Capítulo 3 Tutorial

Tabla de contenido

Este capítulo proporciona una introducción tutorial a MySQL mostrando cómo usar el programa cliente mysql para crear y utilizar una base de datos sencilla. mysql (a veces denominado "monitor de terminal" o simplemente "Monitor") es un programa interactivo que le permite conectarse a un servidor MySQL, ejecutar consultas y ver Los resultados. mysql también se puede usar en modo por lotes: usted coloca sus consultas en un archivo de antemano, luego dice mysql para ejecutar el contenido del archivo. Ambas formas de usar mysql se tratan aquí.

Para ver una lista de opciones proporcionadas por mysql , invocalo con la opción --help :

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

Este capítulo asume que mysql está instalado en su máquina y que hay un servidor MySQL disponible al que puede conectarse. Si esto no es cierto, comuníquese con su administrador de MySQL. (Si *eres* el administrador, debe consultar las partes relevantes de este manual, como el Capítulo 5, *MySQL Server Administración* .)

Este capítulo describe todo el proceso de configuración y uso de una base de datos. Si solo estas interesado al acceder a una base de datos existente, es posible que desee omitir las secciones que describen cómo crear base de datos y las tablas que contiene.

Debido a que este capítulo es de naturaleza tutorial, se omiten necesariamente muchos detalles. Consultar el relevante secciones del manual para obtener más información sobre los temas que se tratan aquí.

3.1 Conexión y desconexión del servidor

Para conectarse al servidor, normalmente deberá proporcionar un nombre de usuario de MySQL cuando invoque mysql y, muy probablemente, una contraseña. Si el servidor se ejecuta en una máquina distinta a aquella en la que inició sesión,

Página 2

Ingresar consultas

También es necesario especificar un nombre de host. Póngase en contacto con su administrador para averiguar qué parámetros de conexión debe usar para conectarse (es decir, qué host, nombre de usuario y contraseña usar). Una vez que sepa el parámetros, debería poder conectarse así:

```
shell> mysql -h host -u usuario -p
Ingrese contraseña: ********
```

host y *usuario* representan el nombre de host donde se ejecuta su servidor MySQL y el nombre de usuario de su Cuenta MySQL. Sustituya los valores apropiados para su configuración. El ******** representa su contraseña; introdúzcalo cuando mysql muestre el mensaje Enter password : .

Si eso funciona, debería ver información introductoria seguida de un mensaje de mysql> :

```
shell> mysql -h host -u usuario -p
Ingrese contraseña: ********
Bienvenido al monitor MySQL. Los comandos terminan con; o \ g.
Su ID de conexión MySQL es 25338 para la versión del servidor: 5.7.33-estándar

Escriba 'ayuda'; o '\ h' para obtener ayuda. Escriba '\ c' para borrar el búfer.

mysql>
```

El indicador mysql> le dice que mysql está listo para que ingrese sentencias SQL.

Si está iniciando sesión en la misma máquina en la que se está ejecutando MySQL, puede omitir el host y simplemente utilice lo siguiente:

```
shell> mysql -u usuario -p
```

Si, cuando intenta iniciar sesión, recibe un mensaje de error como ERROR 2002 (HY000): No se puede conectarse al servidor MySQL local a través del socket '/tmp/mysql.sock' (2) , significa que el daemon del servidor MySQL (Unix) o el servicio (Windows) no se está ejecutando. Consulte al administrador o consulte el sección del Capítulo 2, *Instalación y actualización de MySQL* que sea apropiada para su sistema operativo.

Para obtener ayuda con otros problemas que se encuentran a menudo al intentar iniciar sesión, consulte la Sección B.3.2, “Errores comunes Al usar programas MySQL ” .

Algunas instalaciones de MySQL permiten a los usuarios conectarse como usuario anónimo (sin nombre) al servidor ejecutándose en el host local. Si este es el caso en su máquina, debería poder conectarse a ese servidor invocando mysql sin ninguna opción:

```
shell> mysql
```

Una vez que se haya conectado correctamente, puede desconectarse en cualquier momento escribiendo SALIR (o \ q) en mysql> rápido:

```
mysql> SALIR
Adiós
```

En Unix, también puede desconectarse presionando Control + D.

La mayoría de los ejemplos de las siguientes secciones asumen que está conectado al servidor. Indican esto por el indicador mysql> .

3.2 Introducción de consultas

Asegúrese de estar conectado al servidor, como se discutió en la sección anterior. Hacerlo no implica selecciona cualquier base de datos con la que trabajar, pero eso está bien. En este punto, es más importante averiguar un poco sobre cómo emitir consultas que para saltar directamente en la creación de tablas, cargar datos en ellas y recuperar datos

Página 3

Ingresar consultas

de ellos. Esta sección describe los principios básicos para ingresar consultas, utilizando varias consultas que puede intente familiarizarse con el funcionamiento de mysql .

Aquí hay una consulta simple que le pide al servidor que le diga su número de versión y la fecha actual. Escríbalo como que se muestra aquí siguiendo el indicador mysql> y presione Entrar:

```
mysql> SELECT VERSION (), CURRENT_DATE;
+ -------------- + -------------- +
| VERSIÓN () | CURRENT_DATE |
+ -------------- + -------------- +
| 5.7.1-m4-log | 2012-12-25 |
+ -------------- + -------------- +
1 fila en conjunto (0.01 seg)
mysql>
```

Esta consulta ilustra varias cosas sobre mysql :

- Una consulta normalmente consta de una instrucción SQL seguida de un punto y coma. (Hay algunas excepciones donde se puede omitir un punto y coma. QUIT , mencionado anteriormente, es uno de ellos. Llegaremos a otros más tarde.)

- Cuando emite una consulta, mysql la envía al servidor para su ejecución y muestra los resultados, luego imprime otro indicador de mysql> para indicar que está listo para otra consulta.

- mysql muestra la salida de la consulta en forma tabular (filas y columnas). La primera fila contiene etiquetas para las columnas. Las siguientes filas son los resultados de la consulta. Normalmente, las etiquetas de columna son los nombres de columnas que obtiene de las tablas de la base de datos. Si está recuperando el valor de una expresión en lugar de un columna de la tabla (como en el ejemplo que se acaba de mostrar), mysql etiqueta la columna usando la expresión misma.

- mysql muestra cuántas filas se devolvieron y cuánto tiempo tardó en ejecutarse la consulta, lo que le da una idea aproximada del rendimiento del servidor. Estos valores son imprecisos porque representan la hora del reloj de pared. (no la CPU ni el tiempo de la máquina) y porque se ven afectados por factores como la carga del servidor y la red latencia. (Para mayor brevedad, la línea "filas en conjunto" a veces no se muestra en los ejemplos restantes de este capítulo.)

Las palabras clave se pueden introducir en cualquier tipo de letra. Las siguientes consultas son equivalentes:

```
mysql> SELECT VERSION (), CURRENT_DATE;
mysql> seleccionar versión (), fecha_actual;
mysql> SeLeCt vErSiOn (), FECHA_Actual;
```

Aquí hay otra consulta. Demuestra que puede usar mysql como una calculadora simple:

```
mysql> SELECCIONAR SIN (PI () / 4), (4 + 1) * 5;
+ ------------------ + --------- +
| SIN (PI () / 4) | (4 + 1) * 5 |
+ ------------------ + --------- +
| 0,70710678118655 | 25 |
+ ------------------ + --------- +
1 fila en conjunto (0.02 seg)
```

Las consultas mostradas hasta ahora han sido declaraciones de una sola línea relativamente breves. Incluso puede ingresar varios declaraciones en una sola línea. Simplemente termine cada uno con un punto y coma:

```
mysql> SELECCIONAR VERSIÓN (); SELECCIONAR AHORA ();
+ ------------------ +
| VERSIÓN() |
+ ------------------ +
| 5.7.10-ndb-7.5.1 |
+ ------------------ +
1 fila en conjunto (0,00 seg)

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


Página 4

```
| AHORA()                                |
+ ---------------------- +
| 2016-01-29 18:02:55 |
+ ---------------------- +
1 fila en conjunto (0,00 seg)
```

No es necesario que una consulta se proporcione en una sola línea, por lo que las consultas largas que requieren varias líneas no son un problema. problema. mysql determina dónde termina su declaración buscando el punto y coma final, no por buscando el final de la línea de entrada. (En otras palabras, mysql acepta entrada de formato libre: recopila líneas de entrada pero no los ejecuta hasta que ve el punto y coma).

Aquí hay una declaración simple de varias líneas:

```
mysql> SELECCIONAR
    -> USUARIO ()
    -> ,
    -> CURRENT_DATE;
+ --------------- + -------------- +
| USUARIO()                       | CURRENT_DATE |
+ --------------- + -------------- +
| jon @ localhost | 2010-08-06 |
+ --------------- + -------------- +
```

En este ejemplo, observe cómo el indicador cambia de mysql> a -> después de ingresar la primera línea de un consulta de varias líneas. Así es como mysql indica que aún no ha visto una declaración completa y es esperando el resto. El mensaje es tu amigo, porque proporciona comentarios valiosos. Si usas eso comentarios, siempre puede estar al tanto de lo que está esperando mysql .

Si decide que no desea ejecutar una consulta que está en proceso de ingresar, cancélela escribiendo \ c :

```
mysql> SELECCIONAR
    -> USUARIO ()
    -> \ c
mysql>
```

Aquí, también, observe el mensaje. Vuelve a mysql> después de escribir \ c , proporcionando comentarios para indicar que mysql está listo para una nueva consulta.

La siguiente tabla muestra cada una de las indicaciones que puede ver y resume lo que significan indica que mysql está en.

Significado rápido	
mysql>	Listo para una nueva consulta
->	Esperando la siguiente línea de consulta de varias líneas
'>	Esperando la siguiente línea, esperando la finalización de una cadena que comienza con una comilla simple (')
">	Esperando la siguiente línea, esperando la finalización de una cadena que comenzó con un doble cita (")
`>	Esperando la siguiente línea, esperando la finalización de un identificador que comenzó con una tilde. (`)
/ *>	Esperando la siguiente línea, esperando la finalización de un comentario que comienza con / *

Las declaraciones de varias líneas suelen producirse por accidente cuando se intenta realizar una consulta en una sola línea, pero olvídese del punto y coma final. En este caso, mysql espera más información:

```
mysql> SELECCIONAR USUARIO ()
    ->
```


Página 5

Si esto le sucede a usted (cree que ha ingresado una declaración pero la única respuesta es un -> mensaje), la mayoría probablemente mysql está esperando el punto y coma. Si no nota lo que le dice el mensaje, puede sentarse allí por un tiempo antes de darse cuenta de lo que necesita hacer. Introduzca un punto y coma para completar el enunciado y mysql lo ejecuta:

```
mysql> SELECCIONAR USUARIO ()
    -> ;
+ --------------- +
| USUARIO() |
+ --------------- +
| jon @ localhost |
+ --------------- +
```

Las indicaciones '> y "> ocurren durante la recolección de cadenas (otra forma de decir que MySQL está esperando para completar una cadena). En MySQL, puede escribir cadenas delimitadas por ' o " caracteres (por ejemplo, 'hola' o "adiós"), y mysql le permite ingresar cadenas que abarcan varias líneas. Cuando ve un mensaje '> o "> , significa que ha ingresado una línea que contiene una cadena que comienza con un ' o " carácter de cita, pero aún no ha introducido la cita coincidente que termina la cadena. Esto a menudo indica que ha omitido inadvertidamente un carácter de cita. Por ejemplo:

```
mysql> SELECT * FROM my_table DONDE nombre = 'Smith Y edad <30;
    '>
```

Si ingresa esta instrucción SELECT , luego presione **Enter** y espere el resultado, no sucede nada. En lugar de preguntarse por qué esta consulta tarda tanto, observe la pista proporcionada por el indicador '> . Te dice que mysql espera ver el resto de una cadena sin terminar. (¿Ves el error en la declaración? La cadena A Smith le falta la segunda comilla simple).

En este punto, ¿qué haces? Lo más sencillo es cancelar la consulta. Sin embargo, no puede simplemente escribir \ c en este caso, porque mysql lo interpreta como parte de la cadena que está recopilando. En cambio, ingrese el cierre carácter de cita (para que mysql sepa que ha terminado la cadena), luego escriba \ c :

```
mysql> SELECT * FROM my_table DONDE nombre = 'Smith Y edad <30;
    '> ' \ c
mysql>
```

El indicador vuelve a cambiar a mysql> , lo que indica que mysql está listo para una nueva consulta.

El indicador `> es similar al indicador '> y "> , pero indica que ha comenzado pero no ha completado un identificador con comillas invertidas.

Es importante saber lo que el '> , "> , y `> significado de estos indicadores, ya que si por error se introduce una cadena sin terminar, cualquier otra línea que escriba parece ser ignorada por mysql, incluida una línea que contiene SALIR . Esto puede resultar bastante confuso, especialmente si no sabe que necesita suministrar el terminal cotizar antes de que pueda cancelar la consulta actual.

Nota

Las declaraciones de varias líneas a partir de este punto se escriben sin la secundaria (-> u otros) mensajes, para que sea más fácil copiar y pegar las declaraciones para probar usted mismo.

3.3 Crear y usar una base de datos

Una vez que sepa cómo ingresar declaraciones SQL, estará listo para acceder a una base de datos.

Suponga que tiene varias mascotas en su casa (su colección de animales) y le gustaría realizar un seguimiento de varios tipos de información sobre ellos. Puede hacerlo creando tablas para almacenar sus datos y cargar

ellos con la información deseada. Luego, puede responder diferentes tipos de preguntas sobre sus animales al recuperar datos de las tablas. Esta sección le muestra cómo realizar las siguientes operaciones:

• Crea una base de datos

• Crea una tabla

• Cargar datos en la tabla

• Recuperar datos de la tabla de varias formas

• Utilice varias tablas

La base de datos de la colección de animales es simple (deliberadamente), pero no es difícil pensar en situaciones del mundo real.
en el que podría utilizarse un tipo similar de base de datos. Por ejemplo, una base de datos como esta podría ser utilizada por
un agricultor para realizar un seguimiento del ganado, o por un veterinario para realizar un seguimiento de los registros de los pacientes. Una colección de animales
La distribución que contiene algunas de las consultas y datos de muestra utilizados en las siguientes secciones se puede
obtenido del sitio web de MySQL. Está disponible en formato de archivo tar comprimido y Zip en https: //
dev.mysql.com/doc/ .

Utilice la instrucción SHOW para averiguar qué bases de datos existen actualmente en el servidor:

```
mysql> MOSTRAR BASES DE DATOS;
+ ---------- +
| Base de datos |
+ ---------- +
| mysql |
| prueba |
| tmp |
+ ---------- +
```

La base de datos mysql describe los privilegios de acceso de los usuarios. La base de datos de prueba a menudo está disponible como
espacio de trabajo para que los usuarios prueben cosas.

La lista de bases de datos que muestra la declaración puede ser diferente en su máquina; MOSTRAR BASES DE DATOS
no muestra las bases de datos para las que no tiene privilegios si no tiene MOSTRAR BASES DE DATOS
privilegio. Consulte la Sección 13.7.5.14, “Declaración SHOW DATABASES” .

Si la base de datos de prueba existe, intente acceder a ella:

```
mysql> USE prueba
Base de datos cambiada
```

USE , como SALIR , no requiere un punto y coma. (Puede terminar tales declaraciones con un punto y coma si
me gusta; no hace daño.) La instrucción USE también es especial de otra manera: debe darse en una sola línea.

Puede usar la base de datos de prueba (si tiene acceso a ella) para los ejemplos que siguen, pero
create en esa base de datos puede ser eliminado por cualquier otra persona que tenga acceso a ella. Por esta razón, debe
probablemente pida permiso a su administrador de MySQL para utilizar una base de datos propia. Suponga que usted
quiero llamar el tuyo zoológico . El administrador debe ejecutar una declaración como esta:

```
mysql> OTORGAR TODO EL MENagerie. * TO 'your_mysql_name' @ 'your_client_host';
```

donde your_mysql_name es el nombre de usuario de MySQL que se le asignó y your_client_host es el host
desde el que se conecta al servidor.

3.3.1 Crear y seleccionar una base de datos

Si el administrador crea su base de datos por usted cuando configura sus permisos, puede comenzar a usar
eso. De lo contrario, debe crearlo usted mismo:

Página 7

Crear una tabla

```
mysql> CREATE DATABASE menagerie;
```

En Unix, los nombres de las bases de datos distinguen entre mayúsculas y minúsculas (a diferencia de las palabras clave SQL), por lo que siempre debe consultar
su base de datos como colección de animales , no como colección de animales , MENAGERIE o alguna otra variante. Esto también es cierto
para los nombres de las tablas. (En Windows, esta restricción no se aplica, aunque debe consultar las bases de datos
y tablas que utilizan el mismo tipo de letra en una consulta determinada. Sin embargo, por diversas razones, el
La mejor práctica recomendada es utilizar siempre el mismo tipo de letra que se utilizó cuando se
creado.)

Nota

Si recibe un error como ERROR 1044 (42000): Acceso denegado para el usuario 'micah' @ 'localhost' a la base de datos 'menagerie' al intentar crear una base de datos, esto significa que su cuenta de usuario no tiene los privilegios para hacerlo. Discuta esto con el administrador o consulte la Sección 6.2, “Acceso Control y Gestión de Cuentas ” .

La creación de una base de datos no la selecciona para su uso; debes hacerlo explícitamente. Para hacer de la colección de animales la corriente base de datos, use esta declaración:

```
mysql> USE menagerie
Base de datos cambiada
```

Su base de datos debe crearse solo una vez, pero debe seleccionarla para usarla cada vez que inicie un mysql sesión. Puede hacer esto emitiendo una instrucción USE como se muestra en el ejemplo. Alternativamente, puede seleccionar la base de datos en la línea de comando cuando invoca mysql . Simplemente especifique su nombre después de cualquier conexión parámetros que podría necesitar proporcionar. Por ejemplo:

```
shell> mysql -h host -u usuario -p menagerie
Ingrese contraseña: ********
```

Importante

menagerie en el comando que se acaba de mostrar **no** es su contraseña. Si quieres suministrar su contraseña en la línea de comando después de la opción -p , debe hacerlo sin espacio intermedio (por ejemplo, como -p *contraseña* , no como -p *contraseña*). Sin embargo, No se recomienda poner su contraseña en la línea de comando, porque hacerlo lo expone a espionaje por parte de otros usuarios conectados a su máquina.

Nota

Puede ver en cualquier momento qué base de datos está seleccionada actualmente usando SELECT BASE DE DATOS () .

3.3.2 Crear una tabla

Crear la base de datos es la parte fácil, pero en este punto está vacía, como le dice SHOW TABLES :

```
mysql> MOSTRAR TABLAS;
Conjunto vacío (0,00 seg)
```

La parte más difícil es decidir cuál debe ser la estructura de su base de datos: qué tablas necesita y qué las columnas deben estar en cada uno de ellos.

Quiere una tabla que contenga un registro para cada una de sus mascotas. Esto se puede llamar la mesa de mascotas y debe contener, como mínimo, el nombre de cada animal. Porque el nombre en sí mismo no es muy interesante, la tabla debe contener otra información. Por ejemplo, si más de una persona en su

Página 8

familia tiene mascotas, es posible que desee enumerar el dueño de cada animal. Es posible que también desee grabar algunos información descriptiva como especie y sexo.

¿Qué tal la edad? Eso puede ser interesante, pero no es bueno almacenarlo en una base de datos. Cambios de edad a medida que pasa el tiempo, lo que significa que tendrá que actualizar sus registros con frecuencia. En cambio, es mejor almacenar un fijo valor como la fecha de nacimiento. Luego, siempre que necesite la edad, puede calcularla como la diferencia entre la fecha actual y la fecha de nacimiento. MySQL proporciona funciones para hacer aritmética de fechas, por lo que esto no es difícil. El almacenamiento de la fecha de nacimiento en lugar de la edad también tiene otras ventajas:

- Puede utilizar la base de datos para tareas tales como generar recordatorios para los próximos cumpleaños de mascotas. (Si tu cree que este tipo de consulta es algo tonto, tenga en cuenta que es la misma pregunta que podría hacer en el contexto de una base de datos empresarial para identificar a los clientes a los que necesita enviar felicitaciones de cumpleaños en la actualidad semana o mes, para ese toque personal asistido por computadora).

- Puede calcular la edad en relación con fechas distintas a la fecha actual. Por ejemplo, si almacena la muerte fecha en la base de datos, puede calcular fácilmente la edad de una mascota cuando murió.

Probablemente pueda pensar en otros tipos de información que serían útiles en la tabla de mascotas , pero las identificadas hasta ahora son suficientes: nombre, propietario, especie, sexo, nacimiento y muerte.

Use una declaración CREATE TABLE para especificar el diseño de su tabla:

```
mysql> CREATE TABLE pet (nombre VARCHAR (20), propietario VARCHAR (20),
           especie VARCHAR (20), sexo CHAR (1), FECHA de nacimiento, FECHA de muerte);
```

VARCHAR es una buena opción para las columnas de nombre , propietario y especie porque los valores de las columnas varían en longitud. No es necesario que todas las longitudes en esas definiciones de columna sean iguales, y no es necesario que sean 20 . Usted puede normalmente elija cualquier longitud de 1 a 65535 , la que le parezca más razonable. Si haces un pobre elección y resulta más tarde que necesita un campo más largo, MySQL proporciona una instrucción ALTER TABLE .

Se pueden elegir varios tipos de valores para representar el sexo en los registros de animales, como 'm' y 'f' , o quizás 'masculino' y 'femenino' . Es más sencillo utilizar los caracteres individuales 'm' y 'f' .

El uso del tipo de datos DATE para las columnas de nacimiento y muerte es una opción bastante obvia.

Una vez que haya creado una tabla, MOSTRAR TABLAS debería producir algún resultado:

```
mysql> MOSTRAR TABLAS;
+ --------------------- +
| Mesas en casa de fieras |
+ --------------------- +
| mascota                 |
+ --------------------- +
```

Para verificar que su tabla se creó de la manera que esperaba, use una declaración DESCRIBE :

```
mysql> DESCRIBE pet;
+ --------- + ------------- + ------ + ----- + --------- + - ----- +
| Campo | Tipo                | Nulo | Clave | Por defecto | Extra |
+ --------- + ------------- + ------ + ----- + --------- + - ----- +
| nombre | varchar (20) | SI | | NULL |                            |
| propietario | varchar (20) | SI | | NULL |                       |
| especies | varchar (20) | SI | | NULL |                          |
| sexo | char (1) | SI | | NULL |                                |
| nacimiento | fecha      | SI | | NULL |                          |
| muerte | fecha          | SI | | NULL |                          |
+ --------- + ------------- + ------ + ----- + --------- + - ----- +
```

Puede usar DESCRIBE en cualquier momento, por ejemplo, si olvida los nombres de las columnas en su tabla o qué tipos que tienen.

Para obtener más información sobre los tipos de datos de MySQL, consulte el Capítulo 11, *Tipos de datos* .

3.3.3 Carga de datos en una tabla

Después de crear su tabla, debe completarla. Las sentencias LOAD DATA e INSERT son útiles para esta.

Suponga que los registros de su mascota se pueden describir como se muestra aquí. (Observe que MySQL espera fechas en Formato ' *AAAA-MM-DD* ' ; esto puede diferir de lo que está acostumbrado.)

nombre	propietario	especie	sexo	nacimiento	muerte
Mullido	Harold	gato	F	1993-02-04	
Garras	Gwen	gato	metro	1994-03-17	
Buffy	Harold	perro	F	13/05/1989	
Colmillo	Benny	perro	metro	1990-08-27	
Bowser Diane		perro	metro	1979-08-31	1995-07-29
Alegre	Gwen	pájaro	F	1998-09-11	
Whistler Gwen		pájaro		1997-12-09	

Delgado Benny serpiente metro 1996-04-29

Debido a que está comenzando con una tabla vacía, una manera fácil de completarla es crear un archivo de texto que contiene una fila para cada uno de sus animales, luego cargue el contenido del archivo en la tabla con un solo declaración.

Puede crear un archivo de texto pet.txt que contenga un registro por línea, con valores separados por tabulaciones, y dado en el orden en que se enumeraron las columnas en la instrucción CREATE TABLE . Para valores perdidos (como sexos desconocidos o fechas de muerte para animales que aún viven), puede usar valores NULL . A Represéntelos en su archivo de texto, use \ N (barra invertida, N mayúscula). Por ejemplo, el récord de Whistler the bird se vería así (donde el espacio en blanco entre los valores es un carácter de tabulación única):

Whistler Gwen bird \ N 1997-12-09 \ N

Para cargar el archivo de texto pet.txt en la tabla de mascotas , use esta declaración:

```
mysql> CARGAR INFORMACIÓN LOCAL DE DATOS '/path/pet.txt' EN LA TABLA pet;
```

Si creó el archivo en Windows con un editor que usa \ r \ n como terminador de línea, debe usar este declaración en su lugar:

```
mysql> CARGAR DATOS INFILE LOCAL '/path/pet.txt' EN LA TABLA pet
    -> LÍNEAS TERMINADAS POR '\ r \ n';
```

(En una máquina Apple que ejecuta macOS, es probable que desee utilizar LINES TERMINATED BY '\ r').

Puede especificar el separador de valor de columna y el marcador de final de línea explícitamente en la instrucción LOAD DATA si lo desea, pero los valores predeterminados son tabulación y salto de línea. Estos son suficientes para que la declaración lea el archivo. pet.txt correctamente.

Si la declaración falla, es probable que su instalación de MySQL no tenga la capacidad de archivo local habilitada por defecto. Consulte la Sección 6.1.6, “Consideraciones de seguridad para LOAD DATA LOCAL” , para obtener información sobre cómo cambia esto.

Cuando desee agregar nuevos registros uno a la vez, la instrucción INSERT es útil. En su forma más simple, usted proporciona valores para cada columna, en el orden en que se enumeraron las columnas en CREAR TABLA

Página 10

declaración. Supongamos que Diane tiene un nuevo hámster llamado "Puffball". Puede agregar un nuevo registro usando una declaración INSERT como esta:

```
mysql> INSERTAR EN mascota
    -> VALORES ('Puffball', 'Diane', 'hámster', 'f', '1999-03-30', NULL);
```

Los valores de cadena y fecha se especifican aquí como cadenas entre comillas. Además, con INSERT , puede insertar NULL directamente para representar un valor faltante. No usa \ N como lo hace con LOAD DATA .

En este ejemplo, debería poder ver que habría que escribir mucho más para cargar su registros utilizando inicialmente varias instrucciones INSERT en lugar de una sola instrucción LOAD DATA .

3.3.4 Recuperar información de una tabla

La instrucción SELECT se usa para extraer información de una tabla. La forma general de la declaración es:

```
SELECT what_to_select
FROM which_table
DONDE condiciones_para_ satisfacer ;
```

what_to_select indica lo que desea ver. Puede ser una lista de columnas o * para indicar "todas columnas ". *which_table* indica la tabla de la que desea recuperar datos. La cláusula WHERE es opcional. Si está presente, *condition_to_satisfy* especifica una o más condiciones que las filas deben satisfacer para calificar para la recuperación.

3.3.4.1 Seleccionar todos los datos

La forma más simple de SELECT recupera todo de una tabla:

```
mysql> SELECT * FROM pet;
+ ---------- + -------- + --------- + ------ + ------------ + ------------ +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ ---------- + -------- + --------- + ------ + ------------ + ------------ +
| Fluffy | Harold | gato | f | 1993-02-04 | NULO |
| Garras | Gwen | gato | m | 1994-03-17 | NULO |
| Buffy | Harold | perro | f | 1989-05-13 | NULO |
| Fang | Benny | perro | m | 1990-08-27 | NULO |
| Bowser | Diane | perro | m | 1979-08-31 | 1995-07-29 |
| Chirpy | Gwen | pájaro | f | 1998-09-11 | NULO |
| Whistler | Gwen | pájaro | NULL | 1997-12-09 | NULO |
| Slim | Benny | serpiente | m | 1996-04-29 | NULO |
| Puffball | Diane | hámster f | 1999-03-30 | NULO |
+ ---------- + -------- + --------- + ------ + ------------ + ------------ +
```

Esta forma de SELECCIONAR es útil si desea revisar toda la tabla, por ejemplo, después de haberla cargado con su conjunto de datos inicial. Por ejemplo, puede pensar que la fecha de nacimiento de Bowser no parece Muy bien. Consultando sus documentos genealógicos originales, encuentra que el año de nacimiento correcto debería ser 1989, no 1979.

Hay al menos dos formas de solucionar este problema:

• Edite el archivo pet.txt para corregir el error, luego vacíe la tabla y vuelva a cargarlo usando DELETE y LOAD DATOS :

```
mysql> BORRAR DE mascota;
mysql> CARGAR INFORMACIÓN LOCAL DE DATOS '/path/pet.txt' EN LA TABLA pet;
```

Sin embargo, si hace esto, también debe volver a ingresar el registro de Puffball.

• Corrija solo el registro erróneo con una instrucción UPDATE :

Página 11


```
mysql> ACTUALIZAR CONJUNTO de mascotas nacimiento = '1989-08-31' DONDE nombre = 'Bowser';
```

La ACTUALIZACIÓN cambia solo el registro en cuestión y no requiere que vuelva a cargar la tabla.

3.3.4.2 Seleccionar filas particulares

Como se muestra en la sección anterior, es fácil recuperar una tabla completa. Simplemente omita la cláusula WHERE de la instrucción SELECT . Pero normalmente no desea ver la tabla completa, especialmente cuando se vuelve grande. En cambio, normalmente está más interesado en responder una pregunta en particular, en cuyo caso especifica algunas restricciones sobre la información que desea. Veamos algunas consultas de selección en términos de preguntas. sobre sus mascotas que responden.

Puede seleccionar solo filas particulares de su tabla. Por ejemplo, si desea verificar el cambio que hecho a la fecha de nacimiento de Bowser, seleccione el registro de Bowser así:

```
mysql> SELECT * FROM pet WHERE name = 'Bowser';
+ -------- + ------- + --------- + ------ + ------------ + - ---------- +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ -------- + ------- + --------- + ------ + ------------ + - ---------- +
| Bowser | Diane | perro | m | 1989-08-31 | 1995-07-29 |
+ -------- + ------- + --------- + ------ + ------------ + - ---------- +
```

El resultado confirma que el año se registró correctamente como 1989, no como 1979.

Las comparaciones de cadenas normalmente no distinguen entre mayúsculas y minúsculas, por lo que puede especificar el nombre como 'bowser' , 'BOWSER' , Etcétera. El resultado de la consulta es el mismo.

Puede especificar condiciones en cualquier columna, no solo en el nombre . Por ejemplo, si quiere saber qué animales nacieron durante o después de 1998, pruebe la columna de nacimiento :

```
mysql> SELECT * FROM pet WHERE birth> = '1998-1-1';
+ ---------- + ------- + --------- + ------ + ------------ + ------- +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ ---------- + ------- + --------- + ------ + ------------ + ------- +
```

```
| Chirpy | Gwen | pájaro | f | 1998-09-11 | NULL |
| Puffball | Diane | hámster f | 1999-03-30 | NULL |
+ ---------- + ------- + --------- + ------ + ------------ + ------- +
```

Puede combinar condiciones, por ejemplo, para localizar perras:

```
mysql> SELECCIONAR * DE mascota DONDE especie = 'perro' Y sexo = 'f';
+ ------- + -------- + --------- + ------ + ------------ + - ----- +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ ------- + -------- + --------- + ------ + ------------ + - ----- +
| Buffy | Harold | perro | f | 1989-05-13 | NULL |
+ ------- + -------- + --------- + ------ + ------------ + - ----- +
```

La consulta anterior utiliza el operador lógico AND . También hay un operador OR :

```
mysql> SELECCIONAR * DE mascota DONDE especie = 'serpiente' O especie = 'pájaro';
+ ---------- + ------- + --------- + ------ + ------------ + ------- +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ ---------- + ------- + --------- + ------ + ------------ + ------- +
| Chirpy | Gwen | pájaro | f | 1998-09-11 | NULL |
| Whistler | Gwen | pájaro | NULL | 1997-12-09 | NULL |
| Slim | Benny | serpiente | m | 1996-04-29 | NULL |
+ ---------- + ------- + --------- + ------ + ------------ + ------- +
```

AND y OR pueden estar entremezclados, aunque AND tiene mayor precedencia que OR . Si usa ambos operadores, Es una buena idea usar paréntesis para indicar explícitamente cómo deben agruparse las condiciones:

```
mysql> SELECT * FROM pet WHERE (especie = 'gato' Y sexo = 'm')
```


Pagina 12


```
            O (especie = 'perro' Y sexo = 'f');
+ ------- + -------- + --------- + ------ + ------------ + - ----- +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ ------- + -------- + --------- + ------ + ------------ + - ----- +
| Garras | Gwen | gato | m | 1994-03-17 | NULL |
| Buffy | Harold | perro | f | 1989-05-13 | NULL |
+ ------- + -------- + --------- + ------ + ------------ + - ----- +
```

3.3.4.3 Seleccionar columnas particulares

Si no desea ver filas enteras de su tabla, simplemente nombre las columnas en las que está interesado, separado por comas. Por ejemplo, si desea saber cuándo nacieron sus animales, seleccione el nombre y columnas de nacimiento :

```
mysql> SELECT nombre, nacimiento DE mascota;
+ ---------- + ------------ +
| nombre | nacimiento |
+ ---------- + ------------ +
| Fluffy | 1993-02-04 |
| Garras | 1994-03-17 |
| Buffy | 1989-05-13 |
| Fang | 1990-08-27 |
| Bowser | 1989-08-31 |
| Chirpy | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim | 1996-04-29 |
| Puffball | 1999-03-30 |
+ ---------- + ------------ +
```

Para saber quién tiene mascotas, utilice esta consulta:

```
mysql> SELECCIONAR propietario DE mascota;
+ -------- +
| propietario |
+ -------- +
| Harold |
| Gwen |
| Harold |
| Benny |
| Diane |
| Gwen |
| Gwen |
```

```
| Benny |
| Diane |
+ -------- +
```

Observe que la consulta simplemente recupera la columna del propietario de cada registro, y algunos de ellos aparecen más de una vez. Para minimizar la salida, recupere cada registro de salida único solo una vez agregando la palabra clave DISTINTO :

```
mysql> SELECCIONAR propietario DISTINTO DE mascota;
+ -------- +
| propietario |
+ -------- +
| Benny |
| Diane |
| Gwen |
| Harold |
+ -------- +
```

Puede utilizar una cláusula WHERE para combinar la selección de filas con la selección de columnas. Por ejemplo, para dar a luz fechas solo para perros y gatos, use esta consulta:

```
mysql> SELECCIONAR nombre, especie, nacimiento DE mascota
```

Página 13


```
            DONDE especie = 'perro' O especie = 'gato';
+ -------- + --------- + ------------ +
| nombre | especies | nacimiento |
+ -------- + --------- + ------------ +
| Fluffy | gato | 1993-02-04 |
| Garras | gato | 1994-03-17 |
| Buffy | perro | 1989-05-13 |
| Fang | perro | 1990-08-27 |
| Bowser | perro | 1989-08-31 |
+ -------- + --------- + ------------ +
```

3.3.4.4 Ordenar filas

Es posible que haya notado en los ejemplos anteriores que las filas de resultados se muestran sin ningún orden en particular. Eso A menudo es más fácil examinar el resultado de la consulta cuando las filas se ordenan de una manera significativa. Para ordenar un resultado, utilice una cláusula ORDER BY .

Aquí están los cumpleaños de los animales, ordenados por fecha:

```
mysql> SELECT nombre, nacimiento DE mascota ORDEN POR nacimiento;
+ ---------- + ------------ +
| nombre | nacimiento |
+ ---------- + ------------ +
| Buffy | 1989-05-13 |
| Bowser | 1989-08-31 |
| Fang | 1990-08-27 |
| Fluffy | 1993-02-04 |
| Garras | 1994-03-17 |
| Slim | 1996-04-29 |
| Whistler | 1997-12-09 |
| Chirpy | 1998-09-11 |
| Puffball | 1999-03-30 |
+ ---------- + ------------ +
```

En columnas de tipo de caracteres, la clasificación, como todas las demás operaciones de comparación, se realiza normalmente en un moda que no distingue entre mayúsculas y minúsculas. Esto significa que el orden no está definido para las columnas que son idénticas excepto por su caso. Puede forzar una ordenación que distinga entre mayúsculas y minúsculas para una columna utilizando BINARIO de esta manera: ORDER BY BINARY *col_name* .

El orden de clasificación predeterminado es ascendente, con los valores más pequeños primero. Para ordenar en orden inverso (descendente), agregue la palabra clave DESC al nombre de la columna por la que está ordenando:

```
mysql> SELECCIONAR nombre, nacimiento DESDE mascota ORDEN POR nacimiento DESC;
+ ---------- + ------------ +
| nombre | nacimiento |
+ ---------- + ------------ +
| Puffball | 1999-03-30 |
```

```
| Chirpy | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim | 1996-04-29 |
| Garras | 1994-03-17 |
| Fluffy | 1993-02-04 |
| Fang | 1990-08-27 |
| Bowser | 1989-08-31 |
| Buffy | 1989-05-13 |
+ ---------- + ------------ +
```

Puede ordenar en varias columnas y puede ordenar diferentes columnas en diferentes direcciones. Por ejemplo, para ordenar por tipo de animal en orden ascendente, luego por fecha de nacimiento dentro del tipo de animal en orden descendente (los animales más jóvenes primero), utilice la siguiente consulta:

```
mysql> SELECCIONAR nombre, especie, nacimiento DE mascota
           ORDENAR POR especie, nacimiento DESC;
+ ---------- + --------- + ------------ +
```


Página 14


```
| nombre | especies | nacimiento |
+ ---------- + --------- + ------------ +
| Chirpy | pájaro | 1998-09-11 |
| Whistler | pájaro | 1997-12-09 |
| Garras | gato | 1994-03-17 |
| Fluffy | gato | 1993-02-04 |
| Fang | perro | 1990-08-27 |
| Bowser | perro | 1989-08-31 |
| Buffy | perro | 1989-05-13 |
| Puffball | hámster 1999-03-30 |
| Slim | serpiente | 1996-04-29 |
+ ---------- + --------- + ------------ +
```

La palabra clave DESC se aplica solo al nombre de la columna inmediatamente anterior (nacimiento); no afecta el orden de clasificación de la columna de especies .

3.3.4.5 Cálculos de fecha

MySQL proporciona varias funciones que puede utilizar para realizar cálculos en fechas, por ejemplo, para calcular edades o extraer partes de fechas.

Para determinar cuántos años tiene cada una de sus mascotas, use la función TIMESTAMPDIFF () . Sus Los argumentos son la unidad en la que desea que se exprese el resultado, y las dos fechas para las que tomar la diferencia. La siguiente consulta muestra, para cada mascota, la fecha de nacimiento, la fecha actual y la edad en años. Se utiliza un *alias* (edad) para que la etiqueta de la columna de salida final sea más significativa.

```
mysql> SELECT nombre, nacimiento, CURDATE (),
           TIMESTAMPDIFF (YEAR, birth, CURDATE ()) AS age
           DE mascota;
+ ---------- + ------------ + ------------ + ------ +
| nombre | nacimiento | CURDATE () | edad |
+ ---------- + ------------ + ------------ + ------ +
| Fluffy | 1993-02-04 | 2003-08-19 | 10 |
| Garras | 1994-03-17 | 2003-08-19 | 9 |
| Buffy | 1989-05-13 | 2003-08-19 | 14 |
| Fang | 1990-08-27 | 2003-08-19 | 12 |
| Bowser | 1989-08-31 | 2003-08-19 | 13 |
| Chirpy | 1998-09-11 | 2003-08-19 | 4 |
| Whistler | 1997-12-09 | 2003-08-19 | 5 |
| Slim | 1996-04-29 | 2003-08-19 | 7 |
| Puffball | 1999-03-30 | 2003-08-19 | 4 |
+ ---------- + ------------ + ------------ + ------ +
```

La consulta funciona, pero el resultado podría escanearse más fácilmente si las filas se presentaran en algún orden. Esto se puede hacer agregando una cláusula ORDER BY name para ordenar la salida por nombre:

```
mysql> SELECT nombre, nacimiento, CURDATE (),
           TIMESTAMPDIFF (YEAR, birth, CURDATE ()) AS age
           DE mascota PEDIR POR nombre;
+ ---------- + ------------ + ------------ + ------ +
| nombre | nacimiento | CURDATE () | edad |
+ ---------- + ------------ + ------------ + ------ +
| Bowser | 1989-08-31 | 2003-08-19 | 13 |
```

```
| Buffy | 1989-05-13 | 2003-08-19 | 14 |
| Chirpy | 1998-09-11 | 2003-08-19 | 4 |
| Garras | 1994-03-17 | 2003-08-19 | 9 |
| Fang | 1990-08-27 | 2003-08-19 | 12 |
| Fluffy | 1993-02-04 | 2003-08-19 | 10 |
| Puffball | 1999-03-30 | 2003-08-19 | 4 |
| Slim | 1996-04-29 | 2003-08-19 | 7 |
| Whistler | 1997-12-09 | 2003-08-19 | 5 |
+ ---------- + ------------ + ------------ + ------ +
```

Para ordenar la salida por edad en lugar de por nombre , simplemente use una cláusula ORDER BY diferente :

Página 15


```
mysql> SELECT nombre, nacimiento, CURDATE (),
           TIMESTAMPDIFF (YEAR, birth, CURDATE ()) AS age
           DE MASCOTAS ORDEN POR EDAD;
+ ---------- + ------------ + ------------ + ------ +
| nombre | nacimiento | CURDATE () | edad |
+ ---------- + ------------ + ------------ + ------ +
| Chirpy | 1998-09-11 | 2003-08-19 | 4 |
| Puffball | 1999-03-30 | 2003-08-19 | 4 |
| Whistler | 1997-12-09 | 2003-08-19 | 5 |
| Slim | 1996-04-29 | 2003-08-19 | 7 |
| Garras | 1994-03-17 | 2003-08-19 | 9 |
| Fluffy | 1993-02-04 | 2003-08-19 | 10 |
| Fang | 1990-08-27 | 2003-08-19 | 12 |
| Bowser | 1989-08-31 | 2003-08-19 | 13 |
| Buffy | 1989-05-13 | 2003-08-19 | 14 |
+ ---------- + ------------ + ------------ + ------ +
```

Se puede utilizar una consulta similar para determinar la edad de muerte de los animales que han muerto. Tu determinas cual animales, estos son comprobando si el valor de muerte es NULO . Luego, para aquellos con valores no NULL , calcular la diferencia entre los valores de muerte y nacimiento :

```
mysql> SELECT nombre, nacimiento, muerte,
           TIMESTAMPDIFF (AÑO, nacimiento, muerte) AS edad
           DE mascota DONDE la muerte NO ES NULO ORDEN POR EDAD;
+ -------- + ------------ + ------------ + ------ +
| nombre | nacimiento | muerte | edad |
+ -------- + ------------ + ------------ + ------ +
| Bowser | 1989-08-31 | 1995-07-29 | 5 |
+ -------- + ------------ + ------------ + ------ +
```

La consulta usa death IS NOT NULL en lugar de death <> NULL porque NULL es un valor especial que no se puede comparar utilizando los operadores de comparación habituales. Esto se analiza más adelante. Consulte la Sección 3.3.4.6, “Trabajar con valores NULL” .

¿Qué pasa si quieres saber qué animales cumplen años el próximo mes? Para este tipo de cálculo, año y el día son irrelevantes; simplemente desea extraer la parte del mes de la columna de nacimiento . MySQL proporciona varias funciones para extraer partes de fechas, como AÑO () , MES () y DÍA DEL MES () . MES() es la función apropiada aquí. Para ver cómo funciona, ejecute una consulta simple que muestre el valor de ambos nacimiento y MES (nacimiento) :

```
mysql> SELECCIONAR nombre, nacimiento, MES (nacimiento) DE la mascota;
+ ---------- + ------------ + -------------- +
| nombre | nacimiento | MES (nacimiento) |
+ ---------- + ------------ + -------------- +
| Fluffy | 1993-02-04 | 2 |
| Garras | 1994-03-17 | 3 |
| Buffy | 1989-05-13 | 5 |
| Fang | 1990-08-27 | 8 |
| Bowser | 1989-08-31 | 8 |
| Chirpy | 1998-09-11 | 9 |
| Whistler | 1997-12-09 | 12 |
| Slim | 1996-04-29 | 4 |
| Puffball | 1999-03-30 | 3 |
+ ---------- + ------------ + -------------- +
```

Encontrar animales con cumpleaños en el próximo mes también es sencillo. Suponga que el mes actual es Abril. Entonces, el valor del mes es 4 y puede buscar animales nacidos en mayo (mes 5) así:

```
mysql> SELECT nombre, nacimiento DE mascota DONDE MES (nacimiento) = 5;
```

```
+ ------- + ------------ +
| nombre | nacimiento |
+ ------- + ------------ +
| Buffy | 1989-05-13 |
+ ------- + ------------ +
```


Página 16

Existe una pequeña complicación si el mes actual es diciembre. No puede simplemente agregar uno al mes número (12) y busque animales nacidos en el mes 13 , porque no existe tal mes. En cambio, buscas animales nacidos en enero (mes 1).

Puede escribir la consulta para que funcione sin importar cuál sea el mes actual, para que no tenga que use el número para un mes en particular. DATE_ADD () le permite agregar un intervalo de tiempo a una fecha determinada. Si agrega un mes al valor de CURDATE () , luego extraiga la parte del mes con MONTH () , el resultado produce el mes en el que buscar los cumpleaños:

```
mysql> SELECCIONAR nombre, nacimiento DE mascota
           DONDE MES (nacimiento) = MES (DATE_ADD (CURDATE (), INTERVAL 1 MES));
```

Una forma diferente de lograr la misma tarea es agregar 1 para obtener el mes siguiente después del actual después usando la función de módulo (MOD) para ajustar el valor del mes a 0 si actualmente es 12 :

```
mysql> SELECCIONAR nombre, nacimiento DE mascota
           DONDE MES (nacimiento) = MOD (MES (CURDATE ()), 12) + 1;
```

MES () devuelve un número entre 1 y 12 . Y MOD (algo, 12) devuelve un número entre 0 y 11 . Entonces la adición tiene que ser posterior al MOD () , de lo contrario pasaríamos de noviembre (11) a enero (1).

Si un cálculo utiliza fechas no válidas, el cálculo falla y genera advertencias:

```
mysql> SELECCIONAR '2018-10-31' + INTERVALO 1 DÍA;
+ ------------------------------ +
| '2018-10-31' + INTERVALO 1 DÍA |
+ ------------------------------ +
| 2018-11-01                     |
+ ------------------------------ +
mysql> SELECCIONAR '2018-10-32' + INTERVALO 1 DÍA;
+ ------------------------------ +
| '2018-10-32' + INTERVALO 1 DÍA |
+ ------------------------------ +
| NULO                           |
+ ------------------------------ +
mysql> MOSTRAR ADVERTENCIAS;
+ --------- + ------ + -------------------------------- -------- +
| Nivel | Código | Mensaje                                        |
+ --------- + ------ + -------------------------------- -------- +
| Advertencia | 1292 | Valor de fecha y hora incorrecto: '2018-10-32' |
+ --------- + ------ + -------------------------------- -------- +
```

3.3.4.6 Trabajar con valores NULL

El valor NULL puede resultar sorprendente hasta que te acostumbres. Conceptualmente, NULL significa "un desconocido que falta valor ”y se trata de forma algo diferente a otros valores.

Para probar NULL , use los operadores IS NULL y IS NOT NULL , como se muestra aquí:

```
mysql> SELECT 1 ES NULO, 1 NO ES NULO;
+ ----------- + --------------- +
| 1 ES NULO | 1 NO ES NULO |
+ ----------- + --------------- +
|           0 |               1 |
+ ----------- + --------------- +
```

No puede utilizar operadores de comparación aritmética como = , < o <> para probar NULL . Para demostrar esto por ti mismo, prueba la siguiente consulta:

```
mysql> SELECT 1 = NULL, 1 <> NULL, 1 <NULL, 1> NULL;
+ ---------- + ---------- + ---------- + ---------- +
| 1 = NULO | 1 <> NULL | 1 <NULO | 1> NULO |
```

274

Página 17

Recuperar información de una tabla

```
+ ---------- + ---------- + ---------- + ---------- +
| NULL | NULL | NULL | NULL |
+ ---------- + ---------- + ---------- + ---------- +
```

Debido a que el resultado de cualquier comparación aritmética con NULL también es NULL , no puede obtener ninguna resultados de tales comparaciones.

En MySQL, 0 o NULL significa falso y cualquier otra cosa significa verdadero. El valor de verdad predeterminado de un booleano la operación es 1 .

Este tratamiento especial de NULL es el motivo por el cual, en el apartado anterior, fue necesario determinar qué los animales ya no están vivos usando death IS NULL en lugar de death <> NULL .

Dos valores NULL se consideran iguales en un GROUP BY .

Al hacer un ORDER BY , los valores NULL se presentan primero si hace ORDER BY ... ASC y por último si hacer ORDEN POR ... DESC .

Un error común al trabajar con NULL es asumir que no es posible insertar un cero o un vacío cadena en una columna definida como NOT NULL , pero este no es el caso. Estos son de hecho valores, mientras que NULL significa "no tener valor". Puede probar esto con bastante facilidad usando IS [NOT] NULL como se muestra:

```
mysql> SELECT 0 ES NULO, 0 NO ES NULO, '' ES NULO, '' NO ES NULO;
+ ---------- + --------------- + ------------ + -------- -------- +
| 0 ES NULO | 0 NO ES NULO | '' ES NULO | '' NO ES NULO |
+ ---------- + --------------- + ------------ + -------- -------- +
| 0 | 1 | 0 | 1 |
+ ---------- + --------------- + ------------ + -------- -------- +
```

Por lo tanto, es completamente posible insertar una cadena cero o vacía en una columna NOT NULL , ya que de hecho son NO NULO . Consulte la Sección B.3.4.3, “Problemas con valores NULL” .

3.3.4.7 Coincidencia de patrones

MySQL proporciona coincidencia de patrones SQL estándar, así como una forma de coincidencia de patrones basada en expresiones regulares similares a las que utilizan las utilidades de Unix como vi , grep y sed .

La coincidencia de patrones SQL le permite usar _ para hacer coincidir cualquier carácter individual y % para hacer coincidir un número de caracteres (incluido cero caracteres). En MySQL, los patrones de SQL no distinguen entre mayúsculas y minúsculas de forma predeterminada. Aquí se muestran algunos ejemplos. No use = o <> cuando use patrones SQL. Usa el ME GUSTA o NO LIKE operadores de comparación en su lugar.

Para buscar nombres que comiencen con b :

```
mysql> SELECT * FROM pet DONDE nombre COMO 'b%';
+ -------- + -------- + --------- + ------ + ------------ + - ---------- +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ -------- + -------- + --------- + ------ + ------------ + - ---------- +
| Buffy | Harold | perro | f | 1989-05-13 | NULO |
| Bowser | Diane | perro | m | 1989-08-31 | 1995-07-29 |
+ -------- + -------- + --------- + ------ + ------------ + - ---------- +
```

Para buscar nombres que terminen con fy :

```
mysql> SELECT * FROM pet DONDE nombre COMO '% fy';
+ -------- + -------- + --------- + ------ + ------------ + - ------ +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ -------- + -------- + --------- + ------ + ------------ + - ------ +
| Fluffy | Harold | gato | f | 1993-02-04 | NULL |
| Buffy | Harold | perro | f | 1989-05-13 | NULL |
+ -------- + -------- + --------- + ------ + ------------ + - ------ +
```

275

Página 18

Recuperar información de una tabla

Para buscar nombres que contengan una w :

```
mysql> SELECT * FROM pet DONDE nombre COMO '% w%';
+ ---------- + ------- + --------- + ------ + ------------ + ------------ +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ ---------- + ------- + --------- + ------ + ------------ + ------------ +
| Garras | Gwen | gato | m | 1994-03-17 | NULO |
| Bowser | Diane | perro | m | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen | pájaro | NULL | 1997-12-09 | NULO |
+ ---------- + ------- + --------- + ------ + ------------ + ------------ +
```

Para buscar nombres que contengan exactamente cinco caracteres, utilice cinco instancias del carácter de patrón _ :

```
mysql> SELECT * FROM pet WHERE name LIKE '____';
+ ------- + -------- + --------- + ------ + ------------ + - ----- +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ ------- + -------- + --------- + ------ + ------------ + - ----- +
| Garras | Gwen | gato | m | 1994-03-17 | NULL |
| Buffy | Harold | perro | f | 1989-05-13 | NULL |
+ ------- + -------- + --------- + ------ + ------------ + - ----- +
```

El otro tipo de coincidencia de patrones proporcionada por MySQL utiliza expresiones regulares extendidas. Cuando usted prueba para una coincidencia para este tipo de patrón, use los operadores REGEXP y NOT REGEXP (o RLIKE y NOT RLIKE , que son sinónimos).

La siguiente lista describe algunas características de las expresiones regulares extendidas:

• . coincide con cualquier carácter.

• Una clase de carácter [...] coincide con cualquier carácter entre corchetes. Por ejemplo, [abc] coincide con a , b , o c . Para nombrar un rango de caracteres, use un guión. [az] coincide con cualquier letra, mientras que [0-9] coincide cualquier dígito.

• * coincide con cero o más instancias de lo que le precede. Por ejemplo, x * coincide con cualquier número de x caracteres, [0-9] * coincide con cualquier número de dígitos y . * coincide con cualquier número de cualquier cosa.

• Una coincidencia de patrón de expresión regular tiene éxito si el patrón coincide en cualquier parte del valor que se está probando. (Esto difiere de una coincidencia de patrón LIKE , que solo tiene éxito si el patrón coincide con el valor completo).

• Para anclar un patrón de modo que deba coincidir con el principio o el final del valor que se está probando, utilice ^ en el principio o $ al final del patrón.

Para demostrar cómo funcionan las expresiones regulares extendidas, las consultas LIKE mostradas anteriormente se reescriben aquí para usar REGEXP .

Para buscar nombres que comiencen con b , use ^ para que coincida con el comienzo del nombre:

```
mysql> SELECT * FROM pet DONDE nombre REGEXP '^ b';
+ -------- + -------- + --------- + ------ + ------------ + - ---------- +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ -------- + -------- + --------- + ------ + ------------ + - ---------- +
| Buffy | Harold | perro | f | 1989-05-13 | NULO |
| Bowser | Diane | perro | m | 1989-08-31 | 1995-07-29 |
+ -------- + -------- + --------- + ------ + ------------ + - ---------- +
```

Para forzar una comparación REGEXP para que distinga entre mayúsculas y minúsculas, use la palabra clave BINARY para convertir una de las cadenas en una cadena binaria. Esta consulta solo coincide con la b minúscula al comienzo de un nombre:

```
SELECCIONAR * DE mascota DONDE nombre REGEXP BINARIO '^ b';
```

Para buscar nombres que terminen con fy , use $ para que coincida con el final del nombre:

276

Página 19

Recuperar información de una tabla

```
mysql> SELECT * FROM pet DONDE nombre REGEXP 'fy $';
+ -------- + -------- + --------- + ------ + ------------ + - ------ +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ -------- + -------- + --------- + ------ + ------------ + - ------ +
| Fluffy | Harold | gato | f | 1993-02-04 | NULL |
| Buffy | Harold | perro | f | 1989-05-13 | NULL |
+ -------- + -------- + --------- + ------ + ------------ + - ------ +
```

Para buscar nombres que contengan una w , use esta consulta:

```
mysql> SELECT * FROM pet DONDE nombre REGEXP 'w';
+ ---------- + ------- + --------- + ------ + ------------ + ------------ +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ ---------- + ------- + --------- + ------ + ------------ + ------------ +
| Garras | Gwen | gato | m | 1994-03-17 | NULO |
| Bowser | Diane | perro | m | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen | pájaro | NULL | 1997-12-09 | NULO |
+ ---------- + ------- + --------- + ------ + ------------ + ------------ +
```

Dado que un patrón de expresión regular coincide si aparece en cualquier parte del valor, no es necesario en el consulta anterior para poner un comodín a cada lado del patrón para que coincida con el valor completo como sería verdadero con un patrón SQL.

Para buscar nombres que contengan exactamente cinco caracteres, use ^ y $ para hacer coincidir el principio y el final del nombre, y cinco instancias de . entre:

```
mysql> SELECT * FROM pet WHERE name REGEXP '^ ..... $';
+ ------- + -------- + --------- + ------ + ------------ + - ------ +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ ------- + -------- + --------- + ------ + ------------ + - ------ +
| Garras | Gwen | gato | m | 1994-03-17 | NULL |
| Buffy | Harold | perro | f | 1989-05-13 | NULL |
+ ------- + -------- + --------- + ------ + ------------ + - ------ +
```

También puede escribir la consulta anterior usando el operador { *n* } (" repetir- *n*- veces"):

```
mysql> SELECT * FROM pet DONDE nombre REGEXP '^. {5} $';
+ ------- + -------- + --------- + ------ + ------------ + - ------ +
| nombre | propietario | especies | sexo | nacimiento | muerte |
+ ------- + -------- + --------- + ------ + ------------ + - ------ +
| Garras | Gwen | gato | m | 1994-03-17 | NULL |
| Buffy | Harold | perro | f | 1989-05-13 | NULL |
+ ------- + -------- + --------- + ------ + ------------ + - ------ +
```

Para obtener más información sobre la sintaxis de las expresiones regulares, consulte la Sección 12.8.2, “Expresiones regulares” .

3.3.4.8 Contar filas

Las bases de datos se utilizan a menudo para responder a la pregunta "¿Con qué frecuencia aparece un determinado tipo de datos en una tabla?" Por ejemplo, es posible que desee saber cuántas mascotas tiene o cuántas mascotas tiene cada dueño, o si Es posible que desee realizar varios tipos de operaciones de censo en sus animales.

Contar el número total de animales que tienes es la misma pregunta que "¿Cuántas filas hay en la mascota? ¿mesa?" porque hay un registro por mascota. COUNT (*) cuenta el número de filas, por lo que la consulta debe contar tus animales se ven así:

```
mysql> SELECT COUNT (*) FROM pet;
+ ---------- +
| CONTAR (*) |
+ ---------- +
| 9 |
+ ---------- +
```

277

Página 20

Anteriormente, recuperó los nombres de las personas que tenían mascotas. Puedes usar COUNT () si quieres encontrar saber cuántas mascotas tiene cada dueño:

```
mysql> SELECCIONAR propietario, CONTAR (*) DEL GRUPO DE mascotas POR propietario;
+ -------- + ---------- +
| propietario | CONTAR (*) |
+ -------- + ---------- +
| Benny | 2 |
| Diane | 2 |
| Gwen | 3 |
| Harold | 2 |
+ -------- + ---------- +
```

La consulta anterior usa GROUP BY para agrupar todos los registros de cada propietario . El uso de COUNT () en junto con GROUP BY es útil para caracterizar sus datos en varios grupos. El seguimiento
Los ejemplos muestran diferentes formas de realizar operaciones de censo de animales.

Número de animales por especie:

```
mysql> SELECCIONAR especies, CUENTA (*) DEL GRUPO DE mascotas POR especie;
+ --------- + ---------- +
| especies | CONTAR (*) |
+ --------- + ---------- +
| pájaro | 2 |
| gato | 2 |
| perro | 3 |
| hámster | 1 |
| serpiente | 1 |
+ --------- + ---------- +
```

Número de animales por sexo:

```
mysql> SELECT sex, COUNT (*) FROM pet GROUP BY sex;
+ ------ + ---------- +
| sexo | CONTAR (*) |
+ ------ + ---------- +
| NULL | 1 |
| f | 4 |
| m | 4 |
+ ------ + ---------- +
```

(En esta salida, NULL indica que se desconoce el sexo).

Número de animales por combinación de especies y sexo:

```
mysql> SELECCIONAR especie, sexo, CUENTA (*) DE GRUPO DE mascotas POR especie, sexo;
+ --------- + ------ + ---------- +
| especies | sexo | CONTAR (*) |
+ --------- + ------ + ---------- +
| pájaro | NULL | 1 |
| pájaro | f | 1 |
| gato | f | 1 |
| gato | m | 1 |
| perro | f | 1 |
| perro | m | 2 |
| hámster f | 1 |
| serpiente | m | 1 |
+ --------- + ------ + ---------- +
```

No necesita recuperar una tabla completa cuando usa COUNT () . Por ejemplo, la consulta anterior, cuando realizado solo en perros y gatos, se ve así:

```
mysql> SELECCIONAR especie, sexo, CONTAR (*) DE mascota
           DONDE especie = 'perro' O especie = 'gato'
           GRUPO POR especie, sexo;
```



```
+ --------- + ------ + ---------- +
| especies | sexo | CONTAR (*) |
+ --------- + ------ + ---------- +
```

```
| gato | f |                                    1 |
| gato | m |                                    1 |
| perro | f |                                   1 |
| perro | m |                                   2 |
+ --------- + ------ + ---------- +
```

O, si quisiera la cantidad de animales por sexo solo para animales cuyo sexo se conoce:

```
mysql> SELECCIONAR especie, sexo, CONTAR (*) DE mascota
           DONDE EL sexo NO ES NULO
           GRUPO POR especie, sexo;
+ --------- + ------ + ---------- +
| especies | sexo | CONTAR (*) |
+ --------- + ------ + ---------- +
| pájaro | f |                                  1 |
| gato | f |                                    1 |
| gato | m |                                    1 |
| perro | f |                                   1 |
| perro | m |                                   2 |
| hámster f |                                   1 |
| serpiente | m |                               1 |
+ --------- + ------ + ---------- +
```

Si nombra columnas para seleccionar además del valor COUNT () , debe estar presente una cláusula GROUP BY que nombra esas mismas columnas. De lo contrario, ocurre lo siguiente:

• Si el modo SQL ONLY_FULL_GROUP_BY está habilitado, se produce un error:

```
mysql> SET sql_mode = 'SOLO_GRUPO_FULL_BY';
Consulta OK, 0 filas afectadas (0,00 seg)

mysql> SELECCIONAR propietario, COUNT (*) FROM pet;
ERROR 1140 (42000): en una consulta agregada sin GROUP BY, expresión
# 1 de la lista SELECT contiene la columna no agregada 'menagerie.pet.owner';
esto es incompatible con sql_mode = only_full_group_by
```

• Si ONLY_FULL_GROUP_BY no está habilitado, la consulta se procesa tratando todas las filas como un solo grupo, pero el valor seleccionado para cada columna nombrada no es determinista. El servidor es libre de seleccionar el valor de cualquier fila:

```
mysql> SET sql_mode = '';
Consulta OK, 0 filas afectadas (0,00 seg)

mysql> SELECCIONAR propietario, COUNT (*) FROM pet;
+ -------- + ---------- +
| propietario | CONTAR (*) |
+ -------- + ---------- +
| Harold |                                  8 |
+ -------- + ---------- +
1 fila en conjunto (0,00 seg)
```

Consulte también Sección 12.20.3, “Manejo MySQL de GROUP BY” . Consulte la Sección 12.20.1, “Función agregada Descripciones ” para obtener información sobre el comportamiento de COUNT (*expr*) y optimizaciones relacionadas.

3.3.4.9 Uso de más de una tabla

La tabla de mascotas realiza un seguimiento de las mascotas que tiene. Si desea registrar otra información sobre ellos, como eventos en sus vidas como visitas al veterinario o cuando nacen las camadas, necesita otra mesa. Qué ¿Debería verse esta tabla? Debe contener la siguiente información:

• El nombre de la mascota para que sepa a qué animal pertenece cada evento.

• Una fecha para que sepa cuándo ocurrió el evento.

• Un campo para describir el evento.

• Un campo de tipo de evento, si desea poder categorizar eventos.

Dadas estas consideraciones, la declaración CREATE TABLE para la tabla de eventos podría tener este aspecto:

```
mysql> evento CREATE TABLE (nombre VARCHAR (20), fecha FECHA,
           escriba VARCHAR (15), observe VARCHAR (255));
```

Al igual que con la tabla de mascotas , es más fácil cargar los registros iniciales creando un archivo de texto delimitado por tabulaciones que contenga la siguiente información.

nombre	fecha	tipo	observación
Mullido	1995-05-15	camada	4 gatitos, 3 hembras, 1 macho
Buffy	1993-06-23	camada	5 cachorros, 2 hembras, 3 machos
Buffy	1994-06-19	camada	3 cachorros, 3 hembras
Alegre	1999-03-21	veterinario	pico necesario enderezado
Delgado	1997-08-03	veterinario	costilla rota
Bowser	1991-10-12	perrera	
Colmillo	1991-10-12	perrera	
Colmillo	1998-08-28	cumpleaños	Le di un nuevo juguete para masticar
Garras	1998-03-17	cumpleaños	Le di un nuevo collar antipulgas
Whistler	1998-12-09	cumpleaños	Primer cumpleaños

Cargue los registros así:

```
mysql> LOAD DATA LOCAL INFILE 'event.txt' INTO TABLE event;
```

Según lo que ha aprendido de las consultas que ha ejecutado en la tabla de mascotas , debería poder
para realizar recuperaciones en los registros en la tabla de eventos ; los principios son los mismos. Pero cuando es el evento
tabla por sí sola insuficiente para responder a las preguntas que pueda hacer?

Suponga que desea averiguar las edades a las que cada mascota tuvo su camada. Vimos antes cómo
Calcule las edades a partir de dos fechas. La fecha de la camada de la madre está en la tabla de eventos , pero para calcular su edad
en esa fecha necesita su fecha de nacimiento, que se almacena en la tabla de mascotas . Esto significa que la consulta requiere tanto
mesas:

```
mysql> SELECT pet.name,
           TIMESTAMPDIFF (AÑO, nacimiento, fecha) AS edad,
           observación
           DESDE el evento pet INNER JOIN
               ON pet.name = event.name
           DONDE event.type = 'basura';
+ -------- + ------ + ---------------------------- +
| nombre | edad | observación                                                |
+ -------- + ------ + ---------------------------- +
| Fluffy | 2 | 4 gatitos, 3 hembras, 1 macho |
| Buffy | 4 | 5 cachorros, 2 hembras, 3 machos |
| Buffy | 5 | 3 cachorros, 3 hembras                                     |
+ -------- + ------ + ---------------------------- +
```

Hay varias cosas a tener en cuenta sobre esta consulta:

• La cláusula FROM une dos tablas porque la consulta necesita extraer información de ambas.

Página 23

• Al combinar (unir) información de varias tablas, debe especificar cómo se registran en una tabla
se puede hacer coincidir con los registros en el otro. Esto es fácil porque ambos tienen una columna de nombre . La consulta
utiliza una cláusula ON para hacer coincidir los registros en las dos tablas según los valores de los nombres .

La consulta usa INNER JOIN para combinar las tablas. Un INNER JOIN permite filas de cualquier tabla
aparecer en el resultado si y solo si ambas tablas cumplen las condiciones especificadas en la cláusula ON . En esto
Por ejemplo, la cláusula ON especifica que la columna de nombre en la tabla de mascotas debe coincidir con la columna de nombre
en la mesa del evento . Si un nombre aparece en una tabla pero no en la otra, la fila no aparecerá en el resultado.
porque falla la condición en la cláusula ON .

• Dado que la columna de nombre aparece en ambas tablas, debe especificar a qué tabla se refiere cuando
refiriéndose a la columna. Esto se hace anteponiendo el nombre de la tabla al nombre de la columna.

No es necesario tener dos tablas diferentes para realizar una combinación. A veces es útil unir una mesa a sí misma, si desea comparar los registros de una tabla con otros registros de la misma tabla. Por ejemplo, para encontrar cría parejas entre sus mascotas, puede unirse a la mesa de mascotas consigo mismo para producir parejas candidatas de machos vivos y hembras de especies similares:

```
mysql> SELECT p1.name, p1.sex, p2.name, p2.sex, p1.species
    -> DE pet COMO p1 INNER JOIN pet COMO p2
    ->     ON p1.species = p2.species
    ->     Y p1.sex = 'f' Y p1.death ES NULO
    ->     AND p2.sex = 'm' AND p2.death ES NULO;
+ -------- + ------ + ------- + ------ + --------- +
| nombre | sexo | nombre | sexo | especies |
+ -------- + ------ + ------- + ------ + --------- +
| Fluffy | f | Garras | m | gato |
| Buffy | f | Fang | m | perro |
+ -------- + ------ + ------- + ------ + --------- +
```

En esta consulta, especificamos alias para que el nombre de la tabla se refiera a las columnas y mantenemos instancia de la tabla con la que está asociada cada referencia de columna.

3.4 Obtener información sobre bases de datos y tablas

¿Qué sucede si olvida el nombre de una base de datos o tabla, o cuál es la estructura de una tabla determinada (por ejemplo, cómo se llaman sus columnas)? MySQL aborda este problema a través de varias declaraciones que proporcionan información sobre las bases de datos y tablas que soporta.

Ya ha visto SHOW DATABASES , que enumera las bases de datos administradas por el servidor. Encontrar Para saber qué base de datos está seleccionada actualmente, use la función DATABASE () :

```
mysql> SELECCIONAR BASE DE DATOS ();
+ ------------ +
| BASE DE DATOS () |
+ ------------ +
| menagerie |
+ ------------ +
```

Si aún no ha seleccionado ninguna base de datos, el resultado es NULO .

Para averiguar qué tablas contiene la base de datos predeterminada (por ejemplo, cuando no está seguro del nombre de una mesa), use esta declaración:

```
mysql> MOSTRAR TABLAS;
+ --------------------- +
| Tables_in_menagerie |
+ --------------------- +
| evento |
| mascota |
+ --------------------- +
```


Página 24

El nombre de la columna en la salida producida por esta declaración es siempre Tables_in_ *nombre_base_de_datos* , donde *db_name* es el nombre de la base de datos. Consulte la Sección 13.7.5.37, "Declaración MOSTRAR TABLAS" , para obtener más información. información.

Si desea conocer la estructura de una tabla, la instrucción DESCRIBE es útil; muestra información sobre cada una de las columnas de una tabla:

```
mysql> DESCRIBE pet;
+ --------- + ------------- + ------ + ----- + --------- + - ----- +
| Campo | Tipo | Nulo | Clave | Por defecto | Extra |
+ --------- + ------------- + ------ + ----- + --------- + - ----- +
| nombre | varchar (20) | SI | | NULL | |
| propietario | varchar (20) | SI | | NULL | |
| especies | varchar (20) | SI | | NULL | |
| sexo | char (1) | SI | | NULL | |
| nacimiento | fecha | SI | | NULL | |
| muerte | fecha | SI | | NULL | |
+ --------- + ------------- + ------ + ----- + --------- + - ----- +
```

El campo indica el nombre de la columna, Type es el tipo de datos de la columna, NULL indica si el La columna puede contener valores NULL , Key indica si la columna está indexada y Default especifica la valor predeterminado de la columna. Extra muestra información especial sobre las columnas: si una columna se creó con la opción AUTO_INCREMENT , el valor será auto_increment en lugar de vacío.

DESC es una forma corta de DESCRIBE . Consulte la Sección 13.8.1, “Declaración DESCRIBE” , para obtener más información.

Puede obtener la instrucción CREATE TABLE necesaria para crear una tabla existente utilizando SHOW Sentencia CREATE TABLE . Consulte la Sección 13.7.5.10, “Instrucción SHOW CREATE TABLE” .

Si tiene índices en una tabla, SHOW INDEX FROM *tbl_name* produce información sobre ellos. Ver Sección 13.7.5.22, “Declaración SHOW INDEX” , para obtener más información sobre esta declaración.

3.5 Usando mysql en modo por lotes

En las secciones anteriores, usó mysql de forma interactiva para ingresar declaraciones y ver los resultados. Usted puede también ejecute mysql en modo por lotes. Para hacer esto, coloque las declaraciones que desea ejecutar en un archivo, luego dígale a mysql que leer su entrada del archivo:

```
shell> mysql < archivo por lotes
```

Si está ejecutando mysql en Windows y tiene algunos caracteres especiales en el archivo que causan problemas, puede hacer esto:

```
C: \> mysql -e " archivo por lotes de origen "
```

Si necesita especificar los parámetros de conexión en la línea de comando, el comando podría verse así:

```
shell> mysql -h host -u usuario -p < archivo por lotes
Ingrese contraseña: ********
```

Cuando usa mysql de esta manera, está creando un archivo de script y luego ejecutando el script.

Si desea que el script continúe incluso si algunas de las declaraciones que contiene producen errores, debe usar el -forzar la opción de línea de comandos.

¿Por qué utilizar un guión? Aquí hay algunas razones:

- Si ejecuta una consulta repetidamente (por ejemplo, todos los días o todas las semanas), convertirla en un script le permite evitar volviéndolo a escribir cada vez que lo ejecuta.

Página 25

Ejemplos de consultas comunes

- Puede generar nuevas consultas a partir de las existentes que son similares copiando y editando archivos de script.
- El modo por lotes también puede ser útil mientras desarrolla una consulta, especialmente para declaraciones de varias líneas. o secuencias de declaraciones múltiples. Si comete un error, no es necesario que vuelva a escribir todo. Solo editar su script para corregir el error, luego dígale a mysql que lo ejecute nuevamente.
- Si tiene una consulta que produce muchos resultados, puede ejecutar el resultado a través de un buscapersonas en lugar de viendo cómo se desplaza desde la parte superior de la pantalla:

  ```
  shell> mysql < archivo por lotes | más
  ```

- Puede capturar la salida en un archivo para su posterior procesamiento:

  ```
  shell> mysql < archivo por lotes > mysql.out
  ```

- Puede distribuir su secuencia de comandos a otras personas para que también puedan ejecutar las declaraciones.
- Algunas situaciones no permiten el uso interactivo, por ejemplo, cuando ejecuta una consulta desde un trabajo cron . En En este caso, debe utilizar el modo por lotes.

El formato de salida predeterminado es diferente (más conciso) cuando ejecuta mysql en modo por lotes que cuando utilícelo de forma interactiva. Por ejemplo, la salida de SELECT DISTINCT species FROM pet se ve así cuando mysql se ejecuta de forma interactiva:

```
+ --------- +
| especies |
+ --------- +
| pájaro |
| gato |
| perro |
| hámster
| serpiente |
+ --------- +
```

En el modo por lotes, la salida se ve así:

```
especies
pájaro
gato
perro
hámster
serpiente
```

Si desea obtener el formato de salida interactivo en modo por lotes, use mysql -t . Para hacer eco en la salida declaraciones que se ejecutan, use mysql -v .

También puede usar scripts desde el indicador de mysql usando el comando fuente o \. mando:

```
mysql> nombre de archivo fuente ;
mysql> \. nombre del archivo
```

Consulte la Sección 4.5.1.5, “Ejecución de sentencias SQL desde un archivo de texto” , para obtener más información.

3.6 Ejemplos de consultas comunes

Aquí hay ejemplos de cómo resolver algunos problemas comunes con MySQL.

Algunos de los ejemplos usan la tienda de mesa para mantener el precio de cada artículo (número de artículo) para ciertos comerciantes. (distribuidores). Suponiendo que cada comerciante tiene un precio fijo único por artículo, entonces (artículo , comerciante) es un clave principal para los registros.

Página 26

El valor máximo de una columna

Inicie la herramienta de línea de comandos mysql y seleccione una base de datos:

```
shell> mysql nombre-de-tu-base de datos
```

Para crear y completar la tabla de ejemplo, use estas declaraciones:

```
Tienda CREAR MESA (
    artículo INT UNSIGNED DEFAULT '0000' NOT NULL,
    distribuidor CHAR (20) DEFAULT '' NOT NULL,
    precio DECIMAL (16,2) DEFAULT '0.00' NOT NULL,
    LLAVE PRIMARIA (artículo, distribuidor));
INSERTAR EN LA TIENDA VALORES
    (1, 'A', 3.45), (1, 'B', 3.99), (2, 'A', 10.99), (3, 'B', 1.45),
    (3, 'C', 1,69), (3, 'D', 1,25), (4, 'D', 19,95);
```

Después de emitir las declaraciones, la tabla debe tener el siguiente contenido:

```
SELECCIONAR * DE la tienda PEDIR POR artículo;

+ --------- + -------- + ------- +
| articulo | distribuidor | precio |
+ --------- + -------- + ------- +
|                 1 | A | 3,45 |
|                 1 | B | 3,99 |
|                 2 | A | 10,99 |
|                 3 | B | 1,45 |
|                 3 | C | 1,69 |
|                 3 | D | 1,25 |
|                 4 | D | 19,95 |
+ --------- + -------- + ------- +
```

3.6.1 El valor máximo de una columna

"¿Cuál es el número de artículo más alto?"

```
SELECCIONE MAX (artículo) COMO artículo DE la tienda;

+ --------- +
| articulo |
+ --------- +
| 4 |
+ --------- +
```

3.6.2 La fila que tiene el máximo de una determinada columna

Tarea: busque el número, el distribuidor y el precio del artículo más caro.

Esto se hace fácilmente con una subconsulta:

```
SELECCIONAR artículo, distribuidor, precio
DE la tienda
DONDE precio = (SELECCIONE MAX (precio) DE la tienda);

+ --------- + -------- + ------- +
| articulo | distribuidor | precio |
+ --------- + -------- + ------- +
| 0004 | D | 19,95 |
+ --------- + -------- + ------- +
```

Otras soluciones son usar LEFT JOIN o ordenar todas las filas descendiendo por precio y obtener solo la primera fila utilizando la cláusula LIMIT específica de MySQL :

```
SELECCIONE s1.artículo, s1.distribuidor, s1.precio
```

Página 27

Máximo de columna por grupo

```
DESDE la tienda s1
IZQUIERDA ÚNETE a la tienda s2 ON s1.price <s2.price
DONDE s2.article ES NULO;

SELECCIONAR artículo, distribuidor, precio
DE la tienda
PEDIR POR precio DESC
LÍMITE 1;
```

Nota

Si hubiera varios artículos más caros, cada uno con un precio de 19,95, el LÍMITE La solución mostraría solo uno de ellos.

3.6.3 Máximo de columnas por grupo

Tarea: Encuentra el precio más alto por artículo.

```
SELECCIONAR artículo, MAX (precio) COMO precio
DE la tienda
GRUPO POR artículo
PEDIDO POR artículo;

+ --------- + ------- +
| articulo | precio |
+ --------- + ------- +
| 0001 | 3,99 |
| 0002 | 10,99 |
| 0003 | 1,69 |
| 0004 | 19,95 |
+ --------- + ------- +
```

3.6.4 Las filas que tienen el máximo grupal de una determinada columna

Tarea: Para cada artículo, busque el distribuidor o distribuidores con el precio más caro.

Este problema se puede resolver con una subconsulta como esta:

```
SELECCIONAR artículo, distribuidor, precio
DESDE la tienda s1
DONDE precio = (SELECCIONAR MAX (s2.precio)
                DESDE la tienda s2
                DONDE s1.article = s2.article)
PEDIDO POR artículo;

+ --------- + -------- + ------- +
| articulo | distribuidor | precio |
+ --------- + -------- + ------- +
| 0001 | B | 3,99 |
| 0002 | A | 10,99 |
| 0003 | C | 1,69 |
| 0004 | D | 19,95 |
+ --------- + -------- + ------- +
```

El ejemplo anterior usa una subconsulta correlacionada, que puede ser ineficiente (consulte la Sección 13.2.10.7, “Subconsultas correlacionadas”). Otras posibilidades para resolver el problema son utilizar una subconsulta no correlacionada en la cláusula FROM o LEFT JOIN .

Subconsulta no correlacionada:

```
SELECCIONE s1.article, distribuidor, s1.price
DESDE la tienda s1
UNIRSE (
```


Página 28


```
    SELECCIONAR artículo, MAX (precio) COMO precio
    DE la tienda
    GRUPO POR artículo) AS s2
    ON s1.article = s2.article Y s1.price = s2.price
PEDIDO POR artículo;
```

ÚNETE A LA IZQUIERDA :

```
SELECCIONE s1.artículo, s1.distribuidor, s1.precio
DESDE la tienda s1
IZQUIERDA ÚNETE a la tienda s2 ON s1.article = s2.article Y s1.price <s2.price
DONDE s2.article ES NULO
ORDEN POR s1.article;
```

El LEFT JOIN funciona sobre la base de que cuando s1.price está en su valor máximo, no hay s2.price con un valor mayor y, por tanto, el valor correspondiente del artículo s2 es NULL . Consulte la Sección 13.2.9.2, “UNIR Cláusula ” .

3.6.5 Uso de variables definidas por el usuario

Puede emplear variables de usuario de MySQL para recordar resultados sin tener que almacenarlos en variables en el cliente. (Consulte la Sección 9.4, “Variables definidas por el usuario”).

Por ejemplo, para encontrar los artículos con el precio más alto y más bajo puede hacer esto:

```
mysql> SELECT @min_price: = MIN (precio), @ max_price: = MAX (precio) DE la tienda;
mysql> SELECT * FROM shop WHERE price = @ min_price O price = @ max_price;
+ --------- + -------- + ------- +
| articulo | distribuidor | precio |
+ --------- + -------- + ------- +
| 0003 | D | 1,25 |
| 0004 | D | 19,95 |
+ --------- + -------- + ------- +
```

Nota

También es posible almacenar el nombre de un objeto de base de datos, como una tabla o un columna en una variable de usuario y luego usar esta variable en una declaración SQL; sin embargo, esto requiere el uso de una declaración preparada. Consulte la Sección 13.5, “Declaraciones preparadas” , para más información.

3.6.6 Uso de claves externas

En MySQL, las tablas InnoDB admiten la verificación de restricciones de clave externa. Consulte el Capítulo 14, *InnoDB Storage Engine* , y la Sección 1.7.2.3, “Diferencias de restricciones FOREIGN KEY” .

No se requiere una restricción de clave externa simplemente para unir dos tablas. Para motores de almacenamiento distintos de InnoDB , es posible al definir una columna utilizar una *cláusula* REFERENCES *tbl_name* (*col_name*) , que tiene ningún efecto real, y *sirve solo como una nota o comentario para usted que la columna en la que está actualmente La definición pretende hacer referencia a una columna de otra tabla* . Es extremadamente importante darse cuenta al usar este sintaxis que:

- MySQL no realiza ningún tipo de verificación para asegurarse de que *col_name* realmente existe en *tbl_name* (o incluso ese *tbl_name en* sí mismo existe).

- MySQL no realiza ningún tipo de acción en *tbl_name* , como eliminar filas en respuesta a acciones tomado en filas en la tabla que está definiendo; en otras palabras, esta sintaxis no induce ON DELETE o EN ACTUALIZAR comportamiento alguno. (Aunque puede escribir una cláusula ON DELETE o ON UPDATE como parte de la cláusula REFERENCES , también se ignora).

Página 29

- Esta sintaxis crea una *columna* ; sí **no** crea ningún tipo de índice o una clave.

Puede usar una columna así creada como una columna de combinación, como se muestra aquí:

```
CREAR TABLA persona (
        id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
        nombre CHAR (60) NOT NULL,
        LLAVE PRIMARIA (id)
);

Camisa CREAR MESA (
        id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
        estilo ENUM ('camiseta', 'polo', 'vestido') NOT NULL,
        color ENUM ('rojo', 'azul', 'naranja', 'blanco', 'negro') NOT NULL,
        propietario SMALLINT SIN FIRMAR NOT NULL REFERENCIAS persona (id),
        LLAVE PRIMARIA (id)
);

INSERTAR EN PERSONA VALORES (NULO, 'Antonio Paz');

SELECT @last: = LAST_INSERT_ID ();

INSERTAR EN LA camisa VALORES
(NULO, 'polo', 'azul', @ último),
(NULO, 'vestido', 'blanco', @ último),
(NULO, 'camiseta', 'azul', @ último);

INSÉRTESE EN VALORES de persona (NULL, 'Lilliana Angelovska');

SELECT @last: = LAST_INSERT_ID ();

INSERTAR EN LA camisa VALORES
(NULO, 'vestido', 'naranja', @ último),
(NULO, 'polo', 'rojo', @ último),
(NULO, 'vestido', 'azul', @ último),
(NULO, 'camiseta', 'blanco', @ último);

SELECCIONAR * DE la persona;
+ ---- + ---------- +
| id | nombre |
+ ---- + ---------- +
| 1 | Antonio Paz |
| 2 | Lilliana Angelovska |
+ ---- + ---------- +

SELECCIONAR * DE la camisa;
+ ---- + --------- + -------- + ------- +
| id | estilo | color | propietario |
+ ---- + --------- + -------- + ------- +
| 1 | polo | azul | 1 |
| 2 | vestido | blanco | 1 |
```

```
| 3 | camiseta | azul | 1 |
| 4 | vestido | naranja | 2 |
| 5 | polo | rojo | 2 |
| 6 | vestido | azul | 2 |
| 7 | camiseta | blanco | 2 |
+ ---- + --------- + -------- + ------- +


SELECCIONE s. * DE la persona p INNER JOIN camisa s
      ON s.owner = p.id
  DONDE p. Nombre COMO 'Lilliana%'
      Y en el color <> 'blanco';

+ ---- + ------- + -------- + ------- +
| id | estilo | color | propietario |
+ ---- + ------- + -------- + ------- +
```


Página 30

Buscando en dos claves

```
| 4 | vestido | naranja | 2 |
| 5 | polo | rojo | 2 |
| 6 | vestido | azul | 2 |
+ ---- + ------- + -------- + ------- +
```

Cuando se usa de esta manera, la cláusula REFERENCES no se muestra en la salida de SHOW CREATE TABLE o DESCRIBA :

```
MOSTRAR CREAR TABLA camisa \ G
*************************** 1. fila ******************** *******
Mesa: camisa
Crear tabla: CREAR TABLA `camisa` (
`id` smallint (5) unsigned NOT NULL auto_increment,
`style` enum ('camiseta', 'polo', 'vestido') NOT NULL,
`color` enum ('rojo', 'azul', 'naranja', 'blanco', 'negro') NOT NULL,
`owner` smallint (5) unsigned NOT NULL,
CLAVE PRIMARIA (`id`)
) ENGINE = MyISAM DEFAULT CHARSET = latin1
```

El uso de REFERENCIAS de esta manera como comentario o "recordatorio" en una definición de columna funciona con MyISAM mesas.

3.6.7 Búsqueda en dos claves

Un quirófano que utiliza una sola tecla está bien optimizado, al igual que el manejo de AND .

El único caso complicado es el de buscar en dos claves diferentes combinadas con OR :

```
SELECT field1_index, field2_index FROM test_table
DONDE field1_index = '1' O field2_index = '1'
```

Este caso está optimizado. Consulte la Sección 8.2.1.3, “Optimización de la combinación de índices” .

También puede resolver el problema de manera eficiente utilizando UNION que combine la salida de dos SELECCIONAR declaraciones. Consulte la Sección 13.2.9.3, “Cláusula UNION” .

Cada SELECT busca solo una tecla y se puede optimizar:

```
SELECT field1_index, field2_index
      FROM test_table DONDE field1_index = '1'
UNIÓN
SELECT field1_index, field2_index
      FROM test_table DONDE field2_index = '1';
```

3.6.8 Cálculo de visitas por día

El siguiente ejemplo muestra cómo puede utilizar las funciones de grupo de bits para calcular el número de días por mes en que un usuario ha visitado una página web.

```
CREAR TABLA t1 (año AÑO, mes INT SIN FIRMAR,
                día INT NO FIRMADO);
INSERTAR EN t1 VALORES (2000,1,1), (2000,1,20), (2000,1,30), (2000,2,2),
```

(2000,2,23), (2000,2,23);

La tabla de ejemplo contiene valores año-mes-día que representan las visitas de los usuarios a la página. Para determinar cuántos días diferentes en cada mes ocurren estas visitas, use esta consulta:

```
SELECCIONE año, mes, BIT_COUNT (BIT_OR (1 << día)) COMO días DESDE t1
        GRUPO POR año, mes;
```

Que devuelve:

288

Página 31

Usando AUTO_INCREMENT

```
+ ------ + ------- + ------ +
| año | mes | días |
+ ------ + ------- + ------ +
| 2000 | 1 | 3 |
| 2000 | 2 | 2 |
+ ------ + ------- + ------ +
```

La consulta calcula cuántos días diferentes aparecen en la tabla para cada combinación de año / mes, con eliminación automática de entradas duplicadas.

3.6.9 Uso de AUTO_INCREMENT

El atributo AUTO_INCREMENT se puede utilizar para generar una identidad única para nuevas filas:

```
CREAR TABLA animales (
        id MEDIUMINT NOT NULL AUTO_INCREMENT,
        nombre CHAR (30) NOT NULL,
        LLAVE PRIMARIA (id)
);

INSERTAR EN los animales (nombre) VALORES
      ('perro'), ('gato'), ('pingüino'),
      ('laxo'), ('ballena'), ('avestruz');

SELECCIONAR * DE animales;
```

Que devuelve:

```
+ ---- + --------- +
| id | nombre |
+ ---- + --------- +
| 1 | perro |
| 2 | gato |
| 3 | pingüino |
| 4 | laxo |
| 5 | ballena |
| 6 | avestruz |
+ ---- + --------- +
```

No se especificó ningún valor para la columna AUTO_INCREMENT , por lo que MySQL asignó números de secuencia automáticamente. También puede asignar explícitamente 0 a la columna para generar números de secuencia, a menos que NO_AUTO_VALUE_ON_ZERO El modo SQL está habilitado. Por ejemplo:

```
INSERT INTO animales (id, nombre) VALORES (0, 'marmota');
```

Si la columna se declara NO NULL , también es posible asignar NULL a la columna para generar la secuencia números. Por ejemplo:

```
INSERT INTO animals (id, name) VALUES (NULL, 'ardilla');
```

Cuando inserta cualquier otro valor en una columna AUTO_INCREMENT , la columna se establece en ese valor y la secuencia se restablece de modo que el siguiente valor generado automáticamente siga secuencialmente desde el mayor valor de columna. Por ejemplo:

```
INSERTE EN animales (id, nombre) VALORES (100, 'conejo');
INSERT INTO animals (id, name) VALUES (NULL, 'mouse');
SELECCIONAR * DE animales;
+ ----- + ---------- +
| id | nombre |
+ ----- + ---------- +
```

```
| 1 | perro |
| 2 | gato |
| 3 | pingüino |
| 4 | flojo |
```


Página 32


```
| 5 | ballena |
| 6 | avestruz |
| 7 | marmota
| 8 | ardilla
| 100 | conejo |
| 101 | ratón |
+ ----- + ---------- +
```

Actualizar el valor de una columna AUTO_INCREMENT existente en una tabla InnoDB no restablece el AUTO_INCREMENT secuencia como lo hace para las tablas MyISAM y NDB .

Puede recuperar el valor AUTO_INCREMENT generado automáticamente más reciente con el LAST_INSERT_ID () función SQL o la función de API de C mysql_insert_id () . Estas funciones son específico de la conexión, por lo que sus valores de retorno no se ven afectados por otra conexión que también está realizando inserciones.

Utilice el tipo de datos entero más pequeño para la columna AUTO_INCREMENT que sea lo suficientemente grande para contener el valor de secuencia máximo que necesitará. Cuando la columna alcanza el límite superior del tipo de datos, el el próximo intento de generar un número de secuencia falla. Utilice el atributo UNSIGNED si es posible para permitir un mayor alcance. Por ejemplo, si usa TINYINT , el número de secuencia máximo permitido es 127. Para TINYINT SIN FIRMAR , el máximo es 255. Consulte la Sección 11.1.2, “Tipos de enteros (valor exacto) - INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT ” para los rangos de todos los tipos de enteros.

> **Nota**
>
> Para una inserción de varias filas, LAST_INSERT_ID () y mysql_insert_id () en realidad devuelve la clave AUTO_INCREMENT de la *primera* de las filas insertadas. Esto permite inserciones de varias filas para que se reproduzcan correctamente en otros servidores en una replicación preparar.

Para comenzar con un valor AUTO_INCREMENT distinto de 1, establezca ese valor con CREATE TABLE o ALTER TABLA , así:

```
mysql> ALTER TABLE tbl AUTO_INCREMENT = 100;
```

Notas InnoDB

Para obtener información sobre el uso de AUTO_INCREMENT específico de InnoDB , consulte la Sección 14.6.1.6, “Manejo AUTO_INCREMENT en InnoDB” .

Notas de MyISAM

- Para las tablas MyISAM , puede especificar AUTO_INCREMENT en una columna secundaria en una índice de columna. En este caso, el valor generado para la columna AUTO_INCREMENT se calcula como MAX (*auto_increment_column*) + 1 WHERE prefijo = *prefijo-dado* . Esto es útil cuando quiere poner los datos en grupos ordenados.

```
CREAR TABLA animales (
        grp ENUM ('pez', 'mamífero', 'pájaro') NOT NULL,
        id MEDIUMINT NOT NULL AUTO_INCREMENT,
        nombre CHAR (30) NOT NULL,
        CLAVE PRIMARIA (grp, id)
) MOTOR = MyISAM;

INSERTAR EN ANIMALES (GRP, nombre) VALORES
        ('mamífero', 'perro'), ('mamífero', 'gato'),
        ('pájaro', 'pingüino'), ('pez', 'laxo'), ('mamífero', 'ballena'),
        ('pájaro', 'avestruz');

SELECCIONAR * DE animales ORDENAR POR grp, id;
```

Página 33

Que devuelve:

```
+ -------- + ---- + --------- +
| grp | id | nombre |
+ -------- + ---- + --------- +
| pescado | 1 | laxo |
| mamífero | 1 | perro |
| mamífero | 2 | gato |
| mamífero | 3 | ballena |
| pájaro | 1 | pingüino |
| pájaro | 2 | avestruz |
+ -------- + ---- + --------- +
```

En este caso (cuando la columna AUTO_INCREMENT es parte de un índice de varias columnas), AUTO_INCREMENT los valores se reutilizan si elimina la fila con el valor AUTO_INCREMENT más grande de cualquier grupo. Esta ocurre incluso para las tablas MyISAM , para las cuales los valores AUTO_INCREMENT normalmente no se reutilizan.

- Si la columna AUTO_INCREMENT es parte de varios índices, MySQL genera valores de secuencia usando el índice que comienza con la columna AUTO_INCREMENT , si hay una. Por ejemplo, si los animales tabla contenía índices PRIMARY KEY (grp, id) e INDEX (id) , MySQL ignoraría el LLAVE PRIMARIA para generar valores de secuencia. Como resultado, la tabla contendría una sola secuencia, no una secuencia por valor de grp .

Otras lecturas

Más información sobre AUTO_INCREMENT está disponible aquí:

- Cómo asignar el atributo AUTO_INCREMENT a una columna: Sección 13.1.18, “CREAR TABLA Declaración ” y Sección 13.1.8,“ Declaración ALTER TABLE ” .
- Cómo se comporta AUTO_INCREMENT según el modo SQL NO_AUTO_VALUE_ON_ZERO : Sección 5.1.10, “Modos SQL del servidor” .
- Cómo usar la función LAST_INSERT_ID () para encontrar la fila que contiene el más reciente Valor AUTO_INCREMENT : Sección 12.16, “Funciones de información” .
- Configuración del valor AUTO_INCREMENT que se utilizará: Sección 5.1.7, “Variables del sistema del servidor” .
- Sección 14.6.1.6, “Manejo AUTO_INCREMENT en InnoDB”
- AUTO_INCREMENT y replicación: Sección 16.4.1.1, “Replicación y AUTO_INCREMENT” .
- Variables del sistema del servidor relacionadas con AUTO_INCREMENT (auto_increment_increment y auto_increment_offset) que se pueden utilizar para la replicación: Sección 5.1.7, “Variables del sistema del servidor” .

3.7 Usando MySQL con Apache

Hay programas que le permiten autenticar a sus usuarios desde una base de datos MySQL y también le permiten escribir sus archivos de registro en una tabla MySQL.

Puede cambiar el formato de registro de Apache para que MySQL lo pueda leer fácilmente poniendo lo siguiente en el archivo de configuración de Apache:

```
LogFormat \
        "\"% h \ ",% {% Y% m% d% H% M% S} t,%> s, \"% b \ ", \"% {Content-Type} o \ ", \
        \ "% U \", \ "% {Referer} i \", \ "% {User-Agent} i \" "
```

Para cargar un archivo de registro en ese formato en MySQL, puede usar una declaración como esta:

Página 34


```
CARGAR DATOS INFILE ' / local / access_log ' EN LA TABLA tbl_name
CAMPOS TERMINADOS POR ',' OPCIONALMENTE ENCERRADOS POR '"' ESCAPADOS POR '\\'
```

La tabla nombrada debe crearse para tener columnas que correspondan a las que la línea LogFormat escribe en el archivo de registro.