

PARTE III

ADMINISTRACIÓN DE MySQL

Tema 10. Instalación y primeros pasos con MySQL

10.1. Introducción

10.2. Instalación

- 10.2.1. Instalación de MySQL para Windows versión 4.1
- 10.2.2. Instalación de MySQL para Windows versión 5.0
- 10.2.3. Instalación de MySQL para Linux

10.3. Tutorial básico

- 10.3.1. Conectarse al servidor mediante el programa cliente en modo consola *mysql*
- 10.3.2. Algunas consultas básicas de interés para el administrador
- 10.3.3. Añadir password al usuario root
- 10.3.4. Crear y usar bases de datos en MySQL
- 10.3.5. Los comandos SHOW y DESC
- 10.3.6. Editar las sentencias desde el cliente en modo consola *mysql*
- 10.3.7. Lanzar *scripts* desde el cliente en modo consola *mysql*
- 10.3.8. Verificar el estado, arrancar y detener el proceso servidor.

Tema 11. Administración de MySQL (I)

11.1. Estructura interna

- 11.1.1. Localización del directorio de datos
- 11.1.2. Estructura del directorio de datos
- 11.1.3. Arquitecturas de almacenamiento de las tablas

11.2. Configuración

- 11.2.1. Usar opciones de configuración a través de la línea de comandos
- 11.2.2. Usar ficheros de opciones (ficheros de configuración)

11.3. Administración de cuentas de usuario y seguridad

- 11.3.1. Introducción a las sentencias GRANT y REVOKE
- 11.3.2. Sintaxis de la sentencia GRANT
- 11.3.3. Sintaxis de la sentencia REVOKE
- 11.3.4. Algunos ejemplos de la sentencia GRANT
- 11.3.5. Eliminar una cuenta de usuario.

11.4. Índices

- 11.4.1. Tipos de índices y su creación
- 11.4.2. Eliminar o cambiar un índice
- 11.4.3. Uso eficiente de los índices

11.5. Backup

- 11.5.1. *mysqldump*
- 11.5.2. *mysqlhotcopy*
- 11.5.3. Importar tablas desde ficheros de texto delimitados
- 11.5.4. Exportar tablas a ficheros de texto delimitados

Tema 12. Administración de MySQL (II)

12.1. Ficheros Log

- 12.1.1.El log de error (error log)
- 12.1.2.El fichero log general (query log file)
- 12.1.3.El fichero log de modificaciones (update log)
- 12.1.4.El fichero de log binario (binary log)

12.2. Ejecutar varios servidores MySQL en la misma máquina

- 12.2.1.Ejecutar varios servidores MySQL en Windows
- 12.2.2.Ejecutar varios servidores MySQL en Linux

12.3. Replicación

- 12.3.1.Ventajas de la replicación
- 12.3.2.Implantación del servicio de replicación
- 12.3.3.Otros datos útiles en el proceso de replicación
- 12.3.4.Posibles arquitecturas de replicación

Tema 10.

Instalación y primeros pasos con MySQL

Javier Robles Cascallar

10.1. Introducción

Comparado con otros sistemas corporativos de bases de datos el sistema de gestión de bases de datos MySQL es relativamente fácil de administrar, utilizar e instalar. Gran parte de la popularidad de MySQL es su simplicidad (además de su velocidad).

Componentes esenciales de la base de datos MySQL con los que un administrador debe estar familiarizado:

- **El servidor MySQL.** Normalmente asociado al proceso `mysqld` (aunque ya veremos que existen otros procesos servidor). El servidor es el que realiza todos los accesos y manipulaciones a las tablas de la base de datos.
- **Los clientes MySQL que acceden al servidor.** Existen numerosos programas de la propia distribución de MySQL y de otros fabricantes que ayudan a establecer la comunicación con el servidor y realizar las tareas administrativas. Los más importantes y que vienen en la propia distribución de MySQL son los siguientes:
 - `mysql`, programa interactivo en modo línea de comandos para lanzar sentencias SQL en modo interactivo o batch (por lotes).
 - `mysqladmin`, es un cliente de administración en modo línea de comandos.
 - `mysqldump`, es una herramienta para hacer copias de seguridad o copiar bases de datos a otro servidor.
 - `mysqlimport`, es una herramienta para importar ficheros de datos.
 - `mysqlshow`, muestra información sobre las bases de datos y las tablas.
- **Otras utilidades que pueden operar independientemente del servidor.**
 - `Myisamchk`, permite realizar tareas de mantenimiento sobre las tablas.
 - `myisampack`, permite generar tablas comprimidas de solo-lectura.
 - `mysqlbinlog`, es una herramienta para procesar los ficheros log en modo binario que genera el servidor

A continuación listamos las tareas más importantes de administración que en un momento u otro tendrían que ser asumidas por la persona o personas responsables de la base de datos:

- Inicio y desconexión del servidor.
- Mantenimiento de cuentas de usuario.
- Mantenimiento del archivo de registro.
- Backup de la base de datos y recuperación de caídas.
- Optimizar el servidor
- Trabajar con múltiples servidores a la vez
- Estar al tanto de las actualizaciones
- Garantizar la seguridad del sistema de archivos y la seguridad de los accesos
- Mantenimiento preventivo

En este documento de administración se intentará dar una aproximación detallada a la resolución de todas estas tareas.

10.2. Instalación

10.2.1 Instalación de MySQL versión 4.1 para entorno Windows con soporte InnoDB.

Nota importante: Esta versión es la que recomienda MySQL para trabajar en un entorno de producción real. Está mucho más probada y depurada que la versión 5. Como consejo, debéis instalar esta versión preferentemente al principio del curso y solamente instalar la versión 5 para probar las nuevas características. Ambas distribuciones pueden ser instaladas en el mismo equipo y convivir operando simultáneamente sin interferirse.

Obtención del fichero donde se encuentra el programa de instalación del servidor MySQL. (*mysql-4.1.10-win32.zip* tamaño 34Mb). Este fichero lo podéis obtener en el CD que se distribuye para este curso o también lo podéis encontrar en la Web oficial de MySQL entre otros muchos sitios.

Una vez descargado se descomprime el fichero zip en una carpeta y se ejecuta el programa setup.exe que inicia el asistente que va guiando al usuario en el proceso de instalación y posteriormente en el proceso de configuración del servidor de base de datos MySQL.

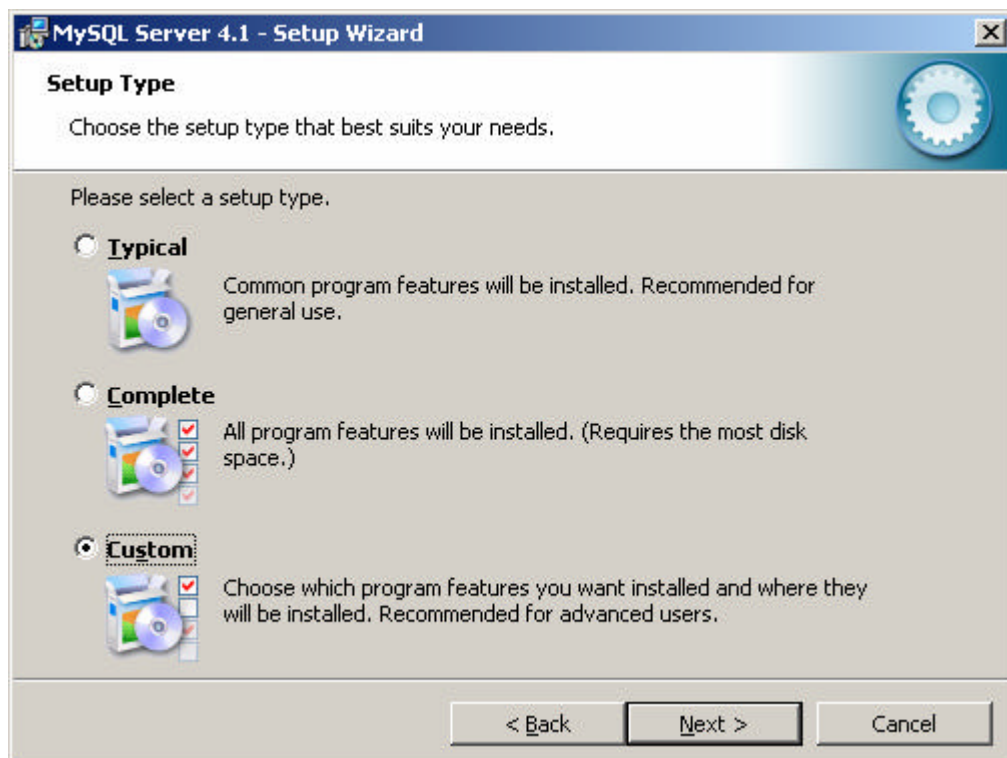
Una vez descargado se ejecuta el programa de instalación y se inicia el asistente que va guiando al usuario en el proceso de instalación y posteriormente en el proceso de configuración del servidor de base de datos MySQL.

Las ventanas del asistente que van apareciendo a lo largo de la instalación se describen a continuación, así como las opciones que conviene que se seleccionen para lograr la instalación que mejor se adapta al contenido del curso.

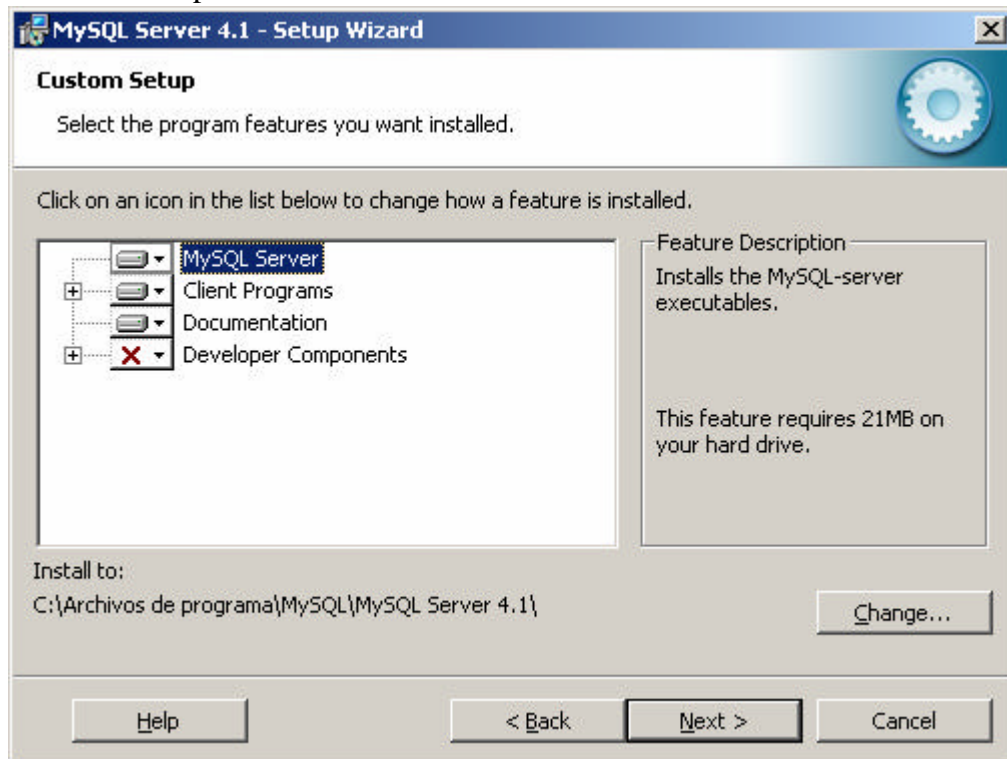
1. Ventana de inicio y presentación del asistente. Pulsamos *Next* para continuar



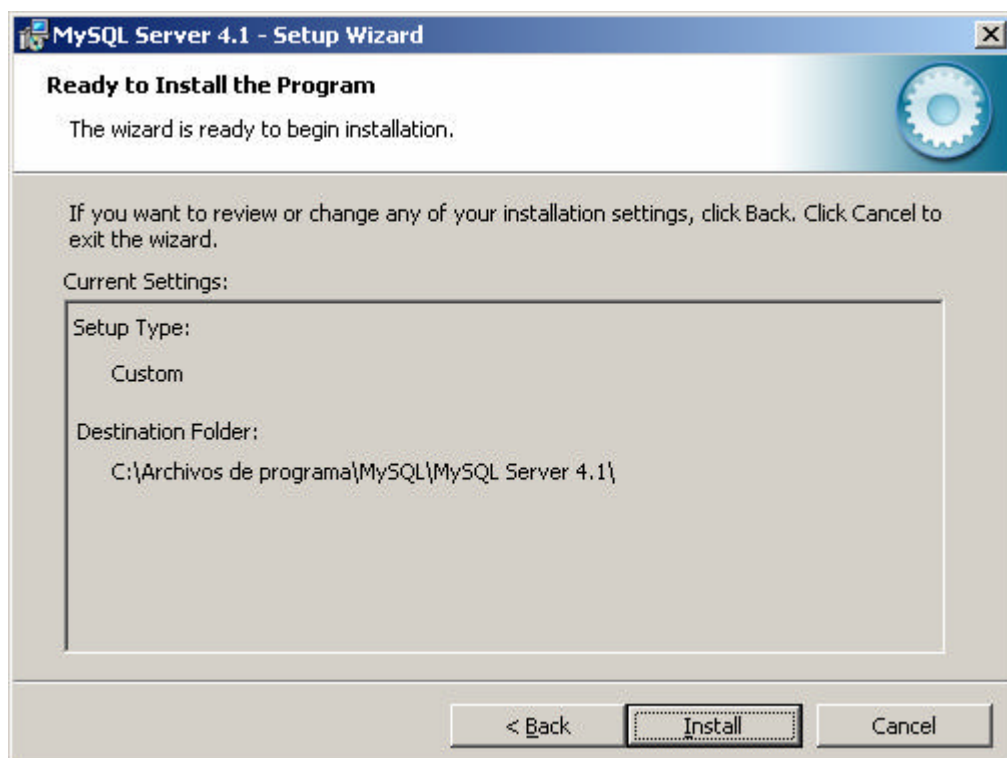
2. Tipo de Instalación. Seleccionamos la instalación personalizada *Custom*



3. Seleccionamos los programas (servidores y clientes) que queremos que se instalen. Lo más conveniente en esta ventana es dejar las opciones que vienen por defecto. Si acaso podemos cambiar (pulsando *Change...*) la carpeta donde se va instalar por defecto la versión.



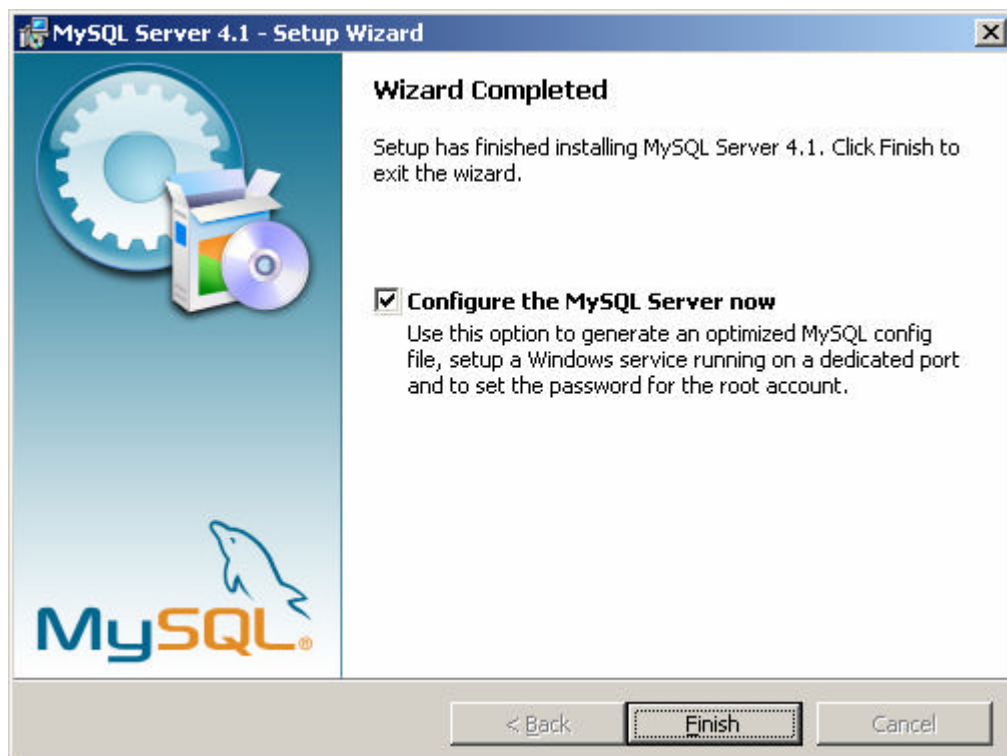
4. Ventana que avisa que ya esta preparado para instalar el producto. Pulsamos *Install*



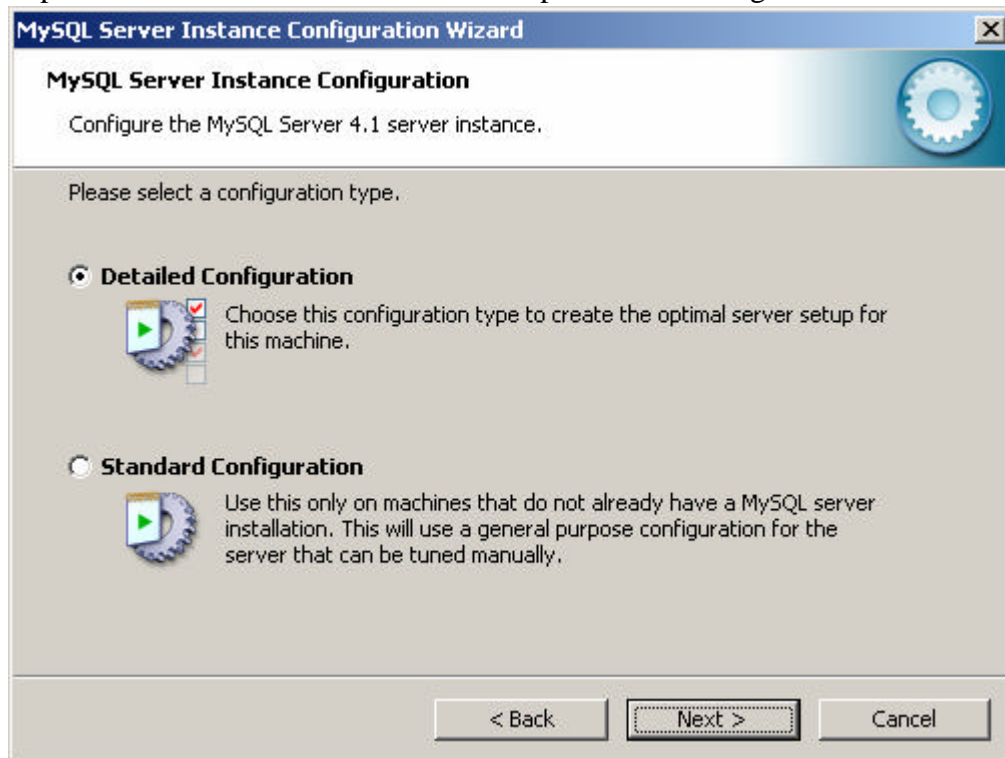
5. Ventana para registrarse en la Web MySQL.com. Podemos dejar este paso si lo deseamos para más adelante. Lo más cómodo y rápido es pulsar por lo tanto *Skip Sign-Up* para saltarse este paso y seguir con la configuración.



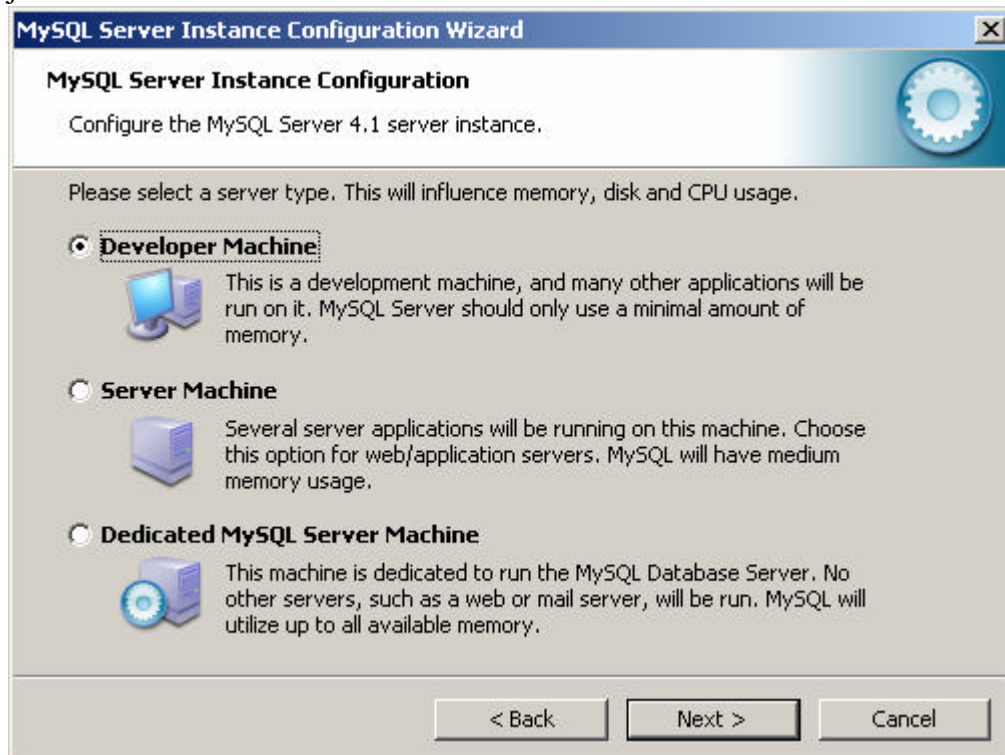
6. Ventana que avisa del final del asistente de instalación. Marcamos la casilla que pone *Configure de MySQL Server now* para iniciar el asistente de configuración.



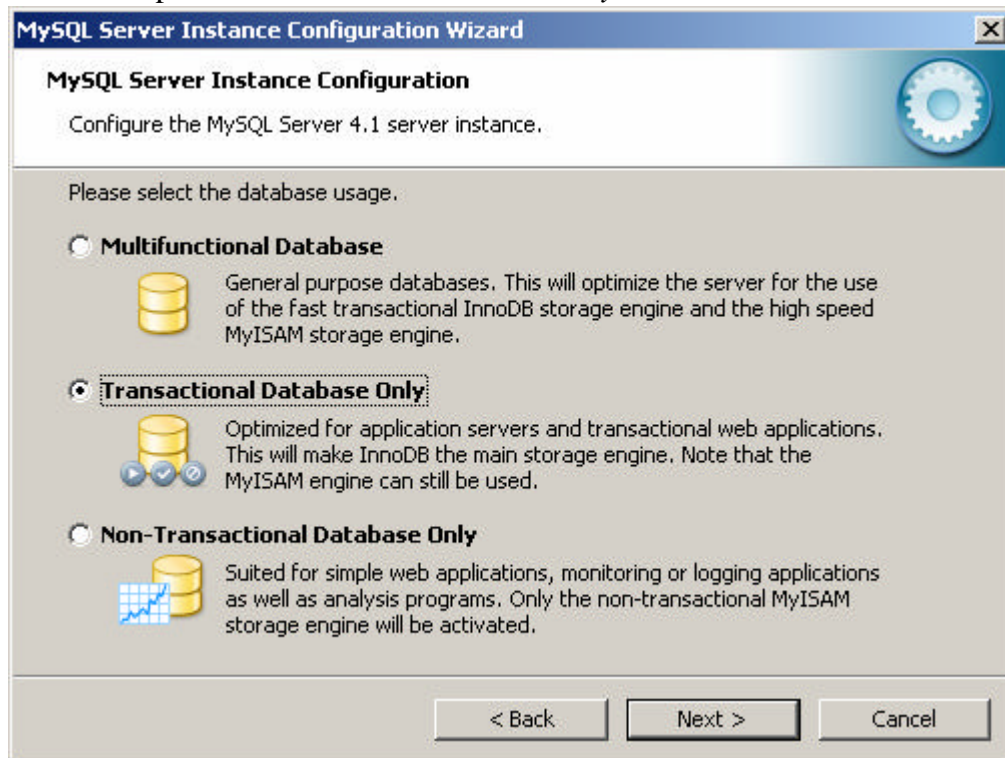
7. Seleccionamos la opción de Configuración Detallada (*Detailed Configuration*). Esta opción permite acceder con más detalle a las opciones de configuración del servidor.



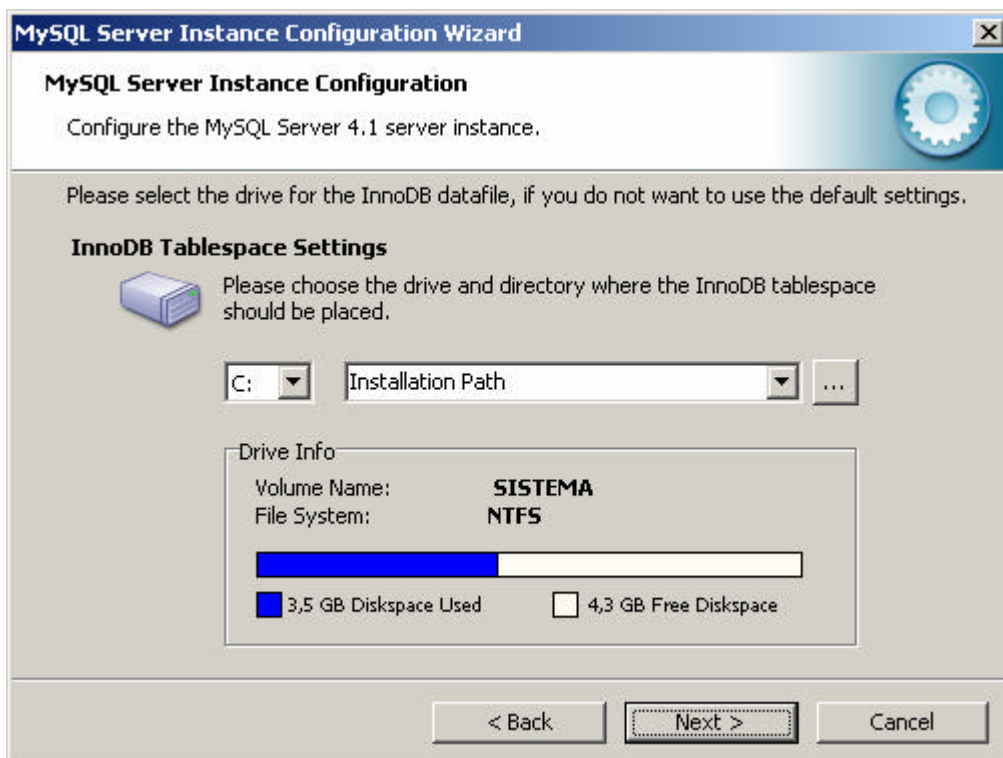
8. Seleccionamos a continuación la opción *Developer Machine*. Esto permite que la instalación de MySQL no le de demasiados recursos de memoria al servidor de MySQL en detrimento de otras aplicaciones que pueden estar funcionando en nuestra estación de trabajo



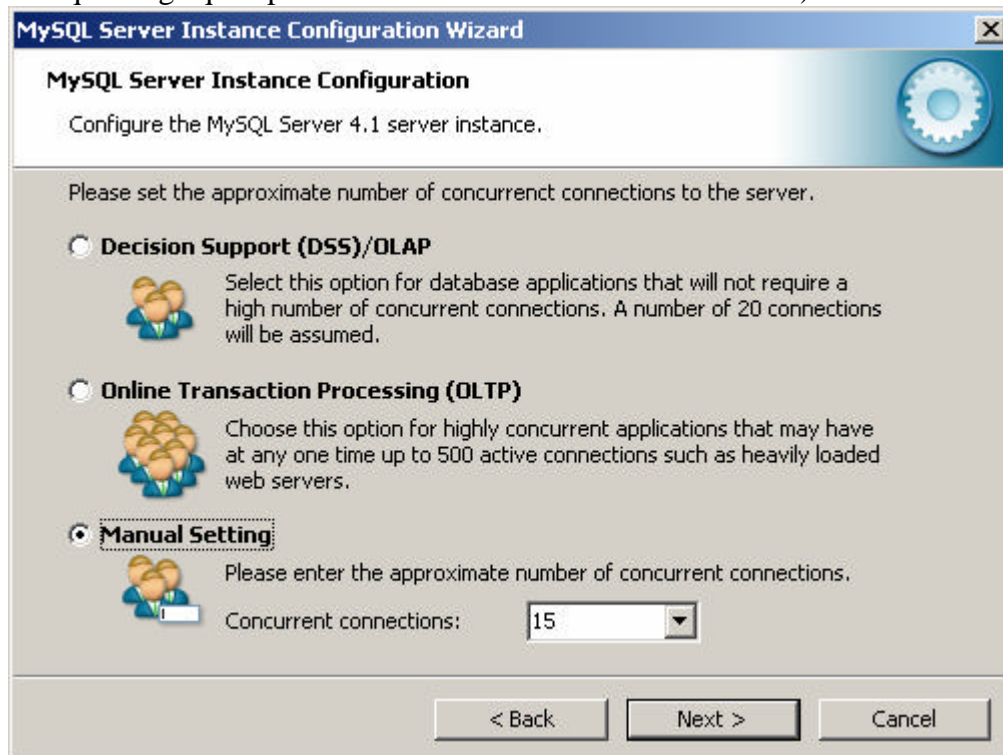
9. Como queremos que la instalación tenga soporte para tablas InnoDB (lo que permite trabajar con integridad referencial entre tablas y control transaccional) seleccionamos en esta ventana la opción *Transactional Database Only*



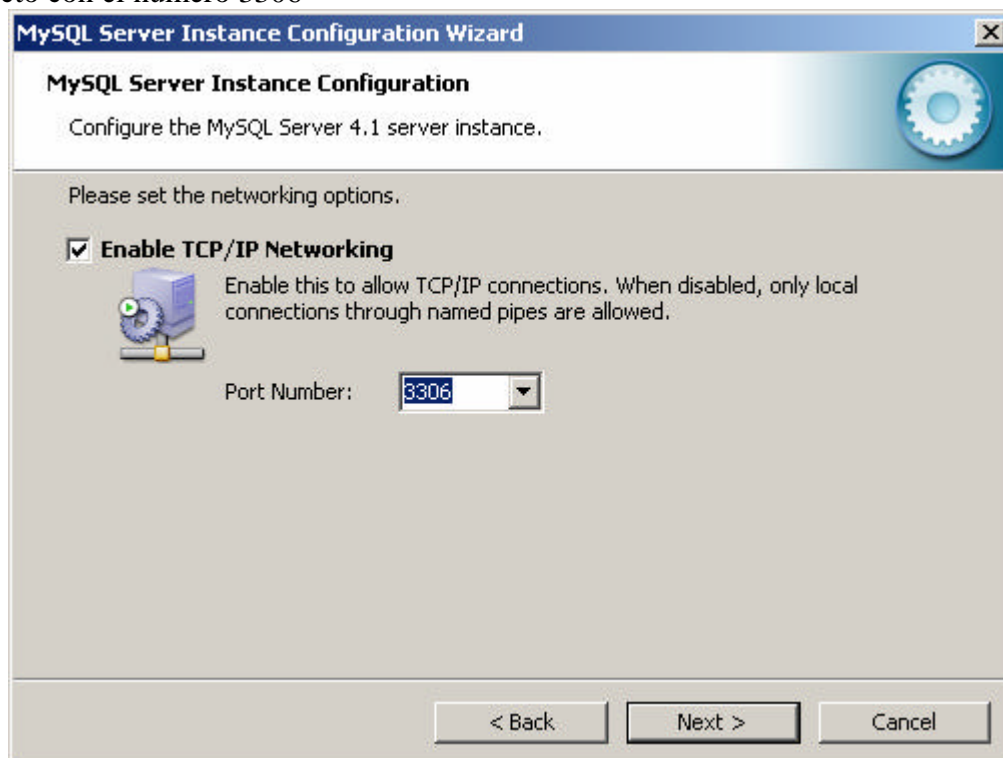
10. Esta ventana selecciona la carpeta donde va a quedar alojado el Tablespace para las tablas InnoDB. Lo más cómodo es dejar la opción que viene por defecto y pulsar *Next*.



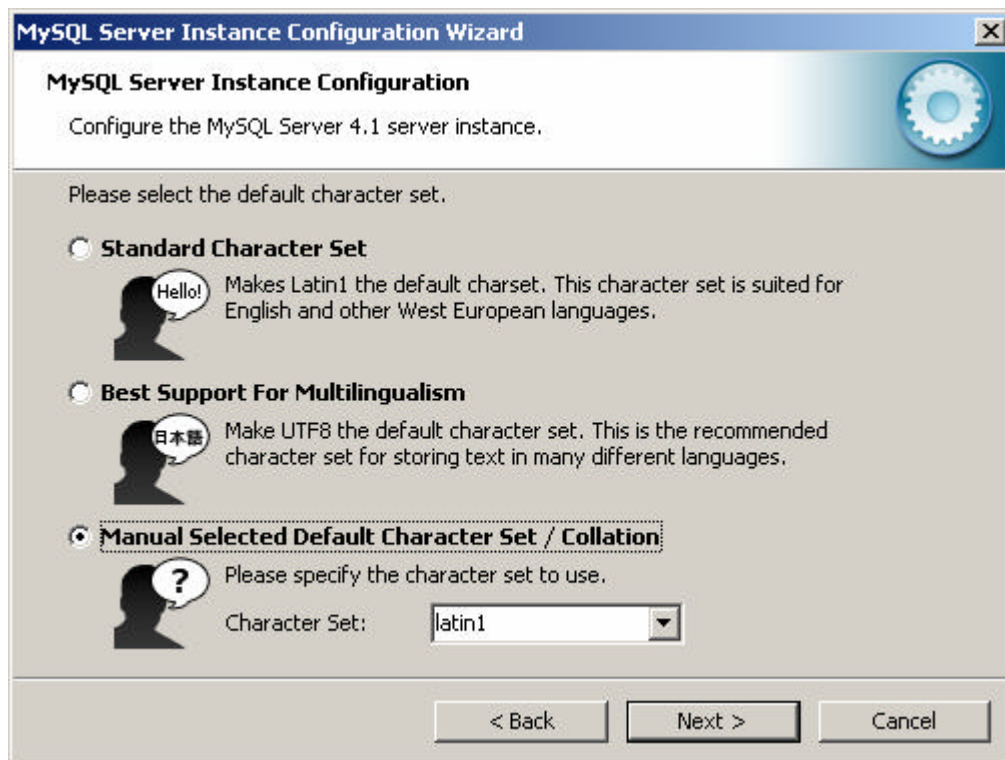
11. Seleccionamos el nº aproximado de conexiones concurrentes. Seleccionamos *Manual Setting* dejando el valor de 15 que viene por defecto. (Podemos incluso reducir el número ya que nuestro servidor de BD va a ser un servidor de prueba y es poco probable que tenga que optimizar el acceso concurrente a los datos).



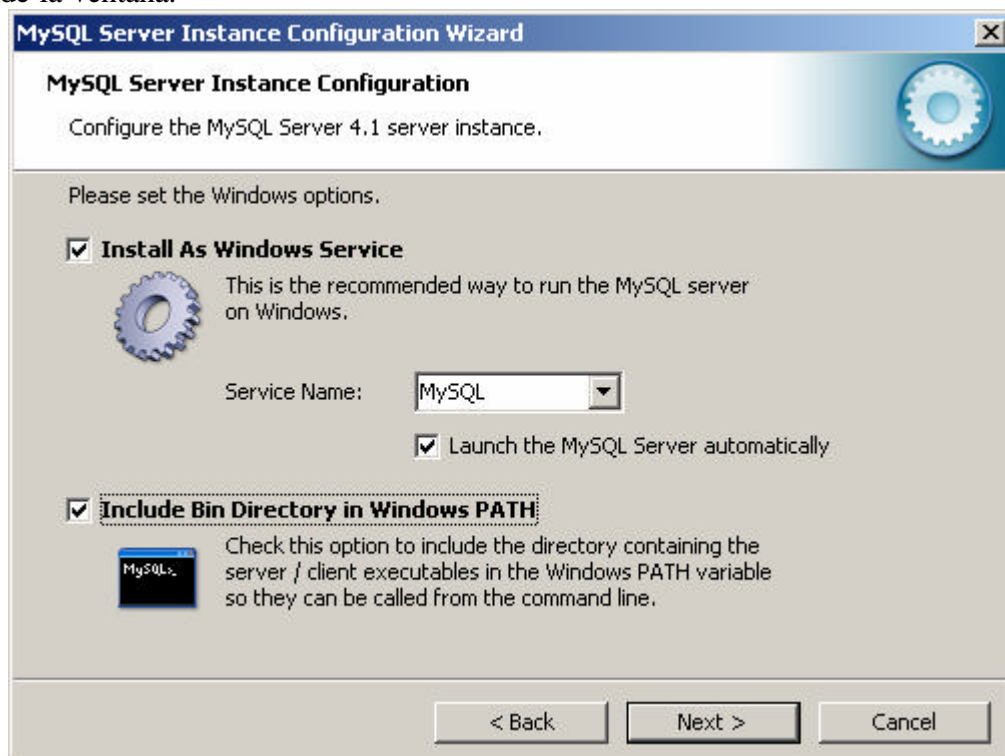
12. Establecer el puerto TCP/IP. Lo más cómodo es dejar el puerto que viene por defecto con el número 3306



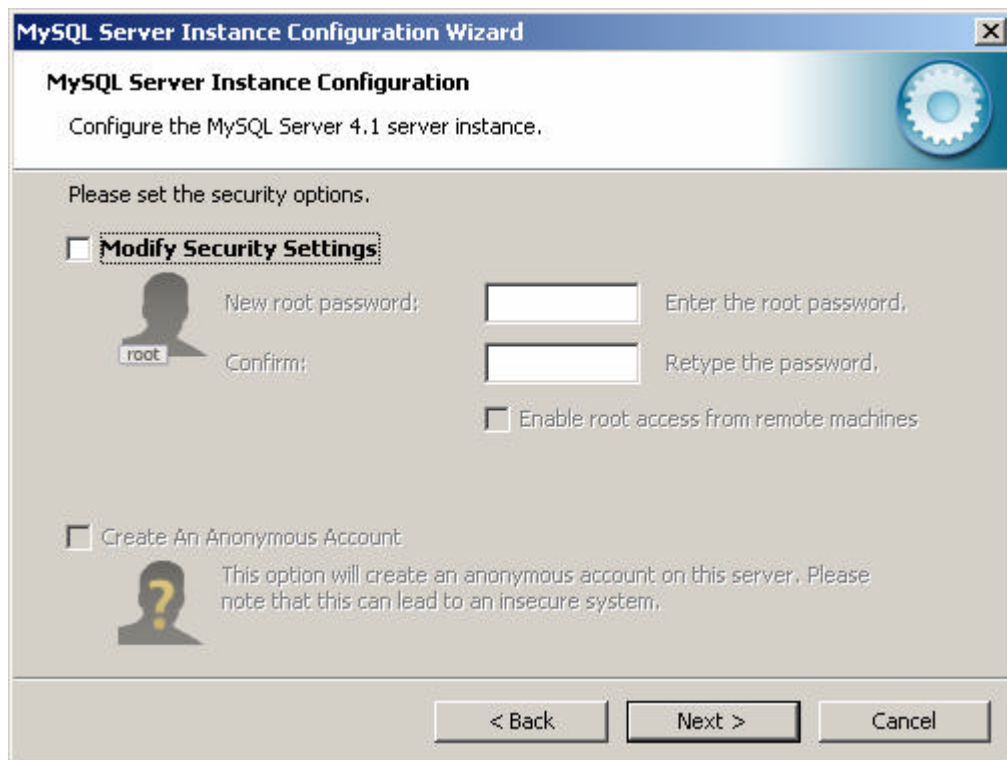
13. Establecer el juego de caracteres por defecto. Debemos seleccionar, si no lo estuviera ya, la opción *Standard Carácter Set*



14. Instalar el arranque del servidor MySQL como un servicio de Windows. Lo más cómodo es que MySQL se configure como un servicio de Windows y que arranque automáticamente cuando arrancamos el equipo. Para ello seleccionamos los dos check box de la ventana.



15. Esta ventana permite configurar las opciones de seguridad iniciales de MySQL. Por defecto MySQL viene con un usuario administrador sin password. Si se desea en esta ventana se puede crear la password de administrador y también la posibilidad de crear un usuario invitado. Lo más sencillo es no modificar las opciones por defecto, por lo tanto si esta marcado el check box de *Modify Security Settings* se desmarca. más adelante ya avanzado el curso se verá como se añade la password al usuario administrador.



The image shows a screenshot of the "MySQL Server Instance Configuration Wizard" window. The title bar reads "MySQL Server Instance Configuration Wizard". The main heading is "MySQL Server Instance Configuration" with a subtext "Configure the MySQL Server 4.1 server instance." and a gear icon. The instruction "Please set the security options." is displayed. There are two main sections, each with a checkbox and a user icon. The first section, "Modify Security Settings", is checked and includes fields for "New root password:" and "Confirm:" with corresponding labels "Enter the root password." and "Retype the password.", and an unchecked checkbox for "Enable root access from remote machines". The second section, "Create An Anonymous Account", is unchecked and includes a warning text: "This option will create an anonymous account on this server. Please note that this can lead to an insecure system." At the bottom, there are three buttons: "< Back", "Next >", and "Cancel".


MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 4.1 server instance.

Please set the security options.


☒ **Modify Security Settings:**

 New root password: Enter the root password.

Confirm: Retype the password.

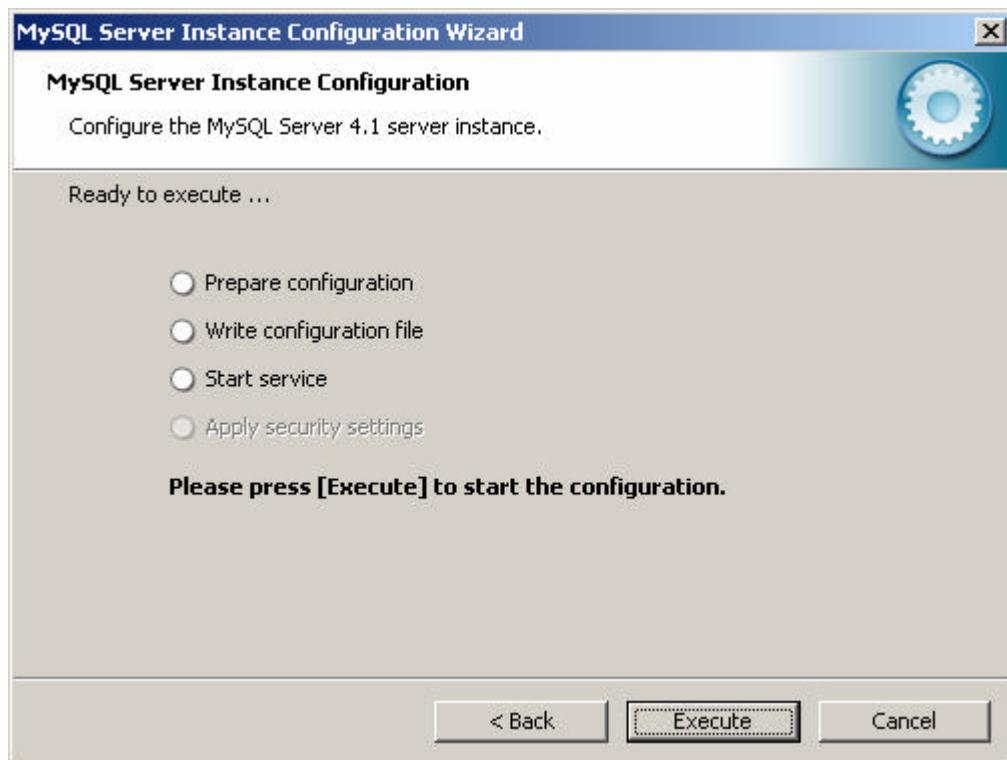
☐ Enable root access from remote machines

☐ Create An Anonymous Account

 This option will create an anonymous account on this server. Please note that this can lead to an insecure system.

< Back Next > Cancel

16. Última ventana del asistente donde nos indica que todo está preparado para generar el fichero de configuración (*my.ini*) y arrancar el servicio de Windows que inicia el servidor MySQL. Pulsamos *Execute* para finalizar el proceso.



10.2.2 Instalación de MySQL versión 5.0 para entorno Windows con soporte InnoDB.

Nota importante: Esta versión es la última release de MySQL y es útil para ver las últimas características y novedades que aporta este servidor de bases de datos. La versión que remienda MySQL para trabajar en un entorno de producción real es la versión 4.1.

Obtención del fichero donde se encuentra el programa de instalación del servidor MySQL. (*mysql-5.0.2-alpha-win.msi* tamaño 37Mb). Este fichero lo podéis obtener en el CD que se distribuye para este curso o también lo podéis encontrar en la Web oficial de MySQL entre otros muchos sitios

Las principales características de la versión 5 que no están soportadas en la versión 4.1.

- Implementación de vistas (VIEWS).
- Procedimientos almacenados (STORED PROCEDURES).
- Soporte para cursores.
- Mejor soporte para las tablas tipo MEMORY
- Mejoras de velocidad.

Una vez descargado se ejecuta el programa de instalación y se inicia el asistente que va guiando al usuario en el proceso de instalación y posteriormente en el proceso de configuración del servidor de base de datos MySQL.

Las ventanas del asistente que van apareciendo a lo largo de la instalación son muy similares a las ya descritas en el apartado 10.2.1 para la instalación de la versión 4.1 y las opciones que conviene seleccionar son las mismas que las ya comentadas en el punto anterior y permiten lograr la instalación que mejor se adapta al contenido del curso.

10.2.3 Instalación de MySQL para Linux.

MySQL viene integrado en la mayor parte de las distribuciones de Linux, lo más probable, por lo tanto, si estamos trabajando en Linux, es que ya esté instalado.

El caso más frecuente con el que nos podemos encontrar es que queramos actualizar a la última versión de MySQL que no viene incluida en nuestra distribución de Linux (por ejemplo si queremos pasar de la 4.1 a la 5.0), en este caso lo más conveniente es desinstalar la aplicación antigua e instalar la nueva versión.

10.2. 3.1 Desinstalar MySQL

- Detener el servidor de MySQL

```
shell> mysql.server stop
```
- Borrar los paquetes de la distribución MySQL instalados en el equipo (rpm)
Se consulta en primer lugar que paquetes hay instalados

```
rpm -qa mysql
```

A continuación se eliminan esos paquetes

```
rpm -e mysql
rpm -e mysql-client
```
- Eliminar manualmente los ficheros de configuración y las bases de datos que la desinstalación de paquetes no elimina.
Eliminar el fichero `/etc/my.cnf`
Eliminar la carpeta `/var/lib/mysql`

10.2.3.2 Instalar MySQL

También comentamos en este apartado la instalación mediante paquetes *rpm* por ser la más cómoda de realizar.

Para realizar una instalación mínima al menos se requiere la instalación de dos paquetes.

```
shell> rpm -i MySQL-server-VERSION.i386.rpm
shell> rpm -i MySQL-client-VERSION.i386.rpm
```

(*VERSION* corresponde al paquete cuya versión queremos instalar)

La instalación por paquetes *rpm* crea automáticamente una cuenta de usuario llamado *mysql* que es el que ejecuta el proceso servidor y crea también las entradas adecuadas en `/etc/init.d/` para arrancar el servidor automáticamente al arrancar el equipo.

10.3. Tutorial Básico

10.3.1. Conectarse al servidor mediante el programa cliente en modo consola *mysql*

El cliente básico de MySQL es el programa *mysql* (*mysql.exe* en Windows) que viene en la distribución y una vez instalado se encuentra en la carpeta */bin* que genera la instalación

Para conectarse al servidor, usualmente necesitamos de un nombre de usuario (login) y de una contraseña (password), y si el servidor al que nos deseamos conectar está en una máquina diferente de la nuestra, también necesitamos indicar el nombre o la dirección IP de dicho servidor. Una vez que conocemos estos tres valores, podemos conectarnos de la siguiente manera:

```
shell> mysql -h NombreDelServidor -u NombreDeUsuario -p
```

Cuando ejecutamos este comando, se nos pedirá que proporcionemos también la contraseña para el nombre de usuario que estamos usando.

Si la conexión al servidor MySQL se pudo establecer de manera satisfactoria, recibiremos el mensaje de bienvenida y estaremos en el prompt de *mysql*:

```
shell>mysql -h 192.168.1.2 -u root -p
```

```
Enter password: *****
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 5563 to server version: 4.1.8-nt-max-log
```

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

```
mysql>
```

Este prompt *mysql>* nos indica que hemos conectado y *mysql* está listo para recibir comandos.

Por defecto la instalación MySQL incorpora dos usuarios el usuario Administrador (*'root'*) y el usuario invitado (*'ODBC'* y también *''*) ambos con privilegios de administración cuando se conectan de modo local al servidor. Por lo tanto es posible acceder con derechos de administrador local del propio equipo sin necesidad de introducir ningún tipo de opción y sin password.

```
shell> mysql
```

Después de que nos hemos conectado de manera satisfactoria, podemos desconectarnos en cualquier momento al escribir *"quit"* o *"exit"*.

Esta forma de acceso como administrador sin password por defecto es útil y cómoda para esta fase de aprendizaje pero altamente insegura y desaconsejable en una instalación de producción convencional.

En la mayoría de los ejemplos siguientes se asume que estamos conectados al servidor, lo cual se indica con el prompt de mysql. (`mysql>`)

Ejemplos de establecimiento de conexión entre cliente y servidor:

Conectarse localmente con el usuario invitado. Normalmente se accede con privilegios de administrador

```
C:\MySQL\bin>mysql
```

Conectarse localmente con el usuario administrador `root` introduciendo la password `micontraseña` en la propia línea de comandos.

```
C:\MySQL\bin>mysql -uroot -pmicontraseña
```

Conectarse localmente con el usuario administrador `root` sin introducir la password en la propia línea de comandos. La password se introduce después de pulsar Enter

```
C:\MySQL410\bin>mysql -uroot -p
Enter password: *****
```

Conectarse a un servidor de otra maquina con el usuario invitado sin password. En este caso no se accede con privilegios de administrador. Lo normal (depende de la distribución) es que solo tenga acceso a la base de datos `test` que viene incluida para pruebas

```
C:\MySQL410\bin>mysql -h192.168.3.33
```

En el tema 11 de esta documentación se trata en profundidad el tema de los usuarios de MySQL y sus privilegios de acceso.

10.3.2 Algunas consultas básicas de interés para el administrador

En este momento debimos de haber podido conectarnos ya al servidor MySQL, aún cuando no hemos seleccionado alguna base de datos para trabajar. Lo que haremos a continuación es escribir algunos comandos para irnos familiarizando con el funcionamiento de mysql.

- Consulta para conocer la versión del servidor y la fecha actual:

```
mysql> SELECT VERSION(), CURRENT_DATE;
+-----+-----+
| VERSION() | CURRENT_DATE |
+-----+-----+
| 4.1.8-nt-max-log | 2005-01-08 |
+-----+-----+
1 row in set (0.33 sec)

mysql>
```

Esta comando ilustra distintas cosas acerca de mysql:

Un comando normalmente consiste de una sentencia SQL seguida por un punto y coma. Cuando emitimos un comando, mysql lo manda al servidor para que lo ejecute, nos muestra los resultados y regresa el *prompt* indicando que está listo para recibir más consultas.

mysql muestra los resultados de la consulta como una tabla (filas y columnas). La primera fila contiene etiquetas para las columnas. Las filas siguientes muestran los resultados de la consulta. Normalmente las etiquetas de las columnas son los nombres de los campos de las tablas que estamos usando en alguna consulta. Si lo que estamos recuperando es el valor de una expresión (como en el ejemplo anterior) las etiquetas en las columnas son la expresión en sí.

mysql muestra cuántas filas fueron regresadas y cuanto tiempo tardó en ejecutarse la consulta, lo cual puede darnos una idea de la eficiencia del servidor, aunque estos valores pueden ser un tanto imprecisos ya que no se muestra la hora del CPU, y porque pueden verse afectados por otros factores, tales como la carga del servidor y la velocidad de comunicación en una red.

Las palabras clave pueden ser escritas usando mayúsculas y minúsculas.

Aquí está un ejemplo que muestra una consulta simple escrita en varias líneas.

-Consulta para conocer el nombre del usuario que ha efectuado la conexión:

```
mysql> SELECT
-> USER();
+-----+
| user() |
+-----+
| root@localhost |
+-----+
1 row in set (0.00 sec)

mysql>
```

En este ejemplo debe notarse como cambia el prompt (de `mysql>` a `->`) cuando se escribe una consulta en varias líneas. Esta es la manera en cómo `mysql` indica que está esperando a que finalice la consulta. Sin embargo si deseamos no terminar de escribir la consulta, podemos hacerlo al escribir `\c` como se muestra en el siguiente ejemplo:

```
mysql> SELECT
      -> USER( ) ,
      -> \c
mysql>
```

De nuevo, se nos regresa el comando el prompt `mysql>` que nos indica que `mysql` está listo para una nueva consulta.

En la siguiente tabla se muestran cada uno de los prompts que podemos obtener y una breve descripción de su significado para `mysql`:

Prompt	Significado
<code>mysql></code>	Listo para una nueva consulta.
<code>-></code>	Esperando la línea siguiente de una consulta multi-línea.
<code>'></code>	Esperando la siguiente línea para completar una cadena que comienza con una comilla sencilla (').
<code>"></code>	Esperando la siguiente línea para completar una cadena que comienza con una comilla doble (").

Los prompts `'>` y `">` ocurren durante la escritura de cadenas. En `mysql` podemos escribir cadenas utilizando comillas sencillas o comillas dobles (por ejemplo, `'hola'` y `"hola"`), y `mysql` nos permite escribir cadenas que ocupen múltiples líneas. De manera que cuando veamos el prompt `'>` o `">`, `mysql` nos indica que hemos empezado a escribir una cadena, pero no la hemos finalizado con la comilla correspondiente.

10.3.3 Añadir password al usuario root

MySQL por defecto viene con la cuenta de administrador sin password.

Realmente MySQL incluye dos cuentas de `root` (administrador) una para conectarse localmente y otra para conectarse desde cualquier máquina externa al servidor. Eso lo podemos ver haciendo el siguiente *select* sobre la tabla de usuarios que se encuentra en la base de datos `mysql` (diccionario de datos del servidor).

```
mysql> select user,host,password from mysql.user;
+-----+-----+-----+
| user | host      | password |
+-----+-----+-----+
| root | localhost |          |
| root | %         |          |
|      | localhost |          |
|      | %         |          |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

En este *select* se ve claramente como las cuentas de `root` y de invitado están sin password, por lo tanto el sistema es totalmente inseguro, solo válido para pruebas.

Lo anterior nos indica que la primera vez que se inicialice el servidor la primera cosa que se debe hacer es especificar un password para el usuario `root` (esto es para el administrador). Esto se puede realizar de la siguiente manera:

```
shell> mysql -u root
mysql> SET PASSWORD FOR root=PASSWORD('new_password');
```

Con esta sentencia le hemos añadido un password a la cuenta de `root` que sirve para conectar desde cualquier máquina.

Si también le queremos añadir un password a la cuenta de `root` para conectar en modo `localhost` ejecutaremos la siguiente instrucción:

```
mysql> SET PASSWORD FOR root@localhost=PASSWORD('new_password');
```

Volvemos a repetir la *select* sobre la tabla usuarios para ver como ha cambiado después de ejecutar las sentencias anteriores.

```
mysql> select user,host,password from mysql.user;
+-----+-----+-----+
| user | host      | password |
+-----+-----+-----+
| root | localhost | *8C6D142DE6424E095EC1E1E756974E907289CE6B |
| root | %         | *8C6D142DE6424E095EC1E1E756974E907289CE6B |
|      | localhost |          |
|      | %         |          |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

Nota1: `SET PASSWORD FOR` se puede usar para establecer nueva contraseña para cualquier usuario, no solo `root`.

Nota 2: La función `PASSWORD('new_password')` genera la password encriptada.

10.3.4 Crear, usar y acceder a bases de datos en MySQL

Ahora que conocemos como escribir y ejecutar sentencias, es tiempo de crear y acceder a una base de datos.

Esta sección muestra como ver las bases de datos a las que tenemos acceso en el servidor, crear una base de datos nueva , posicionar al usuario en una determinada base de datos y acceder a tablas de una base de datos cuando estamos situados en otra base de datos.

Primeramente usaremos la sentencia `SHOW DATABASES` para ver cuáles son las bases de datos existentes en el servidor al que estamos conectados:

```
mysql> SHOW DATABASES ;
+-----+
| Database |
+-----+
| mysql    |
| test     |
+-----+
2 rows in set (0.00 sec)

mysql>
```

Es probable que la lista de bases de datos que veamos sea diferente en nuestro caso, pero seguramente las bases de datos `"mysql"` y `"test"` estarán entre ellas. En particular, la base de datos `"mysql"` es necesaria, ya que ésta tiene la información de los privilegios de los usuarios de MySQL. La base de datos `"test"` se crea durante la instalación de MySQL con el propósito de servir como área de trabajo para los usuarios que inician en el aprendizaje de MySQL.

Para acceder a la base de datos `"test"` se usa el comando `USE`:

```
mysql> USE test
Database changed
mysql>
```

Observar que `USE`, al igual que `QUIT`, no requieren el uso del punto y coma, aunque si se usa éste, no hay ningún problema. El comando `USE` es especial también de otra manera: debe ser usado en una sola línea.

```
mysql> USE prueba
ERROR 1049: Unknown database 'prueba'
mysql>
```

El mensaje anterior indica que la base de datos no ha sido creada, por lo tanto necesitamos crearla.

```
mysql> CREATE DATABASE prueba;
Query OK, 1 row affected (0.00 sec)

mysql> USE prueba
Database changed
mysql>
```

Ahora ya se pueden lanzar sentencias `CREATE TABLE` para crear tablas sobre la base de datos `prueba` que hemos creado y hemos accedido. (Ver la sentencia `CREATE TABLE` en el manual de SQL de este curso)

Bajo el sistema operativo Unix/Linux, los nombres de las bases de datos son sensibles al uso de mayúsculas y minúsculas (no como las palabras clave de SQL), por lo tanto debemos de tener cuidado de escribir correctamente el nombre de la base de datos. Esto es cierto también para los nombres de las tablas.

Al crear una base de datos no se selecciona ésta de manera automática; debemos hacerlo de manera explícita, por ello usamos el comando `USE` en el ejemplo anterior.

La base de datos se crea sólo una vez, pero nosotros debemos seleccionarla cada vez que iniciamos una sesión con *mysql*. Por ello es recomendable que se indique la base de datos sobre la que vamos a trabajar al momento de invocar al monitor de MySQL. Por ejemplo:

```
shell>mysql -h 192.168.1.2 -u root -p prueba

Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17 to server version: 4.1.8-nt-max-log

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

mysql>
```

Observar que "prueba" no es la contraseña que se está proporcionando desde la línea de comandos, sino el nombre de la base de datos a la que deseamos acceder. Si deseamos proporcionar la contraseña en la línea de comandos después de la opción "-p", debemos de hacerlo sin dejar espacios (por ejemplo, `-phola123`, no como `-p hola123`). Sin embargo, escribir nuestra contraseña desde la línea de comandos no es recomendado, ya que es bastante inseguro.

Para ver en un momento dado en que base de datos nos encontramos situados podemos lanzar la siguiente consulta:

```
mysql> select database();
+-----+
| database() |
+-----+
| prueba     |
+-----+
1 row in set (0.00 sec)
```

```
mysql>
```

Para acceder a una tabla de una base de datos cuando nos encontramos situados en otra base de datos es necesario anteponer en la sentencia el nombre de la base de datos seguido de un punto. Por ejemplo accedo mediante el comando `USE` a la base de datos `test` y a continuación hago un *select* de la tabla `departamentos` de la base de datos `prueba`.

```
mysql> use test
Database changed
mysql> select * from prueba.departamentos;
```

DEP_NO	DNOMBRE	LOCALIDAD
10	CONTABILIDAD	BARCELONA
20	INVESTIGACION	VALENCIA
30	VENTAS	MADRID
40	PRODUCCION	SEVILLA

```
4 rows in set (0.10 sec)
```

```
mysql>
```

Esto se podrá hacer siempre que el usuario tenga privilegios suficientes como para acceder a ambas bases de datos (en este ejemplo a `prueba` y a `test`). (El tema de los privilegios de acceso se ve en el punto 6 de este manual).

10.3.5 El comando SHOW y el comando DESC

Estos dos comandos son muy útiles cuando trabajamos con el cliente de MySQL en entorno de comandos puesto que no tenemos el soporte visual que proporcionan los clientes gráficos.

Para ver un listado con las tablas dentro de una determinada base de datos podemos lanzar la sentencia:

```
mysql> show tables;
+-----+
| Tables_in_prueba |
+-----+
| departamentos    |
| empleados        |
+-----+
2 rows in set (0.10 sec)
```

Para ver la descripción de una tabla en sus columnas y tipos asociados, además de poder ver los atributos que aceptan nulos y las opciones por defecto debemos usar el comando DESC o DESCRIBE que realmente es un sinónimo de la sentencia SHOW COLUMNS FROM.

```
mysql> desc departamentos;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| DEP_NO     | int(2)        |      | PRI | 0        |       |
| DNOMBRE    | varchar(14)   | YES  |     | NULL     |       |
| LOCALIDAD  | varchar(10)   | YES  |     | NULL     |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.05 sec)
```

Es muy interesante también poder ver la sentencia de creación de tabla asociada a una determinada tabla de la base de datos. Es una información que puede complementar la proporcionada por el comando DESC.

```
mysql> show create table departamentos \G
***** 1. row *****
      Table: departamentos
Create Table: CREATE TABLE `departamentos` (
  `DEP_NO` int(2) NOT NULL default '0',
  `DNOMBRE` varchar(14) default NULL,
  `LOCALIDAD` varchar(10) default NULL,
  PRIMARY KEY (`DEP_NO`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

La opción \G permite visualizar la salida de la consulta en una sola fila.

Conviene fijarse que esta es una manera donde rápidamente se puede visualizar el tipo de almacenamiento de la tabla (en el ejemplo se ve que la tabla es de tipo MyISAM.; para más información sobre los tipos de almacenamiento soportados por MySQL ver el punto 4.3 de este manual) que es una información que no proporciona DESC directamente.

Otra sentencia que también permite obtener información acerca de las tablas es `SHOW TABLE STATUS [FROM nombre_base_de_datos] [LIKE 'patron']`. Siguiendo con el ejemplo de la tabla `DEPARTAMENTOS`:

```
mysql> show table status like 'departamentos' \G
***** 1. row *****
      Name: departamentos
      Engine: MyISAM
      Version: 9
      Row_format: Dynamic
      Rows: 0
      Avg_row_length: 0
      Data_length: 0
      Max_data_length: 4294967295
      Index_length: 1024
      Data_free: 0
      Auto_increment: NULL
      Create_time: 2005-01-05 15:18:40
      Update_time: 2005-01-05 15:18:41
      Check_time: NULL
      Collation: latin1_swedish_ci
      Checksum: NULL
      Create_options:
      Comment:
1 row in set (0.00 sec)
```

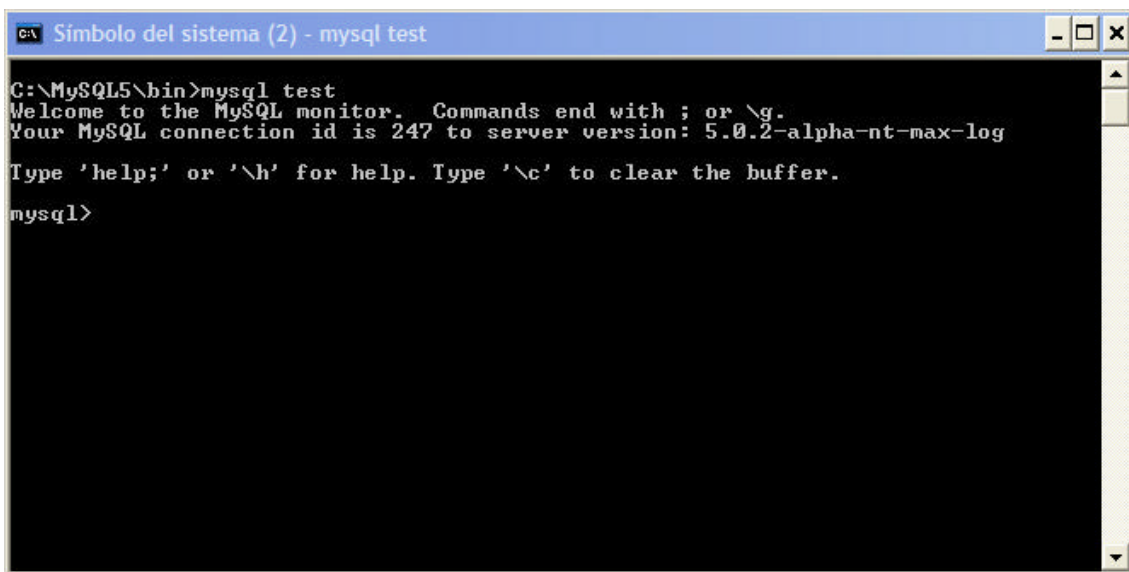
Donde se observa que esta sentencia nos devuelve información de utilidad para el Administrador de la Base de Datos.

10.3.6 Editar las sentencias desde el cliente en modo consola *mysql*

El programa cliente *mysql* no es muy amigable a la hora de editar sentencias largas de SQL antes de lanzarlas. Es por lo tanto conveniente y aconsejable editar dichas sentencias desde un sencillo editor ASCII (*Notepad* o *Wordpad* en entorno Windows) y una vez editadas lanzar las sentencias mediante el usual procedimiento de “copiar y pegar” al cliente *mysql*.

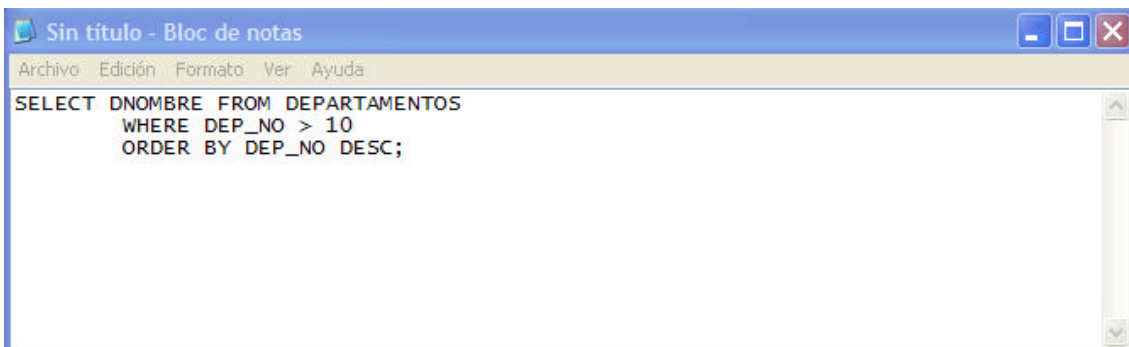
Con el siguiente ejemplo se muestra el proceso desde Windows (en Linux es muy similar).

Paso 1. Se abre una ventana de comandos y se inicia una sesión con el cliente *mysql* sobre la base de datos *test*.



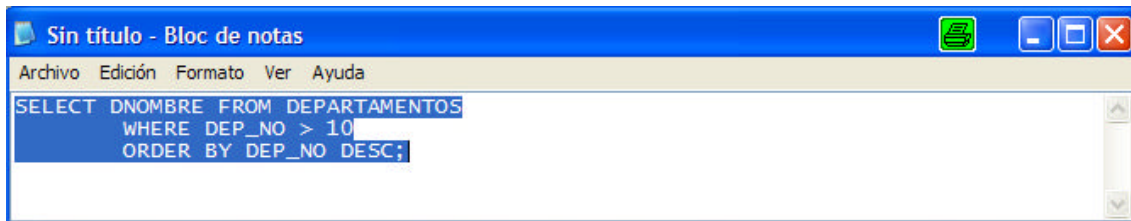
```
C:\MySQL5\bin>mysql test
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 247 to server version: 5.0.2-alpha-nt-max-log
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

Paso 2. Se abre en otra ventana el editor y se edita la sentencia SQL que se va a lanzar al cliente.

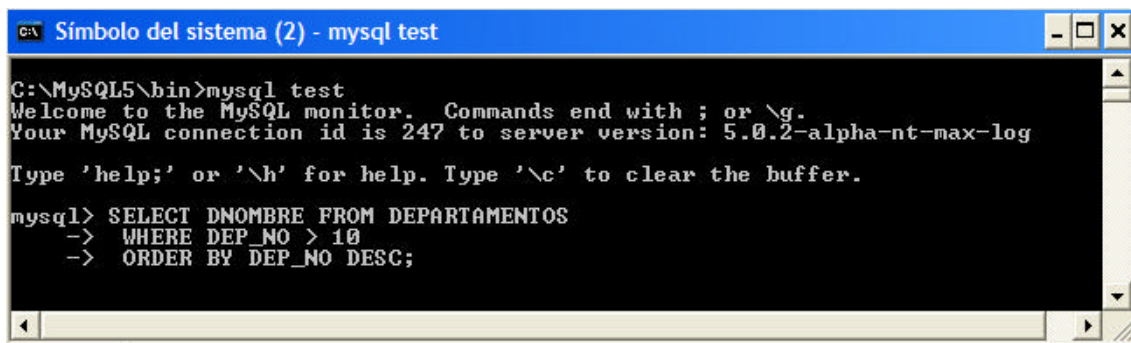


```
Sin título - Bloc de notas
Archivo Edición Formato Ver Ayuda
SELECT DNOMBRE FROM DEPARTAMENTOS
WHERE DEP_NO > 10
ORDER BY DEP_NO DESC;
```

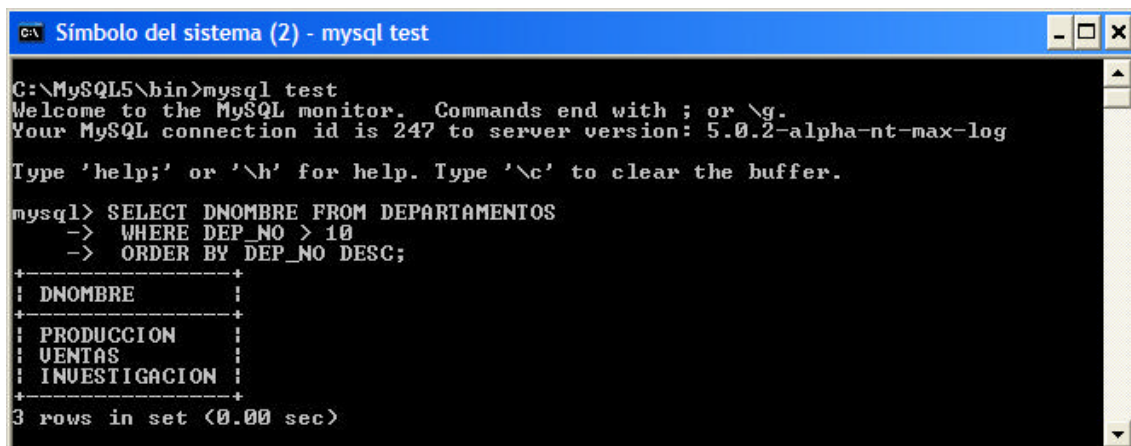
Paso 3. Se marca con el ratón el párrafo con la sentencia para su copia mediante el comando *Menú→Edición→Copiar* o bien pulsando *Ctrl+C*



Paso 4. Se pega la copia anterior pulsando con el botón derecho del ratón sobre cualquier punto de la ventana de comandos y seleccionando la opción *Pegar*.



Paso 5. Se pulsa *Enter* para lanzar la sentencia.



Este proceso, aunque se haya explicado en cinco pasos, una vez que el usuario se ha acostumbrado a la mecánica, es la manera más cómoda de trabajar cuando estamos en entorno de comandos y no disponemos de un cliente gráfico.

10.3.7 Lanzar *scripts* desde el cliente en modo consola *mysql*.

Normalmente hasta ahora hemos usado *mysql* de manera interactiva para ejecutar algunas consultas y ver los resultados. Sin embargo, es posible usar *mysql* en modo batch. Para hacer esto tenemos que poner los comandos que deseamos ejecutar dentro de un archivo, y entonces decirle a *mysql* que lea los comandos de dicho archivo:

```
shell> mysql < archivo-batch.sql
```

Si se usa *mysql* de esta manera, se está creando un pequeño script, y posteriormente se está ejecutando dicho script. La extensión `.sql` no es obligatoria pero si aconsejable para reconocer los scripts de *sql*. Al ejecutar las sentencias y comandos que se encuentran en el script, es posible que suceda algún error. Si se desea que se continúen ejecutando las demás sentencias, a pesar de que haya ocurrido un error, se tiene que usar la opción `--force`

```
shell> mysql --force < archivo-batch.sql
```

Así mismo, es posible especificar los parámetros de conexión desde la línea de comandos. Por ejemplo:

```
shell> mysql -h 192.168.2.1 -u root -p < archivo-batch.sql
```

Algunas cuantas razones para usar scripts en lugar de trabajar en modo interactivo:

- Si se ejecutan un cierto número de consultas frecuentemente (cada día, o cada semana), al hacer un script nos evitamos tener que volver a teclear cada una de las consultas.
- Se pueden generar nuevas consultas que sean similares a las existentes al copiar y editar estos scripts.
- Al escribir consultas de varias líneas, los scripts ayudan bastante para que no se tengan que escribir todas las líneas nuevamente si se comete algún error.
- Se puede guardar la salida en un archivo para revisarla posteriormente.

```
shell> mysql < archivo-batch.sql > salida-del-script.txt
```

- Se pueden distribuir los scripts a otras personas para que puedan ejecutar también nuestros comandos y sentencias.
- En algunas situaciones no se permite el uso interactivo de *mysql*. (En el caso de las tareas programadas). En este caso, es indispensable usar el modo batch.

Cabe mencionar que el formato de la salida es diferente (más conciso) cuando se ejecuta *mysql* en modo batch, que cuando se usa de manera interactiva.

También se puede lanzar el script una vez conectados a *mysql* mediante el comando `source` (o el método abreviado `\.`)

```
mysql> source archivo-batch.sql
```

o el equivalente abreviado `mysql> \. archivo-batch.sql`

10.3.8 Verificar el estado, arrancar y detener el proceso servidor.

En la mayoría de las instalaciones de MySQL, en sus distintas versiones, se selecciona por defecto la opción de arrancar automáticamente el proceso servidor cuando se arranca el host sobre el que está instalado. Pero es fundamental para un administrador saber verificar, en un momento dado, el estado del servidor (en ejecución o detenido) y conocer perfectamente cual es el mecanismo de arranque y/o parada en el caso que sea necesario efectuarlo.

Nota:

El proceso (demonio) servidor en MySQL se denomina `mysqld` o una de sus variantes `mysqld-opt`, `mysqld-nt`, `mysqld-max`, `mysqld-max-nt`. (para ver las diferencias entre un proceso y otro acceder al manual de referencia puntos 2.2.1.4 para Windows y 2.2.2 para Linux).

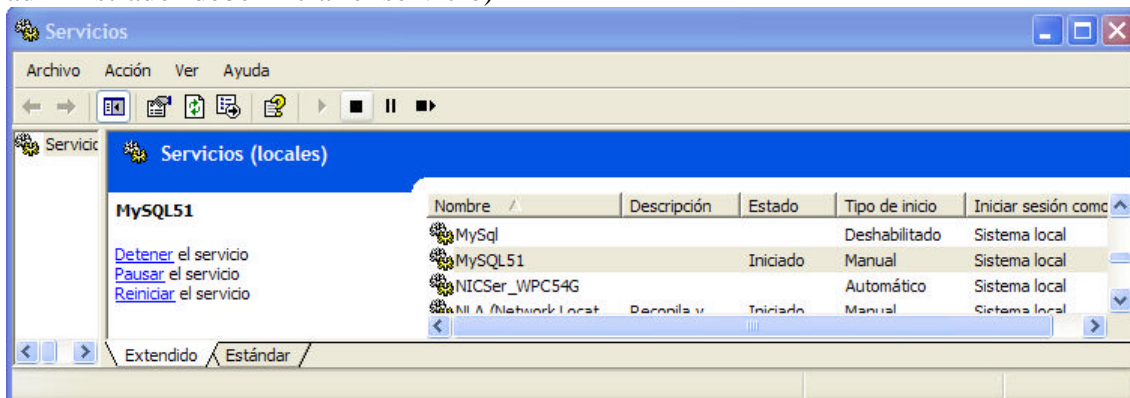
Verificar el estado del servidor en Windows.

En Windows (XP, 2000,...) lo más común es tener el servidor MySQL instalado como un servicio de Windows. Basta con acceder a la ventana de servicios y comprobar su estado.

Para acceder a la ventana de servicios de Windows rápidamente

Menú Inicio → Ejecutar → `services.msc`

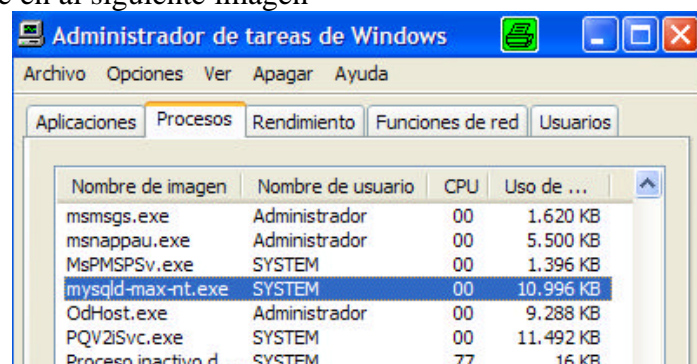
En la siguiente imagen se ve la ventana de servicios donde aparece el servicio asociado al servidor *MySQL51* que esta *Iniciado* y tiene un tipo de inicio *Manual* (el administrador debe iniciar el servicio)



En el caso de no tener MySQL asociado a un servicio o tener MySQL instalado en W98 a versión anterior lo mejor es acceder al Administrador de tareas de Windows

(*Menu Inicio → Ejecutar → `taskmgr`*)

En la pestaña de *Procesos* debe aparecer el proceso servidor si es que esta en ejecución tal como aparece en al siguiente imagen



Verificar el estado del servidor en Linux.

En Linux abrimos una terminal y pulsamos el comando

```
ps -A | grep mysqld
```

Si MySQL esta ejecutándose deberá responder con algo parecido a:

```
2752 ? 00:00:00:00 mysqld
```

Arrancar el proceso servidor en Windows.

Si tenemos el proceso servidor asociado a un servicio de Windows (lo más habitual) pulsamos botón derecho del ratón sobre el servicio que queremos iniciar y seleccionamos *Iniciar*.

En caso contrario lo mejor es iniciar el proceso desde una consola de comandos, es muy importante que en el proceso de arranque se indique mediante la opción `defaults-file` donde se encuentra el fichero de configuración (ver el punto de 11.2.2 de la documentación del curso para más información sobre el fichero de configuración). El comando será algo parecido a lo siguiente

```
C:\MySQL410\bin\mysqld-max-nt --defaults-file="C:\MySQL410\my.ini
```

Arrancar el proceso servidor en Linux.

En Linux lo más habitual es tener automatizado el arranque del servidor de MySQL cuando se inicia el equipo. En el caso de que el servicio este detenido tendremos que arrancar el proceso de forma manual.

En Linux hay muchas formas de arrancar el servidor MySQL. (Ver el punto 5.1 del Manual de Referencia de MySQL). Una de las formas más simples y seguras es arrancar el servidor a través del script de arranque `mysqld_safe` que suele encontrarse en la carpeta `/usr/bin` en la mayoría de las distribuciones. Por lo tanto abrimos una terminal y pulsamos el comando

```
/usr/bin/mysqld_safe &
```

Detener el proceso servidor en Windows.

Si tenemos el proceso servidor asociado a un servicio de Windows (lo más habitual) pulsamos botón derecho del ratón sobre el servicio que queremos iniciar y seleccionamos *Detener*.

En caso contrario lo mejor es usar el programa `mysqladmin` desde la consola de comandos de Windows y añadirle la orden `shutdown`

Ejemplo con usuario administrador de MySQL sin password

```
C:\> mysqladmin shutdown
```

Ejemplo con usuario administrador de MySQL con password

```
C:\>mysqladmin -uroot -ppassword shutdown
```

Detener el proceso servidor en Linux.

En Linux lo más cómodo es usar el programa `mysqladmin` exactamente igual como se ha comentado en el apartado anterior para detener el proceso servidor en Windows

Tema 11.

Administración de MySQL (I)

Javier Robles Cascallar

11.1. Estructura interna

Cada propio sistema de base de datos tiene una manera de organizar los datos y la información que manipula y por lo tanto MySQL tiene su forma particular que se va a detallar en este punto del documento.

11.1.1 Localización del directorio de datos

El directorio de datos hace referencia a la ubicación de la información que maneja el servidor (mysqld). En este directorio (carpeta) se almacenan tanto las bases de datos como los ficheros de estado (archivos de estado) que proporcionan la información sobre la operación de los servidores.

En Windows la ubicación típica suele ser `c:\mysql\data`. Si no se conoce la ubicación hay varias maneras de encontrarla:

- Utilizando `mysqladmin variables`.

En Windows :

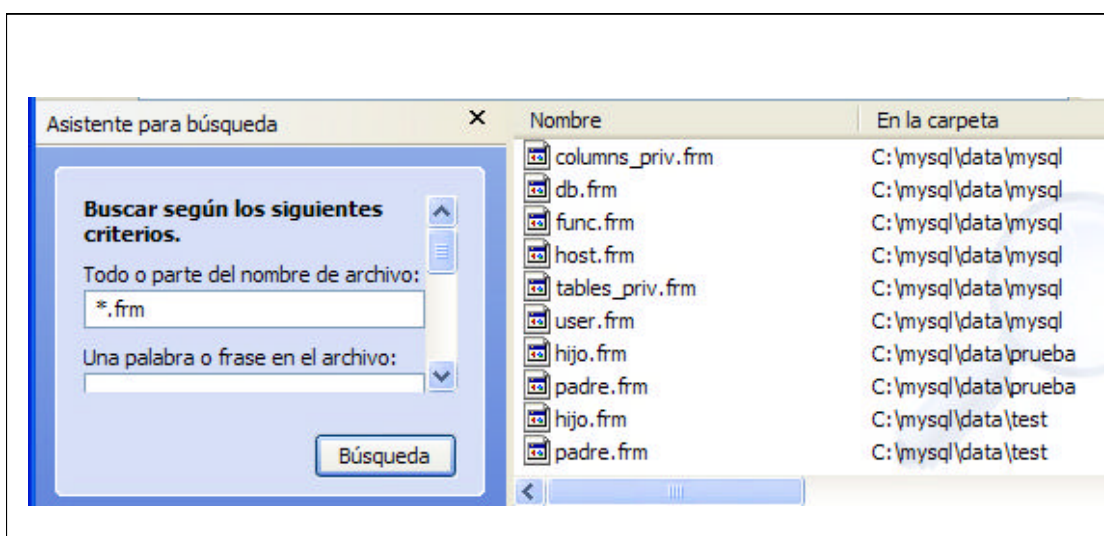
```
C:\mysql\bin>mysqladmin variables | more
```

[el uso de `more` es para detener la salida pantalla a pantalla]
[Buscamos en la salida generada la siguiente línea donde indica el valor de la variable `datadir`]

```
| datadir | C:\mysql\data\
```

- Buscando los archivos de tipo `.frm` (descripción) que forman parte de cualquier instalación de MySQL

En Windows:



11.1.2 Estructura del directorio de datos

Las bases de datos de MySQL se encuentran organizadas en una estructura en árbol que es implementado aprovechando la estructura jerárquica de archivos de los sistemas operativos sobre los que trabaja (Windows, UNIX, Linux,...). De tal manera que en líneas generales cada base de datos se corresponde con una carpeta bajo el directorio de datos y las tablas de esa base de datos se corresponden con los archivos de esa carpeta. Si se crea una base de datos llamada `prueba` las tablas de esa base de datos se almacenarán en la carpeta `DATADIR\prueba` (siendo `DATADIR` el directorio de datos por ejemplo `c:\mysql\data`)

```
C:\MySQL5\data>tree
Listado de rutas de carpetas
El número de serie del volumen es 000002C6 1229:8B69
C:.
+---mysql
+---prueba
+---test
```

11.1.3 Arquitecturas de almacenamiento de las tablas.

En este apartado es necesario detenerse en la forma que MySQL realiza el almacenamiento interno de los datos. MySQL tiene 6 formas de almacenamiento o indexación de la información que son las siguientes.

- MyISAM
- MERGE
- MEMORY (HEAP)
- BDB (Berkeley DB)
- ISAM
- InnoDB

MyISAM es la que se emplea por defecto en una instalación típica de MySQL a partir de la versión 3.23 (es una versión mejorada de ISAM)

InnoDB es la que se emplea y se instala en este curso por que permite control efectivo de transacciones, capacidad de bloqueo a nivel de fila y soporta la utilización de claves ajenas.

El resto de métodos de almacenamiento es menos utilizado y no se va a entrar en este documento a detallar sus características. (Remitimos al lector al manual de referencia de MySQL)

11.1.3.1 Tablas MyISAM

Cada tabla de la base de datos existe como tres archivos en la carpeta de la base de datos correspondiente: un archivo de formulario (o descripción), un archivo de datos y un archivo índice. El nombre base de cada archivo coincide con el nombre de la tabla y solo la extensión diferencia cada tipo de archivo.

Tipo de archivo	Extensión	Contenido del archivo
Archivo de formulario	.frm	Describe la estructura de la tabla (columnas, tipo de columnas, etc..)
Archivo de datos	.MYD	Contiene las filas de la tabla (los datos)
Archivo de índice	.MYI	Contiene los índices que se han creado sobre el archivo de datos.

Cuando se ejecuta un `CREATE TABLE nombre_tabla` el servidor crea los tres archivos (los de datos e índices también se crean aunque indicando que todavía están vacíos)

Por ejemplo si en una base de datos llamada `mi_club` creo dos tablas llamadas `cuotas` y `socios` se obtiene el siguiente listado de archivos en la carpeta `mi_club`

```
C:\MySQL5\data\mi_club>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 1229-8B69

Directorio de C:\MySQL5\data\mi_club

17/01/2005  12:21    <DIR>          .
17/01/2005  12:21    <DIR>          ..
26/12/2004  22:14                8.740 cuotas.frm
26/12/2004  22:51                432 cuotas.MYD
26/12/2004  22:51                3.072 cuotas.MYI
26/12/2004  22:14                9.034 socios.frm
26/12/2004  22:15                680 socios.MYD
26/12/2004  22:21                4.096 socios.MYI
                6 archivos                26.119 bytes
                2 dirs  24.336.830.464 bytes libres
```

Opciones de almacenamiento para tablas MyISAM:

- *fixed*: Se selecciona automáticamente cuando la tabla no contiene columnas tipo `VARCHAR`, `TEXT` o `BLOB` o también si la tabla se crea con la opción `ROW_FORMAT=FIXED` en la sentencia de creación de tabla (si se fuerza la creación de una tabla como `fixed` las columnas de longitud variable pasan a tener longitud fija). Esto permite trabajar con tablas de longitud fija de registro. Se puede acceder a las filas a través únicamente de su número de fila sin necesidad de desplazamiento (offset) lo cual permite que los índices asociados a este tipo de tablas sean más pequeños, rápidos y eficientes. También son tablas que no necesitan optimizarse ni reconstruirse puesto que no generan fragmentación.
- *dynamic*: Se selecciona automáticamente este tipo cuando la tabla contiene, al menos, alguna columna tipo `VARCHAR`, `TEXT` o `BLOB` o también si la tabla se crea con la opción `ROW_FORMAT=DYNAMIC` en la sentencia de creación de tabla. Estas tablas son de longitud de registro variable por lo tanto cada fila contiene una cabecera indicando su longitud. Todas las columnas de cadena (`VARCHAR` o `CHAR`) son de

tamaño variable incluso aquellas definidas como fijas (CHAR) con longitud superior a cuatro caracteres.

Este tipo de almacenamiento tiene la ventaja de ahorrar espacio en disco. Tiene por el contrario la desventaja de que los `deletes` y `updates` generan fragmentación y acaban degradando su uso por lo que en tablas con alto número de transacciones es conveniente, cada cierto tiempo, realizar un proceso de optimización de la tabla o desfragmentación. La optimización de tablas se puede realizar de dos formas. Lanzando una sentencia `OPTIMIZE TABLE` o ejecutando la utilidad *myisamchk* (que se encuentra como programa ejecutable en la carpeta `bin` de la distribución de MySQL).

Ejemplo de `OPTIMIZE TABLE`:

```
mysql> optimize table telefonos2;
+-----+-----+-----+-----+
| Table          | Op      | Msg_type | Msg_text |
+-----+-----+-----+-----+
| test.telefonos2 | optimize | status   | OK       |
+-----+-----+-----+-----+
1 row in set (0.02 sec)
```

- *Compressed*: Es un formato para tablas de solo-lectura generado con la herramienta *myisampack* que se encuentra en la carpeta `bin` de la distribución de MySQL.

Este tipo de tablas se caracterizan por su ahorro de espacio en disco lo que minimiza el uso de disco y es muy conveniente para dispositivos de almacenamiento lentos como los CD-ROM.

Ejemplo de uso de *myisampack* sobre una tabla de nombre `telefonos2` que contiene 10000 registros:

```
C:\mysql\bin>myisampack ..\data\test\telefonos2
Compressing ..\data\test\telefonos2.MYD: (10000 records)
- Calculating statistics
- Compressing file
66.69%
```

Para volver a obtener la tabla original (descomprimida y en modo lectura y escritura) usamos la herramienta *myisamchk* de la siguiente forma:

```
C:\mysql\bin>myisamchk -u ..\data\test\telefonos2
- recovering (with keycache) MyISAM-table
'..\data\test\telefonos2'
Data records: 10000
```

11.1.3.2 Tablas MERGE

Las tablas tipo MERGE no contienen datos por si mismas sino que hacen referencia a dos o más tablas tipo MyISAM a través de una cláusula `UNION`.

```

CREATE TABLE partel (
  campo1 INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  campo2 CHAR(20))
  TYPE=MYISAM;
CREATE TABLE parte2 (
  campo1 INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  campo2 CHAR(20))
  TYPE=MYISAM;

INSERT INTO partel (campo2) VALUES ('uno'),('dos'),('tres');
INSERT INTO parte2 (campo2) VALUES ('cuatro'),('cinco'),('seis');
CREATE TABLE total (
  campo1 INT NOT NULL AUTO_INCREMENT,
  campo2 CHAR(20), INDEX(campo1))
  TYPE=MERGE UNION=(partel,parte2) INSERT_METHOD=LAST;

```

Después de crear las tablas podemos seleccionar todas las filas de ambas tablas a partir de la tabla total

```

mysql> select * from total;
+-----+-----+
| campo1 | campo2 |
+-----+-----+
|      1 | uno    |
|      2 | dos    |
|      3 | tres   |
|      1 | cuatro |
|      2 | cinco  |
|      3 | seis   |
+-----+-----+
6 rows in set (0.00 sec)

```

```
mysql>
```

Físicamente al crear una tabla tipo MERGE como una unión de varias tablas MyISAM se generan dos ficheros uno de ellos con extensión .frm y el otro fichero con extensión .MRG. En el ejemplo anterior si listamos los ficheros de su carpeta obtenemos:

Directorio de C:\MySQL\data\test

```

13/01/2005  21:15    <DIR>          .
13/01/2005  21:15    <DIR>          ..
13/01/2005  21:15             8.598 partel.frm
13/01/2005  21:16             75 partel.MYD
13/01/2005  21:15             2.048 partel.MYI
13/01/2005  21:15             8.598 parte2.frm
13/01/2005  21:16             75 parte2.MYD
13/01/2005  21:15             2.048 parte2.MYI
13/01/2005  21:15             8.598 total.frm
13/01/2005  21:15             74 total.MRG
              8 archivos          30.114 bytes
              2 dirs  24.379.023.360 bytes libres

```

```
C:\MySQL\data\test>
```

Por supuesto que una vez definida la tabla MERGE total podemos seguir accediendo y modificando de forma independiente cada una de las tablas que forman total. Es decir se puede perfectamente lanzar las siguientes dos sentencias:

```
update total set campo2='primero' where campo2='uno';  
update parte2 set campo2='quinto' where campo2='cinco';
```

Una condición indispensable para crear tablas MERGE es que las tablas MyISAM de partida sean de idéntica estructura.

Borrar una tabla MERGE mediante `drop` no supone borrar las tablas que la componen, solo se borra la unión establecida entre las tablas.

11.1.3.3 Tablas MEMORY (HEAP)

Las tablas tipo HEAP (sinónimo de MEMORY) se caracterizan por almacenar todos los datos de la tabla en memoria. Nunca se accede a disco en una tabla MEMORY para acceder o guardar datos. Si se reinicia el servidor (¡ojo! el servidor, no la sesión de cliente) todas las tablas tipo MEMORY estarán vacías de filas aunque conservan sus estructura.

Ejemplo de creación de una tabla HEAP

```
CREATE TABLE emp_heap TYPE=MEMORY  
SELECT * FROM empleados;
```

O también

```
CREATE TABLE emp_heap ENGINE=HEAP  
SELECT * FROM empleados;
```

Estas tipo de tablas se caracterizan por ser extremadamente rápidas puesto que no necesitan ningún acceso a disco y son muy útiles para crear tablas temporales.

Las tablas tipo HEAP únicamente guardan en disco un fichero con extensión `.frm` que almacena la estructura de la tabla.

11.1.3.4 Tablas InnoDB

InnoDB realmente por si solo es un motor de bases de datos muy completo que ha sido embebido dentro de MySQL.

Las características de las tablas InnoDB son las siguientes:

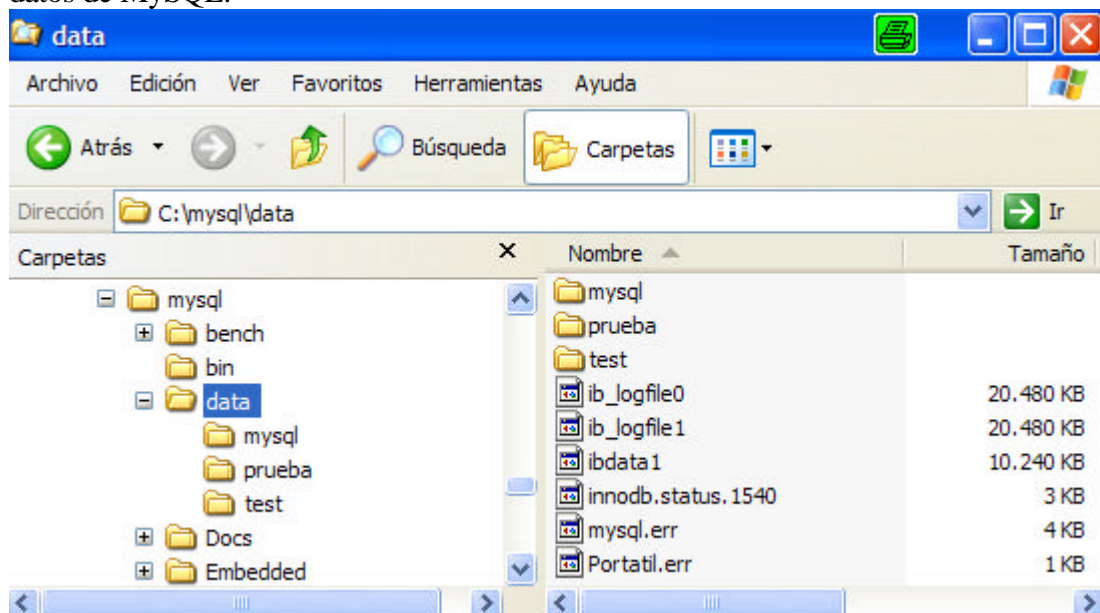
- Control de transacciones.
- Integridad referencial. Permite definir claves ajenas entre tablas InnoDB relacionadas para asegurarse de que un registro no puede ser eliminado de una tabla si aún está siendo referenciado por otra tabla.
- Bloqueo a nivel de filas. Al usar tablas MyISAM, y tener consultas muy grandes que requieren de mucho tiempo, simplemente no se podían ejecutar más

consultas hasta que terminarán las consultas que estaban en ejecución. En cambio, las tablas InnoDB usan bloqueo a nivel de filas para mejorar de manera considerable el rendimiento.

- **SELECTs sin bloqueo.** Como si el bloqueo a nivel de filas no fuera suficiente, el motor InnoDB usa una técnica conocida como multi-versioning (similar a la base de datos PostgreSQL) que elimina la necesidad de hacer bloqueos en consultas SELECT muy simples. Ya no es necesario molestarse porque una simple consulta de sólo lectura está siendo bloqueada por otra consulta que está haciendo cambios en una misma tabla.

La diferencia fundamental de InnoDB con respecto a MyISAM es que InnoDB almacena todos los datos e índices de todas las tablas de todas las bases de datos en un mismo tablespace que consiste un número reducido de archivos del sistema operativo formado por ficheros de datos (data files) y ficheros log (log files).

Si no se cambia la configuración de InnoDB que viene por defecto lo habitual es que se cree un archivo de datos llamado `ibdata1` de 10 Mbytes auto extensible y dos archivos de log llamados `ib_logfile0` y `ib_logfile1` todos ellos en el directorio de datos de MySQL.



Por lo tanto a diferencia del almacenamiento mediante MyISAM lo que hace InnoDB es en la carpeta correspondiente al nombre de la base de datos crear un único fichero con extensión `.frm` donde se describe la estructura de la tabla y los datos e índices quedan almacenados en el tablespace definido.

Por ejemplo si en una base de datos llamada `mi_club` creo dos tablas con almacenamiento InnoDB llamadas `cuotas` y `socios` se obtiene el siguiente listado de archivos en la carpeta `mi_club`

```
C:\MySQL5\data\mi_club>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 1229-8B69
```

Directorio de C:\MySQL5\data\mi_club

```
17/01/2005  12:21    <DIR>          .
17/01/2005  12:21    <DIR>          ..
26/12/2004  22:14                8.740 cuotas.frm
26/12/2004  22:14                65 db.opt
26/12/2004  22:14                9.034 socios.frm
              3 archivos                17.839 bytes
              2 dirs  24.336.830.464 bytes libres
```

Es importante advertir en este punto que si tenemos instalado MySQL con soporte InnoDB ello no implica que todas las tablas creadas tengan que ser de tipo InnoDB. El administrador o cualquier usuario con privilegios de creación de tablas podrá crear indistintamente tablas MyISAM o tablas InnoDB. El administrador se encargará de indicar en el fichero de configuración que tipo de tabla es la que se crea por defecto cuando no se especifica el tipo en la sentencia de SQL.

Es muy sencillo pasar una tabla de un tipo de almacenamiento a otro mediante el uso de la sentencia `ALTER TABLE` que permite modificar la estructura de almacenamiento de la tabla.

El siguiente ejemplo nos indica como a partir de la tabla `empleados` que inicialmente es de tipo InnoDB podemos convertirla a tipo MyISAM mediante un `ALTER TABLE` para volver a dejarla como estaba mediante un segundo `ALTER TABLE`:

```
ALTER TABLE empleados TYPE=MYISAM;
```

```
ALTER TABLE empleados TYPE=INNODB;
```

Las opciones de configuración de las tablas InnoDB son muchísimas y se sale de los objetivos de este documento el dar una visión, aunque sea somera, de todas ellas. Como en otras ocasiones remitimos al lector al Manual De MySQL (Concretamente a los apartados *16.4 InnoDB Configuration* y *16.5 InnoDB Startup Options*)

Finalmente la siguiente tabla nos muestra una comparativa entre los tipos de almacenamiento más importantes

Comparativa

	MyISAM	InnoDB	MEMORY
Control de transacciones	-	X	-
Claves ajenas	-	X	-
Nivel de bloqueo	Tabla	fila	tabla
indices BTREE	X	X	-
Indices FULLTEXT	X	-	-
Indices GIS, RTREE	4.1.0	-	-
Unicode	4.1.0	4.1.2	-
Merge	X	-	-
Almacenamiento de lectura comprimido	X	-	-
Uso de disco	Bajo	alto	-
Uso relativo de RAM	Bajo	alto	bajo

11.2. Configuración

los programas que componen MySQL (mysqld, mysqladmin, etc.) admiten muchas posibilidades de configuración. Dichas opciones pueden ser especificadas a través de la línea de comandos o bien en ficheros de opciones. Algunas de las opciones es necesario establecerlas configurando variables de entorno. Las especificadas a través de la línea de comandos tienen preferencias sobre los ficheros de opciones y estos sobre los valores de variables de entorno.

11.2.1 Usar opciones de configuración a través de la línea de comandos.

Las opciones introducidas a través de la línea de comandos deben seguir las siguientes reglas.

- Siempre se introducen después del nombre del programa.
- La opción siempre va precedida de un guión o dos guiones (- , --) dependiendo de si se utiliza la forma corta o la forma larga (muchas de las opciones admiten las dos formas). Por ejemplo ambas líneas a continuación generan la misma salida que es la ayuda *on-line* sobre el programa incluida la lista de opciones admitida:

```
C:\mysql\bin> mysql --help
C:\mysql\bin> mysql -?
```

- Algunas de las opciones van seguidas de un valor (el valor que toma la opción) si se usa la versión larga para el nombre de la opción el valor debe ir precedido del signo '='. Si se usa la versión corta no es necesario el signo '=' se puede poner un espacio para separar el nombre del valor pero tampoco es obligatorio puede ir el nombre de la opción seguida del valor que toma la opción . Por ejemplo las siguientes tres líneas son equivalentes y significan que se esta ejecutando el programa mysql desde un equipo cliente para acceder a un host llamado *miservidordb*:

```
C:\mysql\bin>mysql --host= miservidordb
C:\mysql\bin>mysql -h miservidordb
C:\mysql\bin>mysql -hmiservidordb
```

Una excepción a esta regla del espacio es la opción que permite introducir la password con la que realiza la conexión al servidor. Solo admite dos formas

```
--password=contraseña o -pcontraseña
no está admitida la entrada
-p contraseña
```

Esto es debido a que si se introduce la opción -p sin introducir ninguna contraseña el programa a continuación pide que se introduzca una. Si se admitiera el espacio el programa no puede saber si lo que viene a continuación es el nombre de la contraseña o el nombre de otra opción.

- Algunos de los programas tienen variables (parámetros operativos) que son configurables. La opción `-set-variable` permite configurar este tipo de variables. Se puede ver esto con el siguiente ejemplo:

El programa *mysql* tiene una variable que se llama *max_allowed_packet* que indica el tamaño máximo del buffer de comunicación que se establece entre cliente y servidor. Si se quiere ejecutar el programa *mysql* estableciendo un tamaño de buffer en 16 Mbytes se lanza la siguiente orden:

```
C:\mysql\bin>mysql --set-variable=max_allowed_packet=16M
```

No se va a entrar en este documento a detallar todas las posibles opciones que admiten los programas de MySQL a través de la línea de comandos. Para ello remitimos al lector al *Manual de Referencia de MySQL* donde puede localizar toda la información necesaria (Muy interesante, si se desea profundizar, es el punto *5.2.1 mysqld Command-Line Options*. Donde se explica de forma detallada las posibilidades de configuración del servidor *mysqld* a través de la línea de comandos, de hecho, y como se verá más adelante, toda la información de este punto es válida para configurar los “ficheros de opciones” del punto siguiente).

11.2.2 Usar ficheros de opciones (ficheros de configuración).

Los ficheros de opciones o ficheros de configuración sirven para que los programas MySQL puedan cargar siempre las mismas opciones de configuración al arrancar, evitando de esta manera que se lancen muchas de las opciones de configuración de forma manual desde la línea de comandos.

El listado de los programas MySQL que soportan ficheros de configuración es el siguiente:

<i>myisamchk</i> ,	<i>mysqlcheck</i> ,
<i>myisampack</i> ,	<i>mysqld_safe</i> ,
<i>mysql</i> ,	<i>mysqldump</i> ,
<i>mysql.server</i> ,	<i>mysqld</i> ,
<i>mysqladmin</i> ,	<i>mysqlhotcopy</i> ,
<i>mysqlbinlog</i> ,	<i>mysqlimport</i> ,
<i>mysqlcc</i> ,	<i>mysqlshow</i> .

Los programas MySQL pueden consultar más de un fichero de configuración (o no consultar ninguno) dependiendo de la ubicación en la que se encuentren dentro del sistema operativo correspondiente.

Bajo Windows se comprueban los siguientes ficheros en el orden dado:

Nombre fichero	Contenidos
<i>WINDIR\my.ini</i>	Opciones globales
<i>C:\my.cnf</i>	Opciones globales

WINDIR hace referencia a la ruta del sistema de Windows (típicamente '*C:\Windows*' o '*C:\WinNT*').

Bajo UNIX/Linux se comprueban los siguientes ficheros en el orden dado:

Nombre fichero	Contenidos
/etc/my.cnf	Opciones globales
DATADIR/my.cnf	Opciones específicas del servidor
~/my.cnf	Opciones específicas del usuario

DATADIR hace referencia al directorio de datos de MySQL.

La sintaxis de estos ficheros de configuración es muy similar a lo ya visto en el punto anterior mediante entradas a través de la línea de comandos.

Las líneas en blanco se ignoran. Las líneas que no están en blanco deben ser de alguno de los siguientes tipos:

```
#comentario  
;comentario
```

Las líneas comentadas pueden empezar con '#' y ';'.

```
[grupo]
```

grupo es el nombre del programa MySQL sobre el que a continuación y sobre el fichero se establecen las opciones de configuración. El nombre de grupo *client* permite especificar opciones que se aplican de forma común a todos los programas cliente.

```
nombre-opcion
```

Es el equivalente a `--nombre-opcion` en la línea de comandos.

```
nombre-opcion=valor
```

Es el equivalente a `--nombre-opcion=valor` en la línea de comandos.

```
set-variable = nombre_variable=valor
```

Es el equivalente a `--set-variable = nombre_variable=valor` en la línea de comandos. Los espacios son válidos alrededor del primer signo '=' pero no son permitidos alrededor del segundo.

Ejemplo de fichero de opciones my.ini

```
#Opciones de configuración para el servidor mysqld
[mysqld]
basedir=C:/mysql
datadir=C:/mysql/data

#la siguiente línea permite poner los mensajes de error
#en castellano (comprobar el path)
language=C:/mysql/share/spanish

# descomentar solo si se desea cambiar el puerto por defecto
#port=3306

# Configuración del soporte InnoDB
#####
# El nombre del tablespace es ibdata1 de 10Mbytes
# y auto extensible
innodb_data_file_path = ibdata1:10M:autoextend
#
# El tamaño para el buffer pool suele ser un 50%
# de la memoria ram del equipo
set-variable = innodb_buffer_pool_size=70M
# El tamaño adicional de pool se usa para aloacar
#el diccionario de datos
set-variable = innodb_additional_mem_pool_size=10M
#
# Opciones para la configuración de los ficheros log
# se suele reservar el 25% del espacio reservado para el buffer pool
set-variable = innodb_log_file_size=20M
set-variable = innodb_log_buffer_size=8M
#La siguiente línea lo normal es ponerla a valor 1
innodb_flush_log_at_trx_commit=1
#La siguiente línea indica que las tablas por defecto se
#crean en modo InnoDB
default-table-type=innodb
#####

[WinMySQLadmin]
Server=C:/mysql/bin/mysqld-max-nt.exe
user=admin
password=admin
```

11.3. Administración de cuentas de usuario y seguridad

Este apartado proporciona información acerca de cómo el administrador MySQL puede cumplir una de las tareas más importantes que tiene encomendada y es la responsabilidad de mantener seguros los contenidos de las bases de datos para que los registros sean accesibles únicamente por aquellos usuarios con autorización para ello

El administrador debe ser capaz de establecer cuentas de usuario de MySQL indicando

- que usuarios pueden conectar al servidor
- desde donde pueden conectar
- que pueden hacer mientras están conectados

El sistema de privilegios de MySQL asegura que todos los usuarios puedan hacer exactamente las cosas que les son permitidas. Cuando se hace una conexión al servidor MySQL, la identidad es determinada por el **host desde el cual se hace la conexión** y el **nombre de usuario**. MySQL considera ambos el hostname y el nombre de usuario en la identificación por esta razón se asume que un usuario dado se puede conectar desde diferentes partes.

Por ejemplo: el usuario Pepe que se conecta desde *cnice.mecd.es* no necesita ser la misma persona para conectarse como usuario Pepe desde *educa.madrid.org* MySQL maneja esto para permitir que se distingan los usuarios en diferentes hosts que a menudo tienen el mismo nombre: se puede otorgar un conjunto de privilegios para conectarse desde *cnice.mecd.es* y otro conjunto de privilegios para la conexión desde *educa.madrid.org*. El control de acceso al servidor se desarrolla en dos partes:

- El servidor chequea si está permitida la conexión.
- Asumiendo que existe una conexión, el servidor chequea cada petición que se emita para ver si se tiene suficientes privilegios para ejecutarla.

11.3.1 Introducción a las sentencias GRANT y REVOKE

A partir de la versión 3.22.11 de MySQL se introducen dos sentencias que son capaces de realizar la tarea de la administración de cuentas de usuario. Dichas sentencias son GRANT y REVOKE.

GRANT crea los usuarios y especifica sus privilegios y REVOKE elimina estos privilegios.

El “diccionario de datos” de MySQL se encuentra en el “directorio de datos” en una base de datos llamada *mysql*. En esta base de datos aparecen cuatro tablas que recogen la información necesaria para especificar los usuarios y sus privilegios. GRANT y REVOKE proporcionan una alternativa más cómoda a la manipulación directa del contenido de estas tablas. Las tablas son las siguientes:

Nombre de tabla	Contenido
user	Usuarios que pueden conectar con el servidor
db	Privilegios a nivel de base de datos
tables_priv	Privilegios a nivel de tabla
columns_priv	Privilegios a nivel de columna

11.3.2 Sintaxis de la sentencia GRANT

```
GRANT privilegios (columnas)
      ON nivel_privilegio
      TO usuario@host IDENTIFIED BY "contraseña"
      WITH GRANT OPTION
```

Para usar la sentencia tenemos que rellenar las partes que están en cursiva en la sentencia anterior y que se explican brevemente a continuación

- Privilegios.

La tabla de los privilegios más importantes que se pueden conceder a un usuario es la que se muestra a continuación

Privilegio	Significado del privilegio
Privilegios aplicados a bases de datos, tablas y columnas	
ALTER	Permite alterar tablas e índices (ALTER TABLE)
CREATE	Permite crear bases de datos y tablas (CREATE TABLE)
CREATE TEMPORARY TABLES	Permite crear tablas temporales (CREATE TEMPORARY TABLE)
DELETE	Permite borrar registros de las tablas existentes (DELETE)
DROP	Permite eliminar bases de datos y tablas (DROP TABLE)
INDEX	Permite crear y eliminar índices asociados a tablas (CREATE INDEX y DROP INDEX)
INSERT	Permite insertar nuevos registros en tablas (INSERT)
SELECT	Permite consultar información de tablas (SELECT)
UPDATE	Permite modificar el contenido de registros existentes (UPDATE)
Privilegios administrativos	
FILE	Permite leer y escribir archivos en el servidor (SELECT ... INTO OUTFILE y LOAD DATA INFILE)
PROCESS	Permite ver información sobre los procesos abiertos en el servidor (SHOW FULL PROCESSLIST)
RELOAD	Permite solicitar al servidor que cargue de nuevo en memoria las tablas del diccionario de datos (FLUSH)
REPLICATION CLIENT	Permite al usuario preguntar donde esta el servidor esclavo o el maestro

REPLICATION SLAVE	Permiso necesario para que el usuario pueda acceder al fichero binary log del maestro desde el esclavo
SHUTDOWN	Permite cerrar el servidor (<code>mysqladmin shutdown</code>)
Privilegios especiales	
ALL [PRIVILEGES]	Permite agrupar todos los privilegios en uno solo.
USAGE	Es un privilegio especial que significa “ausencia de privilegios”

- Nivel de privilegio.

Los privilegios pueden aplicarse a cuatro niveles

1. Nivel global. Se aplican los privilegios a todas las bases de datos y todas las tablas del servidor.
La forma de indicar el nivel global será la siguiente:
`GRANT privilegios ON *.* ...`
2. Nivel de base de datos. Se aplican los privilegios sobre todas las tablas de la base de datos nombrada en la sentencia.
La forma de indicar el nivel de base de datos será la siguiente:
`GRANT privilegios ON nombre_bd.* ...`
3. Nivel de tabla. Se aplican los privilegios sobre todas las columnas de la tabla nombrada de la base de datos nombrada.
La forma de indicar el nivel de tabla será la siguiente:
`GRANT privilegios ON nombre_bd.nombre_tabla ...`
4. Nivel de columna. Se aplican los privilegios sobre la/s columna/s seleccionadas de una tabla determinada.

- Usuario@Host.

El nombre del usuario al que se le están asignando los privilegios. Consiste en un nombre de usuario y un nombre de host (se especifica quien puede conectarse y desde donde). Esto posibilita tener dos usuarios de idéntico nombre que se conectan desde diferentes localizaciones y por lo tanto se les pueden administrar privilegios distintos dependiendo de donde se conecten.

Los nombres de usuario de MySQL son independientes de los nombres de usuarios de los Sistemas Operativos del equipo sobre el que están intentando efectuar la conexión al servidor de base de datos.

Se puede permitir que un usuario se conecte desde un host específico o desde un conjunto de ellos.

Como nombre de host se pueden utilizar direcciones IP o nombres DNS

Esta permitido el uso de caracteres comodín ('%') para direcciones IP o nombres DNS.

La siguiente tabla muestra algunos ejemplos de nombres de usuario y nombres de host que permiten ver las posibilidades de conexión que existen con las diferentes combinaciones que se pueden dar:

host	Usuario	Conexiones permitidas con la combinación usuario@host
'www.mecd.es'	'juan'	juan, conectando desde el host www.mecd.es
'www.mecd.es'	' '	Cualquier usuario, conectando desde el host www.mecd.es
'%'	'juan'	juan, conectando desde cualquier host
'%'	' '	Cualquier usuario, conectando desde cualquier host
'%.mecd.es'	'juan'	juan, conectando desde algún host del dominio mecd.es
'x.y.%'	'juan'	juan, conectando desde los host x.y.es, x.y.com, x.y.org, etc.. (Se puede usar aunque no parece muy útil)
'144.155.166.177'	'juan'	juan, conectando desde el host con IP 144.155.166.177
'144.155.166.%'	'juan'	juan, conectando desde cualquier host de la subred 144.155.166
'144.155.166.0/255.255.255.0'	'juan'	Otra forma de poner el ejemplo anterior

- Contraseña.

La cláusula IDENTIFIED BY no es obligatoria, pero su ausencia, si el usuario no existe, implica la creación de un usuario nuevo sin contraseña (lo cual por razones evidentes de seguridad no es muy aconsejable). Si el usuario existe la contraseña que acompaña la cláusula sustituye la anterior y si el usuario existe y no se usa la cláusula IDENTIFIED BY permanece vigente la que tenía antes.

- Cláusula WITH GRANT OPTION.

Significa que el usuario tiene la posibilidad de transmitir sus privilegios a otros usuarios. Por supuesto esta cláusula es opcional y se utiliza para permitir delegar capacidades de transferencia a otros usuarios.

11.3.3 Sintaxis de la sentencia REVOKE

La sintaxis de REVOKE es muy parecida a GRANT excepto que TO es reemplazado por FROM y no aparecen las cláusulas IDENTIFIED BY y WITH GRANT OPTION.

```
REVOKE privilegios (columnas)
      ON nivel_privilegio
      FROM usuario
```

La parte `usuario` de la sentencia REVOKE debe coincidir con la parte `usuario` de la sentencia GRANT original cuyos privilegios se quieren revocar. Por el contrario la parte `privilegios` no necesita coincidir por que puede que el administrador quiera revocar solo parte de los privilegios que se le concedió al usuario inicialmente.

A partir de la versión 4.1.2 de MySQL existe una forma más cómoda de eliminar de golpe todos los privilegios de un usuario (o varios)

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM usuario,usuario,...;
```

Eliminar todos los privilegios no supone la eliminación del usuario ya que este aun permanece como un registro en la tabla `user` lo que significa que el usuario puede conectar con el servidor.

11.3.4 Algunos ejemplos de la sentencia GRANT

```
shell> mysql --user=root mysql
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@localhost
      IDENTIFIED BY 'algo' WITH GRANT OPTION;
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@"%"
      IDENTIFIED BY 'algo' WITH GRANT OPTION;
mysql> GRANT RELOAD, PROCESS ON *.* TO admin@localhost;
mysql> GRANT USAGE ON *.* TO dummy@localhost;
```

Aquí se han creado tres nuevos usuarios: **monty**: Un superusuario el cual puede conectarse al servidor desde cualquier parte, pero debe usar un password (como ser algo). Notar que se debe emitir GRANT para `monty@localhost` y `monty@"%"`. Si no se agrega la entrada con `localhost`, el usuario anónimo entra por el `localhost` que es creado por `mysql_install_db` tomando prioridad cuando se conecte desde un host local, porque éste posee un valor de campo `Host` más específico y viene primero en el orden de serie de la tabla `user`. **admin**: Un usuario que puede conectarse desde el `localhost` sin un password y al cual se le otorga los privilegios de administración `reload` y `process`. A este usuario se le permite ejecutar los comandos `mysqladmin reload`, `mysqladmin refresh` y `mysqladmin flush*`, así como también `mysqladmin processlist`. **dummy**: Un usuario que puede conectarse sin un password, pero sólo desde host local. Los privilegios globales son todos colocados a "N", el tipo de privilegio `USAGE` permite colocar un usuario con ningún privilegio. Este asume que se deben otorgar privilegios específicos para las bases de datos.

El siguiente ejemplo agrega un usuario **cliente** el cual puede conectarse desde los siguientes hosts: `localhost`, `server.domain` y `casa.gov`. Éste desea acceder a la base de datos **Bancaria** sólo desde el **hostlocal**, a la base de datos **Costos** sólo desde el host **casa.gov** y a la base de datos **Clientes** desde los tres host. Se usará el password **mipass** para los tres hosts. Se otorgaran estos privilegios usando ambos métodos.

```
shell> mysql --user=root mysql
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
      ON Bancaria.*
      TO cliente@localhost
      IDENTIFIED BY 'mipass';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
      ON Costos.*
      TO cliente@casa.gov
      IDENTIFIED BY 'mipass';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
      ON Clientes.*
      TO cliente@"%"
      IDENTIFIED BY 'mipass';
```


Si se desea dar acceso a un usuario específico desde cualquier máquina en un dominio dado, se puede usar tanto la cláusula GRANT como INSERT de la siguiente forma:

```
mysql> GRANT ...  
      ON *.*  
      TO myusername@"%.mydomainname.com"  
      IDENTIFIED BY 'mypassword';
```

11.3.5 Eliminar una cuenta de usuario.

Para eliminar completamente una cuenta de usuario la sentencia que se utiliza es DROP USER (antes de la versión 4.1.1 se usaba la sentencia DELETE).

Para poder eliminar completamente un usuario este ha tenido que ser revocado previamente de todos sus privilegios.

Si no estamos seguros de los privilegios que tiene un determinado usuario podemos usar la sentencia SHOW GRANTS para verlo. Por ejemplo

```
mysql> SHOW GRANTS FOR 'admin'@'localhost';
```

Si aparece un listado con privilegios debemos usar la sentencia REVOKE para quitárselos antes de poder utilizar DROP USER.

```
mysql> DROP USER 'admin'@'localhost';
```

11.4. Índices.

Los índices son usados para encontrar rápidamente los registros que tengan un determinado valor en alguna de sus columnas. Sin un índice, MySQL tiene que iniciar con el primer registro y leer a través de toda la tabla para encontrar los registros relevantes. Aún en tablas pequeñas, de unos 1000 registros, es por lo menos 100 veces más rápido leer los datos usando un índice, que haciendo una lectura secuencial o “escaneo completo de una tabla”.

En particular, debemos evitar los escaneos completos de tablas por las siguientes razones:

Sobrecarga de CPU. El proceso de chequear cada uno de los registros en una tabla es insignificante cuando se tienen pocos datos, pero puede convertirse en un problema a medida que va aumentando la cantidad de registros en nuestra tabla.

Concurrencia. Mientras MySQL está leyendo los datos de una tabla, éste la bloquea, de tal manera que nadie más puede escribir en ella, aunque si pueden leerla. Cuando MySQL está actualizando o eliminando filas de una tabla, éste la bloquea, y por lo tanto nadie puede al menos leerla.

Sobrecarga de disco. En una tabla muy grande, un escaneo completo consume una gran cantidad de entrada/salida en el disco.

En resumen, lo mejor es tratar de que los escaneos completos de tablas sean mínimos - especialmente si nuestra aplicación necesita escalabilidad en tamaño, número de usuarios, o ambos.

Cuando indexamos una columna en particular, MySQL crea otra estructura de datos (un índice) que usa para almacenar información extra acerca de los valores en la columna indexada. Los valores indexados son llamados frecuentemente claves. Aunque esta es la manera simple de explicar los índices, realmente es un poco más complejo, ya que MySQL almacena todas las claves del índice en una estructura de datos de árbol. Esta estructura de datos de árbol le permite a MySQL encontrar claves muy rápidamente.

11.4.1 Tipos de índices y su creación

Existen cuatro tipos de índices que podemos utilizar en MySQL:

- de clave primaria
- únicos
- de texto completo
- ordinarios

Cada uno de ellos será explicado a continuación.

Índices de clave primaria

Una clave primaria, como ya sabemos, es un índice sobre uno o más campos donde cada valor es único y ninguno de los valores son NULL.

Para crear un índice de clave primaria tenemos básicamente dos opciones:

1. Crear el índice de clave primaria al momento de crear la tabla. En este caso se usa, como ya sabemos la opción **PRIMARY KEY** al final de la definición de los campos, con una lista de los campos que serán parte del índice.
2. Crear una clave primaria en una tabla existente con el uso del comando **ALTER TABLE**:

La sentencia **DESC** es útil para identificar las columnas que son clave primaria en la tabla

```
mysql> DESC usuarios;
```

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI	0	
nombre	varchar(50)	YES		NULL	
apellidos	varchar(70)	YES		NULL	

3 rows in set (0.00 sec)

El campo **id** no tiene un valor **YES** en la columna **Null**, lo que indica que este campo ya no podrá almacenar valores nulos. Se puede observar también que se tiene un valor **PRI** en la columna **Key**, lo que indica que este campo es una clave primaria.

Las claves primarias pueden constar de más de un campo. Hay algunas veces en las que un solo campo no puede identificar de manera única a un registro.

Índices ordinarios

Son índices que no son primarios y permiten valores duplicados (a menos que los campos hayan sido especificados como **UNIQUE**).

Para crear un índice ordinario tenemos básicamente tres opciones:

1. Podemos crear un índice ordinario al mismo tiempo que creamos la tabla con el uso de la opción **INDEX**.

```
CREATE TABLE nombreTabla(campo1 tipoDato, campo2 tipoDato,...
INDEX [nombreIndice] (campo1 [,campo2...]));
```

2. De igual manera, podemos crear el índice con el uso de la sentencia **ALTER TABLE** si es que la tabla ya existe.

```
ALTER TABLE nombreTabla ADD INDEX [nombreIndice] (campo1  
[,campo2...]);
```

3. También es posible usar la sentencia `CREATE INDEX` para crear un índice en una tabla existente.

```
CREATE INDEX nombreIndice ON nombreTabla(campo1 [,campo2...]);
```

De todas las formas se pide el nombre del índice, sin embargo con la sentencia `CREATE INDEX` el nombre es obligatorio.

Por ejemplo, para la siguiente definición de tabla:

```
CREATE TABLE usuarios(id int, nombre varchar(50), apellidos  
varchar(70));
```

Se puede crear un índice en la columna `apellidos` con una sentencia `ALTER TABLE`:

```
ALTER TABLE usuarios ADD INDEX idx_apellidos (apellidos);
```

O bien, con una sentencia `CREATE INDEX`:

```
CREATE INDEX idx_apellidos ON usuarios(apellidos);
```

Índices de texto completo

Los índices de texto completo son del tipo `FULLTEXT`, se usan en tablas del tipo `MyISAM`, y pueden contener uno o más campos del tipo `CHAR`, `VARCHAR` y `TEXT`. Un índice de texto completo está diseñado para facilitar y optimizar la búsqueda de palabras clave en tablas que tienen grandes cantidades de información en campos de texto.

Para crear un índice de texto completo tenemos básicamente las mismas tres opciones:

1. `CREATE TABLE nombreTabla(campo1 TIPO, campo2 TIPO,
FULLTEXT [nombreIndice] (campo1 [campo2,...]));`
2. `ALTER TABLE nombreTabla ADD FULLTEXT [nombreIndice] (campo1
[,campo2,...]);`
3. `CREATE FULLTEXT INDEX nombreIndice ON nombreTabla(campo1
[,campo2,...]);`

Como ejemplo consideremos la siguiente definición de tabla:

```
CREATE TABLE usuarios(id int, nombre varchar(50), apellidos  
varchar(70));
```

Podríamos crear un índice `FULLTEXT` en la columna `nombre`, en la columna `apellidos`, o bien, un índice que ocupe ambos campos. A continuación se muestran los tres casos.

```
CREATE FULLTEXT INDEX idx_nombre ON usuarios(nombre);
```

```
CREATE FULLTEXT INDEX idx_apellidos ON usuarios(apellidos);
```

```
CREATE FULLTEXT INDEX idx_nombre_apellidos ON  
usuarios(nombre,apellidos);
```

- *Nota:* Cuando se tienen grandes cantidades de datos, es mucho más rápido cargar los datos en una tabla que no tiene índices de texto completo y después crear los índices necesarios, ya que la carga de datos en una tabla que ya tiene índices de este tipo es un proceso lento.

Índices únicos

Los índices únicos son básicamente como los índices ordinarios, excepto que los valores duplicados no son permitidos.

Para crear un índice UNIQUE se tienen de nuevo las mismas tres opciones:

1.

```
CREATE TABLE nombreTabla(campo1 tipoDato, campo2 tipoDato,..  
    UNIQUE [nombreIndice] (campo1 [,campo2...]));
```
2.

```
ALTER TABLE nombreTabla ADD UNIQUE [nombreIndice] (campo1,  
    campo2) ...
```
3.

```
CREATE UNIQUE INDEX nombreIndice ON nombreTabla(campo1  
    [,campo2...]);
```

Siguiendo con el ejemplo, consideremos de nuevo la siguiente definición de tabla:

```
CREATE TABLE usuarios(id int, nombre varchar(50), apellidos  
varchar(70));
```

Podríamos crear un índice UNIQUE en la columna nombre, y un índice UNIQUE en la columna apellidos.

```
ALTER TABLE usuarios ADD UNIQUE idx_nombre (nombre);
```

```
CREATE UNIQUE INDEX idx_apellidos ON usuarios(apellidos);
```

En el primer caso hacemos uso del comando `ALTER TABLE`, y en el segundo caso creamos el índice con la sentencia `CREATE INDEX`.

11.4.2 Eliminar o cambiar un índice

Algunas veces tendremos la necesidad de cambiar o eliminar un índice. Cuando hagamos algún cambio en el índice, necesitamos eliminar primero el índice y entonces reconstruirlo con la nueva definición.

Para eliminar un índice de clave primaria podemos usar la siguiente sintaxis:

```
ALTER TABLE nombreTabla DROP PRIMARY KEY;
```

Para eliminar un índice ordinario, único, o de texto completo, necesitamos especificar el nombre del índice y usar esta sintaxis:

```
ALTER TABLE nombreTabla DROP INDEX nombreIndice;
```

También es válida esta otra sintaxis:

```
DROP INDEX nombreIndice ON nombreTabla;
```

Si no estamos seguros de cuál es el nombre del índice que deseamos eliminar, podemos hacer uso de la sentencia `SHOW KEYS`:

```
SHOW KEYS FROM nombreTabla;
```

Este es un ejemplo.

```
CREATE TABLE usuarios
(
  id INT NOT,
  nombre VARCHAR(50) NOT NULL,
  apellidos VARCHAR(70) NOT NULL,
  PRIMARY KEY (id),
  INDEX (nombre, apellidos)
);
```

Veamos los índices que existen en esta tabla:

```
mysql> SHOW KEYS FROM usuarios;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	
usuarios	0	PRIMARY	1	id	.
usuarios	1	nombre	1	nombre	.
usuarios	1	nombre	2	apellidos	.

```
3 rows in set (0.00 sec)
```

La tercera columna es la que nos proporciona los nombres de los índices. Podemos observar que al no especificar un nombre al índice ordinario en (nombre, apellidos), se le ha asignado el nombre de la primera columna que forma el índice.

A continuación vamos a eliminar los dos índices que existen en esta tabla:

```
mysql> ALTER TABLE usuarios DROP PRIMARY KEY;
```

```
Query OK, 0 rows affected (0.01 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> ALTER TABLE usuarios DROP INDEX nombre;
Query OK, 0 rows affected (0.00 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Por último, podemos verificar que estos índices ya no existen:

```
mysql> SHOW KEYS FROM usuarios;
Empty set (0.00 sec)
```

11.4.3 Uso eficiente de los índices

Dado que los índices hacen que las consultas se ejecuten más rápido, podemos estar incitados a indexar todas las columnas de nuestras tablas. Sin embargo, lo que tenemos que saber es que el usar índices tiene un precio. Cada vez que hacemos un `INSERT`, `UPDATE`, o `DELETE` sobre una tabla, MySQL tiene que actualizar cualquier índice en la tabla para reflejar los cambios en los datos.

¿Así que, cómo decidimos usar índices o no?. La respuesta es "depende". De manera simple, depende que tipo de consultas ejecutamos y que tan frecuentemente lo hacemos, aunque realmente depende de muchas otras cosas.

La razón para tener un índice en una columna es para permitirle a MySQL que ejecute las búsquedas tan rápido como sea posible (y evitar los escaneos completos de tablas). Podemos pensar que un índice contiene una entrada por cada valor único en la columna. En el índice, MySQL debe contar cualquier valor duplicado. Estos valores duplicados decrementan la eficiencia y la utilidad del índice.

Así que antes de indexar una columna, debemos considerar que porcentaje de entradas en la tabla son duplicadas. Si el porcentaje es demasiado alto, seguramente no veremos alguna mejora con el uso de un índice.

Otra cosa a considerar es qué tan frecuentemente los índices serán usados. MySQL puede usar un índice para una columna en particular únicamente si dicha columna aparece en la cláusula `WHERE` en una consulta.

Si muy rara vez usamos una columna en una cláusula `WHERE`, seguramente no tiene mucha sentido indexar dicha columna. De esta manera, probablemente sea más eficiente sufrir el escaneo completo de la tabla las raras ocasiones en que se use esta columna en una consulta, que estar actualizando el índice cada vez que cambien los datos de la tabla.

Ante la duda, no tenemos otra alternativa que probar. Siempre podemos ejecutar algunas pruebas sobre los datos de nuestras tablas con y sin índices para ver como obtenemos los resultados más rápidamente. Lo único a considerar es que las pruebas sean lo más realistas posibles.

11.5. Backup

MySQL ofrece varias alternativas para hacer copias de seguridad de sus bases de datos y posteriormente restaurarlas.

Para bases de datos que contengan tablas InnoDB lo más aconsejable es el uso de la herramienta cliente `mysqldump`.

Para bases de datos con tablas exclusivamente MyISAM puede ser una alternativa el uso de la herramienta `mysqlhotcopy` (mas rápida que `mysqldump`). La herramienta `mysqlhotcopy` funciona de forma similar a lo que seria la copia directa usando comandos de copia propios del sistema operativo sobre el que se ejecuta MySQL.

Muy importante para llevar una política eficaz de copias de seguridad seguir los siguientes principios:

- Realizar copias de seguridad regularmente
- Activar los logs de actualización (`binary log` se ve en el punto siguiente de este documento). Los logs de actualización permiten, una vez restaurada la copia de una BD dañada, volver a aplicar las operaciones que se realizaron después de efectuado el ultimo proceso de back up. Es decir permite llevar de nuevo las tablas al punto inmediatamente anterior al momento que se decidió restaurar la copia.
- Usar un esquema para nombrar archivos de copia de seguridad coherente y comprensible y establecer una política de rotación de copias.

11.5.1 mysqldump

Esta herramienta permite hacer la copia de seguridad de una o múltiples bases de datos. Además permite que estas copias de seguridad se puedan restaurar en distintos tipos de gestores de bases de datos, sin la necesidad de que se trate de un gestor de MySQL. Esto lo consigue creando unos ficheros, que contienen todas las sentencias SQL necesarias para poder restaurar la tabla, que incluyen desde la sentencia de creación de la tabla, hasta una sentencia de inserción por cada uno de los registros que forman parte de la misma.

`mysqldump` dispone de una amplia variedad de opciones que nos permitirá realizar la copia de la forma más conveniente para el propósito de la misma.

Para poder restaurar la copia de seguridad, bastará con ejecutar todas las sentencias sql que se encuentran dentro del fichero, bien desde la línea de comandos de mysql, o desde la pantalla de creación de sentencias SQL de cualquier entorno gráfico como puede ser el Mysql Control Center.

Las limitaciones de la restauración dependerán de las opciones que se han especificado a la hora de hacer la copia de seguridad, por ejemplo, si se incluye la opción `--add-drop-table` al hacer la copia de seguridad, se podrán restaurar tablas que existen actualmente en el servidor (borrándolas primero). Por lo que es necesario estudiar

primero los procedimientos que se utilizarán tanto en la copia como en la restauración, para que todo salga correcto.

Las opciones de configuración de *mysqldump* son muchas y vienen detalladas en el manual de referencia de MySQL (8.8 *The mysqldump Database Backup Program*). Algunas de las opciones más interesantes que tiene son:

`--add-locks`

Añade LOCK TABLES antes, y UNLOCK TABLE después de la copia de cada tabla. De esta forma se evita que se produzcan modificaciones en la tabla durante el proceso de copia.

`--add-drop-table`

Añade un drop table antes de cada sentencia create

`-A, --all-databases`

Copia todas las bases de datos. Es lo mismo que utilizar `--databases` seleccionando todas.

`-a, --all`

Incluye todas las opciones de creación específicas de Mysql.

`-B, --databases`

Para copiar varias bases de datos. En este caso, no se especifican tablas. El nombre de los argumentos se refiere a los nombres de las bases de datos. Se incluirá USE db_name en la salida antes de cada base de datos.

`-F, --flush-logs`

Escribe en disco todos los logs antes de comenzar con la copia

`-f, --force,`

Continúa aunque se produzca un error de SQL durante la copia.

`-h, --host=.`

Copia los datos del servidor de Mysql especificado. El servidor por defecto es localhost.

`-d, --no-data`

No incluirá ninguna información sobre los registros de la tabla. Esta opción sirve para crear una copia de sólo la estructura de la base de datos.

`--opt`

Lo mismo que `--quick --add-drop-table --add-locks --extended-insert --lock-tables`. Esta opción le debería permitir realizar la copia de seguridad de la base de datos de la forma más rápida y efectiva.

`-pyour_pass, --password[=your_pass]`

Contraseña utilizada cuando se conecta con el servidor. Si no se especifica, ``=your_pass'`, *mysqldump* preguntará la contraseña.

Ejemplos de llamadas a *mysqldump*:

Se quiere realizar una copia de seguridad completa de todas las bases de datos:

```
mysqldump --opt --all-databases >copia_seg_04_12_04.sql
```

Se quiere realizar una copia de varias bases de datos completas(bases de datos test y prueba).

```
mysqldump --opt --databases test prueba > copia_seg_test_prueba.sql
```

Se quiere realizar una copia de algunas tablas de una base de datos (base de datos test tablas articulo y almacen)

```
mysqldump --opt test articulo almacen > copia_seg_test_art_alm.sql
```

Se quiere realizar la copia se seguridad de la base de datos mibase

```
mysqldump --opt mibase > copia_seg_mibase_04_12_04.sql
```

El aspecto de un fichero copia generado por `mysqldump` es el siguiente:

```
-- MySQL dump 10.7
--
-- Host: localhost      Database: test
--
-- Server version      4.1.2-alpha-nt-max-log

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT,
CHARACTER_SET_CLIENT=utf8 */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE="NO_AUTO_VALUE_ON_ZERO" */;

--
-- Current Database: `test`
--

CREATE DATABASE /*!32312 IF NOT EXISTS*/ `test`;

USE `test`;

--
-- Table structure for table `accesslog`
--

DROP TABLE IF EXISTS `accesslog`;
CREATE TABLE `accesslog` (
  `ip` varchar(15) default NULL,
  `fecha` varchar(40) default NULL,
  `operacion` varchar(5) default NULL,
  `fichero` varchar(100) default NULL,
  `protocolo` varchar(10) default NULL,
  `num1` varchar(5) default NULL,
  `num2` varchar(5) default NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- etc,etc,etc...
--
```

Restaurar una copia hecha con *mysqldump*

La manera más directa de restaurar una copia hecha con *mysqldump* es ejecutando el cliente *mysql* desde la línea de comandos y lanzando directamente el script generado previamente por *mysqldump*.

Ejemplo el usuario *root* va a restaurar la copia generada mediante *mysqldump* el 4/12/2004:

```
C:\mysql\bin> mysql -uroot -pPassword < copia_seg_mibase_04_12_04.sql
```

11.5.2 mysqlhotcopy (solo Linux, Unix)

Esta herramienta que viene de serie en la distribución de MySQL permite crear copias de seguridad de las tablas tipo ISAM y MyISAM mientras el servidor esta en ejecución. Funciona estableciendo un bloqueo de solo lectura (read lock) en las tablas que se van a copiar, se efectúa la copia y se libera el bloqueo.

mysqlhotcopy es un script de perl y por defecto **no esta disponible** en las instalaciones para Windows.

Como ejemplo de uso de *mysqlhotcopy* lanzando desde la línea de comandos la siguiente orden

```
$ mysqlhotcopy -uroot -pPassword prueba /tmp
```

obtenemos una copia de las tablas de la base de datos *prueba*. Se habrá creado una subcarpeta *prueba* debajo de la carpeta *tmp* que podemos comprobar ejecutando

```
$ ls -l /tmp/prueba
```

```
total 108
```

-rw-rw----	1	mysql	users	8550	May	3	12:02	tabla1.frm
-rw-rw----	1	mysql	users	25	May	3	12:02	tabla1.MYD
-rw-rw----	1	mysql	users	2048	May	23	12:58	tabla1.MYI
-rw-rw----	1	mysql	users	8924	Mar	4	21:52	tabla2.frm
-rw-rw----	1	mysql	users	7500	Mar	5	21:11	tabla2.MYD
-rw-rw----	1	mysql	users	5120	May	23	12:58	tabla2.MYI
-rw-rw----	1	mysql	users	8550	May	3	12:02	tabla3.frm
-rw-rw----	1	mysql	users	25	May	3	12:02	tabla3.MYD
-rw-rw----	1	mysql	users	2048	May	23	12:58	tabla3.MYI

Restaurar una copia hecha con *mysqlhotcopy*

Para restaurar una copia hecha con *mysqlhotcopy* simplemente tenemos que restaurar los ficheros a la carpeta del directorio de datos correspondiente de MySQL.

Para restaurar la copia de la base de datos *prueba* realizada anteriormente lanzamos la orden

```
$ cp /tmp/prueba/*.* datadir/prueba
```

11.5.3 Importar tablas desde ficheros de texto delimitados

Prácticamente todos los sistemas de gestión de bases de datos tienen la capacidad de importar y exportar tablas a ficheros de texto (ASCII) de campos delimitados. Por lo tanto esta utilidad es muy aconsejable para pasar datos de un SGBD a otro además de para hacer copias de seguridad de tablas individuales.

Un ejemplo de fichero de texto de campos delimitados podría ser el siguiente:

```
1112345;"Martínez Salas,;Fernando";"PROFESOR";2200.00;10
4123005;"Bueno Zarco,;Elisa";"PROFESOR";2200.00;10
4122025;"Montes García, M.Pilar";"PROFESOR";2200.00;10
1112346;"Rivera Silvestre, Ana";"PROFESOR";2050.00;15
9800990;"Ramos Ruiz, Luis";"PROFESOR";2050.00;15
8660990;"De Lucas Fdez, M.Angel";"PROFESOR";2050.00;15
7650000;"Ruiz Lafuente, Manuel";"PROFESOR";2200.00;22
43526789;"Serrano Laguía, María";"PROFESOR";2050.00;45
4480099;"Ruano Cerezo, Manuel";"ADMINISTRATIVO";1800.00;10
1002345;"Albarrán Serrano, Alicia";"ADMINISTRATIVO";1800.00;15
7002660;"Muñoz Rey, Felicia";"ADMINISTRATIVO";1800.00;15
5502678;"Marín Marín, Pedro";"ADMINISTRATIVO";1800.00;22
6600980;"Peinado Gil, Elena";"CONSERJE";1750.00;22
4163222;"Sarro Molina, Carmen";"CONSERJE";1750.00;45
```

Fichero personal.txt

Estos ficheros se caracterizan por estar almacenados en formato ASCII habitualmente con las extensiones .txt o .csv. Tienen los registros de longitud variable y los campos están delimitados por un carácter especial (en este caso “;”). También habitualmente los campos de caracteres vienen entrecomillados.

MySQL proporciona dos métodos para importar directamente este tipo de ficheros a tablas. Una de ellas es la utilidad *mysqlimport*, el otro método es lanzar la sentencia `LOAD DATA INFILE`. En este apartado se comenta únicamente este último método. La utilidad *mysqlimport* se puede ver en detalle en la sección 8.10 del Manual de MySQL.

Sintaxis genérica de la sentencia `LOAD DATA INFILE` :

```
LOAD DATA [LOCAL] INFILE 'path/nombre_fichero.txt'
[REPLACE | IGNORE]
INTO TABLE nombre_tabla
[FIELDS
  [TERMINATED BY '\t']
  [[OPTIONALLY] ENCLOSED BY '']
  [ESCAPED BY '\\'] ]
]
[LINES
  [STARTING BY '']
  [TERMINATED BY '\n']
]
[IGNORE number LINES]
[(col_name,...)]
```

Se observa en esta sintaxis que la mayoría de las cláusulas de esta sentencia son optativas, vamos a comentar las más interesantes.

Si se especifica la cláusula `LOCAL` significa que el fichero es leído por el programa cliente en el equipo donde se está ejecutando el cliente de MySQL desde el

que se esta lanzando la sentencia `LOAD DATA`. Si la cláusula `LOCAL` se omite significa que el fichero debe encontrarse en el host del servidor MySQL .

Para poder ejecutar esta sentencia es necesario crear previamente la tabla sobre la que van a ir los datos. La tabla puede o no estar vacía. Habrá que tener cuidado al hacer una carga masiva de datos con esta sentencia el evitar entradas duplicadas en campos marcados como `UNIQUE` o `PRIMARY KEY`.

En el siguiente ejemplo se va a crear una tabla y lanzar la sentencia `LOAD DATA` que permita cargar los datos del fichero de personal (personal.txt) anterior.

```
USE TEST;
CREATE TABLE IF NOT EXISTS personal(
    dni            int(10),
    apellidos      varchar(30),
    funcion        enum('PROFESOR','ADMINISTRATIVO','CONSERJE'),
    salario        float(6,2),
    cod_centros    int(5),
    CONSTRAINT pk_codigo PRIMARY KEY (dni));

LOAD DATA INFILE 'C:\\temp\\personal.txt'
INTO TABLE personal
FIELDS TERMINATED BY ';' ENCLOSED BY '"'
LINES TERMINATED BY '\\r\\n';
```

Muy importante es construir correctamente las cláusulas `FIELDS` y `LINES` de la sentencia `LOAD DATA`. En la cláusula `FIELDS` indicamos que el carácter delimitador es el “;” y que las cadenas de texto van entrecomilladas. En la cláusula `LINES` indicamos que el carácter de final de línea es ‘\\r\\n’. (Los ficheros de texto de Windows suelen llevar \\r\\n como carácter de final de línea mientras que los ficheros de texto en Linux es únicamente \\n)

Para poder lanzar una sentencia `LOAD DATA` es necesario ser administrador de la base de datos o al menos tener concedido el privilegio `FILE`.

11.5.4 Exportar tablas a ficheros de texto delimitados

La sentencia que permite exportar tablas a ficheros ASCII delimitados tiene la siguiente sintaxis

```
SELECT columnas INTO OUTFILE 'path/nombre_fichero.txt'
  [FIELDS
    [TERMINATED BY '\t']
    [[OPTIONALLY] ENCLOSED BY '']
    [ESCAPED BY '\\ ' ]
  ]
  [LINES
    [STARTING BY '']
    [TERMINATED BY '\n']
  ]
FROM nombre_tabla;
```

A diferencia de `LOAD DATA...` la sentencia `SELECT...INTO OUTFILE` siempre genera el fichero en el host donde esta ejecutándose el servidor. Si lo que se desea es que el fichero se cree en el equipo del cliente esta sentencia no nos sirve. Podemos usar en su lugar el siguiente comando

```
shell> mysql -e "SELECT ..." > nombre_fichero
```

En el siguiente ejemplo se va a crear un fichero llamado `personal2.txt` a partir de la tabla a tabla `personal` del punto anterior.

```
SELECT * INTO OUTFILE 'C:\\temp\\personal2.txt'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
FROM personal;
```

Al igual que ocurre con `LOAD DATA` para poder lanzar una sentencia `SELECT ... INTO OUTFILE` es necesario ser administrador de la base de datos o al menos tener concedido el privilegio `FILE`.

Tema 12.

Administración de MySQL (II)

Javier Robles Cascallar

12.1. Los ficheros Log

12.1.1 El log de error (error log)

Contiene información indicando cuando se arranca y detiene el servidor y cualquier error crítico que ocurra mientras el servidor se esta ejecutando.

A no ser que se especifique de forma explícita otro nombre distinto en el fichero de configuración (`my.ini` o `my.cnf`) el log de error llevará el nombre del propio host seguido de la extensión `.err` (`nombre_host.err`) y estará ubicado en el directorio de datos de MySQL. En sistemas operativos Windows en lugar de llevar el nombre de host el fichero log de error se llama `mysql.err`

Ejemplo del contenido del fichero log de error.

```
MySql: ready for connections.
Version: '4.1.2-alpha-nt-max' socket: '' port: 3306
040702 7:30:47 Aborted connection 2 to db: 'unconnected' user: 'ODBC'
host: 'localhost' (Got an error reading communication packets)
040702 7:58:14 MySql: Normal shutdown

040702 7:58:14 InnoDB: Starting shutdown...
040702 7:58:15 InnoDB: Shutdown completed; log sequence number 0 43634
040702 7:58:15 MySql: Shutdown Complete

MySql: ready for connections.
Version: '4.1.2-alpha-nt-max' socket: '' port: 3306
040702 8:15:43 Aborted connection 1 to db: 'unconnected' user: 'admin'
host: 'localhost' (Got an error reading communication packets)
040702 8:15:49 MySql: Normal shutdown

040702 8:15:49 InnoDB: Starting shutdown...
040702 8:15:51 InnoDB: Shutdown completed; log sequence number 0 43634
040702 8:15:51 MySql: Shutdown Complete

MySql: ready for connections.
Version: '4.1.2-alpha-nt-max' socket: '' port: 3306
040702 8:41:29 MySql: Normal shutdown

040702 8:41:30 MySql: Forcing close of thread 3 user: 'root'

040702 8:41:30 InnoDB: Starting shutdown...
040702 8:41:33 InnoDB: Shutdown completed; log sequence number 0 43634
040702 8:41:33 MySql: Shutdown Complete
```

12.1.2 El fichero log general (query log file)

Para saber todo lo que ha ocurrido en la base de datos es necesario activar en el fichero de configuración (*my.ini* o *my.cnf*) la opción `log[=nombre_fichero]` si no se le pasa nombre el nombre del fichero log coincide con el nombre del servidor seguido de la extensión `.log` (`nombre_host.log`)

Este fichero log puede crecer muy rápidamente. Es desde el sistema operativo desde donde conviene realizar los siguientes pasos cuando el fichero haya alcanzado un tamaño conveniente o cuando haya pasado un número de días suficiente.

- Parar el servidor MySQL
- Renombrar el fichero log y/o moverlo a una carpeta de respaldo
- Iniciar el servidor. Se crea un fichero log nuevo.

12.1.3 El fichero log de modificaciones (update log)

El fichero update log ha sido sustituido en las versiones actuales de MySQL por el fichero binary log que viene comentado a continuación

12.1.4 El fichero de log binario (binary log).

Este fichero ha reemplazado el antiguo fichero update log de versiones anteriores de MySQL.

El binary log recopila toda la información de transacciones efectuadas en la base de datos (modificaciones , inserciones y borrados).

El principal uso del binary log es la posibilidad de efectuar una recuperación de la base de datos ya que contiene toda la información de las modificaciones en la BD.

El binary log también se usa para replicar servidores de datos entre servidores maestros y servidores esclavos.

Tener activo el servicio de binary log supone una pérdida de rendimiento en velocidad del servidor en torno a un 1%. Sin embargo los beneficios proporcionados por la posibilidad de recuperar la información ante un posible fallo o caída del servidor compensan la pérdida de rendimiento.

Activación del binary log

Parar la base de datos.

Editar el fichero *my.ini* o *my.cnf* y añadir la siguiente línea en la sección `[mysqld]` del fichero

```
#Habilitar el binary log
log-bin
```

Arrancar la base de datos.

A partir de la activación se crean por defecto en el directorio de datos de MySQL el fichero binary log que tendrá como nombre el nombre del host seguido de `-bin` junto con una extensión formada por una secuencia de números. Cada vez que se reinicia el servidor se genera un nuevo fichero de log con la secuencia numérica incrementada. También se crea un fichero índice de logs con el mismo nombre que cada uno de los ficheros binary log y con la extensión `.index`. Por ejemplo:

```
26/11/2004 09:27          275 servidorXPASI2-bin.000001
26/11/2004 09:27          56 servidorXPASI2-bin.index
26/11/2004 09:27          170 servidorXPASI2-bin.000002
```

Para examinar los ficheros binary log se puede utilizar la herramienta *mysqlbinlog* que se encuentra en la carpeta `/bin` de la distribución de MySQL.

Ejemplo de uso de *mysqlbinlog* :

Se quiere visualizar las modificaciones realizadas en la base de datos y almacenadas en el fichero de log de nombre `servidorXPASI2-bin.000001`

```
E:\mysql\bin>mysqlbinlog ..\data\servidorXPASI2-bin.000001
```

La salida generada por el programa *mysqlbinlog* es la siguiente:

```
/*!40019 SET @@session.max_insert_delayed_threads=0*/;
# at 4
#041126 9:17:35 server id 1  end_log_pos 95      Start: binlog v 4, server v 5.0.
1-alpha-nt-log created 041126 9:17:35 at startup
# at 95
#041126 9:18:55 server id 1  end_log_pos 173    Query    thread_id=1      exec_tim
e=0      error_code=0
use test;
SET TIMESTAMP=1101457135;
SET @@session.foreign_key_checks=67108864, @@session.sql_auto_is_null=16384, @@s
ession.unique_checks=134217728;
SET @@session.sql_mode=0;
delete from accesslog;
# at 173
#041126 9:19:45 server id 1  end_log_pos 275    Query    thread_id=1      exec_tim
e=0      error_code=0
SET TIMESTAMP=1101457185;
delete from telefonos4 where numtel=910201700;
```

Ejecutar una parte del fichero binary log.

En ocasiones y después de restaurar una copia de seguridad habrá que ejecutar de nuevo parte de un fichero binary log para dejar la base de datos tal como se encontraba en el momento de producirse el fallo o el error que motivó el proceso de restauración de la copia de seguridad. Antes de ejecutar el fichero convendrá analizar si interesa ejecutar todo o solo una parte. Para ejecutar un fichero binary log a partir de una determinada posición del fichero hasta otra determinada posición del fichero se utilizan las opciones `--start-position` y `--stop-position`. Lo que se hace es que la salida generada por *mysqlbinlog* sea la entrada del cliente *mysql*. Para ello empleamos el operador `|` desde la línea de comandos como en el siguiente ejemplo:

```
C:\mysql\bin>mysqlbinlog --start-position=95      --stop-position=173
..\data\servidorXPASI2-bin.000001 | mysql
```

Eliminar ficheros binary log.

Una forma de eliminar ficheros binary log es usando la sentencia PURGE MASTER LOGS (o PURGE BINARY LOGS). Se puede hacer de dos formas eliminar por nombre de fichero o por fecha. Ejemplo de uso por nombre de fichero

Se ejecuta el comando show master logs para obtener una lista de los ficheros binary log almacenados.

```
mysql> show master logs;
+-----+
| Log_name |
+-----+
| servidorXPASI2-bin.000001 |
| servidorXPASI2-bin.000002 |
+-----+
2 rows in set (0.00 sec)
```

Se lanza la sentencia que elimina todos los ficheros log anteriores al fichero log 'servidorXPASI2-bin.000002'

```
mysql> purge binary logs to 'servidorXPASI2-bin.000002';
Query OK, 0 rows affected (0.03 sec)
```

Se vuelve a ejecutar el comando show master logs para ver el resultado

```
mysql> show master logs;
+-----+
| Log_name |
+-----+
| servidorXPASI2-bin.000002 |
+-----+
1 row in set (0.00 sec)
```

Para eliminar por fecha es muy similar a lo anterior cambiando la sentencia por la siguiente por ejemplo

```
PURGE MASTER LOGS BEFORE '2004-11-22 22:46:26';
```

Nota: Por defecto cualquier modificación en cualquier base de datos queda registradas en el binary log. A veces puede ser interesante solamente registrar modificaciones en determinadas bases de datos o ignorar modificación en otras, para ello se utilizan las opciones de configuración

```
binlog-do-db=nombre_bd
binlog-ignore-db=nombre_bd
```

Estas entradas se ponen en el fichero *my.ini* una vez habilitado el binary log

12.2. Ejecutar varios servidores MySQL en la misma máquina

En ocasiones puede ser que necesitemos ejecutar varios procesos `mysqld` en la misma máquina, por ejemplo si queremos testear una nueva release de MySQL y no queremos detener nuestro servidor de MySQL que está en funcionamiento o por que se quiere dar a diferentes grupos de usuarios acceso a servidores diferentes de MySQL para que los administren ellos mismos y no disponemos de más equipos.

12.2.1 Ejecutar varios servidores MySQL en Windows

En Windows tenemos la posibilidad de arrancar los servidores manualmente desde la línea de comandos o bien instalarlos como servicios de Windows. En este apartado vamos a comentar solo la primera opción, es decir, arrancando los servidores manualmente desde la línea de comandos. Normalmente arrancar varios servidores en un mismo equipo suele ser algo provisional por eso no es necesario instalarlo como un servicio de Windows. De todas formas cualquier combinación es valida podemos tener el servidor de MySQL de producción instalado como servicio y arrancar manualmente la nueva release que queremos testear.

Lo fundamental para que el proceso funcione y los servidores no se interfieran entre si es que cada servidor arranque leyendo un fichero de configuración distinto (*my.ini*) esto se consigue fácilmente si en el arranque manual se ejecutan los procesos de servidor con la opción `--defaults-file` apuntando cada proceso servidor a su propio fichero *my.ini*.

En el ejemplo que detallamos a continuación vamos a ejecutar dos procesos servidores uno va a escuchar las peticiones por el puerto 3307 y el por el puerto 3308 (es básico que los puertos sean distintos). Además necesitamos tener dos directorios de datos cada uno de ellos con, al menos, su correspondiente base de datos `mysql` con las tablas del diccionario de datos. (Se puede copiar una base de datos `mysql` de una ubicación a otra simplemente copiando la carpeta `mysql` que se crea tras el proceso de instalación del software MySQL bajo la carpeta `data`). Los directorios de datos van a ser por ejemplo `c:\mysql1\misdatos1` y `c:\mysql2\misdatos2` respectivamente. (Es conveniente insistir en este punto que deben existir al menos las subcarpetas `c:\mysql1\misdatos1\mysql` y `c:\mysql2\misdatos2\mysql` cada una con sus correspondientes tablas del diccionario de datos)

Se crean los dos ficheros de configuración. Uno de ellos será por ejemplo `c:\mysql1\my1.ini` con la siguiente información:

```
[mysqld]
datadir = c:\mysql1\misdatos1
port = 3307
```

El segundo lo llamamos `c:\mysql2\my2.ini` con la siguiente información:

```
[mysqld]
datadir = c:\mysql2\misdatos2
port = 3308
```

Importante. Para arrancar los servidores necesitamos una consola de comandos para cada servidor, ya que no vuelve a aparecer el prompt una vez lanzado el proceso.

Abrimos una ventana de comandos y lanzamos el comando:

```
C:\> C:\mysql\bin\mysqld --defaults-file= c:\mysql1\my1.ini
```

Abrimos una segunda ventana para al siguiente orden:

```
C:\> C:\mysql\bin\mysqld-max --defaults-file= c:\mysql2\my2.ini
```

Para detener los procesos servidor necesitamos abrir una tercera ventana y ejecutar `mysqladmin` de la siguiente forma:

```
C:\> C:\mysql\bin\mysqladmin --port=3307 shutdown
C:\> C:\mysql\bin\mysqladmin --port=3308 shutdown
```

Para ejecutar el cliente y acceder a las distintas bases de datos tendremos siempre que tener en cuenta el número de puerto.

Para comprobar que funcionan los dos servidores ejecutamos dos clientes uno accediendo a cada servidor desde dos ventanas de comandos distintas. Suponemos que estamos conectando desde el mismo equipo (modo localhost) como administradores sin password:

```
C:\> C:\mysql\bin\mysql --port=3307
C:\> C:\mysql\bin\mysql --port=3308
```

12.2.2 Ejecutar varios servidores MySQL en Linux

La manera más sencilla en Linux (u otro servidor UNIX) es compilar los servidores con un diferentes puerto TCP , con diferentes ficheros de socket y también con diferentes directorios base lo que implica que tendrán diferentes directorios de datos.

Para simplificar suponemos que ya tenemos instalado un servidor con la configuración que viene por defecto en la instalación (puerto 3306, fichero de socket en la dirección `/tmp/mysql.sock`).

Para configurar un nuevo servidor usamos el comando `configure` de esta manera:

```
shell> ./configure --with-tcp-port=port_number \
               --with-unix-socket-path=file_name \
               --prefix=/usr/local/mysql-5.0.2
```

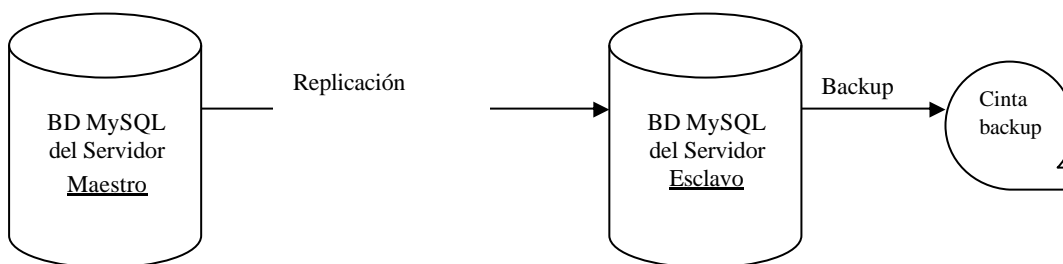
donde *port_number* y *file_name* deben corresponder a valores diferentes de los ya instalados por defecto y la dirección asociada a `--prefix` no debe ser la misma de la que tiene el servidor ya instalado..

12.3. Replicación

En las primeras versiones de MySQL el proceso de replicación era una tarea de implantación compleja y que tenía sus dificultades por que el proceso estaba defectuosamente documentado.

A partir de la versión 4.1 el proceso de replicación se ha simplificado considerablemente.

La forma más básica de replicación de bases de datos consta de dos servidores un *servidor maestro* y un *servidor esclavo*. El servidor maestro registra todas las modificaciones de la base de datos en su fichero de log binario o *binary log* (ver apartado), el servidor esclavo se conecta con el servidor maestro accede al fichero binary log y ejecuta las sentencias de modificación que en el están registradas en su base de datos local. De esta manera se consigue que ambas bases de datos (maestro y esclavo) sean coincidentes y estén sincronizadas.



12.3.1 Ventajas de la replicación

Facilita el concepto de distribución de la información. Permite mantener copias idénticas de la base de datos en sitios que pueden estar muy alejados geográficamente y no requieren estar permanentemente conectados.

Permitir el balanceo de carga. La replicación es una excelente manera de implementar un balanceo de carga efectivo entre los servidores. Teniendo uno o más servidores esclavos podemos distribuir las peticiones de consulta entre todos ellos evitando la sobrecarga en un único servidor.

Copias de seguridad y recuperación de la información. Resulta más sencillo implantar un sistema de copia de seguridad sobre un servidor esclavo que sobre un servidor maestro con gran demanda de solicitud de información por parte de los clientes. Además los clientes nunca se ven penalizados por que no es necesario detener el servicio para generar el proceso de copia.

Aumento de la fiabilidad ante caídas. Un servidor esclavo siempre puede convertirse en maestro en caso de que el servidor maestro sufra una caída por algún motivo.

12.3.2 Implantación del servicio de replicación

En este apartado se pretende explicar el método de implantación de un servicio de replicación en dos posibles escenarios. En primer lugar se planteará la replicación en un servidor de base de datos recién instalado y que por lo tanto aun no contiene datos propios a excepción de la base de datos *test* y la base de datos *mysql*. A continuación se comentará como instalar el servicio de replicación en un servidor que ya ha estado un tiempo funcionando para que cause el menor perjuicio por interrupciones en los usuarios.

12.3.2.1 Replicación en un nuevo servidor.

Los pasos necesarios para llevar a cabo el proceso en el servidor maestro y esclavo (*master and slave*) son los siguientes:

1. Crear una cuenta de usuario encargado de la replicación en el maestro.
2. Añadir las modificaciones necesarios en los ficheros de configuración del maestro y el esclavo (*my.ini* o *my.cnf*).
3. Reiniciar el servidor maestro y verificar la creación del fichero de log binario (*binary log*)
4. Reiniciar el servidor esclavo y comprobar que la replicación funciona correctamente.

Paso 1. Creación de la cuenta de usuario en el servidor maestro.

El usuario se debe crear en el servidor maestro y será el encargado de ejecutar el servicio. El usuario nuevo debe ser creado con los privilegios mínimos para poder realizar el proceso de replicación conectándose desde el usuario esclavo. Estos privilegios son `REPLICATION SLAVE` y `REPLICATION CLIENT`. Llamamos al usuario nuevo *repl* con password *repl*. La dirección IP del servidor esclavo será por ejemplo 192.168.1.2 y la del servidor maestro 192.168.1.1.

```
mysql> GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO
repl@"192.168.1.0/255.255.255.0" IDENTIFIED BY 'repl';
Query OK, 0 rows affected (0.00 sec)
```

Después de crear la cuenta verificamos que se ha creado correctamente:

```
mysql> SHOW GRANTS FOR repl;
+-----+
| Grants for repl@"192.168.1.0/255.255.255.0" |
+-----+
| GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'repl'@'...' IDENTIFIED BY ... |
+-----+
```

Paso 2. Modificación de los ficheros de configuración en el maestro y el esclavo.

Cada servidor debe ser identificado con un entero que le sirve de identificador. Esto es necesario por las distintas combinaciones de maestros esclavos que pueden darse. En

este caso que estamos estudiando se asignará al maestro el identificador 1 y al esclavo el identificador 2.

Por lo tanto en el fichero `my.ini` del maestro en la sección `[mysqld]` se añadirán las dos líneas siguientes.

```
log-bin  
  
server-id = 1
```

La opción `log-bin` activa el *binary log* en el maestro (fundamental para que la replicación tenga lugar).

En el fichero `my.ini` del esclavo es necesario añadir más información. La identificación del esclavo, la dirección del maestro, el usuario y password del maestro encargado de efectuar la replicación y el puerto. Las líneas son por lo tanto las siguientes:

```
server-id = 2  
  
master-host = master.example.com  
  
master-user = repl  
  
master-password = c0pyIT!  
  
master-port = 3306
```

Paso 3. Reiniciar el maestro.

Una vez modificado los ficheros de configuración podemos reiniciar el maestro y verificar una vez arrancado el servidor que se han activado los ficheros `binary log`. Basta con echar un vistazo a la carpeta de datos de MySQL (*datadir*) y ver si aparecen dos ficheros con la siguiente apariencia:

```
Nombre-host-bin.000001  
Nombre-host-bin.index
```

Paso 4. Reiniciar el esclavo.

Una vez reiniciado el maestro basta con reiniciar el servidor esclavo para que la replicación entre en funcionamiento. Para comprobar que el servicio de replicación está en marcha conviene echar un vistazo al fichero log de error (`nombre-host.err`) que debe tener al final la siguiente información:

```
021103 17:58:10 Slave I/O thread: connected to master  
'repl@192.168.1.1:3306',  
  
replication started in log 'nombre-host-bin.000001' at position 4
```

También podemos verificar el correcto funcionamiento de la replicación haciendo cualquier modificación en la base de datos del servidor maestro y ver como queda replicada en el servidor esclavo.

12.3.2.2 Replicación en un servidor existente.

Este es el caso más frecuente y el que añade un poco más de trabajo que el caso anterior sobre servidores nuevos por que requiere hacer una copia completa de los datos que se quieran mantener replicados del servidor maestro en el servidor esclavo en el momento justo antes de poner en marcha la replicación (si la copia se realiza on-line se denomina *snapshot*).

12.3.3 Otros datos útiles en el proceso de replicación.

En el servidor esclavo podemos parar y arrancar el proceso de replicación sin necesidad de reiniciar el servidor de base de datos. Basta con usar estas sentencias;

```
mysql> stop slave; (para parar el proceso de replicación).
mysql> start slave; (para reiniciarlo).
```

La siguiente sentencia permite ver desde el esclavo el estado del proceso de replicación (la opción \G permite una línea por columna y facilita la visualización en pantalla)

```
mysql> SHOW SLAVE STATUS \G

***** 1. row *****
      Slave_IO_State: Waiting for master to send event
      Master_Host: 192.168.1.1
      Master_User: repl
      Master_Port: 3306
      Connect_retry: 60
      Master_Log_File: nombre-host-master-log.004
      Read_Master_Log_Pos: 635904807
      Relay_Log_File: nombre-host-slave-log.004
      Relay_Log_Pos: 846096118
      Relay_Master_Log_File: binary-log.004
      Slave_IO_Running: Yes
      Slave_SQL_Running: Yes
      Replicate_do_db:
      Replicate_ignore_db:
      Last_errno: 0
      Last_error:
      Skip_counter: 0
      Exec_master_log_pos: 635904807
      Relay_log_space: 846096122
1 row in set (0.00 sec)
```

El parámetro `Slave_IO_State` permite ver rápidamente si el proceso de replicación esta en marcha y funcionando. En el caso de que este parado e servicio de replicación la salida `SHOW SLAVE STATUS \G` queda

```
***** 1. row *****
      Slave_IO_State:
      Master_Host: 192.168.1.1
      Master_User: repl
      Master_Port: 3306
      etc.,etc....
```

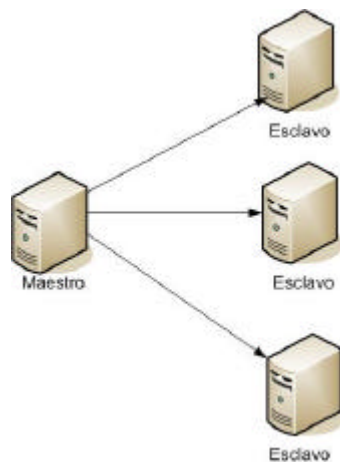

12.3.4 Posibles arquitecturas de replicación.

Una vez visto que establecer un servicio de replicación en MySQL es relativamente sencillo, se pueden combinar los servicios de replicación en múltiples arquitecturas para con ello dar solución a muy diversos tipos de problemas o mejoras en el rendimiento que se puedan plantear.

Es conveniente que todas las arquitecturas de replicación sigan las siguientes reglas básicas:

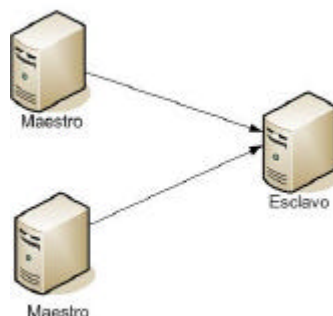
1. Cada esclavo debe tener un identificador único.
2. Un esclavo solo debe tener un maestro.
3. Un maestro puede tener varios esclavos.
4. Esclavos pueden ser a su vez maestros de otros esclavos.

Tipo 1. Un maestro y varios esclavos.



Esta situación es típica de una BD que tiene pocas sentencias de modificación pero si muchas sentencias de consulta. El trabajo de servir las sentencias de consulta se puede distribuir entre los esclavos estableciendo un balanceo de carga. De hecho los servidores esclavos pueden estar ejecutando a su vez otros servicios como Apache.

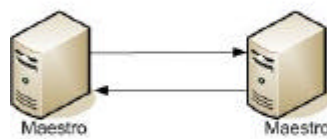
Tipo 2. Un esclavo con dos maestros.



En ocasiones puede ser interesante tener un solo host que haga de esclavo simultáneamente de varios maestros cuyos datos no guarden relación entre si. La principal ventaja de este tipo de arquitectura es el ahorro de costes por que se utiliza

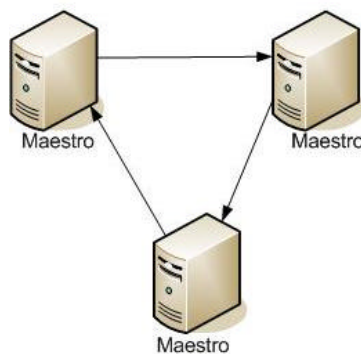
una sola máquina para respaldo de varios servidores . Esta arquitectura aparentemente viola una de las reglas básicas comentadas anteriormente que dice que un servidor esclavo solo debe tener a lo sumo un maestro asociado. Para evitar esta violación de la regla lo que se hace es crear sobre la misma máquina dos instancias de MySQL distintas trabajando sobre distintos puertos TCP. Cada instancia (servicio) de MySQL es responsable de replicar un maestro distinto. No existe ningún impedimento para que un único host pueda ejecutar simultáneamente varias instancias de MySQL siempre que tenga suficiente espacio en disco y potencia de CPU.

Tipo 3. Configuración dual con dos maestros.



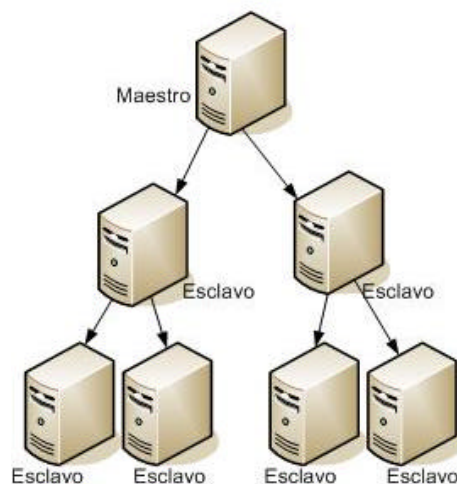
Esta configuración es útil cuando, por ejemplo, existe una organización dividida en dos partes separadas geográficamente por una gran distancia (conexión WAN) y ambas partes de la organización necesitan acceso de lectura/escritura a los mismos datos. Con este tipo de arquitectura ninguna de las partes de la organización se quedan sin acceso a los datos cuando haya problemas en la línea WAN de conexión, y una vez reestablecida, los equipos se ponen al corriente de sus respectivos cambios entre ellos

Tipo 4. Anillo de maestros.



Realmente el caso anterior (configuración dual con dos maestros) es un caso particular de la arquitectura de anillo de maestros donde hay al menos tres o más equipos maestros formando un anillo en el que cada equipo es esclavo de uno de sus vecinos y maestro del otro. La ventaja de esta configuración es la misma que en el caso anterior de configuración dual de cercanía geográfica de los datos que evita las incomodidades de la pérdida de conexión WAN. La desventaja es que estas arquitecturas son frágiles. Si un equipo falla por alguna razón el anillo se rompe y el servicio no queda restablecido hasta que todos los nodos no estén operativos.

Tipo 5. Piramidal



Otro tipo de organización puede requerir otro tipo de arquitectura. Por ejemplo, una gran base de datos centralizada que se replica a cada una de las pequeñas oficinas geográficamente dispersas donde se efectúan consultas de datos mientras que la carga transaccional recae en la base de datos de la oficina central. En este caso lo que parece más manejable es un solo equipo maestro inicial que se va replicando sucesivamente en forma piramidal a los distintos equipos en cada una de las sedes de la organización.