Universidad Técnica Federico Santa María
Departamento de Electrónica
Programación de Sistemas

MySQL

My struct query lenguage

René Andrés Harb Hoecker
9821086-5
Licanray@elo.utfsm.cl

Indice

Universidad Técnica Federico Santa María
Departamento de Electrónica
Programación de Sistemas

RESUMEM

En el siguiente trabajo veremos que son las bases de datos y profundizaremos en un a muy usada en estos días “MySql”, veremos donde radican los principios de esta Gran y poderosa base de datos ,explicaremos porque MySql es un sistema de SGBD(sistema de Gestión de base de datos).

No queda fuera de este trabajo hablar de la instalación y compilación para su puesta en marcha, además veremos un ejemplo de esta base de datos aplicada a los servicios de la nueva era cibernética en el modelo **Cliente Servidor.**

También se presentara un ejemplo más concreto de esta base de datos ya entrando al código duro que presenta este lenguaje estructurado de programación.

INTRODUCCION

Un sistema de bases de datos no es más que un sistema para archivar en computador o dicho de otra manera un lugar donde se almacena un conjunto de archivos de datos computarizados, el cual permite al usuario realizar las siguientes operaciones:

- Agregar archivos nuevos (vacíos) a la base de datos.
- Insertar datos nuevos en archivos ya existentes.
- Obtener, actualizar, borrar datos de archivos ya existentes.
- Eliminar archivos ya existentes (vacíos o no) de la base de datos.

Existen diferentes tipo de bases de datos, siendo una de la más usadas por su simplicidad y organización, las de tipo ***relacional (Mysql)***. Una base de datos relacional indica que es una base de datos que se apoya en el modelo relacional, es decir, el dato es representado en forma de ***relaciones o tablas***. Las tablas en un sistema relacional son consideradas como una estructura de datos lógica. En general, cada tabla presenta las siguientes características:

- Cada columna tiene un valor simple.
- Todas las filas (registros) son del mismo tipo.
- Las columnas (campos) no tienen un orden particular.
- Las filas tiene un campo o atributo identificador (o un conjunto de campos) que forma su clave primaria.
- Las filas no tienen un orden en particular.

Mysql esta basado en SQL (***lenguaje de consulta estructurado)*** ,este a su vez es un lenguaje de base de datos normalizado, utilizado por el motor de base de datos de Microsoft Jet.

SQL se utiliza para crear objetos QueryDef, como el argumento de origen del método OpenRecordSet y como la propiedad RecordSource del control de datos. También se puede utilizar con el método Execute para crear y manipular directamente las bases de datos Jet y crear consultas SQL de paso para manipular bases de datos remotas **cliente - servidor**.

1.1. Componentes del SQL

El lenguaje SQL está compuesto por **comandos, cláusulas, operadores** y **funciones de agregado.** Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

1.2 Comandos

Existen dos tipos de comandos SQL:

DLL (***Lenguaje de manipulación de datos***) que permiten crear y definir nuevas bases de datos, campos e índices.

DML (**Lenguaje de definición de datos**) que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.

Comandos DLL

Comando	Descripción
CREATE	Utilizado para crear nuevas tablas, campos e índices
DROP	Empleado para eliminar tablas e índices
ALTER	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

Comandos DML

Comando	Descripción

SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado
INSERT	Utilizado para cargar lotes de datos en la base de datos en una única operación.
UPDATE	Utilizado para modificar los valores de los campos y registros especificados
DELETE	Utilizado para eliminar registros de una tabla de una base de datos

1.3 Cláusulas

Las cláusulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular.

Cláusula	Descripción
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico

1.4 Operadores Lógicos

Operador	Uso
AND	Es el "y" lógico. Evalua dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdar si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.

1.5 Operadores de Comparación

Operador	Uso

<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor ó Igual que
>=	Mayor ó Igual que
=	Igual que
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en la comparación de un modelo
In	Utilizado para especificar registros de una base de datos

1.6 Funciones de Agregado

Las funciones de agregado se usan dentro de una cláusula SELECT en grupos de registros para devolver un único valor que se aplica a un grupo de registros.

Función	Descripción
AVG	Utilizada para calcular el promedio de los valores de un campo determinado
COUNT	Utilizada para devolver el número de registros de la selección
SUM	Utilizada para devolver la suma de todos los valores de un campo determinado
MAX	Utilizada para devolver el valor más alto de un campo especificado
MIN	Utilizada para devolver el valor más bajo de un campo especificado

Universidad Técnica Federico Santa María
Departamento de Electrónica
Programación de Sistemas

MySql

MySql data base fue desarrollado por los Laboratorios MySql AB , actuales desarrolladores de ésta. Es por excelencia la base de datos de código abierto més popular en el mercado mundial y
Se estima que en el mundo hay más de cuatro millones de instalaciones .
Muchas compañías como Yahoo!, Lucent Tecnologías, Sony Pictures Digital Entertainment, Motorota, NASA, Silicón Graphics, HP, Xerox , Cisco, etc.. confían 100 % en MySql por ser esta una de la mas rápidas y robustas bases de datos en la actualidad

MySQL como sistema de gestión de bases de datos

MySQL es un **sistema de gestión de bases de datos** (SGBD) SQL que inicialmente buscó una compatibilidad con la API de mSQL. Sus principales objetivos han sido la velocidad y la robustez.
En el mundo de GNU/Linux es MySQL junto a PostgresSQL los principales SGBD de uso libre y con código fuente. Las grandes compañías de bases de datos como Software AG están comenzando a ver en GNU/Linux un nuevo mercado y están portando sus grandes sistemas a GNU/Linux. ADABAS es un SGBD del nivel de Oracle y está portado a Linux.
MySQL está en fase de pleno desarrollo. Se están publicando de forma regular nuevas versiones del sistemas, así como herramientas que son básicas en cualquier SGBD actual:

Algunos dispositivos de acceso al SGBD

-Dispositivo JDBC para acceder desde Java
- Dispositivo ODBC para acceder utilizando la API ODBC
- APIs de programación para C, Perl, C++, Python y TCL
- Acceso desde PHP
- Entornos visuales de gestión de la base de datos
- Control de acceso basado en una base de datos de administración

Una de las cosas más importantes de MySQL es que las bases de datos las almacena creando un directorio por cada una de ellas, y dentro de dicho directorio, crea tres ficheros por tabla donde se almacenan los registros de la tabla y la definición de la tabla. El copiar una base de datos es tan fácil como copiar un directorio por lo que hacer copias de seguridad(Backups) de las mismas es algo de lo más sencillo.

Universidad Técnica Federico Santa María
Departamento de Electrónica
Programación de Sistemas

Proceso de Instalación de MySQL

El primer paso es obtener MySQL, compilarlo e instalarlo. Para ello podemos obtener la última versión de http://www.mysql.com

En el momento de escribir este artículo dicha versión era la 3.21.29-gamma

Como se habitual lo que obtenemos es un archivo **.tgz**, que deberemos descomprimir para poder continuar con la instalación: **tar xvfz mysql-3.21.29.tar.gz.**

Entramos en el directorio ***mysql-3.21.29*** desde donde compilaremos el programa. Antes de compilar hay que ejecutar el programa *configure* que se encarga de adaptar las fuentes a

nuestro sistema. A este programa le podemos pasar parámetros de donde se encuentran ciertas librerías e incluso configurar ciertas opciones del programa.

En **Debian 1.3.1 la** versión de Perl instalada es demasiado antigua para el funcionamiento de la parte de Perl de MySQL provocando problemas a la hora de compilar. Por ello la desactivaremos ya que en nuestro caso es innecesaria, algo que pudiera no ser necesario para el lector cuyo sistema tuviera una versión de Perl más actualizada. La orden a ejecutar es: **configure - -without-perl - -enable-shared**. Con la opción **de - -enable-shared** indicamos que queremos que se creen librerías compartidas para el acceso a MySQL.

Si no hay problemas en este paso con un simple **make** se creará el demonio servidor de bases de datos (**mysqld**), los clientes de acceso y programas de gestión junto con las librerías necesarias para poder acceder desde nuestros programas en C a la base de datos.

Tras finalizar la compilación pasamos a instalar MySQL. Para ello necesitaremos ponernos en el papel de **superusuario** y ejecutar**: make install.** Con esta orden se copian bajo **/usr/local/** en los direcotorios adecuados los binarios y las librerías. En especial cabe resaltar que las bases de datos se crean dentro del directorio **/usr/local/var**. Para crear la primera base de datos conocida como mysql y que contiene los permisos de acceso a las bases de datos, ejecutamos desde el directorio actual la orden**: scripts/mysql_install_db.** Con ello se creará esta primera base de datos en **/usr/local/var/mysql** y se arrancará el demonio de la base de datos.

Con esto queda finalizada la compilación e instalación de MySQL. Para arrancar el demonio servidor de bases de datos hay que ejecutar **/usr/local/libexec/mysqld**, aunque dicho demonio está ya arrancado tras el último paso de la instalación.

Entre las herramientas que acompañan a MySQL cabe destacar ***mysql*** que se encuentra en **/usr/local/bin** y que permite ejecutar comandos SQL sobre la base de datos.

BASES DE DATOS EN INTERNET BAJO GNU/LINUX

En Internet y en el mundo existen una gran cantidad de bases de datos como por ejemplo

Oracle , Sybase , MSQL 1.x and 2.x , MySQL, Solid , Generic ODBC , Adabas D FilePro , Velocis , Dbase , UNIX dbm.

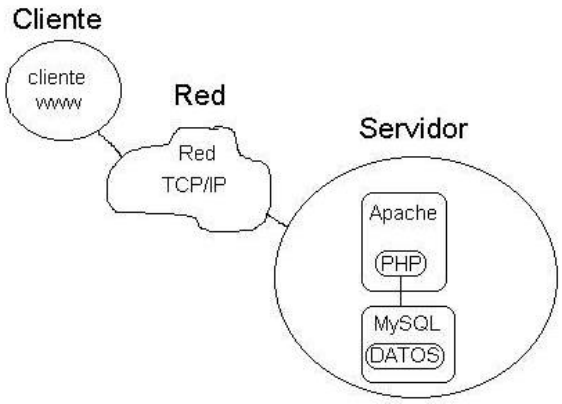
Modelo cliente Servidor

Aquí se presenta un ejemplo de un sistema completo de publicación de datos en Internet, utilizando el navegador de Internet como cliente, y como servidor de bases de datos a Mysql que será accedido mediante el servidor Web Apache con una extensión llamada PHP.

El sistema estará compuesto por cinco elementos principales:

- GNU/Linux como sistema operativo abierto
  - Apache como servidor de Web
  - PHP como módulo de ampliación de Apache para acceder a la base de datos
  - **MySQL como base de datos**

  Todos los elementos del sistema se pueden obtener sin ningún tipo de coste por lo que el montaje inicial del sistema y su uso no conllevan ninguna inversión. Es más, a excepción de MySQL, de la que hay que comprar licencias en el caso de que se quiera vender sistemas que la incluyan, todos los demás elementos tienen licencia GPL .(General Public License)



Los pasos que describen la interacción entre el cliente y la base de datos son:

1. El cliente carga una página HTML con un formulario, rellena los datos y los envía al servidor

2. A través de la red TCP/IP (Internet normalmente) los datos llegan al servidor, y son enviados a un programa, en este caso un programa PHP.

3. El servidor detecta que los datos se envían a una página PHP por lo que informa al módulo de PHP del programa a ejecutar y le pasa los datos del cliente

4. El módulo de PHP ejecuta el programa, el cual accederá a MySQL utilizando de nuevo una comunicación TCP/IP (en nuestro caso local)

5. MySQL procesa la petición del programa PHP y le envía de vuelta los resultados.

6. El módulo PHP recibe los resultados y a través del servidor Apache, envía una página HTML con los resultados al cliente

7. El cliente recibe la página HTML resultado de sus petición, a través de la red TCP/IP

MySQL es la que mejor soporte tiene por parte de PHP junto con PostgreSQL, debido a que ambas bases de datos son de libre distribución, con el código fuente, y con un acceso muy sencillo a través del lenguaje C.

Creando una base de datos con MySQL

Como ya vimos en el proceso de Instalación de MySQL quedamos con la base de datos lista para ocuparla. MySQL se basa en SQL por lo que los comandos son en su mayoría los mismos, cabe destacar que SQL no hace distinción entre mayúsculas y minúsculas, en cambio MySQL si lo hace a excepción de las palabras reservadas. A continuación veremos como :

Cómo crear una base de datos

Cómo crear una tabla

Cómo cargar los datos en la tabla

Cómo extraer información de la tabla de varias maneras

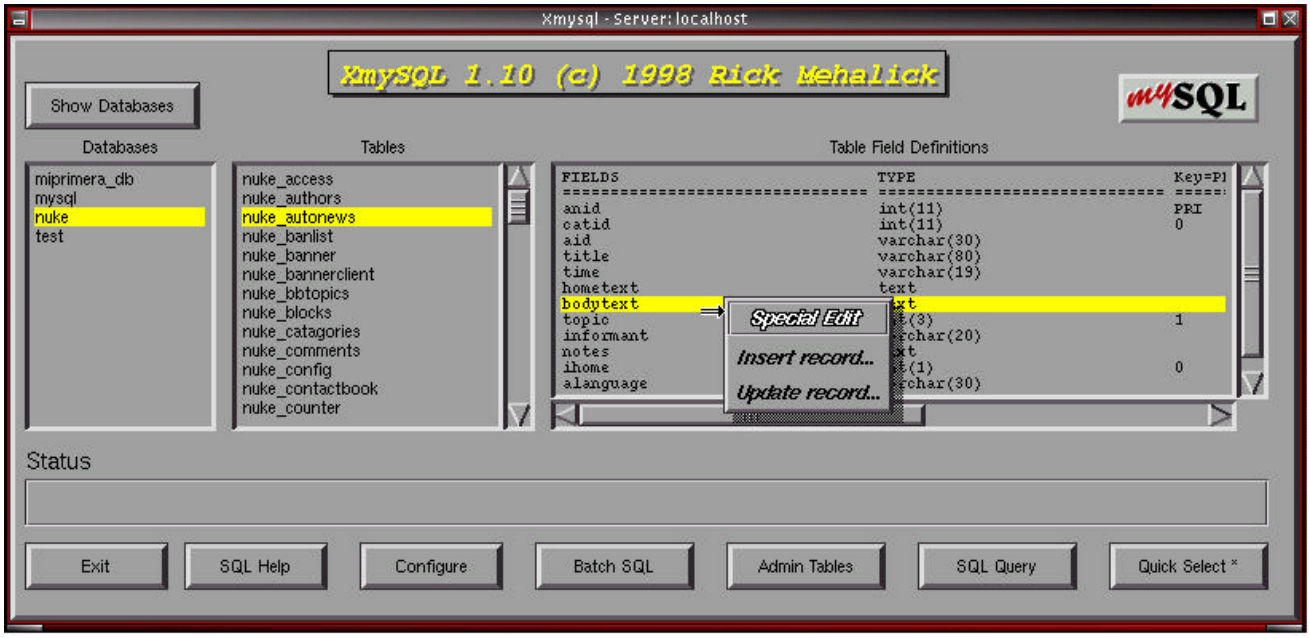
Cómo usar múltiples tablas

```
mysql> SHOW DATABASES;
+--------------+
| Database     |
+--------------+
| miprimera_db |
| mysql        |
| nuke         |
| test         |
+--------------+
4 rows in set (0.00 sec)
mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

mysql> show tables;
+------------------+
| Tables_in_test   |
+------------------+
| datos_personales |
+------------------+
1 row in set (0.00 sec)

mysql> create table programacion_de_sistemas2 (nombre varchar(30),sexo
varchar(10) );
Query OK, 0 rows affected (0.00 sec)

mysql> show tables;
+---------------------------+
| Tables_in_test            |
+---------------------------+
| datos_personales          |
| programacion_de_sistemas  |
| programacion_de_sistemas2 |
+---------------------------+
3 rows in set (0.00 sec)
```

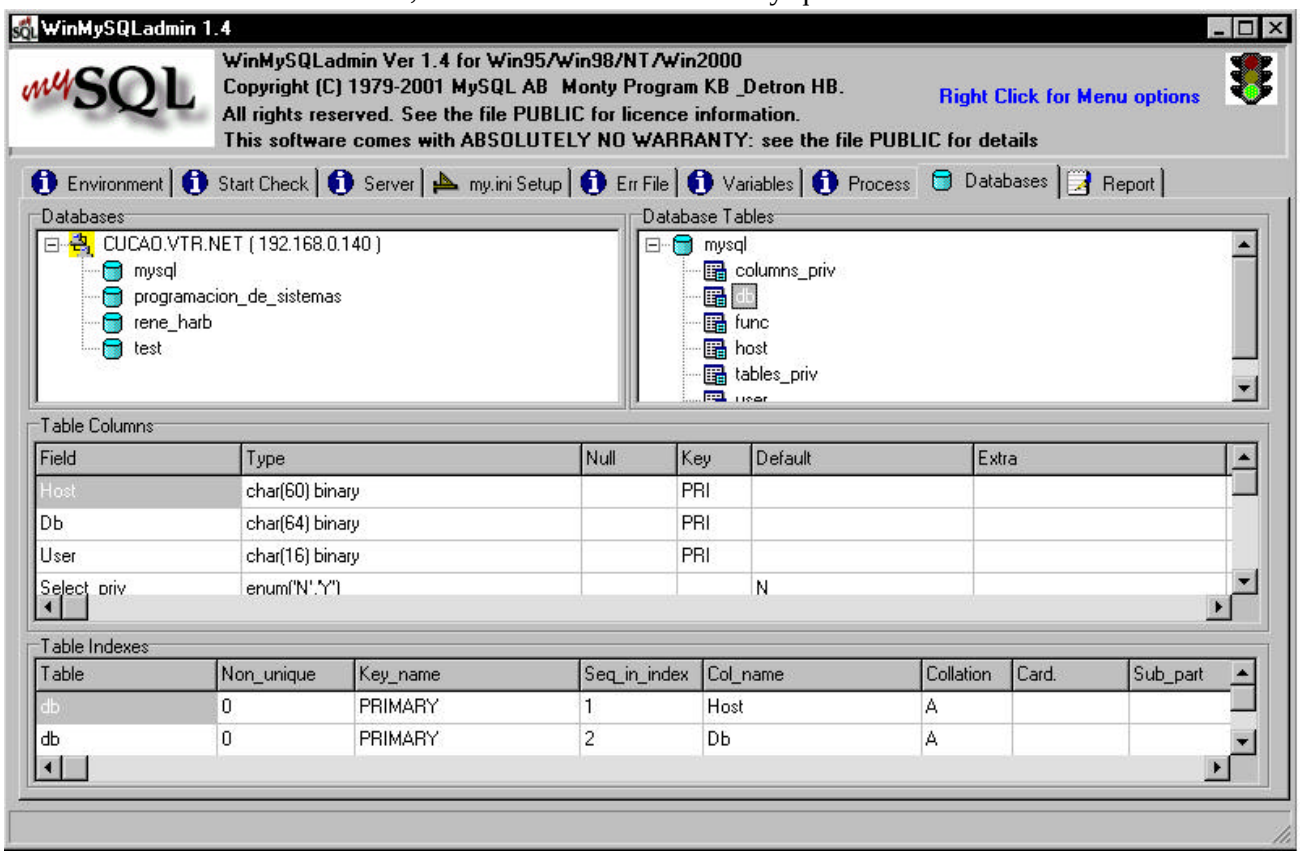
```
mysql> describe programacion_de_sistemas2;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| nombre | varchar(30) | YES  |     | NULL    |       |
| sexo   | varchar(10) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Más detalles en el ANEXO

Existen tambien programas graficos para administrar a MySQL. Podemos mensionar por ejemplo a xmysql bajo plataforma Lnux



, en windows se tiene a WinMysqladmin



Conclusiones

La robustez y eficiencia hacen de **MySQL** una de la más ocupadas en el mundo. Su facilidad para crear respaldos la hacen sobresalir de otras bases de datos. El hecho de ser una base de datos que tiene una filosofía muy sofisticada la cual consta en crear tablas lógicamente estructuradas ó más bien dicho el hecho de ser un sistema de base de datos **RELACIONAL** la hace una de las bases de datos más poderosas del mundo y de mayor prestigio. GNU / LINUX, MySQL y otros software libres juntos al código abierto empiezan crecer en un campo muy importante y que puede ayudarle a multiplicar una confianza que comienza a palparse dentro del mundo empresarial.

Bibliografía o referencias

http://www.mysql.com/

http://mmmysql.sourceforge.net/

http://www.devshed.com/Server_Side/MySQL

http://www.cybercursos.net/

http://www.linuxplanet.com/linuxplanet/tutorials/1046/1/

http://www.edatanew.com/

http://www.xmysql.com

http://web.wt.net/~dblhack/

Anexo

Conectando y desconectando del servidor
==========

```
shell> mysql -h host -u user -p
Enter password: *******
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 459 to server version: 3.22.20a-log

Type 'help' for help.

mysql>
```

El prompt te indica que mysql ya está listo para la introducción de comandos.

Algunas instalaciones MySQL permiten a los usuarios conectarse como usuarios "anonymous" (sin
nombre) al servidor ejecutándose en el host local. Si este es el caso en tu máquina, deberías ser
capaz de conectar a ese servidor invocando mysql sin ninguna opción:

```
shell> mysql
```

Una vez que hayas conectado con éxito, puedes desconectarte en cualquier momento tecleando QUIT en
el prompt mysql> :

```
mysql> QUIT
Bye
```

También puedes desconectar tecleando **control-D**.

Haciendo consultas
==========

En este punto, es más importante averiguar un poco sobre cómo lanzar consultas que lanzarse directamente a la creación de tablas, cargar datos en ellas y recuperar los datos de las mismas. Esta sección describe los principios básicos de la entrada de comandos.
Aquí presentamos un comando simple que pide al servidor que nos diga su número de versión y fecha actual.

```
mysql> SELECT VERSION(), CURRENT_DATE;
+-----------+--------------+
| version() | CURRENT_DATE |
+-----------+--------------+
| 3.22.23b  | 2002-10-27   |
+-----------+--------------+
1 row in set (0.06 sec)
```

Esta consulta ilustra muchas cosas sobre mysql:

* Un comando consiste normalmente de una sentencia SQL seguida por un punto y coma. (Existen algunas

  excepciones donde no es necesario el punto y coma. QUIT, mencionado más adelante, es una de ellas. Conoceremos otras más adelante.)

* Cuando lanzas un comando, mysql lo envía al servidor para su ejecución y muestra los resultados, después imprime otro mysql> para indicar que está listo para otro comando.

* mysql muestra la salida de una consulta como una tabla (filas y columnas). La primera fila contiene etiquetas para las columnas. Las siguientes filas son el resultado de la consulta. Normalmente, las etiquetas de las columnas son los nombres de las columnas que has obtenido de la base de datos. Si pides el valor de una expresión en vez de una columna de una tabla (como en el ejemplo anterior), mysql etiqueta la columna usando la propia expresión.
* mysql muestra el número de filas que se han dado como resultado, y cuánto tiempo llevó la ejecución de la consulta, lo que te da una idea aproximada del rendimiento del servidor. Estos valores son imprecisos porque representan tiempo real (no tiempo de CPU o máquina), y porque están afectados por factores como la carga del servidor y la latencia de la red. (Por cuestiones de brevedad, la línea "rows in set" no se mostrará en los ejemplos posteriores de este capítulo.)

Las palabras clave pueden ser tecleadas en cualquier combinación mayúscula/minúscula. Las siguientes consultas son equivalentes:

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

Mysql como una calculadora sencilla:

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+-------------+---------+
| SIN(PI()/4) | (4+1)*5 |
+-------------+---------+
|    0.707107 |      25 |
+-------------+---------+
```

```
mysql> SELECT VERSION(); SELECT NOW();
+-----------+
| version() |
+-----------+
| 3.22.23b  |
+-----------+

+---------------------+
| NOW()               |
+---------------------+
| 2002-10-27 17:33:16 |
+---------------------+
```

Un comando no necesita ser dado todo en una sóla línea, así pues, los comandos largos que requieran varias lineas no son un problema. mysql determina cuando termina tu sentencia buscando el punto y coma final, no buscando el final de la línea de entrada. (En otras palabras, mysql acepta entrada de libre formato: recoleta las líneas de entrada pero no las ejecutahasta que vea el punto y coma.
Aquí tenemos un simple ejemplo de múltiples líneas:

```
mysql> SELECT
    -> USER()
    -> ,
    -> CURRENT_DATE;
+----------------+--------------+
| USER()         | CURRENT_DATE |
+----------------+--------------+
| root@localhost | 2002-10-27   |
+----------------+--------------+
```

En este ejemplo, observa como cambia el prompt de mysql> a -> una vez que has insertado la primera línea de una consulta multi-línea. Esta es la forma en que mysql indica que no ha encontrado una sentencia completa y que está esperando por el resto. El prompt es tu amigo, dado que ofrece una retroalimentación (feedback) significativa. Si usas ese feedback, siempre sabrás a qué está esperando mysql.

Si decides que no quieres ejecutar un comando que está en proceso de introducción, puedes cancelarlo
tecleando \c :

```
mysql> SELECT
    -> USER
    -> \c
mysql>
```

Observa aquí también el prompt. Ha vuelto a mysql> tras haber tecleado \c, ofreciendo un feedback que indica que mysql está listo para un nuevo comando.
La siguiente tabla muestra cada uno de los prompts que puedes ver y resume qué es lo que significan y el estado en el que se encontrará mysql:

Prompt Significado
mysql> Listo para un nuevo comando
-> Esperando una nueva línea de una consulta multi-línea
'> Esperando la siguiente línea, se ha insertado una línea que comienza con (')
"> Esperando la siguiente línea, se ha insertado una línea que comienza con (")

Las sentencias multi-línea ocurren comúnmente "por accidente" cuando intentas lanzar un comando en una única línea, pero olvidas el punto y coma del final. En este caso, mysql espera más entrada:

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

Si esto es lo que te ocurre (crees que has introducido una sentencia pero la única respuesta es un prompt como ->, lo más probable es que mysql esté esperando por el punto y coma. Si no observas qué es lo que te dice el prompt, podrías quedarte esperando un buen rato antes de enterarte de qué es lo que sucede. Introduce un punto y coma para completar la sentencia, y mysql la ejecutará:

```
mysql> SELECT USER()
    -> ;
+----------------+
| USER()         |
+----------------+
| root@localhost |
+----------------+
```

Los prompts '> y "> ocurren durante la recogida de strings. En MySQL, puedes escribir strings encerrados por comillas simples (')
o dobles (") (por ejemplo, 'hola' o "adios"), y mysql te permite introducir también strings que se cortan en múltiples líneas. Cuando veas un p rompt como '> ó ">, significa que has introducido una línea que contenía un string que comenzaba por (') o ("), pero que no has introducido aún la comilla (simple o doble) de cierre. Esto está bien si realmente estabas introduciendo un string multi-línea, pero no es lo más normal. Lo que sí es más normal, es que los prompts '> ó "> indiquen que te has olvidado del caracter de cierre " ó ' . Por ejemplo:

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

Si tecleas esta sentencia SELECT, después pulsas ENTER y esperas por el resultado, no sucederá nada. En lugar de preocuparte, "?por qué tarda tanto

esta consulta?", observa la pista que te ofrece el prompt "> . Esto te indica que mysql espera ver el resto de un string que aún no ha terminado. (?Ves el error en la sentencia? La cadena "García ha perdido las comillas de cierre.)

Llegados a este punto, ?qué puedes hacer?. Lo más fácil es cancelar el comando. Sin embargo, no puedes teclear simplemente \c en este ejemplo, dado que mysql !lo interpretará como parte del string que está leyendo! En vez de eso, introduce las comillas de cierre (para que mysql sepa que ya has terminado de introducir el string), y después teclea \c :

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

El prompt vuelve a cambiar a mysql>, indicando que mysql está listo para un nuevo comando.
Es importante saber qué significan los prompts '> y ">, dado que si introduces por error un string
sin cerrar, cualquier otra línea que introduzcas serán ignoradas por mysql - !incluyendo una línea
que contenga QUIT! Esto puede ser bastante confuso, especialmente si no sabes que debes introducir
la comilla de cierre antes de poder cancelar el comando actual.

CREANDO Y USANDO UNA BASE DE DATOS

Ahora que sabes como introducir comandos. Supon que tienes varias mascotas en tu casa (tu pequeño "zoo") y que te gustaría llevar un control de varios tipos de información sobre estos animales. Puedes hacerlo creando tablas que guarden tus datos y cargandolas con la información deseada. Después puedes responder a diferentes series de preguntas sobre tus animales extrayendo los datos de las tablas. Esta sección explica cómo hacer todo esto:

* Cómo crear una base de datos
* Cómo crear una tabla
* Cómo cargar los datos en la tabla
* Cómo extraer información de la tabla de varias maneras
* Cómo usar múltiples tablas

La base de datos del zoo será simple (deliberadamente), pero no es difícil pensar en situaciones del
mundo real en las que se pudiera utilizar una base de datos similar. Por ejemplo, se podría usar una base

de datos como ésta en una granja para llevar un control del ganado, o por un veterinario para
controlar el historial de sus pacientes.

Usa la sentencia SHOW para averiguar qué bases de datos existen actualmente en el servidor:

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
```

Probablemente, la lista de las bases de datos será diferente en tu máquina, pero las bases de datos mysql y test es probable que se encuentren en esa lista. Se requiere la base de datos mysql pues describe los privilegios de acceso de los usuarios. La base de datos test se ofrece como campo de
pruebas para que los usuarios prueben ahí sus teorías.
Si la base de datos test existe, intenta acceder a ella:

mysql> USE test
Database changed
Observa que USE, como QUIT, no requiere un punto y coma. (Puedes terminar este tipo de sentencias con un punto y coma si quieres, pero no es necesario.) La sentencia USE es especial en otro sentido, también: debe ser tecleada en una sola línea.
Puedes usar la base de datos test (si tienes acceso a ella) para los ejemplos que siguen, pero cualquier cosa que crees en dicha base de datos puede ser eliminada por cualquiera que tenga acceso a ella. Por esta razón, deberías pedir a tu administrador MySQL permisos para usar una base de datos
propia. Suponte que le quieres llamar zoo. El administrador necesitará jecutar entonces la siguiente orden:

mysql> GRANT ALL ON zoo.* TO tu_nombre;
donde tu_nombre es el nombre de usuario MySQL que tengas asignado.
ejemplo:

mysql> GRANT ALL ON zoo.* TO chessy@localhost;
Query OK, 0 rows affected (0.08 sec)

Creando y seleccionando una base de datos
==

Si el administrador creó la base de datos para tí cuando te configuró los permisos, puedes comenzar a usarla.

Creando una base de datos zoo

```
[chessy@bishito chessy]$ mysql -u chessy
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 3.22.23b

Type 'help' for help.

mysql> CREATE DATABASE zoo;
Query OK, 1 row affected (0.02 sec)
```

Bajo Unix, los nombres de bases de datos son sensibles a las mayúsculas/minúsculas (a diferencia de los comandos SQL), así que deberás referirte siempre a tu base de datos con el nombre zoo, no como Zoo, ZOO o cualquier otra variante. Es es así también para las tablas. (Bajo Windows, esta restricción desaparece, aunque deberías referirte a las bases de datos y a las tablas usando la misma sintaxis en tus consultas.)
Crear una base de datos no la selecciona para su uso, debes hacerlo explícitamente. Para hacer que la base de datos zoo sea tu base de datos de trabajo, usa el comando:

```
mysql> USE zoo;
Database changed
```

Tu base de datos sólo necesita ser creada una vez, pero debes seleccionarla para usarla cada vez que
comiences una sesión mysql. Puedes hacerlo lanzando un comando USE como se ha visto en el
ejemplo. Alternativamente, puedes seleccionar la base de datos desde la línea de comandos cuando
lanzas mysql. Simplemente especifica su nombre tras los parámetros de conexión que hayas
escrito. Por ejemplo:

```
shell> mysql -h host -u user -p zoo
Enter password: ********
```

Observe que en la línea de comandos del ejemplo, zoo no es tu password. Si quieres introducir tu password como parámetro en la línea de comandos tras la opción -p, debes hacerlo sin teclear un espacio en blanco intermedio (es decir, como -pmi_password, no como -p mi_password). Sin embargo, no es recomendable poner tu password en la línea de comandos, pues hacerlo lo expone a posibles miradas de otros usuarios conectados a tu máquina.

Creando una tabla
==========

Crear una tabla es la parte fácil, pero hasta este momento está vacía, como te dice la orden SHOW
TABLES:

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

La parte más dura consiste en decidir cual va a ser la estructura de tu base de datos: qué tablas necesitarás, y qué columnas tendrá

cada una de ellas.Querrás seguramente una tabla que contenga un registro por cada una de tus mascotas. Esta tabla puede llamarse mascotas, y debería contener, como mínimo, el nombre de cada animal. Dado que el nombre por sí solo no es muy interesante, la tabla debería contener otra información. Por ejemplo, si más de una persona de tu familia tiene mascotas, probablemente quieras listar el propietario de cada animal. También querrás guardar información descriptiva básica como puede ser la especie y el sexo de cada mascota.
¿Qué pasa con la edad? Podría ser de interés, pero no es una buena cosa a guardar en una base de datos. La edad cambia a medida que pasa el tiempo, lo que significa que tendrás que actualizar tus registros a menudo. En vez de eso, es mejor almacenar un valor fijo como la edad de nacimiento. Después, cada vez que necesites saber la edad, puedes calcularla como la diferencia
entre la fecha actual y la fecha de nacimiento. MySQL ofrece funciones para realizar cálculos aritméticos entre fechas, por lo que esto no es difícil. Almacenar la fecha de nacimiento en lugar de la edad tiene también otras ventajas:

* Puedes usar la base de datos para generar recordatorios de cumpleaños de mascotas. (Si crees que este tipo de consulta es algo tonta, observa que es la misma pregunta que necesitarás hacer en el contexto de una base de datos de un negocio para identificar clientes a los que pronto necesitarás mandar un saludo por su cumpleaños, para ese toque personal asistido por ordenador :-)
* Puedes calcular la edad en relación a fechas distintas a la fecha actual. Por ejemplo, si almacenas la fecha de muerte en la base de datos, puedes calcular fácilmente lo vieja que era una mascota cuando murió.

Seguramente puedas pensar en otros tipos de información que sería útil en la tabla mascota, pero los identificados hasta ahora son suficientes por el momento: nombre, propietarios, especie, sexo, fecha de nacimiento y muerte.

Usa una sentencia CREATE TABLE para especificar la estructura de tu tabla:

```
mysql> CREATE TABLE mascota (nombre VARCHAR(20), propietario VARCHAR(20),
    -> especie VARCHAR(20), sexo CHAR(1), nacimiento DATE, muerte DATE);
```

VARCHAR es una buena elección para las columnas nombre, propietario y especie dado que los valores de estas columnas variarán su longitud. Las longitudes de estas columnas no necesitan ser iguales, y no necesitan ser 20. Puedes elegir

cualquier longitud entre 1 y 255, cualquiera que te parezca razonable. (Si realizar una elección pobre y resulta que más adelante necesitas un campo mayor, MySQL ofrece una sentencia ALTER TABLE.

El sexo del animal puede ser representado en una variedad de formas, por ejemplo, "m" y "f", o quizás "masculino" y "femenino". Es más simple usar un único caracter, "m" ó "f".
El uso del tipo de datos DATE para las columnas de nacimiento y muerte es una opción bastante
obvia.
Ahora que ya has creado una tabla, SHOW TABLES debería producir alguna salida:

```
mysql> SHOW TABLES;
+---------------+
| Tables in zoo |
+---------------+
| mascota       |
+---------------+
```

Para verificar que tu tabla fue creada de la forma que esperabas, usa una sentencia DESCRIBE:

```
mysql> DESCRIBE mascota;
+-------------+-------------+------+-----+---------+-------+
| 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    |       |
| muerte      | date        | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

Puedes usar DESCRIBE en cualquier momento, por ejemplo, si olvidas los nombres de las columnas de tu
tabla o a qué tipo de datos pertenecen.

Cargando datos en una tabla

Una vez creada tu tabla, necesitas poblarla. Las sentencias LOAD DATA e INSERT son útiles para esto.

Suponte que tus registros de mascotas pueden ser descritos como se muestra más abajo. (Observa que
MySQL espera que las fechas se introduzcan en formato AAAA-MM-DD; esto podría ser diferente a lo que
estás acostumbrado.)

nombre	propietario	especie	sexo	nacimiento	muerte
Fluffy	Harold	gato	f	1993-02-04	/n
Claws	Gwen	gato	m	1994-03-17	/n
Buffy	Harold	perro	f	1989-05-13	/n
Fang	Benny	perro	m	1990-08-27	/n
Bowser	Diane	perro	m	1998-08-31	1995-07-29
Chirpy	Gwen	pájaro	f	1998-09-11	/n
Whistler	Gwen	pájaro	/n	2000-12-09	/n
Slim	Benny	serpiente	m	1996-04-29	/n

Dado que estás comenzando con una tabla vacía, una forma sencilla de poblarla consiste en crear un fichero de texto conteniendo una fila para cada uno de tus animales, y después cargar el contenido del fichero en la tabla con una sola sentencia.

Puedes crear un fichero de texto "mascota.txt" conteniendo un registro por línea, con valores separados por tabuladores, y dados en el orden en el que las columnas fueron listadas en la sentencia CREATE TABLE. Para valores perdidos (como sexos desconocidos, o fechas de muerte de animales que aún están vivos), puedes usar valores NULL. Para representar estos en tu fichero de texto, use \N. Por ejemplo, el

registro para Whistler el pájaro sería algo como esto (donde el espacio en blanco entre valores es un simple caracter de tabulación):

```
Whistler    Gwen      pájaro      \N      2000-12-09      \N
```

Para cargar el fichero de texto "mascota.txt" en la tabla mascota, usa este comando:

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

Se puede especificar el valor de separación de columna y el marcador de final de línea explícitamente en la sentencia LOAD DATA , pero por defecto equivalen a TAB y LF (intro). Estos valores por defecto son suficientes para que la sentencia que lee el fichero "mascota.txt" funcione correctamente.

Al nuevos registros uno a uno, la sentencia INSERT es muy útil. En su forma más simple, ofreces valores para cada columna, en el orden en el que las columnas fueron listadas en la sentencia CREATE TABLE.
Ahora Diane consige un nuevo hamster llamado Puffball. Podrías añadir un nuevo registro usando una sentencia INSERT como esta:

```
mysql> INSERT INTO mascota
    -> VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);
```

Los valores string y fecha se espefican encerrados entre comillas.
Con INSERT puedes insertar NULL directamente para representar un valor perdido. No usamos \N como hacíamos con LOAD DATA.

Extrayendo información de una tabla
==========

La sentencia SELECT se usa para recabar información de una tabla. La forma general de la sentencia es:

```
SELECT qué_seleccionar
FROM de_qué_tabla
WHERE condiciones_a_satisfacer
```

qué_seleccionar indica qué es lo que quieres seleccionar. Puede ser una lista de columnas, o * para indicar "todas las columnas". de_qué_tabla indica la tabla de la que quieres extraer datos. La claúsula WHERE es opcional. Si está presente, condiciones_a_satisfacer especifica las codiciones que las filas deben cumplir para estar presentes en el resultado de la selección.

Seleccionando todos los datos
==========

La forma más simplede SELECT recoge toda la información de una tabla:

```
mysql> SELECT * FROM mascota;
+----------+-------------+-----------+------+------------+------------+
| nombre   | propietario | especie   | sexo | nacimiento | muerte     |
+----------+-------------+-----------+------+------------+------------+
| Bluffy   |  Harold     | gato      | f    | 1993-02-04 | NULL       |
| Claws    |  Gwen       | gato      | m    | 1994-03-17 | NULL       |
| Buffy    |  Harold     | perro     | f    | 1989-05-13 | NULL       |
| Fang     |  Benny      | perro     | m    | 1990-08-27 | NULL       |
| Bowser   |  Diane      | perro     | m    | 1998-08-31 | 1995-07-29 |
| Chirpy   |  Gwen       | pájaro    | f    | 1998-09-11 | NULL       |
| Whistler |  Gwen       | pájaro    | NULL | 2000-12-09 | NULL       |
| Slim     |  Benny      | serpiente | m    | 1996-04-29 | NULL       |
| Puffball | Diane       | hamster   | f    | 1999-03-30 | NULL       |
+----------+-------------+-----------+------+------------+------------+
```

Esta forma de SELECT es útil si quieres revisar tu tabla al completo, por ejemplo, tras haberla cargado con tu conjunto inicial de datos. Como suele suceder, la salida ya muestra un error en tu fichero de datos: Bowser !parece haber nacido tras su muerte! Consultando tus papeles originales sobre el pedigree del perro, descubres que la fecha correcta de nacimiento es 1989, no 1998.

```
mysql> DELETE from mascota;
mysql> LOAD DATA LOCAL INFILE "mascota.txt" INTO TABLE mascota;
```

Sin embargo, si haces esto, debes re-escribir el registro para Puffball.

* Arreglar sólo el registro erróneo con la sentencia UPDATE:

```
mysql> UPDATE mascota SET nacimiento="1989-08-31" WHERE nombre="Bowser";
```

Como se muestra más arriba, es fácil recuperar el cuerpo de una data. Pero típicamente no querrás hacer eso, en particular cuando la tabla sea muy larga. Generalmente, estarás más interesado en responder a una pregunta en particular, en cuyo caso deberás especificar algunas restricciones en la información que deseas. Veamos algunas consultas de selección en términos de

preguntas sobre tus mascotas que se deben responder.

Seleccionando filas en particular
==========
Si quieres verificar el cambio que has realizado a la fecha de nacimiento de Bowser,selecciona el registro de Bowser de la siguiente forma:

```
mysql> SELECT * FROM mascota WHERE nombre="Bowser";
+--------+-------------+---------+------+------------+------------+
| nombre | propietario | especie | sexo | nacimiento | muerte     |
+--------+-------------+---------+------+------------+------------+
| Bowser |  Diane      | perro   | m    | 1989-08-31 | 1995-07-29 |
+--------+-------------+---------+------+------------+------------+
```

La salida confirma que el año está correctamente registrado como 1989, no 1998.

```
mysql> SELECT * FROM mascota WHERE especie="perro" AND sexo="f";
+--------+-------------+---------+------+------------+--------+
| nombre | propietario | especie | sexo | nacimiento | muerte |
+--------+-------------+---------+------+------------+--------+
| Buffy  |  Harold     | perro   | f    | 1989-05-13 | NULL   |
+--------+-------------+---------+------+------------+--------+
```

```
mysql> SELECT * FROM mascota WHERE (especie="gato" AND sexo="m")
    -> OR (especie="perro" AND sexo="f");
+--------+-------------+---------+------+------------+--------+
| nombre | propietario | especie | sexo | nacimiento | muerte |
+--------+-------------+---------+------+------------+--------+
| Claws  |  Gwen       | gato    | m    | 1994-03-17 | NULL   |
| Buffy  |  Harold     | perro   | f    | 1989-05-13 | NULL   |
+--------+-------------+---------+------+------------+--------+
```

Seleccionando columnas en particular
==========
Si no quieres ver filas completas de tu tabla, simplemente nombra las columnas en las cuales estás interesado, separadas por comas. Por ejemplo, si quieres saber cuándo nacieron tus animales, selecciona las columnas nombre y nacimiento:

```
mysql> SELECT nombre, nacimiento FROM mascota;
+----------+------------+
| nombre   | nacimiento |
+----------+------------+
| Bluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Buffy    | 1989-05-13 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Chirpy   | 1998-09-11 |
| Whistler | 2000-12-09 |
| Slim     | 1996-04-29 |
| Puffball | 1999-03-30 |
+----------+------------+
```

```
mysql> SELECT nombre, especie, nacimiento FROM mascota
    -> WHERE especie = "perro" OR especie = "gato";
+--------+---------+------------+
| nombre | especie | nacimiento |
+--------+---------+------------+
| Bluffy | gato    | 1993-02-04 |
| Claws  | gato    | 1994-03-17 |
| Buffy  | perro   | 1989-05-13 |
| Fang   | perro   | 1990-08-27 |
| Bowser | perro   | 1989-08-31 |
+--------+---------+------------+
```

Aquí mostramos las fechas de nacimiento de los animales, ordenadas por fecha:

```
mysql> SELECT nombre, nacimiento FROM mascota ORDER BY nacimiento;
+----------+------------+
| nombre   | nacimiento |
+----------+------------+
| Buffy    | 1989-05-13 |
| Bowser   | 1989-08-31 |
| Fang     | 1990-08-27 |
| Bluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Slim     | 1996-04-29 |
| Whistler | 2000-12-09 |
| Chirpy   | 1998-09-11 |
| Puffball | 1999-03-30 |
+----------+------------+
```

(Dado el tamaño maximo y formato del trabajo esto me impidio colocar mas codigo. De hecho existe muchas mas opciones las cuales se veran en la presentacion de este trabajo)

Para ordenar de forma inversa, añade la palabra reservada DESC (descendente) al nombre de la columna por la que estás ordenando:

```
mysql> SELECT nombre, nacimiento FROM mascota ORDER BY nacimiento DESC;
+----------+------------+
| nombre   | nacimiento |
+----------+------------+
| Puffball | 1999-03-30 |
| Chirpy   | 1998-09-11 |
| Whistler | 2000-12-09 |
| Slim     | 1996-04-29 |
| Claws    | 1994-03-17 |
| Bluffy   | 1993-02-04 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Buffy    | 1989-05-13 |
+----------+------------+
```

Puedes ordenar por múltiples columnas. Por ejemplo, para ordenar por tipo de animal, después por fecha de nacimiento dentro del mismo tipo de animal estando los animales más jóvenes primero, usa la siguiente consulta:

```
mysql> SELECT nombre, especie, nacimiento FROM mascota ORDER BY especie, nacimiento
DESC;
+----------+-----------+------------+
| nombre   | especie   | nacimiento |
+----------+-----------+------------+
| Claws    | gato      | 1994-03-17 |
| Bluffy   | gato      | 1993-02-04 |
| Puffball | hamster   | 1999-03-30 |
| Chirpy   | pájaro    | 1998-09-11 |
| Whistler | pájaro    | 2000-12-09 |
| Fang     | perro     | 1990-08-27 |
| Bowser   | perro     | 1989-08-31 |
| Buffy    | perro     | 1989-05-13 |
| Slim     | serpiente | 1996-04-29 |
+----------+-----------+------------+
```

Observa que la palabra reservada DESC se aplica sólo al nombre de columna que preceda a la palabra reservada (nacimiento); los valores especie siguen siendo ordenados en forma ascendente.