros a l'esquerra els espais buits. Això d'acord amb el valor especificat per **M** en la declaració del tipus. Una columna amb l'atribut **zerofill** és alhora **unsigned**, encara que no s'especifiqui.

Exemple

```
create table nombres (
x int(4) zerofill not null,
y int(5) unsigned
);
```

L'ordre anterior crea una taula amb dues columnes. Totes dues ocuparan un espai de 4 bytes, però en mostrar-se, la columna *x* ocuparà un espai de 4 dígits i la columna *y*, de 5.

Tant **zerofill** com **unsigned** s'han d'escriure sempre abans que qualsevol altre atribut de columna.

Tipus enters

Tipus	Espai d'emmagatzemament	Significat
tinyint[(M)]	1 byte	Enter molt petit
smallint[(M)]	2 bytes	Enter petit
mediumint[(M)]	3 bytes	Enter mitjà
int[(M)]	4 bytes	Enter
bigint[(M)]	8 bytes	Enter gran

Nombres amb punt flotant

MySQL té els tipus **float** i **double**, de 4 i 8 bytes d'emmagatzemament. A més, inclou el tipus decimal, que s'emmagatzema com una cadena de caràcters i no en format binari.

Nombres amb punt flotant

Tipus	Espai d'emmagatzemament	Significat
float	4 bytes	Simple precisió
double	8 bytes	Doble precisió
decimal	M + 2 bytes	Cadena de caràcters que representa un nombre flotant

3.2.2. Cadenes de caràcters

Cadenes de caràcters

Tipus	Equivalent	Mida màxim	Espai d'emmagatzemament
char[(M)]		M bytes	M bytes
varchar[(M)]		M bytes	L+1 bytes
tinytext	tinyblob	2^8–1 bytes	L+1 bytes
text	blob	2^{16}–1 bytes	L+2 bytes
mediumtext	mediumblob	2^{24}–1 bytes	L+3 bytes
longtext	longblob	2^{32}–1 bytes	L+4 bytes
enum('v1','v2',. . .)		65535 valors	1 o 2 bytes
set('v1','v2',. . .)		64 valors	1 a 8 bytes

Si observem la taula, veiem que l'únic tipus de dada que sempre utilitza la mida especificada per M és el tipus **char**. Per aquest motiu, s'ofereix el tipus **varchar** que ocupa només l'espai requerit pel valor de la columna.

Exemple

```
create table persona(
comentari char(250),
encàrrec varchar(250)
);
```

La columna *comentari* ocuparà 250 bytes d'espai d'emmagatzemament, sense importar el valor emmagatzemat. Al contrari, la columna *encàrrec* ocuparà només l'espai necessari segons el valor assignat; per exemple, la cadena "*Instal·lar MySQL*" té 14 bytes de longitud, i el camp *encàrrec* ocuparia 15 bytes per emmagatzemar-la.

Els tipus text i blob són equivalents, però text respecta les majúscules, minúscules i caràcters accentuats en l'ordenació.

Exemple de l'ús dels tipus enumerats o enum

```
create table persona(
estat_civil enum('solter','casat','vidu','divorciat')
);
```

La columna *estat_civil* de la taula en la sentència anterior, només podrà emmagatzemar els valors *'solter*', *'casat*', *'vidu*', *'divorciat*', que especifica el tipus **enum**. La columna ocuparà l'espai d'un byte, ja que els valors enum són representats internament per números.

3.2.3. Dates i hores

Dates i hores

Tipus	Espai d'emmagatzemament	Rang
date	3 bytes	'1000-01-01' al '9999-12-31'
time	3 bytes	'-838:59:59' a '838:59:59'
datetime	8 bytes	'1000-01-01 00:00:00' a '9999-12-31 23:59:59'
timestamp[(M)]	4 bytes	19700101000000 a l'any 2037
year[(M)]	1 byte	1901 a 2155

3.3. Modificar taules

3.3.1. Agregar i eliminar columnes

Alterar l'estructura d'una taula és una tasca més freqüent del que es pot imaginar en un principi. La sentència ***alter table*** permet una àmplia gamma de maneres de modificar una taula. La sentència següent, on modifiquem la taula *personal* creada en la secció anterior, ens recorda una mica l'estructura de la sentència ***create table***:

```
mysql> alter table personal add (
-> mascota char(30) default 'gos',
-> passatemps char (20) not null
-> );
```

Nota

Sempre és possible consultar l'estructura d'una taula amb l'ordre ***describe table***.

Després d'executar la sentència anterior, apareixen dues noves columnes en la taula. Si volem agregar una sola columna, podem usar la sintaxi següent:

```
mysql> alter table personal add capital int not null
-> after name;
```

Aquest format d'**alter table** permet, a més, inserir les columnes abans (**before**) o després (**after**) d'una columna concreta.

Les columnes que no es vulguin es poden eliminar amb l'opció **drop**:

```
mysql> alter table personal drop passatemps;
```

3.3.2. Modificar columnes

La modificació d'una columna amb l'opció **modify** és semblant a tornar a definir-la:

```
mysql> alter table personal modify
-> mascota char (14) default 'gat';
```

Després de la sentència anterior, els atributs i tipus de la columna han canviat pels especificats. El que no es pot canviar amb aquesta sintaxi és el nom de la columna. Per a això, s'ha d'utilitzar l'opció **change:**

```
mysql> alter table personal change name
-> nom char(20);
```

La columna que es deia *name* canvia a *nom*.

Amb la mateixa ordre **alter table** podem realitzar fins i tot l'ordenació física d'una taula segons una columna específica:

```
mysql> alter table personal order by name;
Query OK, 0 rows affected (0.06 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Nota

En general, una taula no pot durar gaire temps amb un `order` respecte a una columna, ja que les insercions no es realitzaran respectant l'ordre establert. Només és útil aplicar aquesta ordre en taules que no seran actualitzades.

Finalment, podem canviar de nom la taula:

```
mysql> alter table personal rename gent;
```

rename table

L'ordre **rename table** *nom_ vell* to *nom_ nou* és una manera alternativa de canviar el nom a una taula.

3.4. Altres opcions

3.4.1. Copiar taules

Tot i que no hi ha una ordre explícita per a copiar taules d'una base de dades a una altra, és possible utilitzar l'ordre **rename table** per a aquest propòsit; n'hi ha prou d'especificar la base de dades a què pertany una taula:

```
mysql> rename table base_uno.taula to base_dos.taula;
```

També és possible crear una taula nova amb el contingut d'una altra ja existent (copiant-ne les dades):

```
mysql> create table nova_taula select * from altra_taula;
```

La sentència següent és equivalent, però no copia les dades de la taula origen:

```
mysql> create table nova_taula like altra_taula;
```

3.4.2. Taules temporals

MySQL permet crear taules temporals, visibles exclusivament en la sessió oberta, i guardar dades entre consultes. La creació d'una taula temporal només requereix la utilització de la paraula ***temporary*** en qualsevol format de l'ordre ***create table***. La utilitat de les taules temporals es limita a consultes complexes que han de generar resultats intermedis que hem de consultar (fer 'join' amb elles) diverses vegades o en consultes separades. Internament, MySQL genera també taules temporals per a resoldre determinades consultes:

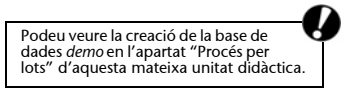
```
mysql> create temporary table nova_taula ...
```

4. Consultes

Com ja hem explicat, les consultes sobre la base de dades s'executen mitjançant sentències SELECT introduïdes en el mateix programa client i els resultats es presenten en forma de taula.

4.1. La base de dades demo

En aquesta secció utilitzarem la base de dades *demo* que hem creat amb l'ordre source *demo.sql*. Així que, abans d'estudiar les consultes en MySQL, revisarem breument l'estructura d'aquesta base de dades, que consta de les taules següents:



Podeu veure la creació de la base de dades *demo* en l'apartat "Procés per lots" d'aquesta mateixa unitat didàctica.

```
mysql> show tables;
+----------------+
| Tables_in_demo |
+----------------+
|  guany         |
|   preus        |
|  productes     |
|  proveidors    |
+----------------+
```

Les quatre taules representen, de manera fictícia, la base de dades d'un distribuïdor d'equips de processament. Estan dissenyades per a servir d'exemple als casos presentats en aquest capítol, per la qual cosa no necessàriament seran útils en la vida real.

En el nostre exemple imaginari representem la situació següent.

- El nostre venedor té una relació de proveïdors que venen els seus productes a crèdit, en efectiu o de totes dues maneres. Les compres a crèdit paguen interessos, però són útils perquè no sempre és possible pagar en efectiu. S'utilitza una columna de tipus conjunt per a *pagament*, que pot prendre els valors *'crèdit'*, *'efectiu'* o tots dos:

```
create table proveidors (
empresa varchar(20) not null,
pago set('crèdit','efectiu'),
primary key (empresa)
);
```

Els productes que es distribueixen són parts d'equip de còmput. Per a la majoria dels productes en el mercat, els fabricants suggereixen un preu de venda al públic que, encara que no és obligatori, els consumidors no estan disposats a pagar més. Les claus dels productes són assignades per a control intern amb

un nombre consecutiu. Amb aquestes especificacions, la taula *productes* es defineix de la manera següent:

```
create table productes (
part varchar(20),
tipus varchar(20) ,
especificacio varchar (20) ,
psuggerit float(6,2),
clau int(3) zerofill not null auto_increment,
primary key (clau)
);
```

- L'empresa defineix una política per als guanys mínims que s'han d'obtenir en vendes: el 5% a l'engròs i el 12% al detall. Aquests valors s'emmagatzemen en la taula *guanys*, on es va decidir incloure una columna de nom *factor*, amb el nombre pel qual es multiplica el preu de compra per a obtenir el preu de venda. Els tipus de venda *'a l'engròs'* i *'al detall'* es defineixen amb un tipus de dades **enum**:

```
create table guany(
venda enum('A l'engròs','Al detall'),
factor decimal (2,2)
);
```

- La llista de preus es defineix a partir de l'empresa proveïdor i el producte, assignant-li un preu. Per aquest motiu, les columnes *empresa* i *clau* es defineixen com a **foreign key**.

```
create table preus (
empresa varchar(20) not null,
clau int(3) zerofill not null,
preu float(6,2),
foreign key (empresa) references proveïdors,
foreign key (clau) references productes
);
```

4.2. Consultar informació

MySQL ofereix un conjunt molt ampli de funcions auxiliars (tant estàndard com pròpies) que ens poden ajudar molt en determinats moments, atès que deixen part de la feina de manipular els resultats al mateix gestor. A causa del ràpid ritme en el desenvolupament d'aquest SGBD, és molt convenient consultar sempre la documentació de la nostra versió per a conèixer-ne les possibilitats concretes.

En el mòdul 3 d'aquest curs ja estudiem detalladament el llenguatge SQL, per la qual cosa no ens estendrem aquí en el seu ús i possibilitats. Únicament mostrarem els aspectes destacables i les facilitats o limitacions que ofereix MySQL respecte a aquest.

4.2.1. Funcions auxiliars

Les funcions auxiliars que podem utilitzar en les nostres consultes (tant en la projecció de les columnes com en condicions en la seva selecció) es poden classificar segons el tipus de dades amb el qual treballen.

Exemple

```
mysql> select concat(part,' ',tipus) as producte,
-> psuggerit as 'preu suggerit',
-> psuggerit + 10 as preu_amb_tramesa
-> from productes;
+--------------------+-----------------+------------------+
|   producte         | preu suggerit   | preu_amb_tramesa |
+--------------------+-----------------+------------------+
| Processador 2 GHz  |    NULL         |    NULL          |
| Processador 2.4 GHz|   35.00         |    45.00         |
| Processador 1.7 GHz|   205.00        |   215.00         |
| Processador 3 GHz  |   560.00        |   570.00         |
|     RAM 128MB      |   10.00         |    20.00         |
|     RAM 256MB      |   35.00         |    45.00         |
| Disc Dur 80 GB     |   60.00         |    70.00         |
| Disc Dur 120 GB    |   78.00         |    88.00         |
| Disc Dur 200 GB    |   110.00        |   120.00         |
| Disc Dur 40 GB     |    NULL         |    NULL          |
| Monitor 1024x876   |   80.00         |    90.00         |
| Monitor 1024x876   |   67.00         |    77.00         |
+--------------------+-----------------+------------------+
12 rows in set (0.00 sec)
```

Alguns exemples de les funcions més usades:

Operadors lògics

Comparació. A part dels estàndards =, !=, <, >, IS NULL, IS NOT NULL, BETWEN, IN, destaquen COALESCE, INTERVAL, LEAST, GREATEST per a treballar amb llistes de valors.

Control del flux d'execució

CASE .. WHEN .. THEN .. ELSE .. END: similar a l'estructura que crearíem mitjançant qualsevol llenguatge de programació:

```
mysql> SELECT CASE WHEN 1>0 THEN 'true' ELSE 'false' END;
+-----------------------------------------+
| CASE WHEN 1>0 THEN 'true' ELSE 'false' END |
+-----------------------------------------+
| true                                    |
+-----------------------------------------+
1 row in set (0.00 sec)
```

- IF(expr1,expr2,expr3): típica estructura condicional, si l'expr1 és certa, torna l'expr2, en cas contrari, l'expr3:

```
mysql> SELECT IF(STRCMP('test','test1'),'no','yes');
+---------------------------------------+
| IF(STRCMP('test','test1'),'no','yes') |
+---------------------------------------+
| no                                    |
+---------------------------------------+
1 row in set (0.00 sec)
```

Funcions per a treballar amb cadenes de caràcters (només alguns exemples)

- CONCAT, INSTR (trobar en una cadena), SUBSTRING, LCASE/RCASE, LENGTH, REPLACE, TRIM, entre d'altres, són funcions similars a les que podem trobar en llenguatges de programació per a manipular cadenes de caràcters.

- QUOTE: delimita una cadena de text correctament per a evitar problemes a l'hora d'usar-la en sentències SQL. La cadena resultant estarà delimitada per cometes simples. Les cometes, el valor ASCII NUL i d'altres potencialment conflictius seran tornats precedits del caràcter '\'.

- ENCODE/DECODE, CRYPT, COMPRESS/UNCOMPRESS, MD5, etc. són funcions que ens poden ajudar molt en l'emmagatzemament de dades sensibles com contrasenyes, etc.

Funcions numèriques

- Els operadors aritmètics clàssics per a realitzar tot tipus d'operacions, suma, resta, divisió, producte, divisió entera, etc.

- Funcions matemàtiques de tot tipus, trigonomètriques, logarítmiques, etc.

Funcions per a treballar amb dates i hores

- Obtenció de dates en qualsevol format: DATE_FORMAT, DATE, NOW, CURRDATE, etc.

- Manipulació i càlculs amb dates: ADDDATE, ADDTIME, CONVERT_TZ, DATE_DIFF, etc.

4.2.2. La sentència EXPLAIN

MySQL ens ofereix també facilitats a l'hora d'avaluar les sentències SQL, gràcies a la sentència EXPLAIN.

Presentem primer l'execució d'una sentència SQL més o menys complexa:

```
mysql> select productes.clau, concat(part,' ',tipus,' ', especificacio) as producte, proveidors.empresa ,
preu , pagament from productes natural join preus natural join proveidors;
+-------+----------------------------+---------+--------+---------------+
| clau  | producte                   | empresa | preu   | pagament      |
+-------+----------------------------+---------+--------+---------------+
|  003  | Processador 1.7 GHz 64 bits | Tecno-k | 198.34 | crèdit        |
|  005  | RAM 128MB 333 MHz          | Tecno-k | 9.27   | crèdit        |
|  006  | RAM 256MB 400 MHz          | Tecno-k | 34.85  | crèdit        |
|  007  | Disc Dur 80 GB 7200 rpm    | Tecno-k | 59.95  | crèdit        |
|  010  | Disc Dur 40 GB 4200 rpm    | Tecno-k | 61.22  | crèdit        |
|  012  | Monitor 1024x876 60 Hz     | Tecno-k | 62.29  | crèdit        |
|  001  | Processador 2 GHz 32 bits  | Patito  | 30.40  | efectiu       |
|  002  | Processador 2.4 GHz 32 bits | Patito  | 33.63  | efectiu       |
|  003  | Processador 1.7 GHz 64 bits | Patito  | 195.59 | efectiu       |
|  005  | RAM 128MB 333 MHz          | Patito  | 9.78   | efectiu       |
|  006  | RAM 256MB 400 MHz          | Patito  | 32.44  | efectiu       |
|  007  | Disc Dur 80 GB 7200 rpm    | Patito  | 59.99  | efectiu       |
|  010  | Disc Dur 40 GB 4200 rpm    | Patito  | 62.02  | efectiu       |
|  001  | Processador 2 GHz 32 bits  | Nacional | 30.82 | crèdit,efectiu |
|  002  | Processador 2.4 GHz 32 bits | Nacional | 32.73 | crèdit,efectiu |
|  003  | Processador 1.7 GHz 64 bits | Nacional | 202.25 | crèdit,efectiu |
|  005  | RAM 128MB 333 MHz          | Nacional | 9.76   | crèdit,efectiu |
|  006  | RAM 256MB 400 MHz          | Nacional | 31.52  | crèdit,efectiu |
|  007  | Disc Dur 80 GB 7200 rpm    | Nacional | 58.41  | crèdit,efectiu |
|  010  | Disc Dur 40 GB 4200 rpm    | Nacional | 64.38  | crèdit,efectiu |
+-------+----------------------------+----------+--------+----------------+
20 rows in set (0.00 sec)
```

Ara utilitzem la sentència EXPLAIN perquè MySQL ens expliqui com ha realitzat aquesta consulta:

mysql> explain select productes.clau, concat(part, ',tipus,' ', especificació) as producte, proveïdors.empresa, preu, pagament from productes natural join preus natural join proveïdors;

```
+------------ +------- +-------------- +-------- +-------- +------------- + ------ + ----------- +
| table       | type   | possible_keys | key     | key_len | ref          | rows   | Extra       |
+------------ +------- +-------------- +-------- +-------- +------------- + ------ + ----------- +
| preus       | ALL    | NULL          | NULL    |  NULL   | NULL         | 20     |             |
| productes   | eq_ref | PRIMARY       | PRIMARY |    4    | preus.clau   |  1     |             |
| proveidors  | ALL    | PRIMARY       | NULL    |  NULL   | NULL         | 3      | where used |
+------------ +------- +-------------- +-------- +-------- +------------- + ------ + ----------- +
3 rows in set (0.00 sec)
```

En cada fila del resultat, ens explica com ha utilitzat els índexs de cada taula involucrada en la consulta. La columna 'type' ens indica el tipus de "join" que ha pogut fer. En el nostre cas, 'eq_ref', 'ref' o 'ref_or_null' indica que s'ha consultat una fila d'aquesta taula per a cada combinació de files de les altres. És un bon senyal, s'estan utilitzant els índexs, tal com indiquen la resta de columnes (en concret, l'atribut 'clau' que és la seva clau primària).

Veiem que en les altres dues taules, el tipus de 'join' és ALL. Això indica que el gestor ha hagut de llegir tota la taula per comprovar les condicions que li hem exigit en la consulta. En el cas de la taula proveïdors, hauria pogut utilitzar la clau primària ('possible_keys'), però no ho ha fet.

Intentarem millorar aquesta consulta. Veiem que en la taula preus no s'ha definit cap índex, cosa que facilitaria la tasca a l'SGBD:

```
mysql> alter table preus add index empresa_idx (empresa);
Query OK, 20 rows affected (0.00 sec)
Records: 20 Duplicates: 0 Warnings: 0

mysql> alter table preus add index clau_idx (clau);
Query OK, 20 rows affected (0.00 sec)
Records: 20 Duplicates: 0 Warnings: 0

mysql> explain select productes.clau, concat(part,' ',tipus,' ', especificacio) as producte,
proveïdors.empresa , preu , pagament from productes natural join preus natural join proveidors;
+------------+ ------ +---------------------+ ----------- +--------+ ------------- +-----+ ------ +
| table      | type   | possible_keys       | key         | key_len | ref           |rows  | Extra |
+------------+ ------ +---------------------+ ----------- +--------+ ------------- +-----+ ------ +
| proveïdors  | ALL    | PRIMARY             | NULL        | NULL    | NULL          | 3    |        |
| preus       | ref    | empresa_idx,clau_idex | empresa_idx | 20      | productes.emp | 7    |        |
| productes   | eq_ref | PRIMARY             | PRIMARY     | 4       | preus.clau    | 1    |        |
+------------+ ------ +---------------------+ ----------- +--------+ ------------- +-----+ ------ +
3 rows in set (0.00 sec)
```

Les coses han canviat de manera substancial. El gestor ha passat de llegir 24 files de dades, a llegir-ne 11. També ha canviat l'ordre de lectura de les taules, fent primer una lectura total de la taula proveïdors (que és inevitable, ja que no hem posat cap condició en el SELECT) i, després, ha aprofitat els índexs definits en 'preus' i en 'productes'.

Veurem més sobre els índexs en el subapartat "Anàlisi i optimització" d'aquesta unitat didàctica.

4.3. Manipulació de files

Per a la manipulació de files disposem de les sentències SQL INSERT, UPDATE i DELETE, l'ús i la sintaxi de les quals ja s'ha vist en el mòdul "El llenguatge SQL" d'aquest curs. En alguns casos, MySQL ens proporciona extensions o modificadors que ens poden ajudar molt en determinades situacions.

- INSERT [DELAYED]. Quan la sentència INSERT pot trigar molt a tornar el resultat (taules molt grans o amb molts índexs que s'han de recalcular en inserir una nova fila) pot ser interessant afegir la paraula clau DELAYED perquè MySQL ens torni el control i realitzi la inserció en segon pla.

- INSERT [[LOW_PRIORITY] | [HIGH_PRIORITY]]. En taules molt ocupades, en les quals molts clients realitzen consultes constantment, una inserció lenta pot bloquejar la resta de clients durant un temps. Mitjançant aquests modificadors podem variar aquest comportament.

- INSERT [IGNORE]. Aquest modificador converteix els errors d'inserció en avisos. Per exemple, si intentem inserir una fila que duplica una clau primària existent, l'SGBD ens tornarà un avís (i no inserirà la nova fila), però el nostre programa client podrà continuar amb la seva funció si el resultat de la inserció no era important per a la seva correcta execució.

- UPDATE [LOW_PRIORITY] [IGNORE]. Es comporten de la mateixa manera que en la sentència INSERT.

- DELETE [QUICK]. Esborra el/els registres sense actualitzar els índexs.

- TRUNCATE. És una manera molt ràpida d'esborrar tots els registres d'una taula, si no necessitem saber el nombre de registres que ha esborrat. DELETE FROM <tabla> té la mateixa funció, però torna el nombre de registres esborrats.

- LAST_INSERT_ID(). Torna l'últim identificador assignat a una columna de tipus AUTO_INCREMENT després d'una sentència INSERT.

5. Administració de MySQL

Les tasques administratives com la instal·lació, gestió d'usuaris, còpies de seguretat, restauracions, entre d'altres, són tasques ineludibles en qualsevol organització. Les polítiques, els recursos i les preferències dels administradors generen una gran varietat d'estils i mecanismes per dur a terme aquestes tasques, per la qual cosa no és possible parlar de mètodes completament estandarditzats en aquestes àrees.

En aquest apartat es revisen les opcions d'ús comú per a l'administració d'un servidor MySQL. Hi ha tantes alternatives que no és possible incloure-les totes en un curs. Per aquest motiu, en aquest capítol es tracten alguns temes d'importància per a l'administrador, des d'una perspectiva general, que permeten obtenir una visió global de les possibilitats pràctiques de les eines administratives.

En aquest sentit, el manual de MySQL és la referència principal per a trobar possibilitats i resoldre dubtes. En especial es recomana llegir els capítols següents:

- Capítol 2. Instal·lació de MySQL.
- Capítol 4. Administració bases de dades.
- Capítol 5. Optimització.

La informació continguda en aquests apartats és molt àmplia i clara, i representa una guia excel·lent per a resoldre dubtes. Així mateix, s'han de tenir en compte les llistes de correu incloses en el lloc oficial www.mysql.com.

Aquest capítol s'inicia amb una breu ressenya del procés d'instal·lació de MySQL. En l'actualitat és possible realitzar la instal·lació a partir de binaris empaquetats que faciliten enormement el procés. L'administració d'usuaris es tracta una mica més detalladament, incloent una breu descripció de les taules del directori de dades. Per als temes de còpies de seguretat i restauració es mostren les ordres i utilitats més usades en la pràctica ometent alguns detalls tècnics poc usuals. L'optimització es tracta de manera molt general, exposant els temes bàsics que en la pràctica es passen per alt.

Finalment, es descriu breument com realitzar la reproducció de dades en un servidor esclau.

5.1. Instal·lació de MySQL

La instal·lació de MySQL no comporta massa problemes, ja que moltes distribucions inclouen paquets amb què realitzar la instal·lació i configuració bàsica. Tanmateix, aquí veurem la instal·lació de MySQL utilitzant el codi font que es pot obtenir a www.mysql.com. Es pot destacar que l'ús d'una versió de MySQL compilada té l'avantatge que, probablement, s'adaptarà molt millor a l'entorn del servidor on s'executarà, proporcionant així un millor rendiment. Per contra,

implicarà més treball en cas que sorgeixin errors en la versió i l'hàgim d'actualitzar. Les instruccions que es descriuen en aquest apartat es basen en la documentació inclosa en la distribució.

En primer lloc, ens hem d'assegurar que comptem amb les llibreries i utilitats necessàries per a compilar els fitxers font. Principalment la llista de verificació ha d'incloure els fitxers següents:

- Compilador gcc
- Llibreries libgc

El procés d'instal·lació inclou els passos següents:

- Descomprimir els arxius font.

```
cd /usr/local/src
tar xzvf mysql-VERSION.tar.gz
cd mysql-VERSION
```

- Configurar la versió de MySQL que obtindrem. L'*script* 'configure' admet molts paràmetres que haurem d'examinar mitjançant l'opció '--help'. Segons els esquemes de taula que necessitem o extensions molt concretes que hàgim d'utilitzar, haurem d'examinar-ne amb compte les opcions. En la versió més simple l'executaríem de la manera següent:

```
./configure –prefix=/usr/local/mysql
```

- Compilar. Procedirem a compilar si no hi ha hagut problemes amb la configuració. El paràmetre -prefix especifica la ruta del sistema de fitxers on s'instal·larà.

```
make
```

- Instal·lar el sistema el servidor ja compilat, mitjançant la instrucció següent:

```
make install
```

- Crear la base de dades inicial del servidor, la que emmagatzemarà els usuaris i privilegis. Aquesta base de dades és imprescindible perquè els usuaris es puguin connectar al servidor.

```
scripts/mysql_istall_db
```

- Crear un nou usuari i el seu grup, perquè el servei s'executi en un entorn de privilegis restringit en el sistema operatiu. En cap cas no es recomana que l'usuari que executi el servei mysqld sigui root.

```
groupadd mysql
useradd -g mysql mysql
```

- Tots els arxius han de ser propietat de root (mysql no s'ha de poder modificar a ell mateix) i del grup mysql. El directori de dades serà de l'usuari mysql perquè pugui treballar amb les bases de dades, fitxers de registre, etc.

```
chown -R root /usr/local/mysql
chgrp -R mysql /usr/local/mysql
chown -R mysql /usr/local/mysql/var
```

- Crear l'arxiu de configuració. La distribució inclou diversos arxius de configuració que serveixen com a plantilla per a adaptar-lo a les nostres necessitats. En aquest cas, utilitzem la configuració mitjana com a plantilla. Opcionalment podem editar l'arxiu /etc/my.cnf.

```
cp support-files/my-medium.cnf /etc/my.cnf
```

- Llançar el servidor.

```
/usr/local/mysql/bin/mysql_safe &
```

- En aquest estat, el servidor no pot servir encara d'SGBD. Per defecte, tindrem creat un usuari '*root*' sense contrasenya que podrà accedir tant des de l'equip local com remotament. El pas següent serà assignar una contrasenya a aquest usuari i repassar els usuaris i privilegis definits. Per a assignar la contrasenya, haurem de fer el següent:

```
mysqladmin -u root password "novapasswd"
mysqladmin -u root -h host_name password "novapasswd"
```

Podem provar el funcionament de l'SGBD connectant amb el client 'mysql':

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

Vegem ara algunes característiques del servidor que acabem d'instal·lar:

- **mysqld**. El primer mètode és llançar-lo directament, se li poden especificar les opcions que l'administrador vulgui.
- **mysqld_safe**. És un *script* que executa *mysqld* i garanteix una configuració segura. És molt més recomanable que executar *mysqld* directament.
- **mysql_server**. És un guió que realitza dues tasques: iniciar el servidor *mysqld* i detenir-lo amb els paràmetres start i stop, respectivament. Utilitza *mysqld_safe* per a llançar el servidor *mysqld*. No és comú trobar-lo amb aquest nom, ja que generalment es copia com l'arxiu /etc/init.d/mysql
- **mysql_multi**. Permet l'execució de múltiples servidors de manera alterna.

Per a aturar el servidor bàsicament tenim dos mètodes:

- **/etc/init.d/mysql stop**. És el mecanisme estàndard en els sistemes tipus Unix. Encara que els directoris poden canviar.
- **$ mysqladmin -u root -p. shutdown**. És la utilitat per a realitzar tasques administratives en un servidor MySQL, en aquest cas li passem el paràmetre 'shutdown' per a aturar el servei.

Perquè els missatges del servidor apareguin en espanyol, s'ha d'executar amb el paràmetre *-language* adequat*:*

```
$ mysqld --language=spanish
```

Una altra opció és agregar a l'arxiu /etc/my.cnf una línia a la secció [mysqld]:

```
[mysqld]
language = /usr/share/mysql/spanish
```

5.2. Usuaris i privilegis

L'accés al servidor MySQL està controlat per usuaris i privilegis. Els usuaris del servidor MySQL no tenen cap correspondència amb els usuaris del sistema

operatiu. Encara que en la pràctica és comú que algun administrador de MySQL assigni els mateixos noms que els usuaris tenen en el sistema, són mecanismes totalment independents i sol ser aconsellable en general.

L'usuari administrador del sistema MySQL es diu *root*. Igual que el superusuari dels sistemes tipus Unix.

A més de l'usuari *root*, les instal·lacions noves de MySQL inclouen l'usuari anònim, que té permisos sobre la base de dades test. Si volem, també podem restringir-lo assignant-li una contrasenya. L'usuari anònim de MySQL es representa per una cadena buida. Veiem una altra manera d'assignar contrasenyes a un usuari, des del client de mysql i com a usuari *root*:

```
mysql> set password for ''@'localhost' = password('novapasswd');
```

L'administració de privilegis i usuaris en MySQL es realitza per mitjà de les sentències següents:

- **GRANT**. Atorga privilegis a un usuari; si no n'hi ha, es crearà l'usuari.
- **REVOKE**. Elimina els privilegis d'un usuari existent.
- **SET PASSWORD**. Assigna una contrasenya.
- **DROP USER**. Elimina un usuari.

5.2.1. La sentència GRANT

La sintaxi simplificada de GRANT consta de tres seccions. No se'n pot ometre cap i l'ordre és important:

- **grant** *llista de privilegis*
- **on** *base de dades.taula*
- **to** *usuari*

Exemple

Creació d'un nou usuari al qual s'atorga alguns privilegis

```
mysql> grant update, insert, select
-> on demo.preus
-> to visitant@localhost ;
```

En la primera línia s'especifiquen els privilegis que seran atorgats, en aquest cas es permet actualitzar (**update**), inserir (**insert**) i consultar (**select**). La segona línia especifica que els privilegis s'apliquen a la taula preus de la base de dades *demo*. En l'última línia s'indica el nom de l'usuari i l'equip des del qual es permetrà la connexió.

L'ordre **grant** crea el compte si no n'hi ha i, si n'hi ha, agrega els privilegis especificats. És possible assignar una contrasenya al compte al mateix temps que es crea i se li atorguen privilegis:

```
mysql> grant update, insert, select
-> on demo.preus
-> to visitant@localhost identified by 'novapasswd';
```

En la mateixa sentència és possible també atorgar permisos a més d'un usuari i assignar-los, o no, contrasenya:

```
mysql> grant update, insert, select
-> on demo.preus
-> to visitant@localhost,
-> yo@localhost identified by 'novapasswd',
-> elteu@equip.remot.com;
```

5.2.2. Especificació de llocs origen de la connexió

MySQL proporciona mecanismes per a permetre que l'usuari realitzi la seva connexió des de diferents equips dins d'una xarxa específica, només des d'un equip, o únicament des del mateix servidor.

```
mysql> grant update, insert, select
-> on demo.preus
-> to visitant@'%.empresa.com';
```

El caràcter % s'utilitza de la mateixa manera que en l'ordre **like**: substitueix qualsevol cadena de caràcters. En aquest cas, es permetria l'accés de l'usuari 'visitant' (amb contrasenya, si la tingués definida) des de qualsevol equip del domini 'empresa.com'. Cal observar que és necessari posar entre cometes el nom de l'equip origen a fi que sigui acceptat per MySQL. Igual que en **like**, es pot utilitzar el caràcter '_'.

Llavors, per a permetre l'entrada des de qualsevol equip a Internet, escriuríem:

```
-> to visitant@'%'
```

Obtindríem el mateix resultat ometent el nom de l'equip origen i escrivint simplement el nom de l'usuari:

```
-> to visitant
```

Els amfitrions vàlids també es poden especificar amb les seves adreces IP.

```
to visitant@192.168.128.10
to visitant@'192.168.128.%'
```

Els caràcters ' % ' i '_' no es permeten en els noms dels usuaris.

5.2.3. Especificació de bases de dades i taules

Després d'analitzar les opcions referents als llocs de connexió permesos, vegem ara com podem limitar els privilegis a bases de dades, taules i columnes.

En el següent exemple atorguem privilegis sobre totes les taules de la base de dades *demo*:

```
mysql> grant all
-> on demo.*
-> to 'visitant'@'localhost';
```

Podem obtenir el mateix resultat d'aquesta manera:

```
mysql> use demo;
mysql> grant all
-> on *
-> to 'visitant'@'localhost';
```

De la mateixa manera, en especificar només el nom d'una taula s'interpretarà que pertany a la base de dades en ús:

```
mysql> use demo;
mysql> grant all
-> on preus
-> to 'visitant'@'localhost';
```

Opcions per a la clàusula on de l'ordre grant

Opció	Significat
.	Totes les bases de dades i totes les taules
base.*	Totes les taules de la base de dades especificada
tabla	Taula especificada de la base de dades en ús
*	Totes les taules de la base de dades en ús

5.2.4. Especificació de columnes

A continuació presentem un exemple on s'especifiquen les columnes sobre les quals s'atorguen privilegis amb l'ordre **grant**:

```
mysql> grant update(preu,empresa)
-> on demo.preus
-> to visitant@localhost;
```

Podem especificar privilegis diferents per a cada columna o grups de columnes:

```
mysql> grant update(preu), select (preu, empresa)
-> on demo.preus
-> to visitant@localhost;
```

5.2.5. Tipus de privilegis

MySQL proporciona una gran varietat de tipus de privilegis.

- Privilegis relacionats amb taules: **alter**, **create**, **delete**, **drop**, **index**, **insert**, **select**, **update**.
- Privilegis administratius: **file**, **proccess**, **super reload**, **replication client**, **grant option**, **shutdown**.
- Privilegis per a finalitats diverses: **lock tables**, **show databases**, **create temporary tables**.

El privilegi **all** atorga tots els privilegis exceptuant el privilegi **grant option**. I el privilegi **usage** no n'atorga cap, la qual cosa és útil quan es vol, per exemple, simplement canviar la contrasenya:

```
grant usage
on *.*
to visitant@localhost identified by 'secret';
```

Tipus de privilegis en MySQL

Tipus de privilegi	Operació que permet
all [privileges]	Atorga tots els privilegis excepte **grant option**
usage	No atorga cap privilegi.
alter	Privilegi per a alterar l'estructura d'una taula.
create	Permet l'ús de **create table**
delete	Permet l'ús de **delete**
drop	Permet l'ús de **drop table**
index	Permet l'ús de **index** i **drop index**
insert	Permet l'ús d'**insert**
select	Permet l'ús de **select**
update	Permet l'ús d'**update**
file	Permet l'ús de **select . . . into outfile** i **load data infile**
process	Permet l'ús de **show full procces list**
super	Permet l'execució d'ordres de supervisió.
reload	Permet l'ús de **flush**
replication client	Permet preguntar la localització de mestre i esclau.
replication slave	Permet llegir els *binlog* del mestre.
grant option	Permet l'ús de **grant** i **revoke**
shutdown	Permet donar de baixa el servidor.
lock tables	Permet l'ús de **lock tables**
show tables	Permet l'ús de **show tables**
create temporary tables	Permet l'ús de **create temporary table**

En entorns grans, és freqüent trobar-se en la necessitat de delegar la feina d'administrar un servidor de bases de dades perquè altres usuaris, a més de l'administrador, puguin responsabilitzar-se d'atorgar privilegis sobre una base de dades particular. Això es pot fer en MySQL amb el privilegi **grant option**:

```
mysql> grant all, grant option
-> on demo.*
-> to operador@localhost;
```

El mateix resultat es pot obtenir amb la sintaxi alternativa següent:

```
mysql> grant all
-> on demo.*
-> to operador@localhost
-> with grant option;
```

D'aquesta manera, l'usuari *operador* podrà disposar de tots els privilegis sobre la base de dades *demo*, incloent-hi el de controlar l'accés a altres usuaris.

5.2.6. Opcions d'encriptació

MySQL pot establir connexions segures encriptant-les mitjançant el protocol SSL*; d'aquesta manera, les dades que es transmeten (tant la consulta, en un sentit, com el resultat, en l'altre) entre el client i el servidor estaran protegides contra intrusos. Per especificar que un usuari s'ha de connectar obligatòriament amb aquest protocol, s'utilitza la clàusula **require**:

** Secure Sockets Layer*

```
mysql> grant all
-> on *.*
-> to visitant@localhost
-> require ssl;
```

Les connexions encriptades ofereixen protecció contra el robatori d'informació, però representen una càrrega addicional per al servei, que ha de desencriptar la petició del client i encriptar la resposta (a més d'un procés més llarg de negociació en connectar), per això, minven el rendiment de l'SGBD.

5.2.7. Límits d'ús

Els recursos físics del servidor sempre són limitats: si es connecten molts usuaris alhora al servidor i realitzen consultes o manipulacions de dades complexes, és probable que pugui decaure el rendiment notablement. Una possible solució a aquest problema és limitar als usuaris el treball que poden demanar al servidor amb tres paràmetres:

- Màxim nombre de connexions per hora.
- Màxim nombre de consultes per hora.
- Màxim nombre d'actualitzacions per hora.

La sintaxi d'aquestes limitacions es mostra a continuació:

```
mysql> grant all
-> on *.*
-> to
-> with MAX_CONECTIONS_PER_HOUR 3
-> MAX_QUERIES_PER_HOUR 300
-> MAX_UPDATES_PER_HOUR 30;
```

5.2.8. Eliminar privilegis

L'ordre **revoke** permet eliminar privilegis atorgats amb **grant** als usuaris. Vegem-ne un exemple representatiu:

```
revoke all
on *.*
from visitant@localhost;
```

En executar aquesta ordre es retiren a l'usuari *visitant* tots els seus privilegis sobre totes les bases de dades, quan es connecta des de *localhost*.

L'ordre anterior no retira tots els privilegis de l'usuari *visitant*, només els hi retira quan es connecta des de *localhost*. Si l'usuari es connecta des d'una altra adreça (i tenia permís per a fer-ho) els seus privilegis romanen intactes.

5.2.9. Eliminar usuaris

Abans de procedir a l'eliminació d'un usuari, és necessari assegurar-se que se li han tret primer tots els privilegis. Una vegada assegurat aquest detall, es procedeix a eliminar-lo mitjançant l'ordre **drop user**:

```
mysql> drop user visitant;
```

5.2.10. La base de dades de privilegis: mysql

MySQL emmagatzema la informació sobre els usuaris i els seus privilegis en una base de dades com qualsevol altra, el nom de la qual és *mysql*. Si explorem la seva estructura, entendrem la manera com MySQL emmagatzema la informació dels usuaris i els seus privilegis:

```
mysql -u root -p
mysql> use mysql;
mysql> show tables;
+-----------------+
| Tables_in_mysql |
+-----------------+
| columns_priv    |
| db              |
| func            |
| host            |
| tables_priv     |
| user            |
+-----------------+
```

```
mysql> show columns from user;
+-----------------+------------------+------+-----+---------+-------+
| Field           | Type             | Null | Key | Default | Extra |
+-----------------+------------------+------+-----+---------+-------+
| Host            | char(60) binary  | PRI  |     |         |       |
| User            | char(16) binary  | PRI  |     |         |       |
| Password        | char(16) binary  |      |     |         |       |
| Select_priv     | enum('N','Y')    | N    |     |         |       |
| Insert_priv     | enum('N','Y')    |      | N   |         |       |
| Update_priv     | enum('N','Y')    | N    |     |         |       |
| Delete_priv     | enum('N','Y')    |      | N   |         |       |
| Create_priv     | enum('N','Y')    | N    |     |         |       |
| Drop_priv       | enum('N','Y')    | N    |     |         |       |
| Reload_priv     | enum('N','Y')    |      | N   |         |       |
| Shutdown_priv   | enum('N','Y')    | N    |     |         |       |
| Process_priv    | enum('N','Y')    | N    |     |         |       |
| File_priv       | enum('N','Y')    |      | N   |         |       |
| Grant_priv      | enum('N','Y')    |      | N   |         |       |
| References_priv | enum('N','Y')    |      | N   |         |       |
| Index_priv      | enum('N','Y')    |      | N   |         |       |
| Alter_priv      | enum('N','Y')    |      |N    |         |       |
+-----------------+------------------+------+-----+---------+-------+
17 rows in set (0.04 sec)

mysql> show columns from db;
+-----------------+------------------+------+-----+---------+-------+
| Field           | Type             | Null | Key | Default | Extra |
+-----------------+------------------+------+-----+---------+-------+
| Host            | char(60) binary  | PRI  |     |         |       |
| Db              | char(64) binary  | PRI  |     |         |       |
| User            | char(16) binary  | PRI  |     |         |       |
| Select_priv     | enum('N','Y')    |      | N   |         |       |
| Insert_priv     | enum('N','Y')    |      | N   |         |       |
| Update_priv     | enum('N','Y')    |      | N   |         |       |
| Delete_priv     | enum('N','Y')    |      | N   |         |       |
| Create_priv     | enum('N','Y')    |      | N   |         |       |
| Drop_priv       | enum('N','Y')    |      | N   |         |       |
| Grant_priv      | enum('N','Y')    |      | N   |         |       |
| References_priv | enum('N','Y')    |      | N   | N       |       |
| Index_priv      | enum('N','Y')    |      | N   |         |       |
| Alter_priv      | enum('N','Y')    |      | N   |         |       |
+-----------------+------------------+------+-----+---------+-------+
13 rows in set (0.00 sec)

mysql> show columns from tables_priv;
+-------------+-----------------------------------------------------------------------------------------------+----+---+-------+-----+
| Field       | Type                                                                                          |Null|Key|Default|Extra|
+-------------+-----------------------------------------------------------------------------------------------+----+---+-------+-----+
| Host        | char(60) binary                                                                               |PRI |   |       |     |
| Db          | char(64) binary                                                                               |PRI |   |       |     |
| User        | char(16) binary                                                                               |PRI |   |       |     |
| Table_name  | char(60) binary                                                                               |    | N |       |     |
| Grantor     | char(77)                                                                                      |    | N |       |     |
| Timestamp   | timestamp(14)                                                                                 |    | N |       |     |
| Table_priv  | set('Select','Insert','Update','Delete','Create','Drop','Grant','References','Index','Alter')|    | N |       |     |
| Column_priv | set('Select','Insert','Update','References')                                                  |N   |   |       |     |
+-------------+-----------------------------------------------------------------------------------------------+----+---+-------+-----+
8 rows in set (0.00 sec)
```

És possible fer modificacions directament sobre aquestes taules i obtenir els mateixos resultats que si utilitzéssim les ordres **grant**, **revoke**, **set password** o **drop user**:

```
mysql> update user
-> set Password = password('novapasswd')
-> where User ='visitant' and Host = 'localhost';
mysql> flush privileges;
```

L'ordre **flush privileges** sol·licita a MySQL que torni a llegir les taules de privilegis. En el moment d'executar-se, el servidor llegeix la informació d'aquestes taules. Però si les taules s'han alterat manualment, no s'assabentarà dels canvis fins que utilitzem l'ordre **flush privileges**.

Taules de la base de dades *mysql*

Taula	Contingut
user	Comptes d'usuari i els seus privilegis globals
db	Privilegis sobre bases de dades
tables_priv	Privilegis sobre taules
columns_priv	Privilegis sobre columnes
host	Privilegis d'altres equips amfitrions sobre bases de dades

L'accés directe a les taules de privilegis és útil en diversos casos; per exemple, per a esborrar un usuari del sistema en les versions de MySQL anteriors a la 4.1.1:

```
mysql> delete from user
-> where User = 'visitant' and Host = 'localhost';
mysql> flush privileges;
```

No és possible eliminar tots els privilegis d'un usuari mitjançant una sola ordre **revoke**.

Exemple

S'atorguen drets a un usuari amb dues ordres **grant**.

Si observem el contingut de la base de dades de privilegis, podem entendre el comportament de les ordres **grant** i **revoke**. Primer assignem privilegis per usar l'ordre **select** a l'usuari visitant amb dues ordres **grant**: la primera li permet l'ingrés des del servidor *lanostra-ong.org* i la segona li atorga el mateix tipus de privilegi, però des de qualsevol equip a Internet.

```
mysql> grant select
-> on *.*
-> to visitant@lanostra-ong.org;
Query OK, 0 rows affected (0.01 sec)
mysql> grant select
-> on *.*
-> to visitant@'%';
Query OK, 0 rows affected (0.00 sec)
```

Consultant la taula *user* de la base de dades de privilegis, podem observar els valors 'Y' a la columna del *privilegi select*.

```
mysql> select user,host,select_priv from user
-> where user = 'visitant';
```

```
+-----------+-----------------+-------------+
|   user    |      host       | select_priv |
+-----------+-----------------+-------------+
| visitant  | lanostra-ong.org|      Y      |
| visitant  |        %        |      Y      |
+-----------+-----------------+-------------+
2 rows in set (0.00 sec)
```

Ara sol·licitem eliminar el *privilegi select* de totes les bases de dades i de tots els equips a Internet:

```
mysql> revoke all
-> on *.*
-> from visitant@'%';
Query OK, 0 rows affected (0.00 sec)
mysql> select user,host,select_priv from user
-> where user = 'visitant';
```

```
+-----------+-----------------+-------------+
|   user    |      host       | select_priv |
+-----------+-----------------+-------------+
| visitant  | lanostra-ong.org|      Y      |
| visitant  |        %        |      N      |
+-----------+-----------------+-------------+
2 rows in set (0.01 sec)
```

En la taula *user* observem que, efectivament, s'ha eliminat el privilegi per a visitant@'%', però no per a 'visitant@lanostra-ong.org'. MySQL considera que són adreces diferents i respecta els privilegis atorgats a l'un quan es modifica l'altre.

5.3. Còpies de seguretat

Cap sistema no és perfecte ni està fora de perill d'errors humans, talls en el subministrament del corrent elèctric, desperfectes en el maquinari o errors de programari; així que una tasca més que recomanable de l'administrador del servidor de bases de dades és realitzar còpies de seguretat i dissenyar un pla de contingència. S'han de fer assajos del pla per a assegurar-ne el bon funcionament i, si s'hi descobreixen anomalies, fer els ajustos necessaris.

No hi ha una recepta universal que ens indiqui com s'han de portar les nostres còpies de seguretat de dades. Cada administrador ha de dissenyar la manera de fer-ho del seu sistema d'acord amb les seves necessitats, recursos, riscos i el valor de la informació.

MySQL ofereix diverses alternatives de còpia de seguretat de la informació. La primera que podem esmentar consisteix simplement a copiar els arxius de dades. Efectivament, és una opció vàlida i senzilla.

En primera instància són necessaris dos requisits per a dur-la a terme:

- Conèixer la ubicació i estructura del directori de dades.
- Parar el servei MySQL mentre es realitza la còpia.

Quant a la ubicació i estructura del directori, recordem que la distribució de MySQL ubica el directori de dades en /usr/local/mysql/var, les distribucions GNU/Linux basades en paquets com DEB o RPM ubiquen, en general, les dades en /var/lib/mysql.

Si per algun motiu no trobem el directori de dades, podem consultar-ne la ubicació a MySQL. L'ordre **show variables** ens mostra totes les variables disponibles, n'hi ha prou de fer un filtre amb la clàusula **like**:

```
mysql> show variables like 'datadir';
+---------------+-----------------+
| Variable_name |      Value      |
+---------------+-----------------+
|    datadir    | /var/lib/mysql/ |
+---------------+-----------------+
1 row in set (0.00 sec)
```

Una vegada localitzats els arxius, detenim l'execució del servidor; una manera senzilla d'assegurar-nos que la base de dades no serà modificada mentre acabem la còpia:

```
$ mysqladmin -u root -p shutdown
```

Finalment, copiem el directori complet amb totes les bases de dades:

```
$ cp -r /var/lib/mysql/ /algun_dir/
```

Podem triar altres maneres de copiar-lo o comprimir-lo, d'acord amb les nostres preferències i necessitats:

```
$ tar czf mysql-backup.tar.gz /var/lib/mysql
```

Si volem copiar només una base de dades, copiem el directori amb el mateix nom de la base de dades:

```
$ cp -r /var/lib/mysql/demo/ /algun_dir/suport_demo/
```

També és possible fer còpia de seguretat d'una sola taula.

```
$ cp -r /var/lib/mysql/demo/productes.* / algun_dir/backup_demo/
```

Com podem observar, l'organització de la base de dades en MySQL és molt simple:

- Totes les bases de dades s'emmagatzemen en un directori, anomenat directori de dades(*datadir*).

- Cada base de dades s'emmagatzema com un subdirectori del directori de dades.
- Cada taula s'emmagatzema en un arxiu, acompanyat d'altres arxius auxiliars amb el mateix nom i diferent extensió.

El problema d'aquest mecanisme és que hem d'aturar el servei de bases de dades mentre realitzem el suport.

5.3.1. mysqlhotcopy

Un mecanisme que permet realitzar la còpia dels arxius del servidor sense necessitat de detenir el servei és l'*script* '*mysqlhotcopy*'. L'*script* està escrit en Perl i bloqueja les taules mentre realitza el suport per evitar-ne la modificació. S'usa de la manera següent:

```
$ mysqlhotcopy demo /algun_directori
```

En aquest cas, crearà un directori /algun_directori/demo amb tots els arxius de la base de dades.

L'ordre *mysqlhotcopy* pot rebre només el nom d'una base de dades com a paràmetre:

```
$ mysqlhotcopy demo
```

En aquest cas, crearà un directori /var/lib/mysql/demo_copy.

Aquest mètode no funciona per a taules amb el mecanisme d'emmagatzemament tipus InnoDB.

5.3.2. mysqldump

Les dues opcions anteriors representen còpies binàries de la base de dades. L'ordre *mysqldump*, en canvi, realitza un bolcatge de les bases de dades però traduint-les a SQL; és a dir, lliura un arxiu de text amb totes les ordres necessàries per a tornar a reconstruir les bases de dades, les seves taules i les seves dades. És el mètode més útil per a copiar o distribuir una base de dades que s'haurà d'emmagatzemar en altres servidors.

```
$ mysqldump demo > demo.sql
```

L'ordre *mysqldump* ofereix multitud de paràmetres per a modificar-ne el comportament o el tipus de bolcatge generat: per defecte, genera sentències SQL, però pot generar fitxers de dades tipus CSV o altres formats. També podem especificar-li que faci el bolcatge de totes les bases de dades o que només bolqui les dades i no la creació de les taules, etc.

Les primeres línies de l'arxiu *demo.sql* segons l'exemple anterior tindrien l'aspecte següent:

```
~$ mysqldump demo | head -25
-- MySQL dump 8.21
--
-- Host: localhost Database: demo
---------------------------------------------------------
-- Server version 3.23.49-log
--
-- Table structure for table 'guany'
--
DROP TABLE IF EXISTS guany;
CREATE TABLE guany (
venda enum('A l'engròs','Al detall') default NULL,
factor decimal(4,2) default NULL
) TYPE=MyISAM;
--
Dumping data for table 'guany'
--
INSERT INTO guany VALUES ('a l'engròs',1.05);
INSERT INTO guany VALUES ('al detall',1.12);--
```

L'avantatge d'utilitzar *mysqldump* és que permet que els arxius puguin ser llegits (i modificats) en un simple editor de textos, i poden ser utilitzats per a migrar la informació a un altre SGBD que suporti SQL. A més, suporta tots els tipus de taules. El desavantatge és que el seu processament és lent i els arxius que s'obtenen són molt grans.

5.3.3. Restaurar a partir de suports

En algun moment, sigui pel motiu que sigui, necessitarem realitzar la restauració de les nostres bases de dades.

Si tenim una còpia binària del directori de dades, n'hi haurà prou de copiar-la al directori original i reiniciar el servidor:

```
# mysqladmin -u root -p shutdown
# cp /algun_dir/suport-mysql/* /var/lib/mysql
# chown -R mysql.mysql /var/lib/mysql
# mysql_safe
```

És important restaurar també l'amo i el grup dels arxius de dades, per a tenir els accessos correctament establerts. En aquest exemple s'adopta el supòsit que l'usuari mysql és el que executa el servidor *mysqld*.

La restauració d'un arxiu SQL obtingut amb *mysqldump*, es realitza des del client *mysql*, la base de dades ha d'existir, ja que l'arxiu *demo.sql* no la crea per defecte:

```
$ mysql demo -u root -p < demo.sql
```

5.4. Reparació de taules

En determinades circumstàncies d'ús molt freqüent, com la inserció i esborrament massius de dades, coincidint amb bloquejos del sistema o ompliment de l'espai en disc o altres circumstàncies, és possible que una taula o alguns dels seus índexs es corrompin.

Podem consultar l'estat d'integritat d'una taula amb l'ordre **check table**, que realitza algunes verificacions sobre la taula a la recerca d'errors i ens lliura un informe amb les columnes d'informació següents:

```
mysql> check table preus;
+--------------+-------+----------+----------+
|    Table     |  Op   | Msg_type | Msg_text |
+--------------+-------+----------+----------+
| demo.preus   | check |  status  |    OK    |
+--------------+-------+----------+----------+
1 row in set (0.03 sec)
```

La columna *Op* descriu l'operació que es realitza sobre la taula. Per a l'ordre **check table** aquesta columna sempre té el valor *check* perquè aquesta és l'operació que es realitza. La columna *Msg_type* pot contenir un dels valors *status*, *error*, *info*, o *warning*. I la columna *Msg_text* és el text que reporta d'alguna situació trobada en la taula.

És possible que la informació lliurada inclogui diverses files amb diversos missatges, però l'últim missatge sempre ha de ser el missatge *OK* de tipus *status*.

En altres ocasions, **check table** no farà la verificació de taula, al seu lloc lliurarà com a resultat el missatge *Table is already up to date*, que significa que el gestor de la taula indica que no cal revisar-la.

MySQL no permet realitzar consultes sobre una taula danyada i enviarà un missatge d'error sense desplegar resultats parcials:

```
mysql> select * from preus;
ERROR 1016: No puc obrir arxiu: 'preus.MYD'. (Error: 145)
```

Per a obtenir informació del significat de l'error 145, usarem la utilitat en línia d'ordres *perror*:

```
$ perror 145
145 = Table was marked as crashed and should be repaired
```

Després d'un missatge com l'anterior, és el moment de realitzar una verificació de la integritat de la taula per a obtenir el report.

```
mysql> check table preus extended;
+-------------+------+----------+----------------------------------------+
|    Table    |  Op  | Msg_type |              Msg_text                  |
+-------------+------+----------+----------------------------------------+
|demo.preus   |check |  error   |Size of datafile is:450 Should be:452 |
|demo.preus   |check |  error   |               Corrupt                  |
+-------------+------+----------+----------------------------------------+
2 rows in set (0.00 sec)
```

En aquest cas localitzem dos errors en la taula. L'opció **extended** és un dels cinc nivells de comprovació que es pot sol·licitar per a verificar una taula.

Tipus de verificació

Tipus	Significat
quick	No revisa les files a la recerca de referències incorrectes.
fast	Només verifica les taules que no van ser tancades adequadament.
changed	Verifica només les taules modificades des de l'última verificació o que no s'han tancat apropiadament.
medium	Revisa les files per verificar que els esborraments lligats són correctes, verifica les sumes de comprovació de les files.
extended	Realitza una recerca completa en totes les claus de cada columna. Garanteix el 100% de la integritat de la taula.

La sentència **repair table** realitza la reparació de taules tipus MyISAM corruptes:

```
mysql> repair table preus;
+-------------+------+---------+----------------------------------------+
|    Table    |  Op  | Msg_type|              Msg_text                  |
+-------------+------+---------+----------------------------------------+
|demo.preus   |repair|   info  | Wrong bytesec: 0-17-1 at 168;Skipped |
|demo.preus   |repair| warning | Number of rows changed from 20 to 7  |
|demo.preus   |repair|  status |                  OK                    |
+-------------+------+---------+----------------------------------------+
3 rows in set (0.04 sec)
```

El segon missatge informa de la pèrdua de 13 files durant el procés de reparació. Això significa, com és natural, que l'ordre **repair table** és útil només en casos d'extrema necessitat, ja que no garanteix la recuperació total de la informació. A la pràctica, sempre és millor realitzar la restauració de la informació utilitzant les còpies de seguretat. En cas de desastre, s'ha de conèixer el motiu que origina la corrupció de les taules i prendre les mesures adequades per a evitar-ho. Respecte a l'estabilitat de MySQL, es pot confiar en el fet que molt probablement mai no serà necessari utilitzar l'ordre **repair table**.

L'ordre **optimize table** pot realitzar també algunes correccions sobre una taula.

5.4.1. myisamchk

El programa **myisamchk** és una utilitat en línia d'ordres que s'inclou amb la distribució de MySQL i serveix per a reparar taules tipus MyISAM. Per a utilitzar-lo amb seguretat el servidor no s'ha d'estar executant i es recomana realitzar un suport del directori de dades abans de la seva utilització.

myisamchk rep com a paràmetre principal els arxius .MYI corresponents a les taules que cal revisar; és a dir, no coneix la ubicació del directori de dades. Per exemple, si el directori de dades està ubicat en /val/lib/mysql, les següents serien dues maneres de realitzar una comprovació dels arxius de la base de dades *demo*:

```
# myisamchk /var/lib/mysql/demo/*.MYI
# cd /var/lib/lib/mysql/demo
# myisamchk *.MYI
```

Es poden revisar totes les bases de dades utilitzant '*' per a denominar el directori de la base de dades:

```
# myisamchk /var/lib/mysql/*/*.MYI
```

Per a realitzar una comprovació ràpida, el manual suggereix utilitzar l'ordre següent:

```
# myisamchk --silent --fast *.MYI
```

I per a realitzar la correcció de les taules corruptes, el manual suggereix la sintaxi següent:

```
# myisamchk --silent --force --update-state -O key_buffer=64M \
-O sort_buffer=64M -O read_buffer=1M -O write_buffer=1M *.MYI
```

Les opcions donades per -O es refereixen a l'ús de memòria, que permeten accelerar de manera notòria el procés de reparació:

`--force` reinicia myisamchk amb el paràmetre `--recover` quan troba algun error.

`--updatestate` emmagatzema informació sobre el resultat de l'anàlisi en la taula MYI.

Nota

A la pràctica amb aquestes opcions s'aconsegueixen corregir els errors més comuns. Per a conèixer altres opcions de recuperació amb **myisamchk**, podeu consultar el manual que acompanya la distribució de MySQL.

5.5. Anàlisi i optimització

El disseny de MySQL li permet funcionar amb un rendiment notable, tanmateix, es poden cometre fàcilment errors que disminueixin la capacitat de resposta del servidor. També es poden realitzar alguns ajustos a la configuració de MySQL que n'incrementa el rendiment.

5.5.1. Indexació

La indexació és la principal eina per a optimitzar el rendiment general de qualsevol base de dades. És també la més coneguda pels usuaris de servidors MySQL i, paradoxalment, la seva no-utilització és una de les principals causes de baix rendiment en servidors de bases de dades.

> Molts administradors i dissenyadors simplement semblen oblidar usar índexs per a optimitzar els accessos a les bases de dades. D'altra banda, algunes persones tendeixen a indexar-ho tot, esperant que d'aquesta manera el servidor acceleri qualsevol tipus de consulta que se li sol·liciti. En realitat, aquesta pràctica pot causar una disminució en el rendiment, sobretot respecte a insercions i modificacions.

Per a veure els avantatges d'utilitzar índexs, analitzarem en primer terme una simple recerca en una taula sense cap índex:

- El constant accés d'escriptura d'una taula la manté desordenada.
- L'ordenació d'una taula és una operació costosa: el servidor s'hauria d'aturar un temps considerable per ordenar les seves taules.
- Moltes taules tenen més d'un criteri d'ordenació: ordenar segons una columna implica desordenar-ne una altra.
- La inserció i eliminació de dades sense alterar l'ordre en una taula és costosa: la inserció d'un registre en una taula gran implicaria una llarga espera en la seva actualització.
- Si s'opta per mantenir la taula desordenada (que és l'opció més viable), una recerca implicaria forçosament un recorregut seqüencial (també denominat *full scan*), registre per registre.

L'ús d'índexs en l'ordenació de les bases de dades ofereix els avantatges següents:

- Permet ordenar les taules per diversos criteris simultàniament.
- És menys costós ordenar un arxiu índex, perquè inclou només referències a la informació i no la informació pròpiament.
- El cost d'inserció i eliminació és més baix.
- Amb els registres sempre ordenats s'utilitzaran algoritmes molt més eficients que el simple recorregut seqüencial en les consultes.

L'ús d'índexs també comporta algun desavantatge:

- Els índexs ocupen espai en disc.

- Tot i tenir registres petits, el fet de mantenir en ordre un índex disminueix la velocitat de les operacions d'escriptura sobre la taula.

Malgrat aquests inconvenients, la utilització d'índexs ofereix més avantatges que desavantatges, sobretot en la consulta de múltiples taules, i l'augment de rendiment és més gran com més gran és la taula.

Considerem, per exemple, una consulta sobre les taules A, B, i C, independentment del contingut de la clàusula **where**, les tres taules s'han de combinar per a fer possible posteriorment el filtratge segons les condicions donades:

```
select *
from A,B,C
where A.a = B.b
and B.b = C.c;
```

Considerem que no són taules grans, que no sobrepassen els 1.000 registres. Si A té 500 registres, B en té 600 i C, 700, la taula resultant de la consulta anterior tindrà 210 milions de registres. MySQL faria el producte cartesià de les tres taules i, posteriorment, es recorreria la relació resultant per buscar els registres que satisfan les condicions donades, encara que al final el resultat inclogui només 1.000 registres.

Si utilitzem índexs MySQL, els empraria d'una manera semblant a la següent:

- Prendria cada un dels registres de A.

- Per a cada registre de A, buscaria els registres en B que complissin la condició A.a = B.b. Com que B està indexat per l'atribut 'b', no necessitaria fer el recorregut de tots els registres, simplement accediria directament al registre que complís la condició.

- Per a cada registre de A i B trobat en el pas anterior, buscaria els registres de C que complissin la condició B.b = C.c. És el mateix cas que en el pas anterior.

Comparant les dues alternatives de recerca, la segona és la que ocuparia prop del 0,000005% del temps original. Per descomptat que només es tracta d'una aproximació teòrica, però adequada per a comprendre l'efecte dels índexs en les consultes sobre bases de dades.

5.5.2. Equilibri

L'índex ideal hauria de tenir les característiques següents:

- Els registres haurien de ser tan petits com sigui possible.
- Només s'han d'indexar valors únics.

Analitzem cada recomanació:

- Com més petits siguin els registres, més ràpidament es podran canviar de lloc (en inserir files, modificar-les o esborrar-les), a més, en un moment determinat, l'índex pot romandre en memòria. Considerem les dues definicions possibles:

```
create table Empresa(
nombre char(30),
telèfon char(20),
index (nom)
);
```

En aquesta taula l'índex es realitza sobre *nom*, que és un camp de 30 caràcters, i s'utilitza com a clau per a fer els '*joins*' amb altres taules.

Ara considereu l'alternativa següent:

```
create table Empresa(
id int ,
nom char(30),
telèfon char(20),
index (id)
);
```

S'agrega una columna que servirà com a identificador de l'empresa. Des del punt de vista de rendiment implica una millora, ja que l'índex es realitza sobre nombres enters, per tant, ocuparà menys espai i funcionarà més ràpid.

Com més petita sigui la columna indexada, més velocitat es tindrà en l'accés a la taula.

- Considerem l'índex següent, creat per a disminuir la necessitat d'efectuar accessos a la taula:

```
create table Empresa(
nom char(30),
crèdit enum{'SI','NO'},
index(crèdit)
);
```

Si considerem que un índex es crea per a evitar la necessitat de recórrer la taula, veurem que l'índex creat és pràcticament inútil, ja que algun dels valors ocorre el 50% o més de les vegades: per a trobar tots els resultats, cal recórrer gran part de la taula. MySQL no utilitza els índexs que impliquen un 30% d'ocurrències en una taula.

Tot i així, i exceptuant casos exagerats com aquest últim, pot ser interessant indexar una taula per algun atribut que no sigui únic, si aquest atribut s'utilitza per a ordenar els resultats. També pot ser convenient crear un índex per diversos atributs simultàniament si s'usen tots en alguna consulta en la clàusula ORDER BY.

Com més petita sigui la repetició de valors en una columna indexada, hi haurà menys necessitat d'accedir a la taula i més eficient serà l'índex.

5.5.3. La *cache* de consultes de MySQL

El servidor MySQL inclou la possibilitat d'utilitzar una *cache** amb els resultats de les últimes consultes per a accelerar la velocitat de resposta. Aquesta solució és útil quan les taules tenen relativament pocs canvis i es realitzen els mateixos tipus de consultes. El funcionament de la *cache* es basa en les premisses següents:

*** Memòria intermèdia d'accés ràpid.**

- La primera vegada que es rep una consulta s'emmagatzema en la *cache*.
- Les següents vegades, la consulta es realitza primer en la *cache*; si té èxit, el resultat s'envia immediatament.

La *cache* té les característiques següents:

- El servidor compara el text de la consulta; encara que tècnicament sigui igual si difereix en ús de majúscules-minúscules o qualsevol altre canvi, no es considera la sol·licitud idèntica i no serà tractada per la *cache*.
- Si alguna taula inclosa en alguna consulta canvia, el contingut de la consulta és eliminat de la *cache*.

La configuració de la *cache* es fa per mitjà de variables globals:

- **query_cache_limit**. No emmagatzema resultats que sobrepassin aquesta mida. Per omissió és d'1 M.
- **query_cache_size**. Mida de la memòria cau expressada en bytes. Per omissió és 0; és a dir, no hi ha memòria cau.
- **query_cache_type**. Pot tenir tres valors: ON, OFF o DEMAND.

Tipus de *cache*

Valor	Tipus	Significat
0	OFF	*Cache* desactivat
1	ON	*Cache* activat
2	DEMAND	Només sota sol·licitud explícita

Quan la memòria cau del servidor està en mode DEMAND, s'ha de sol·licitar explícitament que la consulta utilitzi o no la memòria cau:

```
select sql_cache
select sql_no_cache
```

5.6. Replicació

La reproducció és la còpia sincronitzada entre dos servidors de bases de dades de manera que qualsevol dels dos pot lliurar els mateixos resultats als seus clients.

MySQL inclou la possibilitat de reproducció amb les característiques següents:

- Funciona amb l'esquema *mestre-esclau*: existeix un servidor mestre que porta el control central i un o diversos servidors esclaus que es mantenen sincronitzats amb el servidor mestre.

- La rèplica es realitza mitjançant un registre dels canvis realitzats en la base de dades: no es realitzen les còpies de les bases de dades per a mantenir-les sincronitzades, al seu lloc s'informa de les operacions realitzades al servidor mestre (insert, delete, update...) perquè les realitzin al seu torn els servidors esclaus.

- No és possible fer canvis en els servidors esclaus, són exclusivament per a consultes.

Aquest esquema senzill permet la creació de rèpliques sense més complicacions obtenint els beneficis següents:

- Es distribueix la càrrega de treball.

- El sistema és redundant, per la qual cosa en cas de desastre hi ha menys probabilitats de perdre les dades.

- És possible realitzar els suports d'un esclau sense interrompre el treball del servidor mestre.

5.6.1. Preparació prèvia

L'equip mestre ha de tenir accés per xarxa. Abans de realitzar la configuració dels servidors mestre i esclau és necessari realitzar les tasques següents:

- Assegurar-se que en tots dos està instal·lada la mateixa versió de MySQL.
- Assegurar-se que cap dels servidors no atendrà peticions durant el procés de configuració.
- Assegurar-se que les bases de dades del servidor mestre han estat copiades manualment al servidor esclau, de manera que en tots dos es trobi exactament la mateixa informació.
- Assegurar-se que tots dos atenen connexions per TCP/IP. Per seguretat, aquesta opció està desactivada per omissió. Per a activar-la s'ha de comentar la línia *skip_networking en* l'arxiu de configuració /etc/my.cnf

5.6.2. Configuració del servidor mestre

En el servidor mestre creem un compte d'usuari amb permisos de reproducció per a autoritzar, al servidor mestre, el nou usuari per realitzar rèpliques:

```
mysql> grant replication slave
-> on *.*
-> to replicador@esclau.empresa.com identified by 'secret';
```

Reproductor és el nom del nou usuari.

Esclau.empresa.com és l'adreça del servidor esclau.

'Secret' és la contrasenya.

El servidor mestre portarà un arxiu de registre '*binlog*' on es registraran totes les sol·licituds d'actualització que es realitzin en les bases de dades. Per a activar la creació d'aquest arxiu hem d'editar l'arxiu /etc/my.cnf i agregar les següents línies en la secció [mysqld]:

```
[mysqld]
log-bin
server-id = 1
```

El servidor mestre s'ha d'identificar amb un id; en aquest cas, serà el número 1. A continuació, reiniciem el servidor:

```
/etc/init.d/mysql restart
```

Finalment, consultem el nom de l'arxiu '*binlog*' i la posició de compensació (aquestes dades són necessàries per a configurar l'esclau):

```
mysql> show master status;
+-----------------+----------+--------------+------------------+
|      File       | Position | Binlog_do_db | Binlog_ignore_db |
+-----------------+----------+--------------+------------------+
| mestre-bin.001  |    76    |              |                  |
+-----------------+----------+--------------+------------------+
```

5.6.3. Configuració del servidor esclau

Al servidor esclau, editem l'arxiu /etc/my.cnf i agreguem, igual que en el mestre, l'activació de l'arxiu '*binlog*' i un identificador del servidor (que ha de ser diferent de l'identificador del servidor mestre):

```
[mysqld]
log-bin
server-id = 2
```

Reiniciem el servidor esclau:

```
# /etc/init.d/mysql restart
```

Configurem les dades del mestre al servidor esclau:

```
mysql> change master to
-> master_host = 'mestre.empresa.com',
-> master_user = 'replicador',
-> master_password = 'secret',
-> master_log_file = 'mestre-log.001',
-> master_log_pos = 76;
```

L'últim pas és iniciar el servidor esclau:

```
mysql> start slave;
```

I ja tindrem el servidor esclau funcionant.

5.7. Importació i exportació de dades

Moltes vegades és necessari moure dades d'una aplicació a l'altra, per a això són necessaris formats estàndard que puguin ser escrits per l'aplicació origen i llegits per l'aplicació destinació. El més simple d'aquests formats és el text pla, on cada arxiu és una taula, cada fila és un registre i els valors dels camps se separen per tabuladors.

MySQL pot llegir aquest tipus d'arxius, incloent valors nuls representats per '\N'(N majúscula).s

Utilitzant el client *mysql*, podem introduir les dades de l'arxiu local *proveïdors.txt* en la taula *proveïdors*:

```
mysql> load data local infile 'proveïdors.txt'
-> into table proveïdors;
```

Si s'omet la paraula **local**, MySQL buscarà l'arxiu en el servidor i no en el client.

En un arxiu es poden posar entre cometes els camps, utilitzar comes per a separar-los i acabar les línies amb els caràcters '\r\n' (com en els arxius Windows). L'ordre **load data** té dues clàusules opcionals, **fields**, en les quals s'especifiquen aquests paràmetres:

```
mysql> load data local infile 'prooveedors.txt'
-> fields terminated by ','
-> enclosed by '"'
-> lines terminated by '\r\n';
```

L'opció **enclosed by** pot tenir la forma **optionaly enclosed by**, en cas que els camps numèrics no siguin delimitats.

A més, es poden ometre les primeres línies de l'arxiu si contenen informació d'encapçalaments:

```
mysql> load data local infile 'proveïdors.txt'
-> ignore 1 lines;
```

5.7.1. mysqlimport

La utilitat **mysqlimport** que s'inclou en la distribució pot fer la mateixa feina que **load data**. Aquests són alguns dels seus paràmetres:

```
mysqlimport basededades arxiu.txt
```

Aquests són alguns dels arguments de **mysqlimport** per a realitzar les tasques equivalents a la sentència **load data**:

```
--fields-terminated-by=
--fields-enclosed-by=
--fields-optionally-enclosed-by=
--fields-escaped-by=
--lines-terminated-by=
```

La manera més simple per a exportar dades és readreçant la sortida del client *mysql*. El paràmetre -e permet executar una ordre en mode de processament per lots. MySQL detecta si la sortida és en pantalla o està readreçada a un arxiu i tria la presentació adequada: amb encapçalaments i línies de separació per a la sortida en pantalla, i sense encapçalaments i amb tabuladors per a un arxiu:

```
$ mysql demo -e "select * from proveïdors" > proveïdors.txt
```

La sentència **select** també té una opció per a realitzar la tasca inversa de la sentència **load data**:

```
mysql> select *
-> into outfile "/tmp/proveïdors.txt"
-> fields termitated by ','
-> optionaly enclosed by '"'
-> lines termitates by '\n'
-> from proveidors;
```

5.7.2. mysqldump

La utilitat **mysqldump** realitza el bolcatge de bases de dades i es pot utilitzar per a transportar dades d'una base a una altra que també entengui SQL. Tanmateix, l'arxiu ha de ser editat abans d'utilitzar-se, ja que algunes opcions són

exclusives de MySQL. En general, n'hi ha prou d'eliminar el tipus de taula que s'especifica al final d'una ordre **create table**.

L'ordre següent realitza el buidatge complet de la base de dades *demo*:

```
$ mysqldump demo > demo.sql
```

En alguns casos, les ordres **insert** són suficients i no necessitem les definicions de les taules.

L'ordre següent realitza un buidatge de la taula *proveïdors* de la base de dades *demo* filtrant la sortida amb l'ordre *grep* de Unix que selecciona només les línies que contenen la paraula INSERT. D'aquesta manera, l'arxiu *proveïdors-insert.txt* conté exclusivament ordres **insert**:

```
$ mysqldump demo proveidors | grep INSERT
```

6. Clients gràfics

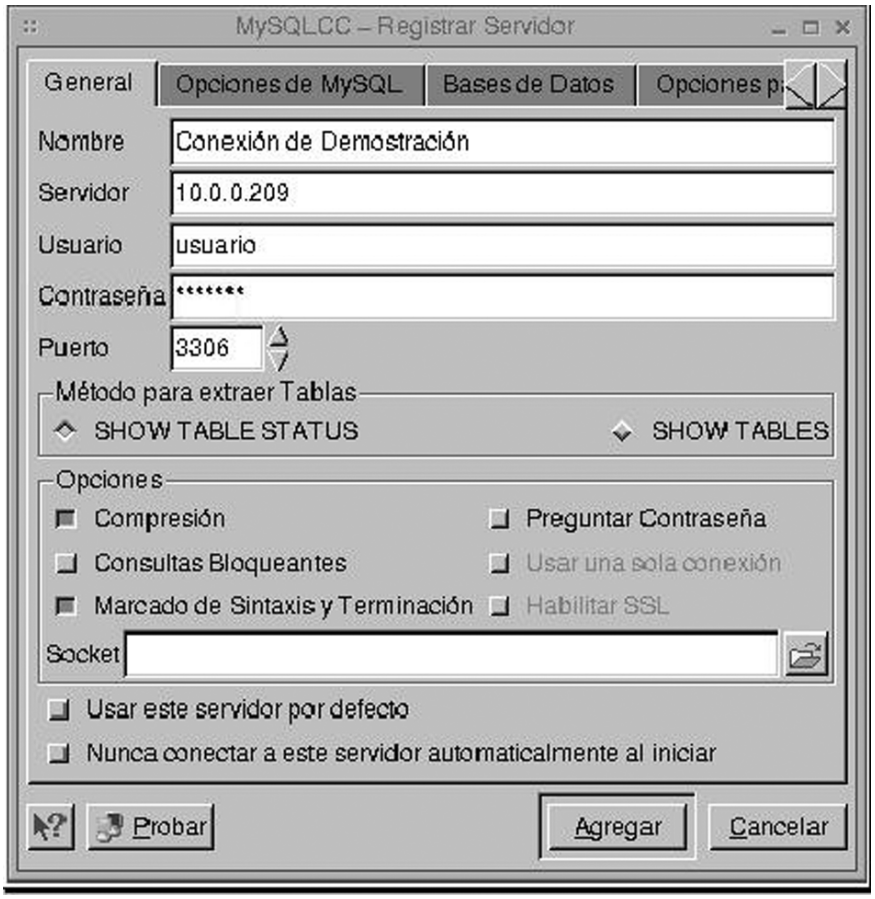
Hi ha múltiples clients d'entorn gràfic que permeten la interacció amb un servidor MySQL. Analitzarem breument els que distribueix l'empresa MySQL AB (*mysqlcc, mysql-query-browser i mysql-administrator*) i que es poden baixar del lloc oficial www.mysql.com.

Nota

Actualment, *mysqlcc* s'ha donat per obsolet en favor dels altres dos.

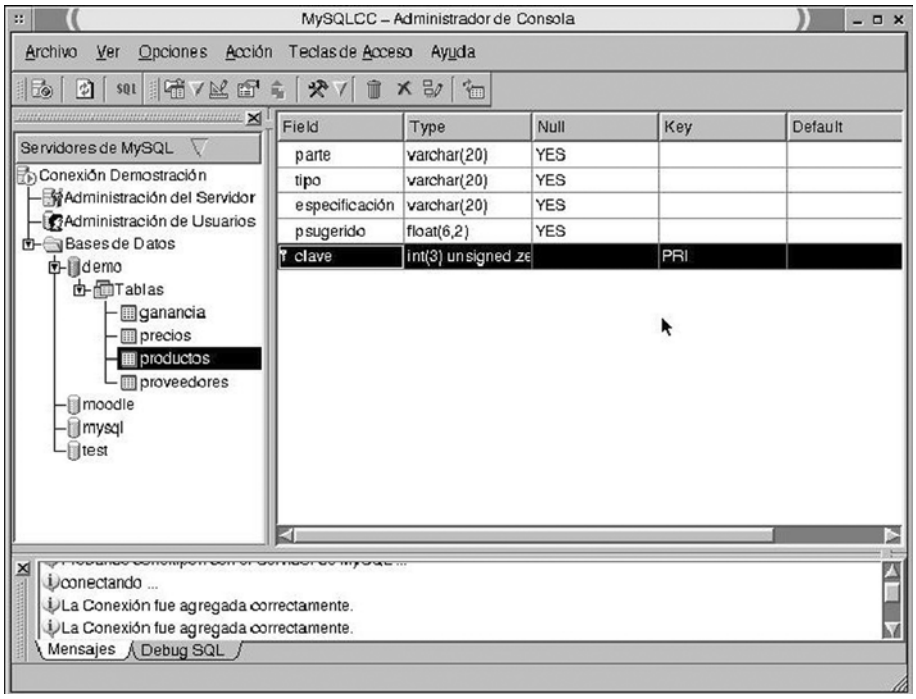
6.1. mysqlcc

En executar-se per primera vegada obrirà el diàleg que permet realitzar el registre d'un nou servidor MySQL:



En la finestra principal es poden apreciar els servidors registrats; en aquest cas només n'hi ha un.

Amb el botó dret del ratolí sobre "Connexió de demostració", es pot activar la connexió. Després d'això, *mysqlcc* mostra les propietats dels elements de la base de dades.

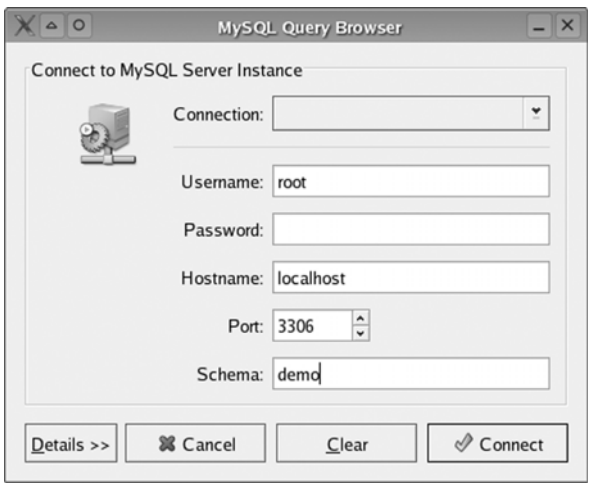


Ara ja estem en disposició de realitzar consultes SQL amb Ctrl-Q (o fent clic sobre la icona 'SQL'). S'obrirà una nova finestra en la qual podrem escriure la consulta que, una vegada escrita, s'executarà en teclejar Ctrl-E. Els resultats es mostraran en forma de taula com en la captura de pantalla anterior.

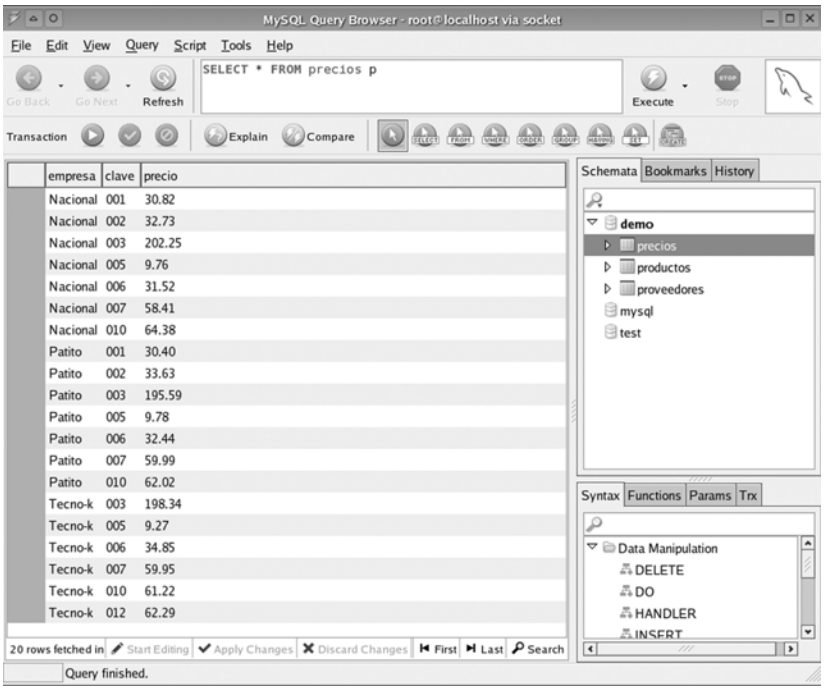
mysqlcc ofereix múltiples opcions per realitzar insercions, eliminacions, configurar tecles d'accés ràpid i una sèrie de característiques d'ús molt intuïtiu. També ofereix prestacions per exportar el resultat d'una consulta a un fitxer de text.

6.2. mysql-query-browser

Tant *mysql-query-browser* com *mysql-administrator* comparteixen la informació referent a les connexions emmagatzemades. La pantalla inicial ens permetrà seleccionar-ne una d'existent o configurar-ne una de nova:



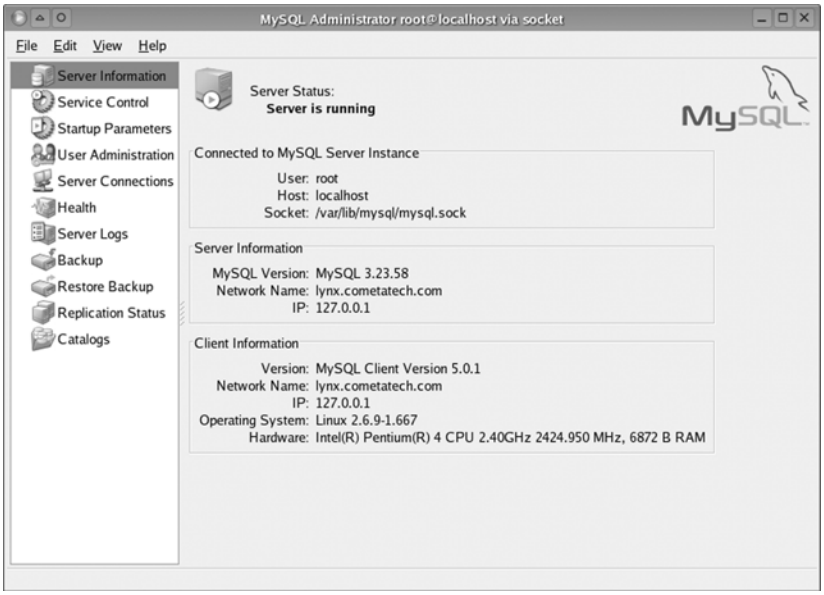
L'aspecte d'aquest programa és millor que el de *mysqlcc* i ofereix prestacions d'ajuda en la generació de consultes, preferits, marcadors, accessos ràpids a *EXPLAIN*, etc.



Així mateix, ofereix diversos formats d'exportació dels resultats d'una consulta i més facilitats a l'hora de navegar pels resultats.

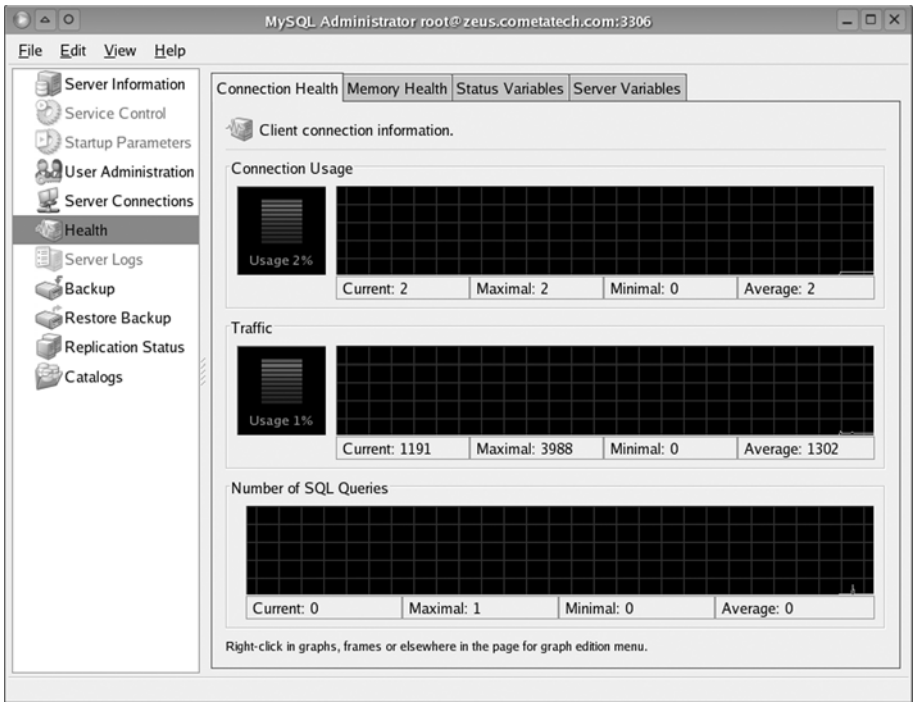
6.3. mysql-administrator

Aquesta eina innovadora és extremadament potent i completa pel que fa a tasques d'administració.



mysql-administrator permet, entre altres coses:

- Engegar i parar l'SGBD.
- Gestionar el fitxer de configuració */etc/my.cnf* de forma gràfica.
- Gestionar usuaris i privilegis.
- Monitoritzar l'ús del gestor que s'està fent, el nombre de connexions, consultes simultànies i tot tipus d'informació estadística.
- Consultar els fitxers de registre (*log*) del servidor.
- Gestionar còpies de seguretat.
- Gestionar la reproducció de bases de dades.
- Crear i esborrar bases de dades (SCHEMA).



Resum

MySQL és un SGBD relacional de fàcil ús i alt rendiment, dues característiques molt valuoses per a un desenvolupador de sistemes: la seva facilitat d'ús permet la creació de bases de dades amb rapidesa i sense gaires complicacions, i el seu alt rendiment el fa summament atractiu per a aplicacions comercials importants o portals web de molt trànsit. Si hi afegim la disponibilitat de codi i la seva llicència doble, es comprèn que MySQL sigui atractiu i accessible per a tothom.

Aquests atributs tenen els seus costos: mantenir-lo amb un alt rendiment fa una mica més lent el seu desenvolupament, per la qual cosa no és el més avançat quant a prestacions i compatibilitat amb estàndards. MySQL manca de característiques que molts altres SGBD tenen. Però no s'ha d'oblidar que està en continu desenvolupament, per la qual cosa futures versions inclouran noves característiques. Per descomptat, per a MySQL és més important l'eficiència que incloure prestacions només per competir o satisfer alguns usuaris.

Bibliografia

DuBois, P. (2003). *MySQL Second Edition: The definitive guide to using, programming, and administering MySQL 4 databases* Indianapolis: Developer's Library.

Manual de MySQL de la distribución (accessible a: http://dev.mysql.com/doc/).

Silberschatz, A.; Korth, H.; Sudarshan, S. (2002). *Fundamentos de Bases de Datos* (4a. ed.). Madrid: McGraw-Hill.