

Manual de DBA

Autores

De Dios De La Cruz Reynaldo Bernard

Juárez Jiménez Efraín

López Jiménez Alejandro

Morales Denis José Antonio

Valencia Ortiz Kairo Iván

Introducción

El objetivo de este manual es mostrar el uso del programa cliente MySQL para crear y usar una sencilla base de datos. MySQL (algunas veces referido como "monitor MySQL ") es un programa interactivo que permite conectarse a un servidor MySQL, ejecutar algunas consultas, y ver los resultados. MySQL puede ser usado también en modo batch: es decir, se pueden colocar toda una serie de consultas en un archivo, y posteriormente decirle a MySQL que ejecute dichas consultas.

Este manual asume que MySQL está instalado en alguna máquina y que se dispone de un servidor MySQL al cual conectarse. Si este no es el caso, se recomienda que contacte con el administrador MySQL. (Si usted es el administrador, es necesario consultar la documentación de MySQL que se refiere a la [instalación y configuración del servidor MySQL](#)). Para ver la lista de opciones proporcionadas por MySQL, se invoca con la opción `--help`:

```
shell> mysql --help
```

A continuación se describe el proceso completo de creación y uso de una base de datos en MySQL.

Si se está interesado sólo en el acceso y uso de una base de datos existente, se pueden omitir las secciones que describen como crear la base de datos y las tablas correspondientes.

Puesto que es imposible que se describan a detalle muchos de los tópicos cubiertos en este artículo, se recomienda que se consulte el [manual de MySQL](#) para obtener más información al respecto.

Tabla de contenido

1 ¿Qué es un DBA?	4
1.1 Habilidades	4
1.2 Deberes	5
1.2.1 Recuperabilidad	5
1.2.2 Integridad	6
1.2.3 Seguridad	6
1.2.4 Disponibilidad	7
1.2.5 Rendimiento	7
1.2.6 Desarrollo/Soporte a pruebas	7
1.3 Funciones	7
2 Instalando MySQL	9
3 Conectándose y desconectándose al servidor MySQL	18
4 Ejecutando algunas consultas	19
5. Creando y usando una base de datos	23
6 Creando una tabla	26
7 Cargando datos en una tabla	29
8 Recuperando información de una tabla	31
8.1 Seleccionando todos los datos	31
8.2 Seleccionando registros particulares	32
8.3 Seleccionando columnas particulares	34
8.4 Ordenando registros	36
Recomendaciones	38
Bibliografía	39

1 ¿Qué es un DBA?

Un administrador de bases de datos (o DBA) tiene la responsabilidad de mantener y operar las bases de datos que conforman el sistema de información de una compañía.

1.1 Habilidades

Debido a la importancia de los datos que están a su cargo, el administrador de bases de datos debe ser experto en TI (tecnología de la información), teniendo particular conocimiento de DBMS (sistemas de administración de bases de datos) y el lenguaje de consulta SQL. También debe tener conocimiento de varios tipos de lenguaje de programación para poder automatizar ciertas tareas.

Una de sus tareas es la de asegurar la integridad del sistema de información de la compañía. Además, es necesario que posea un buen entendimiento de DBMS para optimizar las consultas, ajustar la configuración de DBMS o para sincronizar en forma precisa las herramientas de control del acceso a las bases de datos.

Es posible que el administrador de bases de datos tenga que brindar asistencia técnica a usuarios de las aplicaciones cliente o equipos de desarrollo para solucionar problemas, dar consejos o ayudar a resolver consultas complicadas.

Al trabajar con el jefe de seguridad, el administrador de bases de datos debe crear copias de seguridad, planes y procedimientos de restauración para preservar los datos de los cuales es responsable.

Además de estas habilidades técnicas, el administrador de bases de datos debe poseer un buen entendimiento de las aplicaciones de la compañía y estar dispuesto a atender las necesidades de los usuarios cuando desarrolla o edita una base de datos. En el mejor de los casos, debe tener experiencia en diseño de sistemas de información y modelos UML (Lenguaje unificado de modelos).

El administrador de base de datos (DBA) es la persona responsable de los aspectos ambientales de una base de datos. En general esto incluye:

- **Recuperabilidad** - Crear y probar Respaldos.
- **Integridad** - Verificar ó ayudar a la verificación en la integridad de datos.
- **Seguridad** - Definir y/o implementar controles de acceso a los datos.
- **Disponibilidad** - Asegurarse del mayor tiempo de encendido.
- **Desempeño** - Asegurarse del máximo desempeño incluso con las limitaciones

-
- **Desarrollo y soporte a pruebas** - Ayudar a los programadores e ingenieros a utilizar eficientemente la base de datos.

El diseño lógico y físico de las bases de datos a pesar de no ser obligaciones de un administrador de bases de datos, es a veces parte del trabajo. Esas funciones por lo general están asignadas a los analistas de bases de datos ó a los diseñadores de bases de datos.

1.2 Deberes

Los deberes de un administrador de bases de datos dependen de la descripción del puesto, corporación y políticas de Tecnologías de Información (TI). Por lo general se incluye recuperación de desastres (respaldos y pruebas de respaldos), análisis de desempeño y optimización, y algo de asistencia en el diseño de la base de datos.

Antes de continuar, necesitamos describir brevemente lo que es una "base de datos." Una base de datos es una colección de información, accedida y administrada por un DBMS. Después de experimentar con DBMSs jerárquicos y de red durante los 70's, la industria de IT se vio dominada por DBMSs tales como Oracle y MySQL.

Un DBMS relacional manipula la información a manera de tipos de cosas del mundo real (entidades) en tablas que representan esas entidades. Una tabla es como una hoja de cálculo; cada renglón representa una entidad en particular (instancia), y cada columna representa la información respecto de la entidad (dominio). En ocasiones las entidades están hechas de entidades más pequeñas, como órdenes y líneas de orden.

Una base de datos relacional bien manejada, minimiza la necesidad de las aplicaciones de contener información respecto al almacenamiento físico de los datos que se van a acceder. Para maximizar el aislamiento de los programas de las estructuras de datos, los DBMSs restringen el acceso a los datos mediante el protocolo SQL, un lenguaje no procedural que limita al programador a obtener ciertos resultados.

1.2.1 Recuperabilidad

La Recuperabilidad significa que, si se da algún error en los datos, hay un bug de programa ó de hardware, el DBA (Administrador de base de datos) puede traer de vuelta la base de datos al tiempo y estado en que se encontraba en estado consistente antes de que el daño se causara. Las actividades de recuperación incluyen el hacer respaldos de la base de datos y almacenar esos respaldos de manera que se minimice el riesgo de daño ó pérdida de los mismos, tales como hacer diversas copias en medios de almacenamiento removibles y almacenarlos fuera del área en antelación a un desastre anticipado. La recuperación es una de las tareas más importantes de los DBA's.

La recuperabilidad, frecuentemente denominada "recuperación de desastres", tiene dos formas primarias. La primera son los respaldos y después las pruebas de recuperación. La recuperación de las bases de datos consiste en información y estampas de tiempo junto con bitácoras los cuales se cambian de manera tal que sean consistentes en un momento y fecha en particular. Es posible hacer respaldos de la base de datos que no incluyan las estampas de tiempo y las bitácoras, la diferencia reside en que el DBA debe sacar de línea la base de datos en caso de llevar a cabo una recuperación. Las pruebas de recuperación consisten en la restauración de los datos, después se aplican las bitácoras a esos datos para restaurar la base de datos y llevarla a un estado consistente en un tiempo y momento determinados. Alternativamente se puede restaurar una base de datos que se encuentra fuera de línea sustituyendo con una copia de la base de datos.

Si el DBA (o el administrador) intentan implementar un plan de recuperación de bases de datos sin pruebas de recuperación, no existe la certeza de que los respaldos sean del todo válidos. En la práctica, los respaldos de la mayoría de los RDBMSs son raramente válidos si no se hacen pruebas exhaustivas que aseguren que no ha habido errores humanos ó bugs que pudieran haber corrompido los respaldos.

1.2.2 Integridad

La integridad de una base de datos significa que, la base de datos ó los programas que generaron su contenido, incorporen métodos que aseguren que el contenido de los datos del sistema no se rompa así como las reglas del negocio. Por ejemplo, un distribuidor puede tener una regla la cual permita que solo los clientes individuales puedan solicitar órdenes; a su vez cada orden identifique a uno y solo un proveedor. El servidor Oracle y otros DBMSs relacionales hacen cumplir este tipo de reglas del negocio con limitantes, las cuales pueden ser configuradas implícitamente a través de consultas. Para continuar con este ejemplo, en el proceso de inserción de una nueva orden a la base de datos, esta a su vez tendría que cerciorarse de que el cliente identificado existe en su tabla para que la orden pueda darse.

1.2.3 Seguridad

Seguridad significa la capacidad de los usuarios para acceder y cambiar los datos de acuerdo a las políticas del negocio, así como, las decisiones de los encargados. Al igual que otros metadatos, una DBMS relacional maneja la seguridad en forma de tablas. Estas tablas son las "llaves del reino" por lo cual se deben proteger de posibles intrusos.

1.2.4 Disponibilidad

La disponibilidad significa que los usuarios autorizados tengan acceso a los datos cuando lo necesiten para atender a las necesidades del negocio. De manera incremental los negocios han ido requiriendo que su información esté disponible todo el tiempo (7x24", o siete días a la semana, 24 horas del día). La industria de TI ha respondido a estas necesidades con redundancia de red y hardware para incrementar las capacidades administrativas en línea.

1.2.5 Rendimiento

El rendimiento significa que la base de datos no cause tiempos de respuesta poco razonables. En sistemas muy complejos cliente/servidor y de tres capas, la base de datos es solo uno de los elementos que determinan la experiencia de los usuarios en línea y los programas desatendidos. El rendimiento es una de las mayores motivaciones de los DBA para coordinarse con los especialistas de otras áreas del sistema fuera de las líneas burocráticas tradicionales.

1.2.6 Desarrollo/Soporte a pruebas

Uno de los deberes menos respetados por el administrador de base de datos es el desarrollo y soporte a pruebas, mientras que algunos otros encargados lo consideran como la responsabilidad más importante de un DBA. Las actividades de soporte incluyen la colecta de datos de producción para llevar a cabo pruebas con ellos; consultar a los programadores respecto al desempeño; y hacer cambios a los diseños de tablas de manera que se puedan proporcionar nuevos tipos de almacenamientos para las funciones de los programas.

1.3 Funciones

Las funciones del DBA incluyen:

- **Definición de esquema:** Al compilar las sentencias DDL resultan tablas que son almacenadas permanentemente en el diccionario de datos.
- **Definición de la estructura de almacenamiento y del método de acceso:** Estructuras de almacenamiento y métodos de acceso adecuados se crean escribiendo un conjunto de definiciones que son traducidas por el compilador del lenguaje de almacenamiento y definición de datos.
- **Modificación del esquema y de la organización física:** Las modificaciones, tanto al esquema de la BDD como a la descripción de la organización física de almacenamiento, se logran escribiendo un conjunto de definiciones que son

usadas bien por el compilador de DDL o bien por el compilador del lenguaje de definición de datos.

- **Concesión de autorización para el acceso a los datos:** Esto para regular qué partes de la BDD van a poder ser accedidas por varios usuarios. Especificación de las restricciones de integridad: las restricciones se mantienen en una estructura especial del sistema que consulta el gestor de la BDD cada vez que tiene lugar una actualización en el sistema.

2 Instalando MySQL

Para el año 2016 se tiene un paquete que contiene los siguientes programas y herramientas.

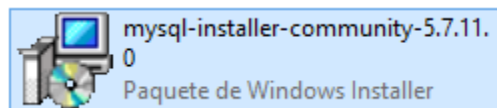
MySQL Installer

MySQL Installer provides an easy to use, wizard-based installation experience for all your MySQL software needs. Included in the product are the latest versions of:

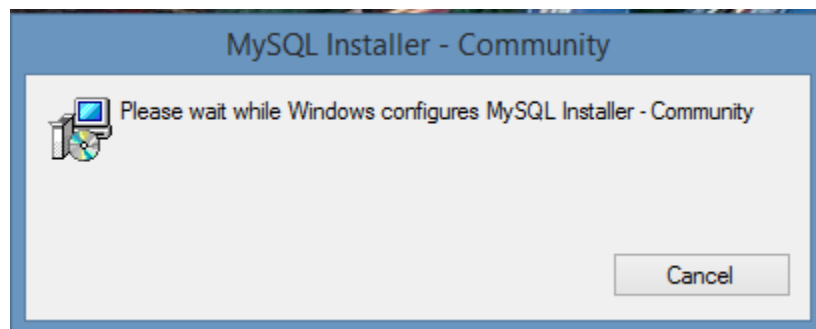
- MySQL Server
- MySQL Connectors
- MySQL Workbench and sample models
- Sample Databases
- MySQL for Excel
- MySQL Notifier
- MySQL for Visual Studio
- Documentation

Por ahora interesa solamente instalar MySQL Server, algunos ejemplos de base de datos y la documentación de MySQL oficial. Para posteriormente conectarse y ejecutar consultas.

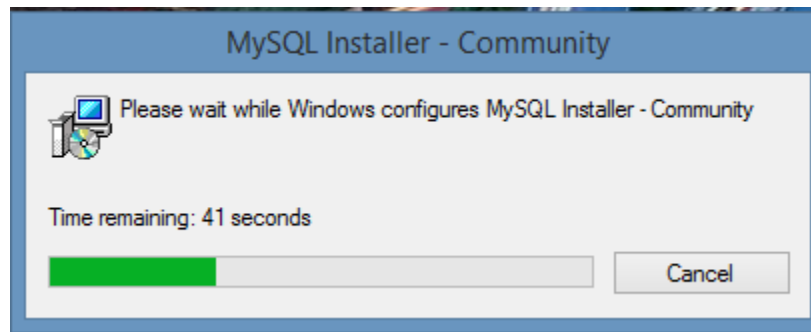
Puede descargar el instalador (paquete ejecutable para su sistema operativo) desde la [página oficial de descarga](#). Una vez descargado el archivo se debe pasar a ejecutar, el nombre del archivo descargado puede ser parecido a este.



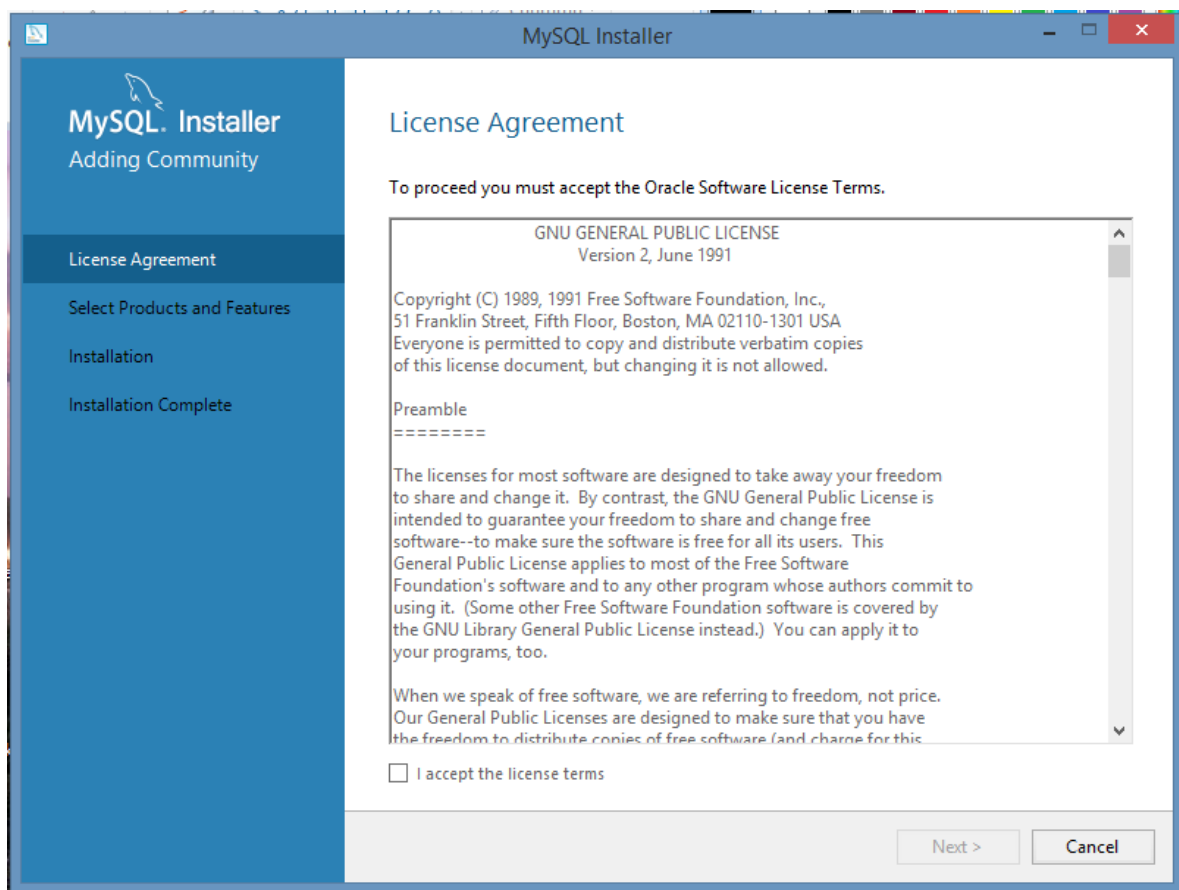
Cuando se ejecute aparecerá la siguiente ventana. Hay que esperar hasta que nos envíe un mensaje preguntando si deseamos darle permiso para ejecutarlo, clic en SI.



Posteriormente aparecerá la ventana de la siguiente forma. Y volverá a preguntar si se desea dar permisos para ejecutar el programa, clic en SI.



Después de unos momentos aparecerá la siguiente ventana.



Dar clic en I accept the license terms, seguido de clic en Next.

La ventana que se mostrará a continuación permitirá seleccionar las características que se deseen instalar, por tanto se elige la opción Custom y clic en Next, como se muestra en la siguiente imagen.

Choosing a Setup Type

Please select the Setup Type that suits your use case.

- ☐ **Developer Default**
Installs all products needed for MySQL development purposes.
- ☐ **Server only**
Installs only the MySQL Server product.
- ☐ **Client only**
Installs only the MySQL Client products, without a server.
- ☐ **Full**
Installs all included MySQL products and features.
- ☒ **Custom**
Manually select the products that should be installed on the system.

Setup Type Description

Allows you to select exactly which products you would like to install. This also allows to pick other server versions and architectures (depending on your OS).

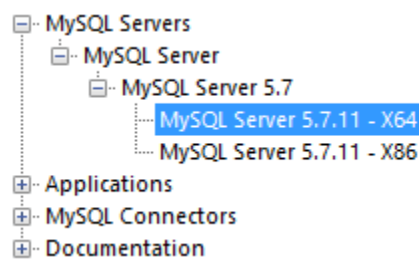
< Back

Next >

Cancel

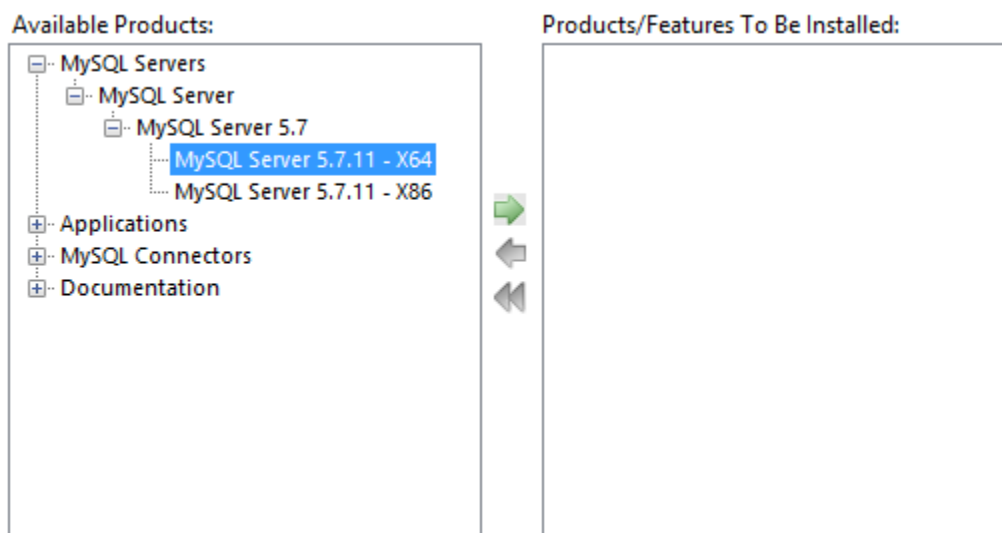
Por ahora solo interesa instalar el sistema gestor de base de datos MySQL por lo que si usted desea instalar algún otro elemento puede elegirlo de la misma manera descrita a continuación.

Cada categoría posee programas o herramientas que podrían ser útil para usted, solo se mencionará como instalar MySQL Server, para ello presione clic en los iconos de más (+) del menú MySQL Servers, como se muestra en la siguiente imagen.

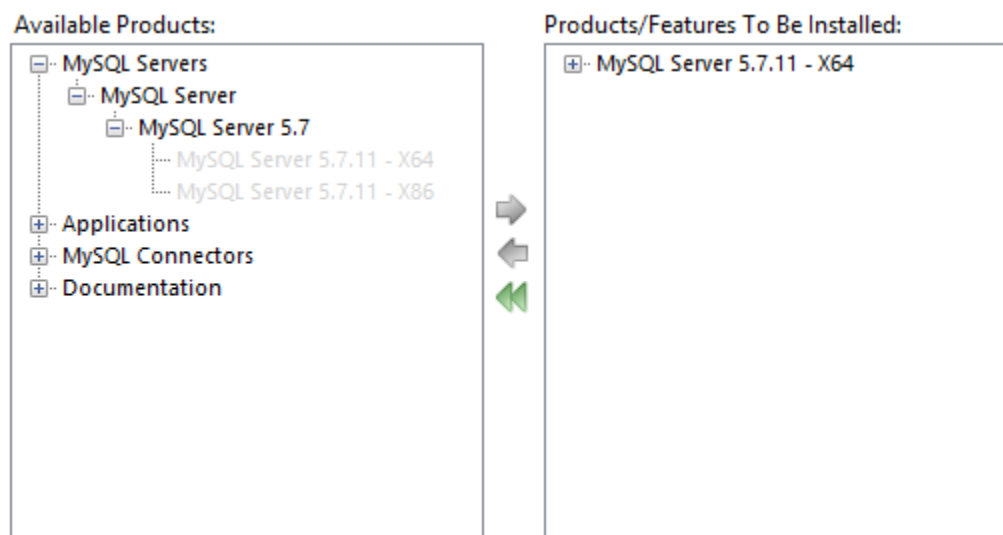


Dependiendo de la arquitectura de su procesador y su sistema operativo deberá elegir el más conveniente. Para este caso x64 (64 bits), si usted posee una computadora de 32 bits entonces debe elegir la opción x86.

A continuación debe presionar sobre la flecha verde con dirección a la derecha.




Esto permitirá pasar el programa a la lista de instalación. Clic en Next




Entonces aparecerá un cuadro donde nos resumirá qué se instalará y si está listo para instalarse. Clic en el botón Execute.


Press Execute to upgrade the following products.

Product	Status	Progress	Notes
 MySQL Server 5.7.11	Ready to Install		

Iniciará su instalación casi inmediatamente.

Product	Status	Progress	Notes
 MySQL Server 5.7.11	Installing		

Una vez terminado nos notificará si se ha podido instalar o ha fallado. Clic en Next.

Product	Status	Progress	Notes
 MySQL Server 5.7.11	Complete		

Nota: Si usted eligió más componentes los pasos se repetirán, es decir, al terminar de instalar un producto (como es puesto en la tabla) pasará a instalar el que sigue de manera secuencial.

A continuación se debe configurar MySQL Server para que la computadora pueda ejecutarlo satisfactoriamente. El mensaje que mostrará entonces es el siguiente, clic en Next.

Product	Status
MySQL Server 5.7.11	Ready to Configure

Dejaremos los valores por defecto y clic en Next, como se muestra a continuación.

Type and Networking

Server Configuration Type

Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.

Config Type:

Connectivity

Use the following controls to select how you would like to connect to this server.

- ☒ TCP/IP Port Number:
- ☒ Open Firewall port for network access
- ☐ Named Pipe Pipe Name:
- ☐ Shared Memory Memory Name:

Advanced Configuration

Select the checkbox below to get additional configuration page where you can set advanced options for this server instance.

- ☐ Show Advanced Options

Seguidamente se pedirá ingresar una contraseña para el usuario root (usuario máximo de MySQL que usualmente es el administrador general). Si usted desea agregar un nuevo usuario puede agregarlo en la parte inferior dando clic en el botón Add User.

Accounts and Roles

Root Account Password

Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password:

Repeat Password:

Password Strength: **Weak**

MySQL User Accounts

Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL Username	Host	User Role	
			<div>Add User</div> <div>Edit User</div> <div>Delete</div>

< Back

Next >

Cancel

Después se pedirá configurar el nombre del servicio y también si se desea iniciar MySQL Server cada vez que la computadora encienda. Por ahora se dejará por defecto, para no preocuparse luego de si MySQL Server está ejecutándose o no, clic en Next.

Windows Service

☒ Configure MySQL Server as a Windows Service

Windows Service Details

Please specify a Windows Service name to be used for this MySQL Server instance. A unique name is required for each instance.

Windows Service Name:

☒ Start the MySQL Server at System Startup

Run Windows Service as ...

The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.

- ☒ **Standard System Account**
Recommended for most scenarios.
- ☐ **Custom User**
An existing user account can be selected for advanced scenarios.

<input type="button" value=" < Back"/>	<input type="button" value=" Next >"/>	<input type="button" value=" Cancel"/>
-------------------------------------------	-------------------------------------------	----------------------------------------

La siguiente ventana mostrará los pasos que se llevarán a cabo para la instalación, clic en Execute.

Apply Server Configuration

Press [Execute] to apply the changes

Configuration Steps Log

- ☐ Stopping Server [if necessary]
- ☐ Writing configuration file
- ☐ Updating firewall
- ☐ Adjusting Windows service [if necessary]
- ☐ Initializing Database [if necessary]
- ☐ Starting Server
- ☐ Applying security settings
- ☐ Creating user accounts
- ☐ Updating Start Menu Link

< Back Execute Cancel

Nota: Si durante la ejecución aparece un mensaje solo presione clic en OK. De cualquier forma si usted lee ese mensaje menciona que la instalación está tardando más de lo normal.

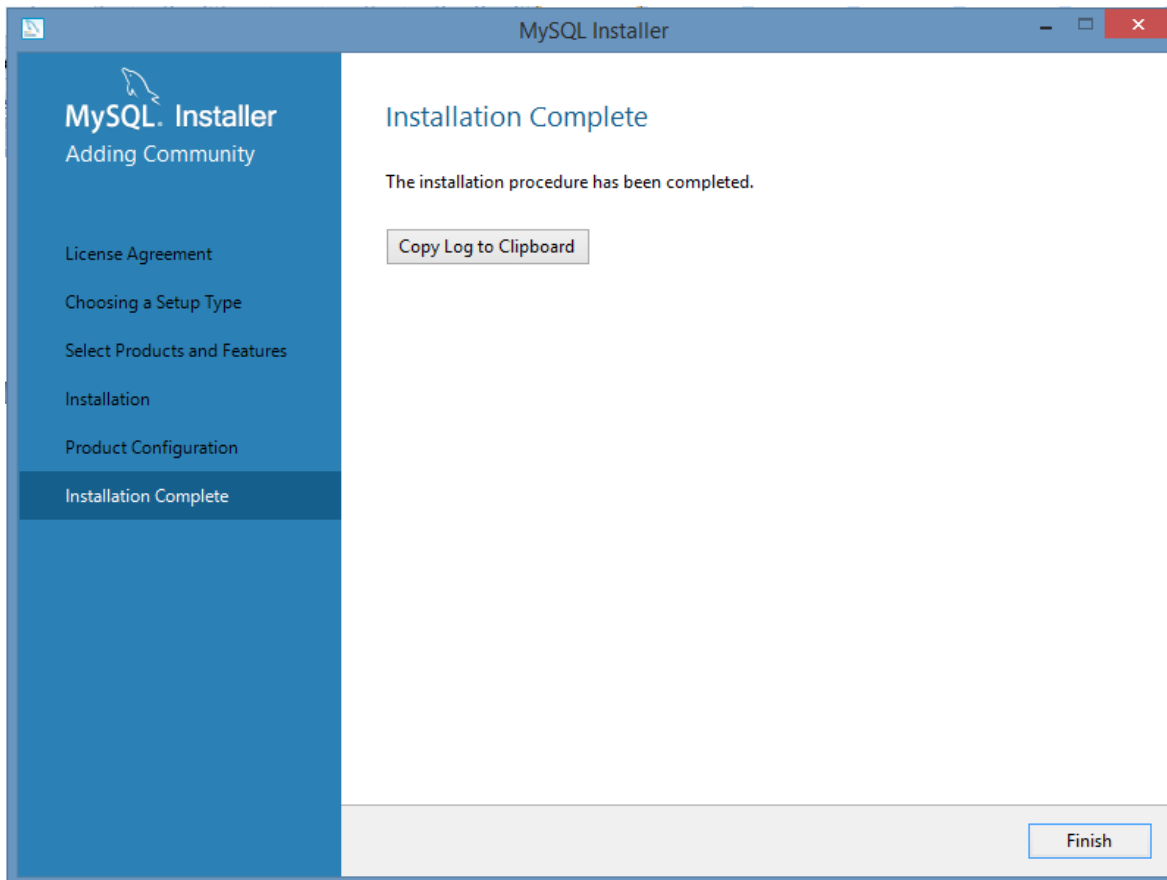
Una vez finalizado todo aparecerá de la siguiente forma. Clic en Finish

- ✔ Stopping Server [if necessary]
- ✔ Writing configuration file
- ✔ Updating firewall
- ✔ Adjusting Windows service [if necessary]
- ✔ Initializing Database [if necessary]
- ✔ Starting Server
- ✔ Applying security settings
- ✔ Creating user accounts
- ✔ Updating Start Menu Link

Por último se mostrará el estado del producto, es decir, de la instalación de MySQL Server en este ejemplo. Clic en Next.

Product	Status
MySQL Server 5.7.11	Configuration Complete.

Enhorabuena, MySQL Server está instalado.



3 Conectándose y desconectándose al servidor MySQL

Para conectarse al servidor, usualmente se necesita de un nombre de usuario (login) y de una contraseña (password), y si el servidor al que se desea conectar está en una máquina diferente, también se necesitará indicar el nombre o la dirección IP de dicho servidor.

Una vez conocidos estos tres valores, se puede conectar de la siguiente manera:

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

Cuando se ejecuta este comando, se pedirá que proporcione también la contraseña para el nombre de usuario que está usando.

Si la conexión al servidor MySQL se pudo establecer de manera satisfactoria, se mostrará el mensaje de bienvenida y prompt de MySQL estará disponible:

```
shell>mysql -h casita -u blueman -p
```

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

```
Welcome to the MySQL monitor. Commands end with ; or \g.
```

```
Your MySQL connection id is 5563 to server version: 3.23.41
```

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

```
mysql>
```

Este prompt indica que mysql está listo para recibir comandos.

Algunas instalaciones permiten que los usuarios se conecten de manera anónima al servidor corriendo en la máquina local. Si es el caso de nuestra máquina, debemos de ser capaces de conectarnos al servidor invocando a mysql sin ninguna opción:

```
shell> mysql
```

Después de que la conexión fue creada de manera satisfactoria, se puede desconectar en cualquier momento al escribir "quit", "exit", o presionar CONTROL + D.

La mayoría de los ejemplos siguientes asume que estamos conectados al servidor, lo cual se indica con el prompt de MySQL.

4 Ejecutando algunas consultas

En este momento la conexión al servidor MySQL es totalmente correcta, aún cuando no se ha seleccionado alguna base de datos para trabajar. A continuación se mostrará cómo escribir algunos comandos para familiarizarse con el funcionamiento de MySQL.

```
mysql> SELECT VERSION(), CURRENT_DATE;
+-----+-----+
| VERSION() | CURRENT_DATE |
+-----+-----+
| 3.23.41 | 2002-10-01 |
+-----+-----+
1 row in set (0.03 sec)
mysql>
```

Este 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 cuánto 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.

Las siguientes consultas son equivalentes:

```
mysql> SELECT VERSION(), CURRENT_DATE;  
mysql> select version(), current_date;  
mysql> SeLeCt vErSiOn(), current_DATE;
```

Aunque hasta este momento se han escrito sentencias sencillas de una sólo línea, es posible escribir más de una sentencia por línea, siempre y cuando estén separadas por punto y coma:

```
mysql> SELECT VERSION(); SELECT NOW();  
+-----+  
| VERSION() |  
+-----+  
| 3.23.41 |  
+-----+  
1 row in set (0.01 sec)  
  
+-----+  
| NOW() |  
+-----+  
| 2002-10-28 14:26:04 |  
+-----+  
1 row in set (0.01 sec)
```

Un comando no necesita ser escrito en una sola línea, así que los comandos que requieran de varias líneas no son un problema. MySQL determinará en dónde finaliza la sentencia cuando encuentre el punto y coma, no cuando encuentre el fin de línea.

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

```
mysql> SELECT  
-> USER(),  
-> CURRENT_DATE;  
+-----+-----+  
| USER() | CURRENT_DATE |  
+-----+-----+  
| bluelman@localhost | 2002-09-14 |  
+-----+-----+  
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 se desea terminar de escribir la consulta, utilice entonces `"\c"` como se muestra en el siguiente ejemplo:

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

De nuevo, se regresa el comando el prompt `mysql>` que 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>).
<code>"></code>	Esperando la siguiente línea para completar una cadena que comienza con una comilla doble (<code>"</code>).

Los comandos multi-línea comúnmente ocurren por accidente cuando la tecla ENTER es presionada, pero se omite escribir el punto y coma. En este caso MySQL se queda esperando para la finalización de la consulta:

```
mysql> SELECT USER()
->
```

Si esto llega a suceder, muy probablemente MySQL estará esperando por un punto y coma, de manera que al poner el punto y coma se podrá completar la consulta y MySQL podrá ejecutarla:

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

Los prompts '>' y '>' ocurren durante la escritura de cadenas. En MySQL se permite escribir cadenas utilizando comillas sencillas o comillas dobles (por ejemplo, 'hola' y "hola"), y MySQL permite escribir cadenas que ocupen múltiples líneas. De manera que cuando se muestre en el prompt '>' o '>', MySQL indica que se ha empezado a escribir una cadena, pero no ha sido finalizada con la comilla correspondiente.

Aunque esto puede suceder si se está escribiendo una cadena muy grande, es más frecuente obtener alguno de estos prompts si inadvertidamente se escribe alguna de estas comillas.

Por ejemplo:

```
mysql> SELECT * FROM mi_tabla WHERE nombre = "Lupita AND edad < 30;  
>
```

Si se ejecuta esta consulta SELECT y entonces ENTER es presionado para ver el resultado, no sucederá nada. En lugar de preocuparse porque la consulta ha tomado mucho tiempo, hay que notar la pista que da MySQL cambiando el prompt. Esto indica que MySQL está esperando que se finalice la cadena iniciada ("Lupita).

En este caso, ¿qué se debe hacer? . La cosa más simple es cancelar la consulta. Sin embargo, no basta con escribir "\c", ya que MySQL interpreta esto como parte de la cadena que estamos escribiendo. En lugar de esto, se escribe antes la comilla correspondiente y después "\c ":

```
mysql> SELECT * FROM mi_tabla WHERE nombre = "Lupita AND edad < 30;  
> " \c  
mysql>
```

El prompt cambiará de nuevo al ya conocido mysql>, indicando que MySQL está listo para una nueva consulta.

Es sumamente importante conocer lo que significan los prompts '>' y '>', ya que si en algún momento aparece alguno de ellos, todas la líneas que se escriban a continuación serán consideradas como parte de la cadena, inclusive cuando se introduce QUIT. Esto puede ser confuso, especialmente si no se sabe que es necesario escribir la comilla correspondiente para finalizar la cadena, para que podamos escribir después algún otro comando, o terminar la consulta para ser ejecutada.

5. Creando y usando una base de datos

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

Supongamos que tenemos diversas mascotas en casa (nuestro pequeño zoológico) y deseamos tener registros de los datos acerca de ellas. Podemos hacer esto al crear tablas que guarden esta información, para que posteriormente la consulta de estos datos sea bastante fácil y de manera muy práctica. Esta sección muestra cómo crear una base de datos, crear una tabla, incorporar datos en una tabla, y recuperar datos de las tablas de diversas maneras.

La base de datos "zoológico" será muy simple (deliberadamente), pero no es difícil pensar de situaciones del mundo real en la cual una base de datos similar puede ser usada.

Primeramente usaremos la sentencia SHOW 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 requerida, ya que ésta tiene la información de los privilegios de los usuarios de MySQL. La base de datos "test" es creada durante la instalación de MySQL con el propósito de servir como área de trabajo para los usuarios que inician en el aprendizaje de MySQL.

Se debe anotar también que es posible que no veamos todas las bases de datos si no tenemos el privilegio SHOW DATABASES. Se recomienda revisar la sección del manual de MySQL dedicada a los comandos GRANT y REVOKE.

Si la base de datos "test" existe, hay que intentar acceder a ella:

```
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: éste debe ser usado en una sola línea.

Podríamos usar la base de datos "test" (si tenemos acceso a ella) para los ejemplos que vienen a continuación, pero cualquier cosa que hagamos puede ser eliminada por cualquier otro usuario que tenga acceso a esta base de datos. Por esta razón, es recomendable que preguntemos al administrador MySQL acerca de la base de datos que podemos usar. Supongamos que deseamos tener una base de datos llamada "zoologico" (nótese que no se está acentuando la palabra) a la cual sólo nosotros tengamos acceso, para ello el administrador necesita ejecutar un comando como el siguiente:

```
mysql> GRANT ALL on zoologico.* TO MiNombreUsuario@MiComputadora  
-> IDENTIFIED BY 'MiContraseña';
```

En donde *MiNombreUsuario* es el nombre de usuario asignado dentro del contexto de MySQL, *MiComputadora* es el nombre o la dirección IP de la computadora desde la que nos conectamos al servidor MySQL, y *MiContraseña* es la contraseña que se nos ha asignado, igualmente, dentro del ambiente de MySQL exclusivamente. Ambos, nombre de usuario y contraseña no tienen nada que ver con el nombre de usuario y contraseña manejados por el sistema operativo (si es el caso).

Si el administrador creó la base de datos al momento de asignar los permisos, podemos hacer uso de ella. De otro modo, nosotros debemos crearla:

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

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

```
mysql> CREATE DATABASE zoologico;  
Query OK, 1 row affected (0.00 sec)  
mysql> USE zoologico  
Database changed  
mysql>
```

Bajo el sistema operativo Unix, 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 casita -u blueman -p zoologico

Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 17 to server version: 3.23.38-nt

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

mysql>
```

Observar que "zoologico" 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.

6 Creando una tabla

Crear la base de datos es la parte más fácil, pero en este momento la base de datos está vacía, como lo indica el comando SHOW TABLES:

```
mysql> SHOW TABLES;  
Empty set (0.00 sec)
```

La parte un tanto complicada es decidir la estructura que debe tener nuestra base de datos: qué tablas se necesitan y qué columnas estarán en cada tabla.

En principio, necesitamos una tabla que contenga un registro para cada una de nuestras mascotas. Ésta puede ser una tabla llamada mascotas, y debe contener por lo menos el nombre de cada uno de nuestros animalitos. Ya que el nombre en sí no es muy interesante, la tabla debe contener alguna otra información. Por ejemplo, si más de una persona en nuestra familia tiene una mascota, es probable que tengamos que guardar la información acerca de quién es el dueño de cada mascota. Así mismo, también sería interesante contar con alguna información más descriptiva tal como la especie, y el sexo de cada mascota.

¿Y qué sucede con la edad?. Esto puede ser también de interés, pero no es una buena idea almacenar este dato en la base de datos. La edad cambia conforme pasa el tiempo, lo cual significa que debemos de actualizar los registros frecuentemente. En vez de esto, es una mejor idea guardar un valor fijo, tal como la fecha de nacimiento. Entonces, cuando necesitemos la edad, la podemos calcular como la diferencia entre la fecha actual y la fecha de nacimiento. MySQL proporciona funciones para hacer operaciones entre fechas, así que no hay ningún problema.

Al almacenar la fecha de nacimiento en lugar de la edad tenemos algunas otras ventajas:

Podemos usar la base de datos para tareas tales como generar recordatorios para cada cumpleaños próximo de nuestras mascotas. Podemos calcular la edad en relación a otras fechas que la fecha actual. Por ejemplo, si almacenamos la fecha en que murió nuestra mascota en la base de datos, es fácil calcular que edad tenía nuestro animalito cuando falleció. Es probable que estemos pensando en otro tipo de información que sería igualmente útil en la tabla "mascotas", pero para nosotros será suficiente por ahora contar con información de nombre, propietario, especie, nacimiento y fallecimiento.

Usaremos la sentencia CREATE TABLE para indicar como estarán conformados los registros de nuestras mascotas.

```
mysql> CREATE TABLE mascotas(  
-> nombre VARCHAR(20), propietario VARCHAR(20),  
-> especie VARCHAR(20), sexo CHAR(1), nacimiento DATE,  
-> fallecimiento DATE);  
Query OK, 0 rows affected (0.02 sec)  
  
mysql>
```

VARCHAR es una buena elección para los campos nombre, propietario, y especie, ya que los valores que almacenarán son de longitud variable. No es necesario que la longitud de estas columnas sea la misma, ni tampoco que sea de 20. Se puede especificar cualquier longitud entre 1 y 255, lo que se considere más adecuado. Si resulta que la elección de la longitud de los campos que hemos hecho no resultó adecuada, MySQL proporciona una sentencia ALTER TABLE que nos puede ayudar a solventar este problema.

El campo sexo puede ser representado en una variedad de formas, por ejemplo, "m" y "f", o tal vez "masculino" y "femenino", aunque resulta más simple la primera opción.

El uso del tipo de dato DATE para los campos nacimiento y fallecimiento debe de resultar obvio.

Ahora que hemos creado la tabla, la sentencia SHOW TABLES debe producir algo como:

```
mysql> SHOW TABLES;  
+-----+  
| Tables_in_zoologico |  
+-----+  
| mascotas |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

Para verificar que la tabla fue creada como nosotros esperábamos, usaremos la sentencia DESCRIBE:

```
mysql> DESCRIBE mascotas;
```

Field	Type	Null	Key	Default	Extra
nombre	varchar(20)	YES		NULL	
propietario	varchar(20)	YES		NULL	
especie	varchar(20)	YES		NULL	
sexo	char(1)	YES		NULL	
nacimiento	date	YES		NULL	
fallecimiento	date	YES		NULL	

```
6 rows in set (0.01 sec)
```

```
mysql>
```

Podemos hacer uso de la sentencia DESCRIBE en cualquier momento, por ejemplo, si olvidamos los nombres ó el tipo de las columnas en la tabla.

7 Cargando datos en una tabla

Después de haber creado la tabla, ahora podemos incorporar algunos datos en ella, para lo cual haremos uso de las sentencias INSERT y LOAD DATA.

Supongamos que los registros de nuestras mascotas pueden ser descritos por los datos mostrados en la siguiente tabla.

Nombre	Propietario	Especie	Sexo	Nacimiento	Fallecimiento
Fluffy	Arnoldo	Gato	f	1999-02-04	
Mau	Juan	Gato	m	1998-03-17	
Buffy	Arnoldo	Perro	f	1999-05-13	
FanFan	Benito	Perro	m	2000-08-27	
Kaiser	Diana	Perro	m	1998-08-31	1997-07-29
Chispa	Omar	Ave	f	1998-09-11	
Wicho	Tomás	Ave		2000-02-09	
Skim	Benito	Serpiente	m	2001-04-29	

Debemos observar que MySQL espera recibir fechas en el formato YYYY-MM-DD, que puede ser diferente a lo que nosotros estamos acostumbrados.

Ya que estamos iniciando con una tabla vacía, la manera más fácil de poblarla es crear un archivo de texto que contenga un registro por línea para cada uno de nuestros animalitos para que posteriormente carguemos el contenido del archivo en la tabla únicamente con una sentencia.

Por tanto, debemos de crear un archivo de texto "mascotas.txt" que contenga un registro por línea con valores separados por tabuladores, cuidando que el orden de las columnas sea el mismo que utilizamos en la sentencia CREATE TABLE. Para valores que no conozcamos podemos usar valores nulos (NULL). Para representar estos valores en nuestro archivo debemos usar \N.

El archivo mascotas.txt contendrá:

Fluffy	Arnoldo	Gato	f	1999-02-04	\N
Mau	Juan	Gato	m	1998-03-17	\N
Buffy	Arnoldo	Perro	f	1999-05-13	\N
FanFan	Benito	Perro	m	2000-08-27	\N
Kaiser	Diana	Perro	m	1998-08-31	1997-07-29

Chispa	Omar	Ave	f	1998-09-11	\N
Wicho	Tomás	Ave	\N	2000-02-09	\N
Skim	Benito	Serpiente	m	2001-04-29	\N

Para cargar el contenido del archivo en la tabla mascotas, usaremos el siguiente comando:

```
mysql> LOAD DATA LOCAL INFILE "mascotas.txt" INTO TABLE mascotas;
```

La sentencia LOAD DATA nos permite especificar cuál es el separador de columnas, y el separador de registros, por default el tabulador es el separador de columnas (campos), y el salto de línea es el separador de registros, que en este caso son suficientes para que la sentencia LOAD DATA lea correctamente el archivo "mascotas.txt".

Si lo que deseamos es añadir un registro a la vez, entonces debemos hacer uso de la sentencia INSERT. En la manera más simple, debemos proporcionar un valor para cada columna en el orden en el cual fueron listados en la sentencia CREATE TABLE. Supongamos que nuestra hermana Diana compra un nuevo hámster nombrado Pelusa. Podemos usar la sentencia INSERT para agregar su registro en nuestra base de datos.

```
mysql> INSERT INTO mascotas  
-> VALUES('Pelusa','Diana','Hamster','f','2000-03-30',NULL);
```

Notar que los valores de cadenas y fechas deben estar encerrados entre comillas. También, con la sentencia INSERT podemos insertar el valor NULL directamente para representar un valor nulo, un valor que no conocemos. En este caso no se usa \N como en el caso de la sentencia LOAD DATA.

De este ejemplo, debemos ser capaces de ver que es un poco más la tarea que se tiene que realizar si inicialmente cargamos los registros con varias sentencias INSERT en lugar de una única sentencia LOAD DATA.

8 Recuperando información de una tabla

La sentencia SELECT es usada para obtener la información guardada en una tabla. La forma general de esta sentencia es:

SELECT LaInformaciónQueDeseamos FROM DeQueTabla WHERE CondiciónASatisfacer

Aquí, *LaInformaciónQueDeseamos* es la información que queremos ver. Esta puede ser una lista de columnas, o un * para indicar "todas las columnas". *DeQueTabla* indica el nombre de la tabla de la cual vamos a obtener los datos. La cláusula WHERE es opcional. Si está presente, la *CondiciónASatisfacer* especifica las condiciones que los registros deben satisfacer para que puedan ser mostrados.

8.1 Seleccionando todos los datos

La manera más simple de la sentencia SELECT es cuando se recuperan todos los datos de una tabla:

```
mysql> SELECT * FROM mascotas;
```

nombre	propietario	especie	sexo	nacimiento	fallecimiento
Fluffy	Arnoldo	Gato	f	1999-02-04	NULL
Mau	Juan	Gato	m	1998-03-17	NULL
Buffy	Arnoldo	Perro	f	1999-05-13	NULL
FanFan	Benito	Perro	m	2000-08-27	NULL
Kaiser	Diana	Perro	m	1998-08-31	1997-07-29
Chispa	Omar	Ave	f	1998-09-11	NULL
Wicho	Tomás	Ave	NULL	2000-02-09	NULL
Skim	Benito	Serpiente	m	2001-04-29	NULL
Pelusa	Diana	Hamster	f	2000-03-30	NULL

9 rows in set (0.00 sec)

Esta forma del SELECT es útil si deseamos ver los datos completos de la tabla, por ejemplo, para asegurarnos de que están todos los registros después de la carga de un archivo.

Por ejemplo, en este caso que estamos tratando, al consultar los registros de la tabla, nos damos cuenta de que hay un error en el archivo de datos (mascotas.txt): parece que Kaiser ha nacido después de que ha fallecido!. Al revisar un poco el pedigree de Kaiser encontramos que la fecha correcta de nacimiento es el año 1989, no 1998.

Hay por lo menos un par de maneras de solucionar este problema:

Editar el archivo "mascotas.txt" para corregir el error, eliminar los datos de la tabla mascotas con la sentencia DELETE, y cargar los datos nuevamente con el comando LOAD DATA:

```
mysql> DELETE FROM mascotas;  
mysql> LOAD DATA LOCAL INFILE "mascotas.txt" INTO TABLE mascotas;
```

Sin embargo, si hacemos esto, debemos ingresar los datos de Pelusa, la mascota de nuestra hermana Diana.

La segunda opción consiste en corregir sólo el registro erróneo con una sentencia UPDATE:

```
mysql> UPDATE mascotas SET nacimiento="1989-08-31" WHERE nombre="Kaiser";
```

Como se mostró anteriormente, es muy fácil recuperar los datos de una tabla completa. Pero típicamente no deseamos hacer esto, particularmente cuando las tablas son demasiado grandes. En vez de ello, estaremos más interesados en responder preguntas particulares, en cuyo caso debemos especificar algunas restricciones para la información que deseamos ver.

8.2 Seleccionando registros particulares

Podemos seleccionar sólo registros particulares de una tabla. Por ejemplo, si deseamos verificar el cambio que hicimos a la fecha de nacimiento de Kaiser, seleccionamos sólo el registro de Kaiser de la siguiente manera:

```
mysql> SELECT * FROM mascotas WHERE nombre="Kaiser";
```

nombre	propietario	especie	sexo	nacimiento	fallecimiento
Kaiser	Diana	Perro	m	1989-08-31	1997-07-29

1 row in set (0.00 sec)

La salida mostrada confirma que el año ha sido corregido de 1998 a 1989.

La comparación de cadenas es normalmente no sensitiva, así que podemos especificar el nombre como "kaiser", "KAISER", etc. El resultado de la consulta será el mismo.

Podemos además especificar condiciones sobre cualquier columna, no sólo el "nombre". Por ejemplo, si deseamos conocer qué mascotas nacieron después del 2000, tendríamos que usar la columna "nacimiento":


```
mysql> SELECT * FROM mascotas WHERE nacimiento >= "2000-1-1";
```

nombre	propietario	especie	sexo	nacimiento	fallecimiento
FanFan	Benito	Perro	m	2000-08-27	NULL
Wicho	Tomás	Ave	NULL	2000-02-09	NULL
Skim	Benito	Serpiente	m	2001-04-29	NULL
Pelusa	Diana	Hamster	f	2000-03-30	NULL

4 rows in set (0.00 sec)

Podemos también combinar condiciones, por ejemplo, para localizar a los perros hembras:

```
mysql> SELECT * FROM mascotas WHERE especie="Perro" AND sexo="f";
```

nombre	propietario	especie	sexo	nacimiento	fallecimiento
Buffy	Arnoldo	Perro	f	1999-05-13	NULL

1 row in set (0.00 sec)

La consulta anterior usa el operador lógico AND. Hay también un operador lógico OR:

```
mysql> SELECT * FROM mascotas WHERE especie = "Ave" OR especie = "Gato";
```

nombre	propietario	especie	sexo	nacimiento	fallecimiento
Fluffy	Arnoldo	Gato	f	1999-02-04	NULL
Mau	Juan	Gato	m	1998-03-17	NULL
Chispa	Omar	Ave	f	1998-09-11	NULL
Wicho	Tomás	Ave	NULL	2000-02-09	NULL

4 rows in set (0.00 sec)

El operador AND y el operador OR pueden ser intercambiados. Si hacemos esto, es buena idea usar paréntesis para indicar como deben ser agrupadas las condiciones:

```
mysql> SELECT * FROM mascotas WHERE (especie = "Gato" AND sexo = "m")
-> OR (especie = "Perro" AND sexo = "f");
```

nombre	propietario	especie	sexo	nacimiento	fallecimiento
Mau	Juan	Gato	m	1998-03-17	NULL
Buffy	Arnoldo	Perro	f	1999-05-13	NULL

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

8.3 Seleccionando columnas particulares

Si no deseamos ver los registros completos de una tabla, entonces tenemos que usar los nombres de las columnas en las que estamos interesados separándolas por coma. Por ejemplo, si deseamos conocer la fecha de nacimiento de nuestras mascotas, debemos seleccionar la columna "nombre" y "nacimiento":

```
mysql> SELECT nombre, nacimiento FROM mascotas ORDER BY
-> nacimiento DESC;
```

nombre	nacimiento
Skim	2001-04-29
FanFan	2000-08-27
Pelusa	2000-03-30
Wicho	2000-02-09
Buffy	1999-05-13
Fluffy	1999-02-04
Chispa	1998-09-11
Mau	1998-03-17
Kaiser	1989-08-31

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

Para conocer quién tiene alguna mascota, usaremos la siguiente consulta:

```
mysql> SELECT propietario FROM mascotas;
```

```
+-----+
```

```
| propietario |
```

```
+-----+
```

```
| Arnoldo |
```

```
| Juan |
```

```
| Arnoldo |
```

```
| Benito |
```

```
| Diana |
```

```
| Omar |
```

```
| Tomás |
```

```
| Benito |
```

```
| Diana |
```

```
+-----+
```

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

Sin embargo, debemos notar que la consulta recupera el nombre del propietario de cada mascota, y algunos de ellos aparecen más de una vez. Para minimizar la salida, agregaremos la palabra clave DISTINCT:

```
mysql> SELECT DISTINCT propietario FROM mascotas;
```

```
+----- +
```

```
| propietario |
```

```
+----- +
```

```
| Arnoldo |
```

```
| Juan |
```

```
| Benito |
```

```
| Diana |
```

```
| Omar |
```

```
| Tomás |
```

```
+----- +
```

```
6 rows in set (0.03 sec)
```

Se puede usar también una cláusula WHERE para combinar selección de filas con selección de columnas. Por ejemplo, para obtener la fecha de nacimiento de los perritos y los gatitos, usaremos la siguiente consulta:

```
mysql> SELECT nombre, especie, nacimiento FROM mascotas
```

```
-> WHERE especie = "perro" OR especie = "gato";
```

nombre	especie	nacimiento
Fluffy	Gato	1999-02-04
Mau	Gato	1998-03-17
Buffy	Perro	1999-05-13
FanFan	Perro	2000-08-27
Kaiser	Perro	1989-08-31

5 rows in set (0.00 sec)

8.4 Ordenando registros

Se debe notar en los ejemplos anteriores que las filas regresadas son mostradas sin ningún orden en particular. Sin embargo, frecuentemente es más fácil examinar la salida de una consulta cuando las filas son ordenadas en alguna forma útil. Para ordenar los resultados, tenemos que usar una cláusula ORDER BY.

Aquí aparecen algunos datos ordenados por fecha de nacimiento:

nombre	nacimiento
Kaiser	1989-08-31
Mau	1998-03-17
Chispa	1998-09-11
Fluffy	1999-02-04
Buffy	1999-05-13
Wicho	2000-02-09
Pelusa	2000-03-30
FanFan	2000-08-27
Skim	2001-04-29

9 rows in set (0.00 sec)

En las columnas de tipo caracter, el ordenamiento es ejecutado normalmente de forma no sensitiva, es decir, no hay diferencia entre mayúsculas y minúsculas. Sin embargo, se puede forzar un ordenamiento sensitivo al usar el operador BINARY.

Para ordenar en orden inverso, debemos agregar la palabra clave DESC al nombre de la columna que estamos usando en el ordenamiento:

```
mysql> SELECT nombre, nacimiento FROM mascotas ORDER BY  
-> nacimiento DESC;
```

nombre	nacimiento
Skim	2001-04-29
FanFan	2000-08-27
Pelusa	2000-03-30
Wicho	2000-02-09
Buffy	1999-05-13
Fluffy	1999-02-04
Chispa	1998-09-11
Mau	1998-03-17
Kaiser	1989-08-31

9 rows in set (0.00 sec)

Podemos ordenar múltiples columnas. Por ejemplo, para ordenar por tipo de animal, y poner al inicio los animalitos más pequeños de edad, usaremos la siguiente consulta:

```
mysql> SELECT nombre, especie, nacimiento FROM mascotas  
-> ORDER BY especie, nacimiento DESC;
```

nombre	especie	nacimiento
Wicho	Ave	2000-02-09
Chispa	Ave	1998-09-11
Fluffy	Gato	1999-02-04
Mau	Gato	1998-03-17
Pelusa	Hamster	2000-03-30
FanFan	Perro	2000-08-27
Buffy	Perro	1999-05-13
Kaiser	Perro	1989-08-31
Skim	Serpiente	2001-04-29

9 rows in set (0.00 sec)

Notar que la palabra clave DESC aplica sólo a la columna nombrada que le precede.

Recomendaciones

En el mundo de la programación, como en la vida misma, hay que ser ordenado y elegante. Hay que seguir unas pautas para que seamos más eficientes y evitarnos de los malos hábitos. A la hora de crear una base de datos pasa exactamente lo mismo. Tenemos que ser cuidadosos a la hora de crear y de ejecutar comandos en una base de datos para evitarnos peligros mucho mayores. Y es que, la base de datos en una web, ya que estamos hablando de MySQL, es una de las partes más importantes del desarrollo. Más aún si almacenamos en ella datos sensibles como datos personales de usuarios y/o clientes.

Es por eso que hay que seguir unas normas que nunca debemos saltarnos a la hora de trabajar con una BD. A continuación os muestro una serie de consejos que deberíamos seguir a rajatabla cuando estemos manejando o tengamos que crear una base de datos MySQL. Muchos de estos consejos ya nos lo dieron en nuestro época de estudiante, pero la rutina los olvida y los oculta en lo más lejano de nuestro inconsciente. Hoy los traemos otra vez a la vida para que no vuelvas a caer en estos errores tan peligrosos.

Bibliografía

<https://es.wikipedia.org/wiki/DBA>

https://es.wikipedia.org/wiki/Administrador_de_base_de_datos

<http://es.ccm.net/contents/320-administrador-de-bases-de-datos>

<http://www.angelfire.com/nf/tecvirtual/cursos/admonbd/DBA1.htm>

<http://searchdatacenter.techtarget.com/es/definicion/Administrador-de-base-de-datos-DBA>

<http://blog.capacityacademy.com/2013/02/18/cuales-son-las-funciones-de-un-administrador-de-base-de-datos-parte-1-de-2/>

http://programacion.net/articulo/10_consejos_para_mysql_1065