

MySQL

02_ Base de datos y tablas



MySQL_ Bases de datos y tablas

Contenido

| | |
|---|----|
| 1. Una base de datos es un conjunto de tablas. | 4 |
| 2. Creación de una tabla y mostrar sus campos | 9 |
| 3. Carga de registros a una tabla y su recuperación | 18 |
| 4. Cláusula limit del comando select. | 20 |
| 5. Tipos de datos básicos de un campo de una tabla. | 25 |
| 6. Recuperación selectiva de algunos campos (select) | 30 |
| 7. Recuperación de registros específicos (select - where) | 34 |
| 8. Borrado de registros de una tabla (delete) | 37 |
| 9. Modificación de registros de una tabla (update) | 43 |
| 10. Operadores Relacionales = <> < <= > >= | 46 |
| 11. Truncar Tabla | 49 |

MySQL_ Bases de datos y tablas

Introducción

MySQL es un sistema de gestión de bases de datos relacionales que permite la creación, manipulación y consulta de bases de datos y tablas a través de comandos SQL.

En MySQL, una base de datos es un contenedor para un conjunto de tablas, que a su vez contienen los datos que se van a almacenar. Para crear una base de datos en MySQL, se utiliza el comando `CREATE DATABASE` seguido del nombre de la base de datos deseada.

Una tabla en MySQL es una estructura de datos que contiene filas y columnas. Las columnas representan los diferentes tipos de datos que se almacenan en la tabla, mientras que las filas representan los datos individuales. Para crear una tabla en MySQL, se utiliza el comando `CREATE TABLE`, que permite especificar los nombres y tipos de datos de las columnas.

Además, MySQL permite la definición de claves primarias y restricciones de integridad referencial para garantizar la consistencia y calidad de los datos almacenados en las tablas.

En resumen, MySQL es una herramienta poderosa para la gestión de bases de datos relacionales y la creación y manipulación de tablas es una parte fundamental de su uso.



MySQL_ Bases de datos y tablas

1. Una base de datos es un conjunto de tablas.

Una base de datos tiene un nombre con el cual accederemos a ella.

Para que el servidor nos muestre las bases de datos existentes, se lo solicitamos enviando la instrucción:

```
show databases;
```

Nos mostrará los nombres de las bases de datos.

Recordemos que para crear una nueva base de datos utilizamos el comando **create**

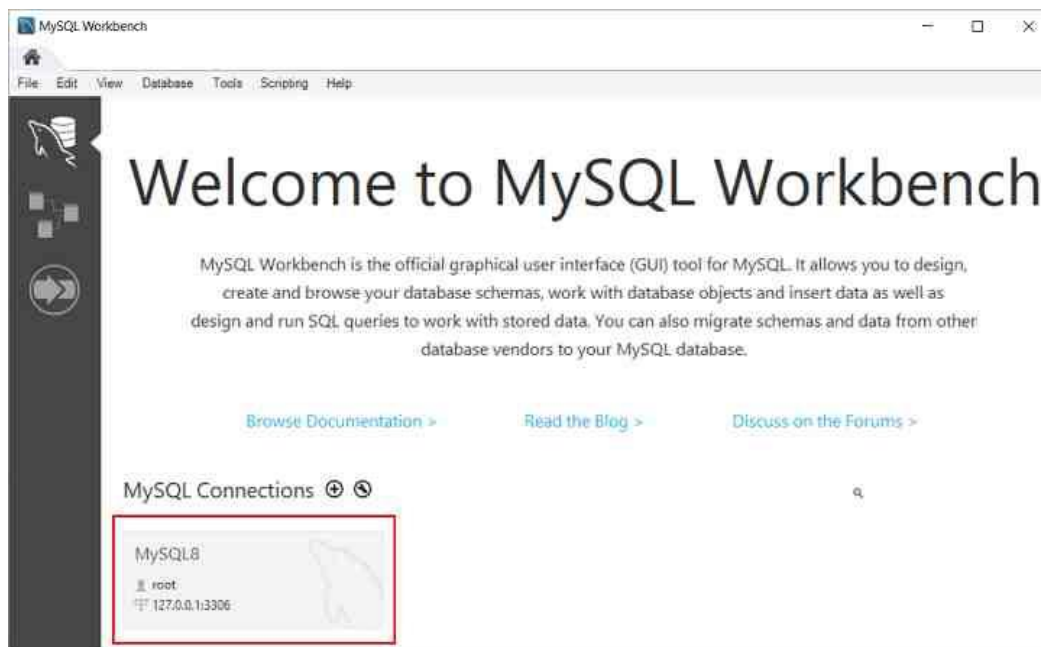
```
create database Ejemplo;
```

Servidor de MySQL instalado en forma local

Si instaló el MySQL en su equipo debemos ingresar desde el menú de opciones de Windows al programa "MySQL Workbench" para desarrollar este curso:

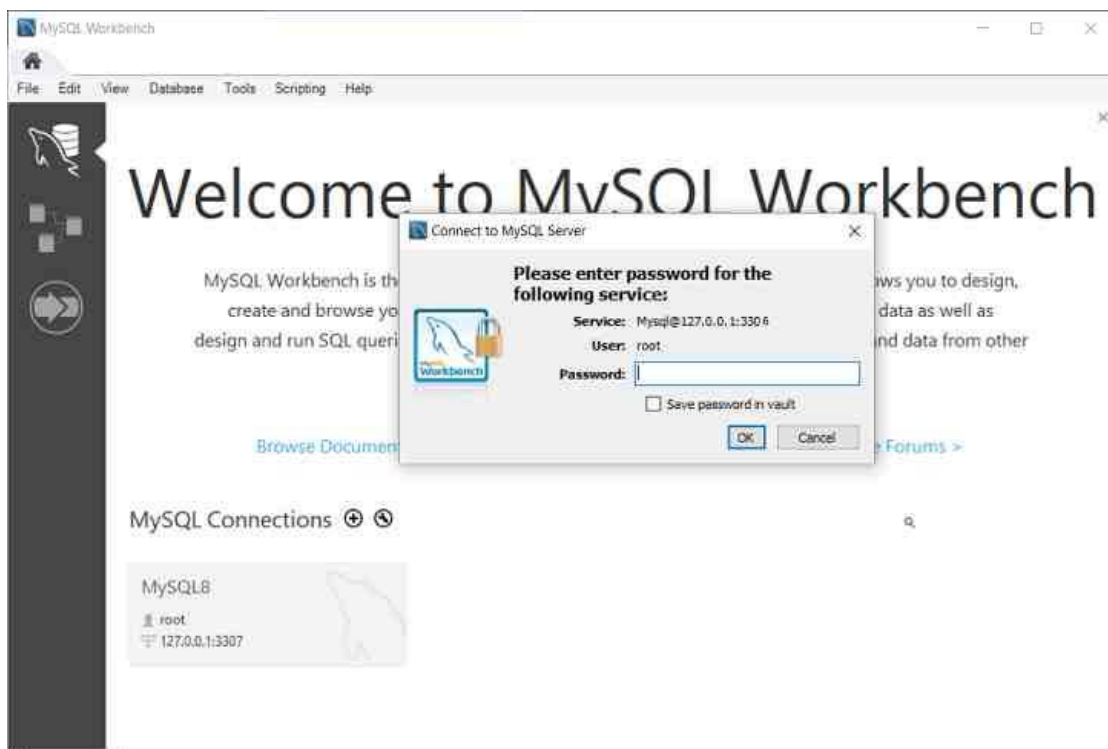


Cuando se inicia el programa "Workbench" debemos conectarnos al servidor de MySQL con el usuario "root" en el servidor local: 127.0.0.1 y el puerto: 3306:

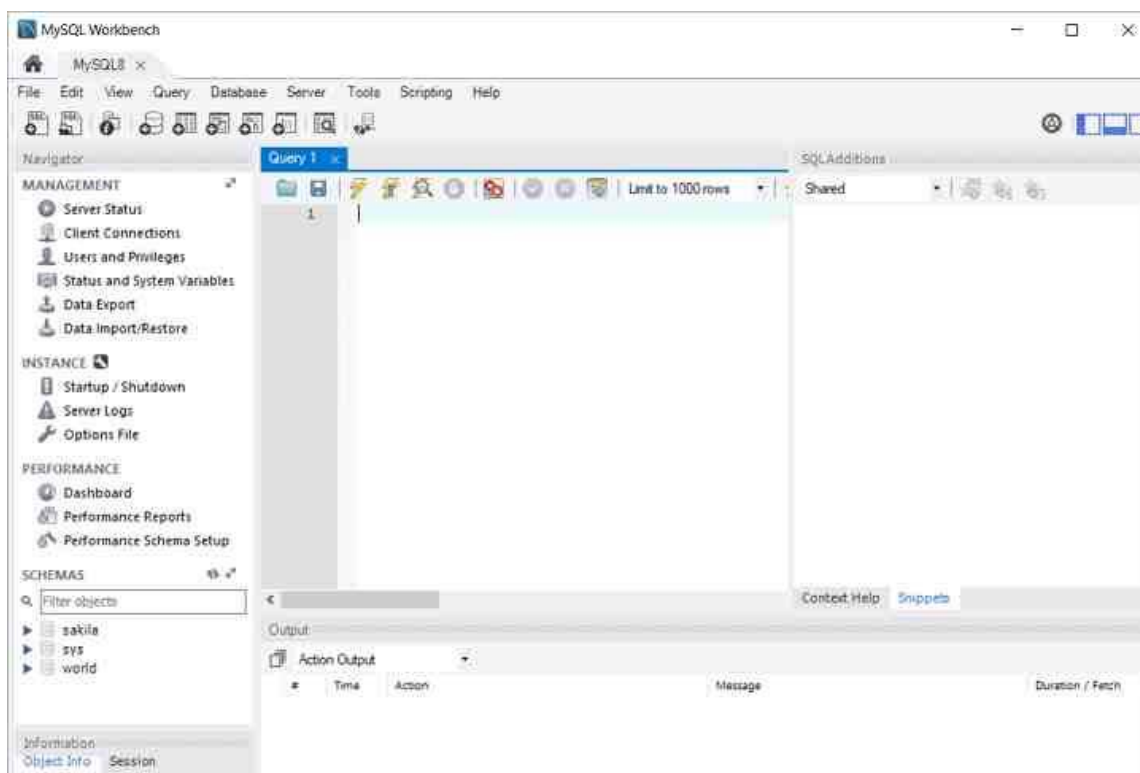


MySQL_ Bases de datos y tablas

Previo a ingresar se nos pide la clave del usuario "root":

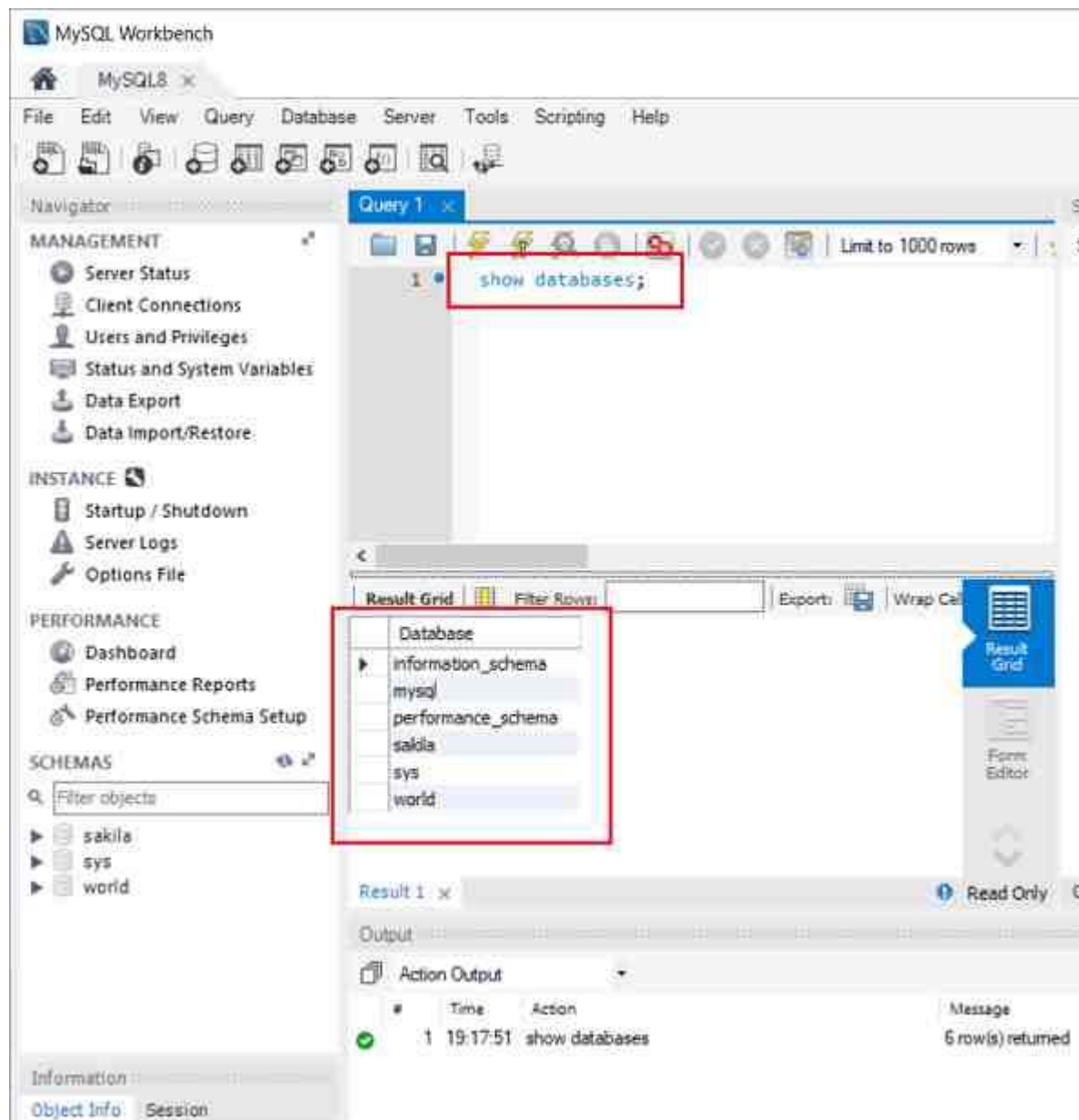


La ventana principal del programa "Workbench" es:



MySQL_ Bases de datos y tablas

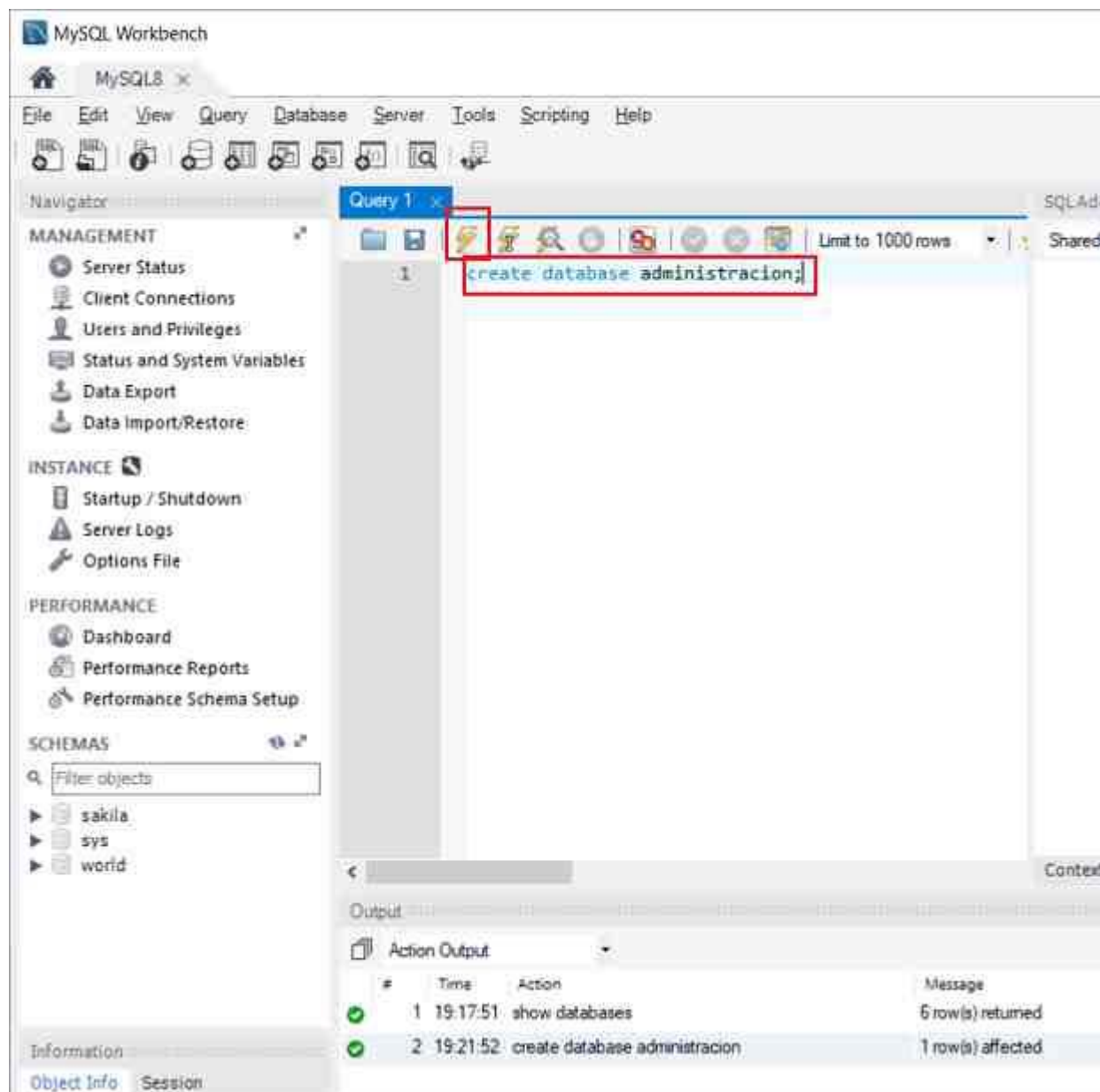
En la ventana de Query 1 escribimos el comando "show databases;":



MySQL_ Bases de datos y tablas

Se muestran 6 bases de datos creadas cuando instalamos el MySQL pero no la base de datos "administración". Debemos nosotros crear la base de datos "administración" mediante el comando "create database administracion;" y presionar el ícono de ejecución:

```
create database administracion;
```



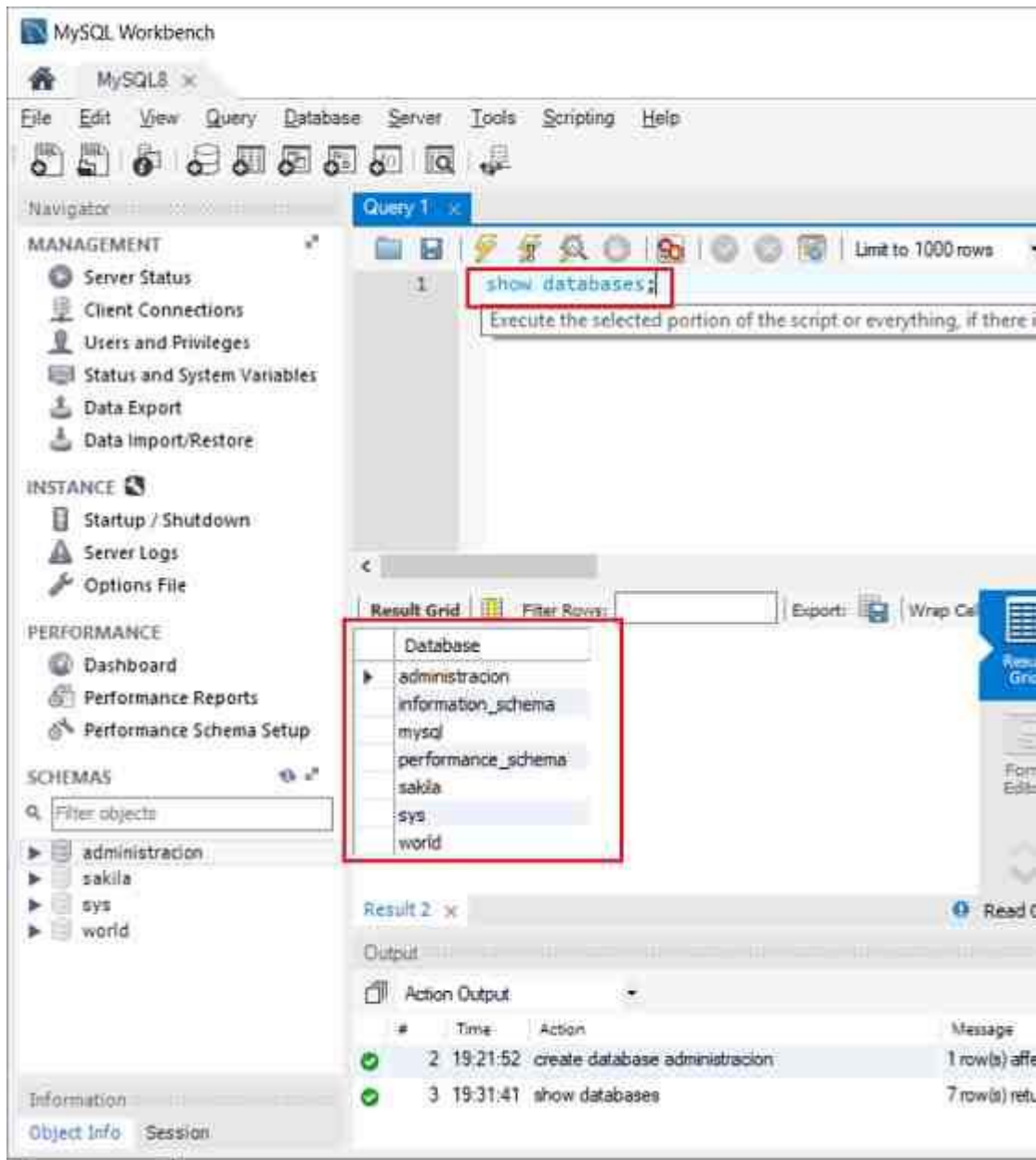
MySQL_ Bases de datos y tablas

Luego presionamos el ícono de actualizar "schemas" y aparecerá listada la nueva base de datos llamada "administración":



MySQL_ Bases de datos y tablas

Si ejecutamos nuevamente el comando "show databases;" debe aparecer la nueva base de datos creada:



MySQL_ Bases de datos y tablas

2. Creación de una tabla y mostrar sus campos

create table - show tables - describe - drop table

Una base de datos almacena sus datos en tablas.

Una tabla es una estructura de datos que organiza los datos en columnas y filas; cada columna es un campo (o atributo) y cada fila, un registro. La intersección de una columna con una fila, contiene un dato específico, un solo valor.

Cada registro contiene un dato por cada columna de la tabla.

Cada campo (columna) debe tener un nombre. El nombre del campo hace referencia a la información que almacenará.

Cada campo (columna) también debe definir el tipo de dato que almacenará.

| Nombre | clave |
|----------------|-------|
| MarioPerez | Mario |
| MariaGarcia | Mary |
| DiegoRodriguez | z8080 |

Gráficamente aquí tenemos la tabla usuarios, que contiene dos campos llamados: nombre y clave. Luego tenemos tres registros almacenados en esta tabla, el primero almacena en el campo nombre el valor "MarioPerez" y en el campo clave "Mario", y así sucesivamente con los otros dos registros.

Las tablas forman parte de una base de datos.

Para ver las tablas existentes en una base de datos tipeamos:

```
show tables;
```

Deben aparecer todas las tablas que han creado los visitantes al sitio web2020

Al crear una tabla debemos resolver qué campos (columnas) tendrá y que tipo de datos almacenarán cada uno de ellos, es decir, su estructura.

La tabla debe ser definida con un nombre que la identifique y con el cual accederemos a ella.

Creamos una tabla llamada "usuarios", tipeamos:

```
create table usuarios (  
    nombre varchar(30),  
    clave varchar(10)  
);
```

MySQL_ Bases de datos y tablas

Si intentamos crear una tabla con un nombre ya existente (existe otra tabla con ese nombre), mostrará un mensaje de error indicando que la acción no se realizó porque ya existe una tabla con el mismo nombre.

Para ver las tablas existentes en una base de datos tipeamos nuevamente:

```
show tables;
```

Ahora aparece "usuarios" entre otras que ya pueden estar creadas.

Cuando se crea una tabla debemos indicar su nombre y definir sus campos con su tipo de dato. En esta tabla "usuarios" definimos 2 campos:

- nombre: que contendrá una cadena de hasta 30 caracteres de longitud, que almacenará el nombre de usuario y
- clave: otra cadena de caracteres de 10 de longitud, que guardará la clave de cada usuario.

Cada usuario ocupará un registro de esta tabla, con su respectivo nombre y clave.

Para ver la estructura de una tabla usamos el comando "describe" junto al nombre de la tabla:

```
describe usuarios;
```

Aparece lo siguiente:

| Field | Type | Null |
|--------|-------------|------|
| nombre | varchar(30) | YES |
| clave | varchar(10) | YES |

Esta es la estructura de la tabla "usuarios"; nos muestra cada campo, su tipo, lo que ocupa en bytes y otros datos como la aceptación de valores nulos etc, que veremos más adelante en detalle.

Para eliminar una tabla usamos "drop table". Tipeamos:

```
drop table usuarios;
```

Si tipeamos nuevamente:

```
drop table usuarios;
```

Aparece un mensaje de error, indicando que no existe, ya que intentamos borrar una tabla inexistente.

Para evitar este mensaje podemos tipear:

```
drop table if exists usuarios;
```

MySQL_ Bases de datos y tablas

En la sentencia precedente especificamos que elimine la tabla "usuarios" si existe.

Servidor de MySQL instalado en forma local.

Cada vez que ingresamos a "Workbench" debemos seleccionar la base de datos con la que vamos a trabajar, mediante la cláusula "use" y posteriormente ejecutar comandos SQL sobre dicha base de datos:

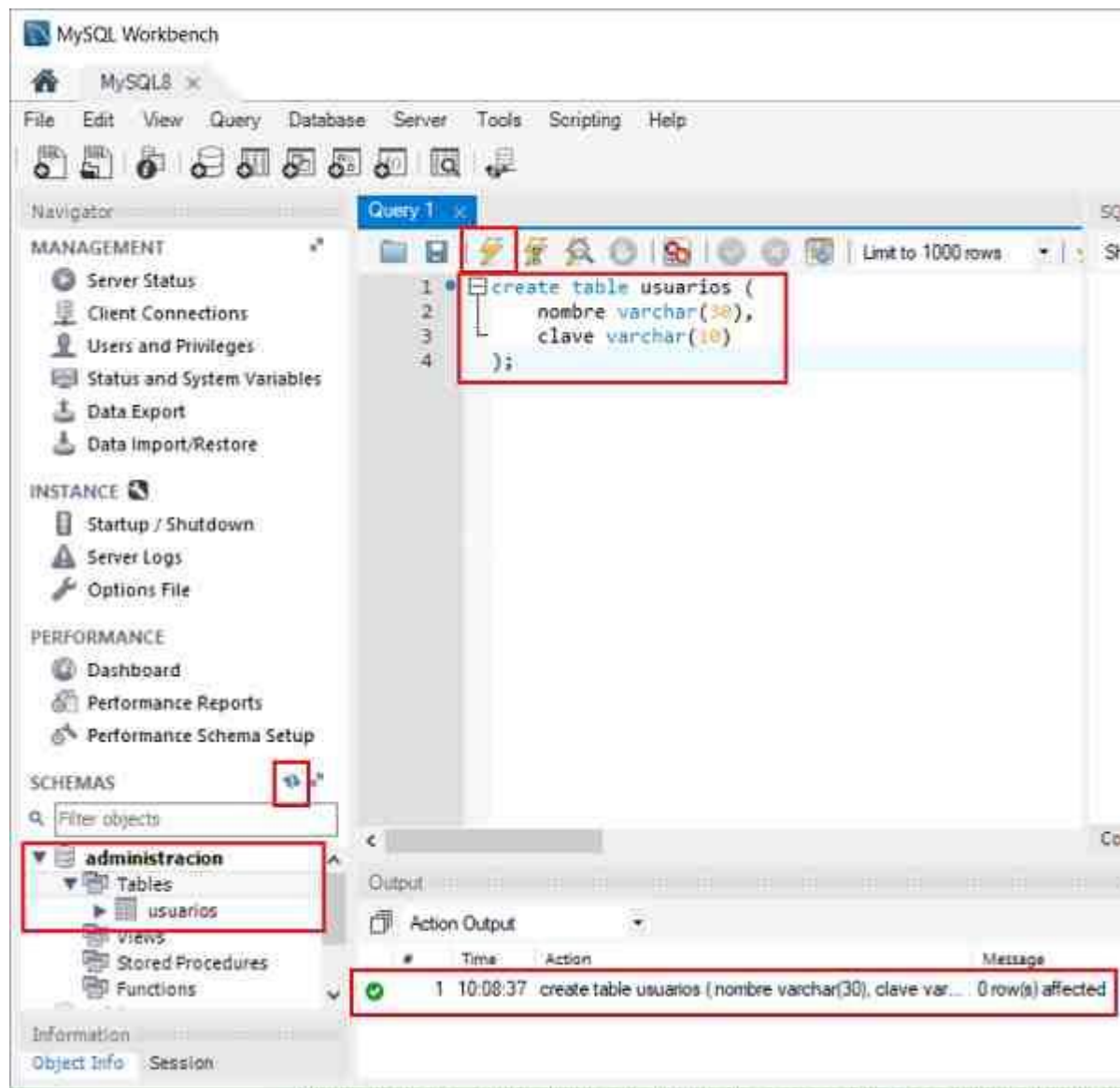
```
use administracion;
```

También podemos hacer doble clic en la ventana "SCHEMAS" para su selección:



MySQL_ Bases de datos y tablas

Creamos la tabla "usuarios":



Cada vez que escribimos un comando "SQL" como en este caso el "create table" debemos presionar el ícono de "rayo" para que lo ejecute, en la parte inferior se nos informa si hay algún error y en la ventana de "SCHEMAS" la debemos actualizar para que se vean los cambios en la base de datos "administracion".

MySQL_ Bases de datos y tablas

Probemos ahora de ejecutar el comando SQL "describe" para conocer la estructura de la tabla "usuarios" que acabamos de crear:

The screenshot shows the MySQL Workbench interface. In the 'Query 1' editor, the command `describe usuarios;` is entered. The 'Result Grid' displays the structure of the 'usuarios' table:

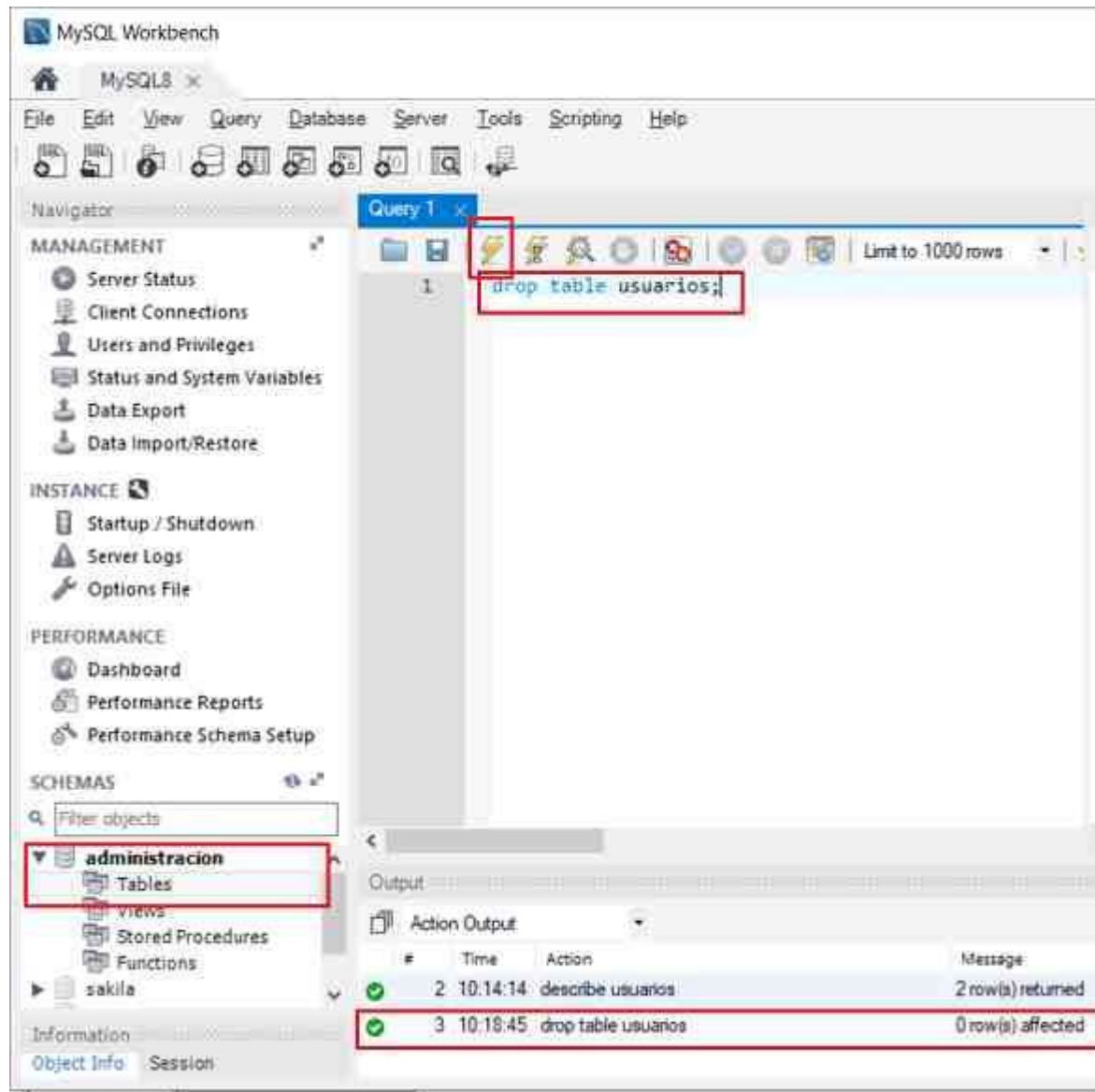
| Field | Type | Null | Key | Default | Extra |
|--------|-------------|------|-----|---------|-------|
| nombre | varchar(30) | YES | | | |
| clave | varchar(10) | YES | | | |

Below the Result Grid, the 'Action Output' pane shows the execution log:

| # | Time | Action | Message |
|---|----------|--|-------------------|
| 1 | 10:08:37 | create table usuarios (nombre varchar(30), clave v... | 0 row(s) affected |
| 2 | 10:14:14 | describe usuarios | 2 row(s) returned |

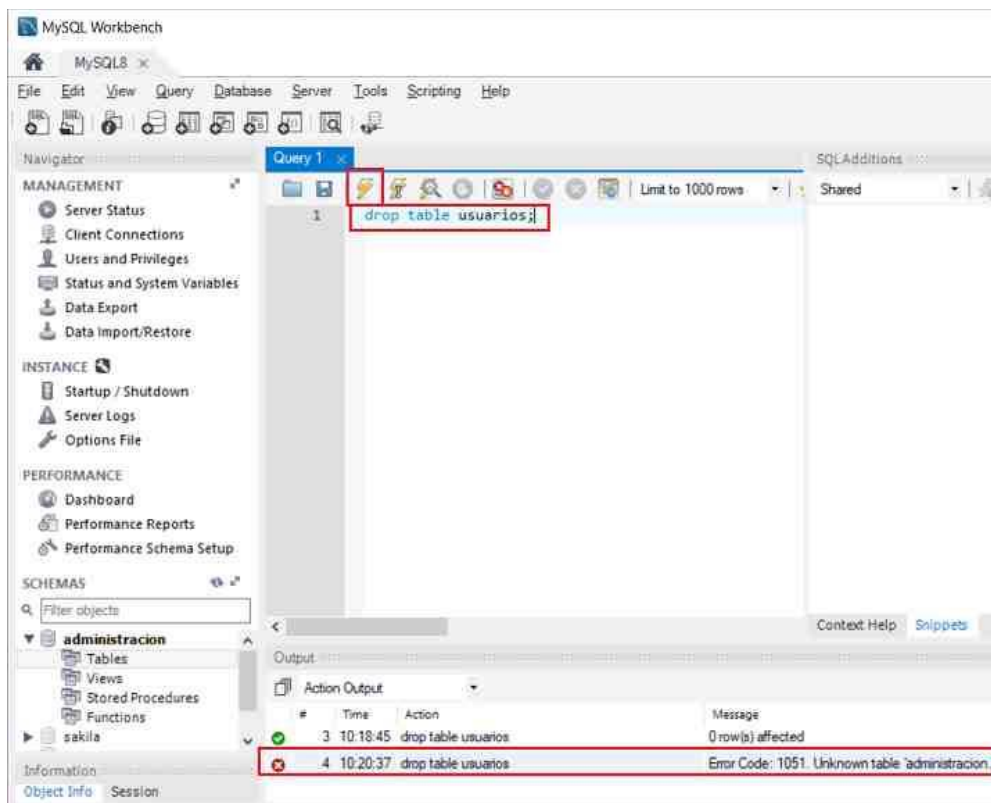
MySQL_ Bases de datos y tablas

Borremos la tabla "usuarios" mediante el comando "drop table" que acabamos de crear:



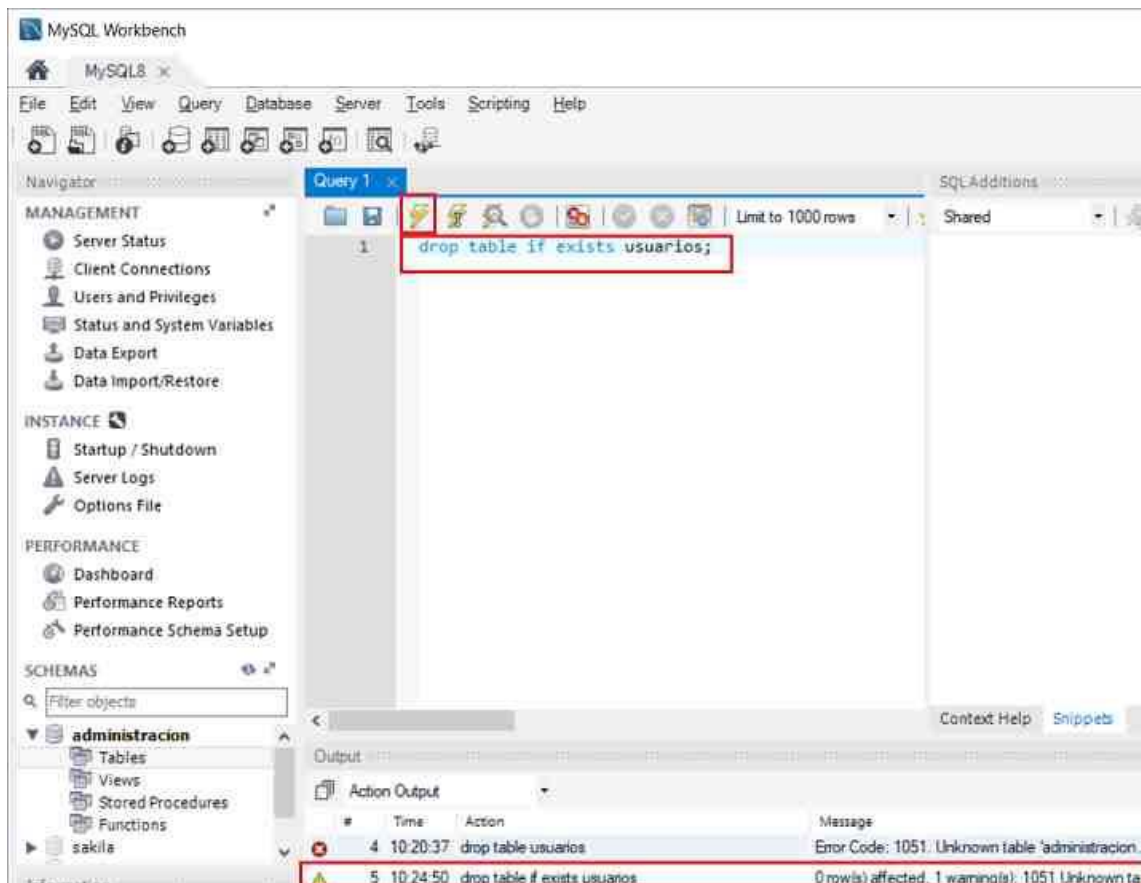
MySQL_ Bases de datos y tablas

Intentemos de borrarla nuevamente a la tabla "usuarios" mediante el comando "drop table", veremos que se nos informa de un error:



MySQL_ Bases de datos y tablas

El resultado es distinto si utilizamos la variante de "drop table if exists usuarios;", en dicho caso no se informa un error ya que valida primero que la tabla exista para su borrado:



3. Carga de registros a una tabla y su recuperación

insert into - select

Un registro es una fila de la tabla que contiene los datos propiamente dichos. Cada registro tiene un dato por cada columna.

Recordemos como crear la tabla "usuarios":

```
create table usuarios (  
    nombre varchar(30),  
    clave varchar(10)  
);
```

Al ingresar los datos de cada registro debe tenerse en cuenta la cantidad y el orden de los campos.

Ahora vamos a agregar un registro a la tabla:

```
insert into usuarios (nombre, clave) values ('MarioPerez','Mario');
```

Usamos "insert into". Especificamos los nombres de los campos entre paréntesis y separados por comas y luego los valores para cada campo, también entre paréntesis y separados por comas.

La tabla usuarios ahora la podemos graficar de la siguiente forma:

| nombre | clave |
|------------|-------|
| MarioPerez | Mario |

Es importante ingresar los valores en el mismo orden en que se nombran los campos, si ingresamos los datos en otro orden, no aparece un mensaje de error y los datos se guardan de modo incorrecto.

Note que los datos ingresados, como corresponden a campos de cadenas de caracteres se colocan entre comillas simples. Las comillas simples son OBLIGATORIAS.

Para ver los registros de una tabla usamos "select":

```
select nombre,clave from usuarios;
```

Aparece un registro.

MySQL_ Bases de datos y tablas

El comando "select" recupera los registros de una tabla. Luego del comando select indicamos los nombres de los campos a rescatar.

Servidor de MySQL instalado en forma local.

Si tenemos el servidor de base de datos MySQL instalado en nuestra computadora ingresemos ahora al programa "Workbench" para ejecutar los comandos SQL que acabamos de ver:

The screenshot shows the MySQL Workbench interface. The left sidebar contains the 'MANAGEMENT' section with options like 'Server Status', 'Client Connections', and 'Users and Privileges'. The 'SCHEMAS' section shows a tree view with 'administracion' selected. The main window displays a SQL query in the 'Query 1' editor:

```
1 drop table if exists usuarios;
2
3 create table usuarios (
4     nombre varchar(30),
5     clave varchar(10)
6 );
7
8 insert into usuarios(nombre, clave) values ('MarioPerez','Marito');
9
10 select nombre, clave from usuarios;
```

Below the query editor, the 'Result Grid' shows the output of the last query:

| nombre | clave |
|------------|--------|
| MarioPerez | Marito |

The bottom panel shows the 'Action Output' section with a table of execution results:

| # | Time | Action | Message |
|----|----------|--|-------------------|
| 9 | 11:43:53 | insert into usuarios(nombre, clave) values ('MarioPerez',... | 1 row(s) affected |
| 10 | 11:43:53 | select nombre, clave from usuarios LIMIT 0, 1000 | 1 row(s) returned |

4. Cláusula limit del comando select.

La cláusula "limit" se usa para restringir los registros que se retornan en una consulta "select".

Recibe 1 ó 2 argumentos numéricos enteros positivos; el primero indica el número del primer registro a retornar, el segundo, el número máximo de registros a retornar. El número de registro inicial es 0 (no 1).

Si el segundo argumento supera la cantidad de registros de la tabla, se limita hasta el último registro.

Ejemplo:

```
select * from libros limit 0,4;
```

Muestra los primeros 4 registros, 0,1,2 y 3.

Si tipeamos:

```
select * from libros limit 5,4;
```

recuperamos 4 registros, desde el 5 al 8.

Si se coloca un solo argumento, indica el máximo número de registros a retornar, comenzando desde 0. Ejemplo:

```
select * from libros limit 8;
```

Muestra los primeros 8 registros.

Para recuperar los registros desde cierto número hasta el final, se puede colocar un número grande para el segundo argumento:

```
select * from libros limit 6,10000;
```

recupera los registros 7 al último.

"limit" puede combinarse con el comando "delete". Si queremos eliminar 2 registros de la tabla "libros" Usamos:

```
delete from libros  
limit 2;
```

Podemos ordenar los registros por precio (por ejemplo) y borrar 2:

```
delete from libros  
order by precio
```

MySQL_ Bases de datos y tablas

```
limit 2;
```

esta sentencia borrará los 2 primeros registros, es decir, los de precio más bajo.

Podemos emplear la cláusula "limit" para eliminar registros duplicados. Por ejemplo, continuamos con la tabla "libros" de una librería, ya hemos almacenado el libro "El aleph" de "Borges" de la editorial "Planeta", pero nos equivocamos y volvemos a ingresar el mismo libro, del mismo autor y editorial 2 veces más, es un error que no controla MySQL. Para eliminar el libro duplicado y que sólo quede un registro de él vemos cuántos tenemos:

```
select * from libros  
where titulo='El aleph' and  
autor='Borges' and  
editorial='Planeta';
```

Luego eliminamos con "limit" la cantidad sobrante (tenemos 3 y queremos solo 1):

```
delete from libros  
where titulo='El aleph' and  
autor='Borges' and  
editorial='Planeta'  
limit 2;
```

Un mensaje nos muestra la cantidad de registros eliminados.

Es decir, con "limit" indicamos la cantidad a eliminar.

Veamos cuántos hay ahora:

```
select * from libros  
where titulo='El aleph' and  
autor='Borges' and  
editorial='Planeta';
```

Sólo queda 1.

MySQL_ Bases de datos y tablas

Servidor de MySQL instalado en forma local.

Ingrese al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
drop table if exists libros;

create table libros(
  codigo int unsigned auto_increment,
  titulo varchar(40) not null,
  autor varchar(30),
  editorial varchar(15),
  precio decimal(5,2) unsigned,
  primary key (codigo)
);

insert into libros (titulo,autor,editorial,precio)
  values('El aleph','Borges','Planeta',15);
insert into libros (titulo,autor,editorial,precio)
  values('Martin Fierro','Jose Hernandez','Emece',22.20);
insert into libros (titulo,autor,editorial,precio)
  values('Antologia poetica','Borges','Planeta',40);
insert into libros (titulo,autor,editorial,precio)
  values('Aprenda PHP','Mario Molina','Emece',18.20);
insert into libros (titulo,autor,editorial,precio)
  values('Cervantes y el quijote','Borges','Paidos',36.40);
insert into libros (titulo,autor,editorial,precio)
  values('Manual de PHP','J.C. Paez','Paidos',30.80);
insert into libros (titulo,autor,editorial,precio)
  values('Harry Potter y la piedra filosofal','J.K. Rowling','Paidos',45.00);
insert into libros (titulo,autor,editorial,precio)
  values('Harry Potter y la camara secreta','J.K. Rowling','Paidos',46.00);
insert into libros (titulo,autor,editorial,precio)
  values('Alicia en el pais de las maravillas','Lewis Carroll','Paidos',null);

-- recuperar 4 libros desde el registro cero:
select * from libros limit 0,4;

-- recuperar 4 libros a partir del registro 5
select * from libros limit 5,4;

-- recuperar 8 libros desde el principio
select * from libros limit 8;
```

MySQL_ Bases de datos y tablas

```
-- para recuperar 10000 registros o hasta el final de la tabla a partir
-- del registro 6
select * from libros limit 6,10000;
```

```
-- para eliminar los 2 registros con precio más bajo
delete from libros
order by precio
limit 2;
```

```
insert into libros (titulo,autor,editorial,precio)
values('El aleph','Borges','Planeta',15);
```

```
insert into libros (titulo,autor,editorial,precio)
values('El aleph','Borges','Planeta',15);
insert into libros (titulo,autor,editorial,precio)
values('El aleph','Borges','Planeta',15);
```

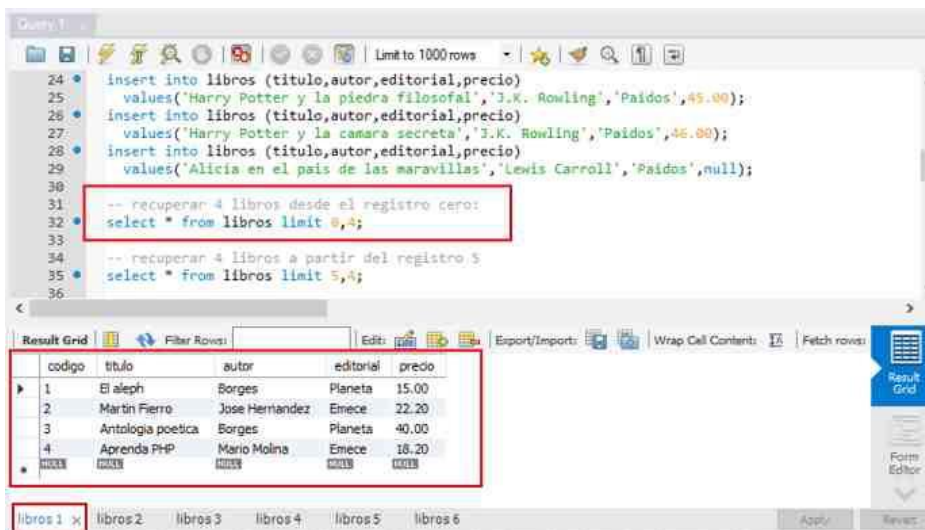
```
select * from libros
where titulo='El aleph' and
autor='Borges' and
editorial='Planeta';
```

```
-- eliminamos 2 registros
delete from libros
where titulo='El aleph' and
autor='Borges' and
editorial='Planeta'
limit 2;
```

```
select * from libros
where titulo='El aleph' and
autor='Borges' and
editorial='Planeta';
```

MySQL_ Bases de datos y tablas

Genera una salida similar a esta:



The screenshot shows a MySQL IDE interface. The top pane displays SQL queries. The bottom pane shows the 'Result Grid' with a table of book data. Red boxes highlight the SQL query for the first four records and the first four rows of the result grid.

```
24 • insert into libros (titulo,autor,editorial,precio)
25 • values('Harry Potter y la piedra filosofal','J.K. Rowling','Paidós',45.00);
26 • insert into libros (titulo,autor,editorial,precio)
27 • values('Harry Potter y la cámara secreta','J.K. Rowling','Paidós',46.00);
28 • insert into libros (titulo,autor,editorial,precio)
29 • values('Alicia en el país de las maravillas','Lewis Carroll','Paidós',null);
30
31 • -- recuperar 4 libros desde el registro cero:
32 • select * from libros limit 0,4;
33
34 • -- recuperar 4 libros a partir del registro 5
35 • select * from libros limit 5,4;
36
```

| codigo | titulo | autor | editorial | precio |
|--------|-------------------|----------------|-----------|--------|
| 1 | El aleph | Borges | Planeta | 15.00 |
| 2 | Martin Fierro | Jose Hernandez | Emece | 22.20 |
| 3 | Antologia poetica | Borges | Planeta | 40.00 |
| 4 | Aprenda PHP | Mario Molina | Emece | 18.20 |
| 5 | ... | ... | ... | ... |

libros 1 x libros 2 libros 3 libros 4 libros 5 libros 6

5. Tipos de datos básicos de un campo de una tabla.

Ya explicamos que al crear una tabla debemos resolver qué campos (columnas) tendrá y que tipo de datos almacenará cada uno de ellos, es decir, su estructura. Estos son algunos tipos de datos básicos:

- **varchar**: se usa para almacenar cadenas de caracteres. Una cadena es una secuencia de caracteres. Se coloca entre comillas (simples): 'Hola'. El tipo "varchar" define una cadena de longitud variable en la cual determinamos el máximo de caracteres. Puede guardar hasta 65535 caracteres (versiones antiguas de MySQL permitían solo 255). Para almacenar cadenas de hasta 30 caracteres, definimos un campo de tipo varchar(30). Si asignamos una cadena de caracteres de mayor longitud que la definida, la cadena se corta. Por ejemplo, si definimos un campo de tipo varchar(10) y le asignamos la cadena 'Buenas tardes', se almacenará 'Buenas tar' ajustándose a la longitud de 10 caracteres.
- **integer**: se usa para guardar valores numéricos enteros, de -2000000000 a 2000000000 aprox. Definimos campos de este tipo cuando queremos representar, por ejemplo, cantidades.
- **float**: se usa para almacenar valores numéricos decimales. Se utiliza como separador el punto (.). Definimos campos de este tipo para precios, por ejemplo.

Antes de crear una tabla debemos pensar en sus campos y optar por el tipo de dato adecuado para cada uno de ellos. Por ejemplo, si en un campo almacenaremos números enteros, el tipo "float" sería una mala elección; si vamos a guardar precios, el tipo "float" es correcto, no así "integer" que no tiene decimales.

MySQL_ Bases de datos y tablas

Servidor de MySQL instalado en forma local.

Crearemos mediante el "Workbench" una tabla en la base de datos "administracion" con una serie de campos de distinto tipo:

```
create table libros(  
    titulo varchar(40),  
    autor varchar(20),  
    editorial varchar(15),  
    precio float,  
    cantidad integer  
);
```

Procederemos a insertar una serie de filas:

```
insert into libros (titulo,autor,editorial,precio,cantidad)  
values ('El aleph','Borges','Emece',45.50,100);  
insert into libros (titulo,autor,editorial,precio,cantidad)  
values ('Alicia en el pais de las maravillas','Lewis Carroll','Planeta',25,200);  
insert into libros (titulo,autor,editorial,precio,cantidad)  
values ('Matematica estas ahi','Paenza','Planeta',15.8,200);
```

Podemos ejecutar todos los comandos al presionar el ícono de "rayo":

The screenshot shows the MySQL Workbench interface. The 'Query 1' tab is active, displaying the following SQL commands:

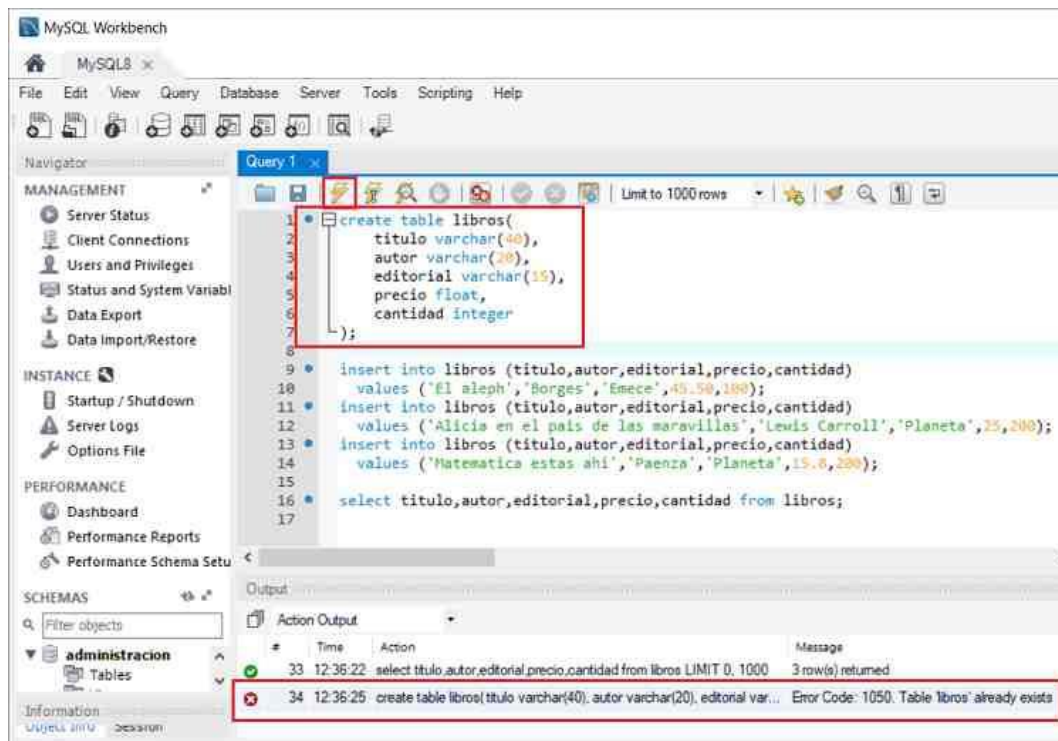
```
1 create table libros(  
2     titulo varchar(40),  
3     autor varchar(20),  
4     editorial varchar(15),  
5     precio float,  
6     cantidad integer  
7 );  
8  
9 insert into libros (titulo,autor,editorial,precio,cantidad)  
10 values ('El aleph','Borges','Emece',45.50,100);  
11 insert into libros (titulo,autor,editorial,precio,cantidad)  
12 values ('Alicia en el pais de las maravillas','Lewis Carroll','Planeta',25,200);  
13 insert into libros (titulo,autor,editorial,precio,cantidad)  
14 values ('Matematica estas ahi','Paenza','Planeta',15.8,200);  
15  
16 select titulo,autor,editorial,precio,cantidad from libros;  
17
```

The 'Result Grid' at the bottom shows the data inserted into the 'libros' table:

| titulo | autor | editorial | precio | cantidad |
|-------------------------------------|---------------|-----------|--------|----------|
| El aleph | Borges | Emece | 45.5 | 100 |
| Alicia en el pais de las maravillas | Lewis Carroll | Planeta | 25 | 200 |
| Matematica estas ahi | Paenza | Planeta | 15.8 | 200 |

MySQL_ Bases de datos y tablas

Tener cuidado de no tratar de ejecutar nuevamente todos los comandos ya que fallará con la primer cláusula "create table" debido a que ya existe la tabla "libros" creada en la base de datos "administracion" (podemos por ejemplo seguir insertando registros en la tabla "libros"):



MySQL_ Bases de datos y tablas

Problema Propuesto

Problema:

Un videoclub que alquila películas en video almacena la información de sus películas en una tabla

llamada "peliculas"; para cada película necesita los siguientes datos:

- nombre, cadena de caracteres de 20 de longitud,
- actor, cadena de caracteres de 20 de longitud,
- duración, valor numérico entero.
- cantidad de copias: valor entero.

1- Elimine la tabla, si existe:

```
drop table if exists peliculas;
```

2- Cree la tabla eligiendo el tipo de dato adecuado para cada campo:

```
create table peliculas(  
  nombre varchar(20),  
  actor varchar(20),  
  duracion integer,  
  cantidad integer  
);
```

3- Vea la estructura de la tabla:

```
describe peliculas;
```

4- Ingrese los siguientes registros:

```
insert into peliculas (nombre, actor, duracion, cantidad)  
values ('Mision imposible','Tom Cruise',120,3);  
insert into peliculas (nombre, actor, duracion, cantidad)  
values ('Mision imposible 2','Tom Cruise',180,2);  
insert into peliculas (nombre, actor, duracion, cantidad)  
values ('Mujer bonita','Julia R.',90,3);  
insert into peliculas (nombre, actor, duracion, cantidad)  
values ('Elsa y Fred','China Zorrilla',90,2);
```

5- Muestre todos los registros:

```
select * from peliculas;
```

Otros problemas:

Una empresa almacena los datos de sus empleados en una tabla "empleados" que guarda los siguientes

datos: nombre, documento, sexo, domicilio, sueldobasico.

MySQL_ Bases de datos y tablas

1- Elimine la tabla, si existe:

```
drop table if exists empleados;
```

2- Cree la tabla eligiendo el tipo de dato adecuado para cada campo:

```
create table empleados(  
  nombre varchar(20),  
  documento varchar(8),  
  sexo varchar(1),  
  domicilio varchar(30),  
  sueldobasico float  
);
```

3- Vea la estructura de la tabla:

```
describe empleados;
```

4- Ingrese algunos registros:

```
insert into empleados (nombre, documento, sexo, domicilio, sueldobasico)  
values ('Juan Perez','22345678','m','Sarmiento 123',300);  
insert into empleados (nombre, documento, sexo, domicilio, sueldobasico)  
values ('Ana Acosta','24345678','f','Colon 134',500);  
insert into empleados (nombre, documento, sexo, domicilio, sueldobasico)  
values ('Marcos Torres','27345678','m','Urquiza 479',800);
```

5- Seleccione todos los registros:

```
select * from empleados;
```

6. Recuperación selectiva de algunos campos (select)

Hemos aprendido cómo ver todos los registros de una tabla:

```
select * from libros;
```

El comando "select" recupera los registros de una tabla. Con el asterisco (*) indicamos que seleccione todos los campos de la tabla que nombramos.

Podemos especificar el nombre de los campos que queremos ver separándolos por comas:

```
select titulo,autor,editorial from libros;
```

En la sentencia anterior la consulta mostrará sólo los campos "titulo", "autor" y "editorial". En la siguiente sentencia, veremos los campos correspondientes al título y precio de todos los libros:

```
select titulo,precio from libros;
```

Para ver solamente la editorial y la cantidad de libros tipeamos:

```
select editorial,cantidad from libros;
```

MySQL_ Bases de datos y tablas

Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y procedamos a crear una tabla, insertar algunas filas y mostrar solo algunas columnas de dicha tabla:

```
drop table if exists libros;

create table libros(
  titulo varchar(100),
  autor varchar(30),
  editorial varchar(15),
  precio float,
  cantidad integer
);

insert into libros (titulo,autor,editorial,precio,cantidad)
  values ('El aleph','Borges','Emece',45.50,100);
insert into libros (titulo,autor,editorial,precio,cantidad)
  values ('Alicia en el pais de las maravillas','Lewis Carroll','Planeta',25,200);
insert into libros (titulo,autor,editorial,precio,cantidad)
  values ('Matematica estas ahi','Paenza','Planeta',15.8,200);

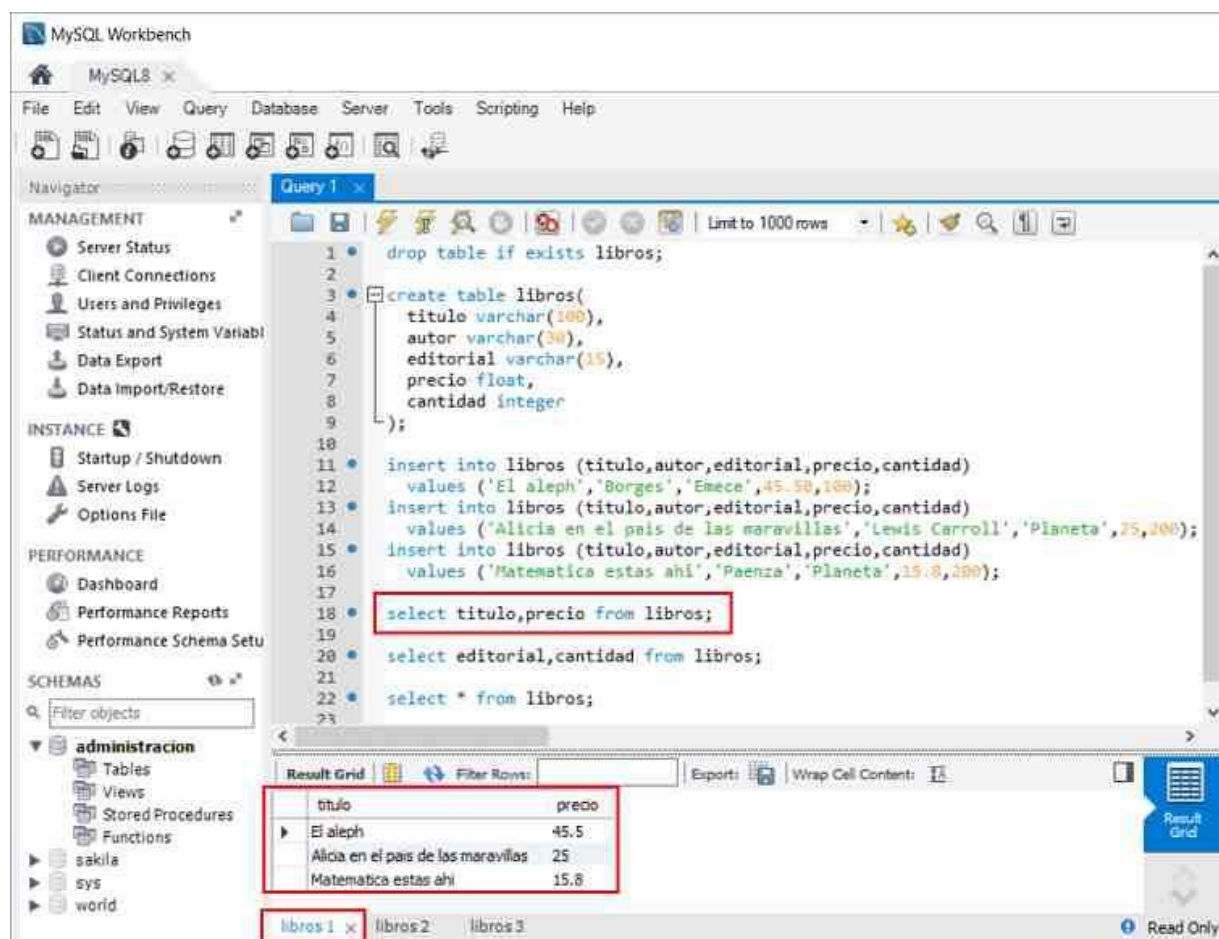
select titulo,precio from libros;

select editorial,cantidad from libros;

select * from libros;
```

MySQL_ Bases de datos y tablas

Cuando lo ejecutamos cada uno de los resultados de los comandos "select" aparece en una pestaña en la parte inferior:



The screenshot shows the MySQL Workbench interface. The main editor displays a SQL script with the following queries:

```
1 drop table if exists libros;
2
3 create table libros(
4     titulo varchar(100),
5     autor varchar(30),
6     editorial varchar(15),
7     precio float,
8     cantidad integer
9 );
10
11 insert into libros (titulo,autor,editorial,precio,cantidad)
12     values ('El aleph','Borges','Emece',45.50,100);
13 insert into libros (titulo,autor,editorial,precio,cantidad)
14     values ('Alicia en el pais de las maravillas','Lewis Carroll','Planeta',25,200);
15 insert into libros (titulo,autor,editorial,precio,cantidad)
16     values ('Matematica estas ahi','Paenza','Planeta',15.8,200);
17
18 select titulo,precio from libros;
19
20 select editorial,cantidad from libros;
21
22 select * from libros;
```

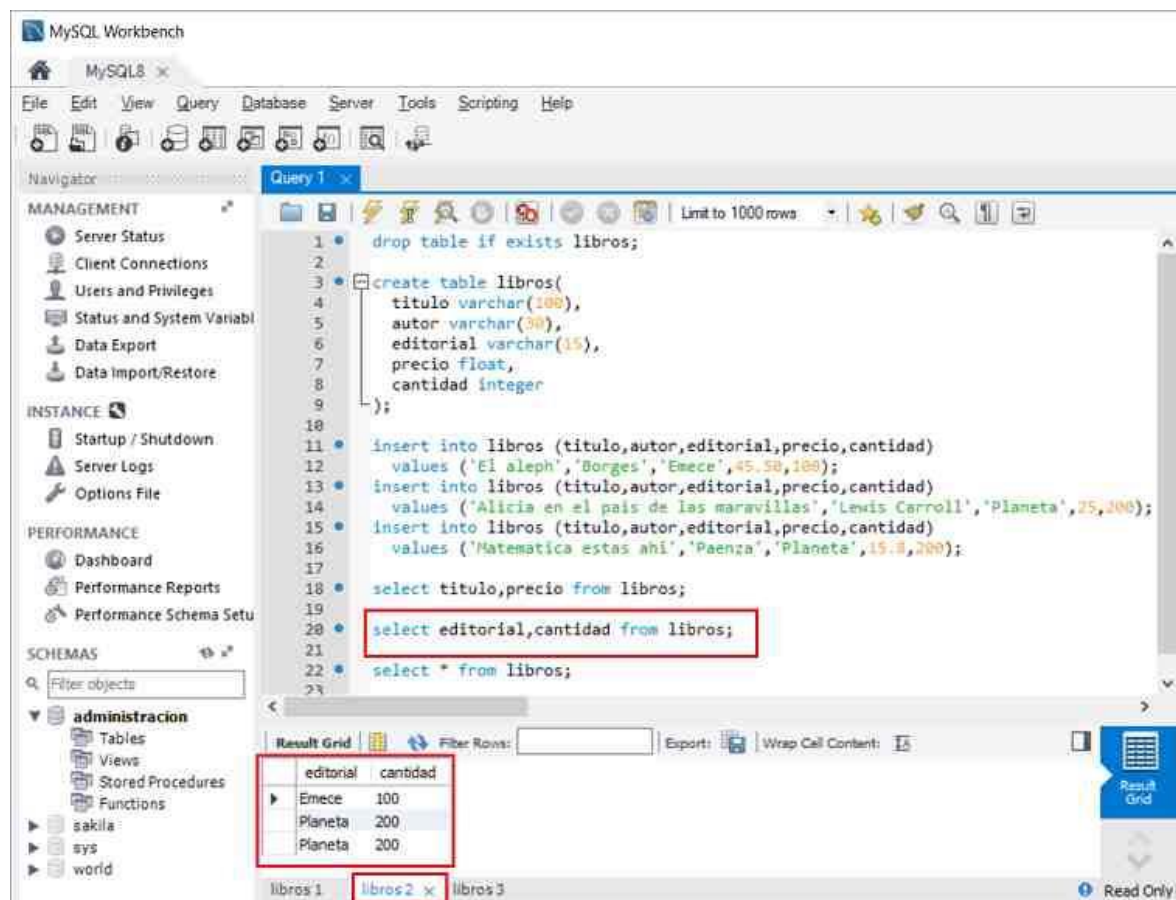
The results of the first SELECT query are displayed in the 'Result Grid' at the bottom. The grid shows the following data:

| titulo | precio |
|-------------------------------------|--------|
| El aleph | 45.5 |
| Alicia en el pais de las maravillas | 25 |
| Matematica estas ahi | 15.8 |

Below the result grid, there are tabs for 'libros 1', 'libros 2', and 'libros 3'. The 'libros 1' tab is currently selected and highlighted with a red box.

MySQL_ Bases de datos y tablas

El resultado del segundo select es:



The screenshot shows the MySQL Workbench interface. The central pane displays a SQL query with the following lines:

```
1 drop table if exists libros;
2
3 create table libros(
4     titulo varchar(100),
5     autor varchar(30),
6     editorial varchar(15),
7     precio float,
8     cantidad integer
9 );
10
11 insert into libros (titulo,autor,editorial,precio,cantidad)
12 values ('El aleph','Borges','Emece',45.50,100);
13 insert into libros (titulo,autor,editorial,precio,cantidad)
14 values ('Alicia en el pais de las maravillas','Lewis Carroll','Planeta',25,200);
15 insert into libros (titulo,autor,editorial,precio,cantidad)
16 values ('Matematica estas ahi','Paenza','Planeta',15.8,200);
17
18 select titulo,precio from libros;
19
20 select editorial,cantidad from libros;
21
22 select * from libros;
```

The query is executed, and the result grid at the bottom shows the output of the second SELECT statement. The result grid has two columns: 'editorial' and 'cantidad'. The data rows are:

| editorial | cantidad |
|-----------|----------|
| Emece | 100 |
| Planeta | 200 |
| Planeta | 200 |

7. Recuperación de registros específicos (select - where)

Hemos aprendido cómo ver todos los registros de una tabla:

```
select nombre, clave from usuarios;
```

El comando "select" recupera los registros de una tabla. Detallando los nombres de los campos separados por comas, indicamos que seleccione todos los campos de la tabla que nombramos.

Existe una cláusula, "where" que es opcional, con ella podemos especificar condiciones para la consulta "select". Es decir, podemos recuperar algunos registros, sólo los que cumplan con ciertas condiciones indicadas con la cláusula "where". Por ejemplo, queremos ver el usuario cuyo nombre es "MarioPerez", para ello utilizamos "where" y luego de ella, la condición:

```
select nombre, clave from usuarios where nombre='MarioPerez';
```

Para las condiciones se utilizan operadores relacionales (tema que trataremos más adelante en detalle). El signo igual(=) es un operador relacional. Para la siguiente selección de registros especificamos una condición que solicita los usuarios cuya clave es igual a 'bocajunior':

```
select nombre, clave from usuarios where clave='bocajunior';
```

Si ningún registro cumple la condición establecida con el "where", no aparecerá ningún registro.

MySQL_ Bases de datos y tablas

Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutamos los siguientes comandos SQL:

```
drop table if exists usuarios;

create table usuarios (
  nombre varchar(30),
  clave varchar(10)
);

describe usuarios;

insert into usuarios (nombre, clave) values ('Leonardo','payaso');
insert into usuarios (nombre, clave) values ('MarioPerez','Mario');
insert into usuarios (nombre, clave) values ('Marcelo','bocajunior');
insert into usuarios (nombre, clave) values ('Gustavo','bocajunior');

select nombre, clave from usuarios;

select nombre, clave from usuarios where nombre='Leonardo';

select nombre, clave from usuarios where clave='bocajunior';

select nombre, clave from usuarios where clave='river';
```

Cuando ejecutamos este conjunto de comandos SQL tenemos los siguientes resultados:

MySQL_ Bases de datos y tablas

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a database named 'administracion' with a table 'usuarios'. The main editor displays a SQL script with the following queries:

```
1 drop table if exists usuarios;
2
3 create table usuarios (
4     nombre varchar(30),
5     clave varchar(30)
6 );
7
8 describe usuarios;
9
10 insert into usuarios (nombre, clave) values ('Leonardo', 'payaso');
11 insert into usuarios (nombre, clave) values ('MarioPerez', 'Marito');
12 insert into usuarios (nombre, clave) values ('Marcelo', 'boca junior');
13 insert into usuarios (nombre, clave) values ('Gustavo', 'boca junior');
14
15 select nombre, clave from usuarios;
16
17 select nombre, clave from usuarios where nombre='Leonardo';
18
19 select nombre, clave from usuarios where clave='boca junior';
20
21 select nombre, clave from usuarios where clave='river';
22
```

The results of the third query are shown in the 'Result Grid' at the bottom:

| nombre | clave |
|----------|--------|
| Leonardo | payaso |

The 'Result Grid' tab is labeled 'usuarios 3'.

MySQL_ Bases de datos y tablas

8. Borrado de registros de una tabla (delete)

Para eliminar los registros de una tabla usamos el comando "delete":

```
delete from usuarios;
```

La ejecución del comando indicado en la línea anterior borra TODOS los registros de la tabla.

Si queremos eliminar uno o varios registros debemos indicar cuál o cuáles, para ello utilizamos el comando "delete" junto con la clausula "where" con la cual establecemos la condición que deben cumplir los registros a borrar. Por ejemplo, queremos eliminar aquel registro cuyo nombre de usuario es 'Leonardo':

```
delete from usuarios where nombre='Leonardo';
```

Si solicitamos el borrado de un registro que no existe, es decir, ningún registro cumple con la condición especificada, no se borrarán registros, pues no encontró registros con ese dato.

El comando delete hay que tener mucho cuidado en su uso, una vez eliminado un registro no hay forma de recuperarlo. Si por ejemplo ejecutamos el comando:

```
delete from usuarios;
```

Si la tabla tiene 1000000 de filas, todas ellas serán eliminadas.

En MySQL hay una variable de configuración llamada SQL_SAFE_UPDATES que puede almacenar los valores 1 (activa) y 0 (desactiva). Cuando tiene el valor 1 no permite ejecutar comandos delete sin indicar un where y que dicho where se relacione a una clave primaria, tema que veremos más adelante.

MySQL_ Bases de datos y tablas

Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutamos los siguientes comandos SQL donde utilizamos entre otros el comando delete:

```
drop table if exists usuarios;

create table usuarios (
  nombre varchar(30),
  clave varchar(10)
);

insert into usuarios (nombre, clave) values ('Leonardo','payaso');
insert into usuarios (nombre, clave) values ('MarioPerez','Mario');
insert into usuarios (nombre, clave) values ('Marcelo','River');
insert into usuarios (nombre, clave) values ('Gustavo','River');

delete from usuarios;

delete from usuarios where nombre='Leonardo';

select nombre,clave from usuarios;

delete from usuarios where clave='River';

select nombre,clave from usuarios;
```

Si ejecutamos este conjunto de comandos SQL se produce un error en el primer comando 'delete' debido a que la variable 'SQL_SAFE_UPDATES' tiene el valor 1 (es decir está activa), en MySQL 8.0 se instala por defecto para la variable 'SQL_SAFE_UPDATES' con el valor 1:

MySQL_ Bases de datos y tablas

The screenshot shows the MySQL Workbench interface. The top pane displays a SQL query with the following statements:

```
11. insert into usuarios (nombre, clave) values ('Gustavo', 'River');
12.
13. delete from usuarios;
14.
15. delete from usuarios where nombre='Leonardo';
16.
17. select nombre,clave from usuarios;
18.
19. delete from usuarios where clave='River';
20.
21. select nombre,clave from usuarios;
22.
23.
```

The bottom pane shows the 'Result Grid' with the following data:

| nombre | clave |
|------------|--------|
| Leonardo | payaso |
| MarioPerez | Marito |
| Marcelo | River |
| Gustavo | River |

Below the result grid, the 'Output' pane shows the 'Action Output' table:

| # | Time | Action | Message |
|---|----------|---|--|
| 6 | 09:48:25 | insert into usuarios (nombre, clave) val... | 1 row(s) affected |
| 7 | 09:48:26 | delete from usuarios | Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a |

MySQL_ Bases de datos y tablas

Soluciones

La idea de que no se puedan ejecutar ciertos comandos 'delete' es para evitar borrados masivos de datos que luego no podemos recuperar.

Tenemos dos soluciones para resolver el problema de los 'delete', la primera es encerrar todo el bloque donde ejecutamos los comandos delete cambiando el esta de la variable 'SQL_SAFE_UPDATES':

El primer método es cambiar el estado de la variable SQL_SAFE_UPDATES en forma temporal:

```
drop table if exists usuarios;

create table usuarios (
  nombre varchar(30),
  clave varchar(10)
);

insert into usuarios (nombre, clave) values ('Leonardo','payaso');
insert into usuarios (nombre, clave) values ('MarioPerez','Mario');
insert into usuarios (nombre, clave) values ('Marcelo','River');
insert into usuarios (nombre, clave) values ('Gustavo','River');

set SQL_SAFE_UPDATES=0;

delete from usuarios;

delete from usuarios where nombre='Leonardo';

select nombre,clave from usuarios;

delete from usuarios where clave='River';

select nombre,clave from usuarios;

set SQL_SAFE_UPDATES=1;
```

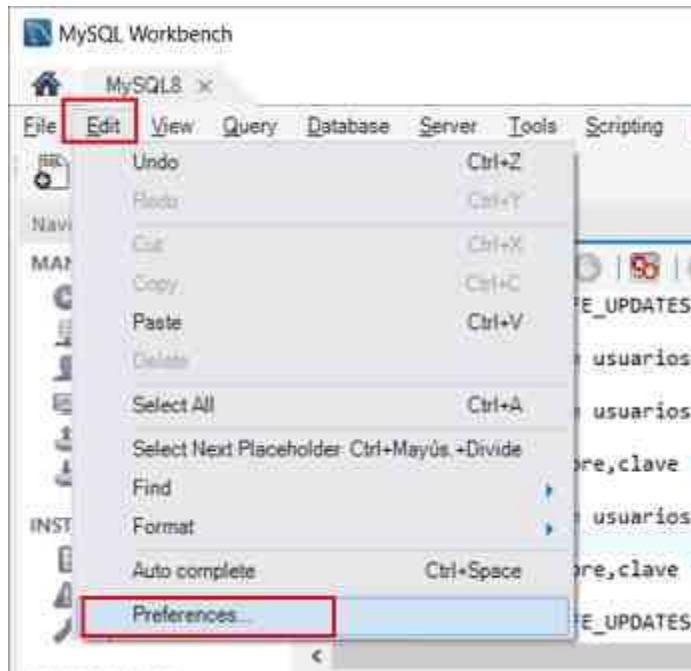
Tengamos en cuenta que disponer la variable SQL_SAFE_UPDATES para que los borrados sean solo seguros es muy conveniente cuando hay programadores que recién están comenzando en SQL y hay datos valiosos ya almacenados.

MySQL_ Bases de datos y tablas

Podemos saber el estado global de la variable 'SQL_SAFE_UPDATES' mediante la consulta:

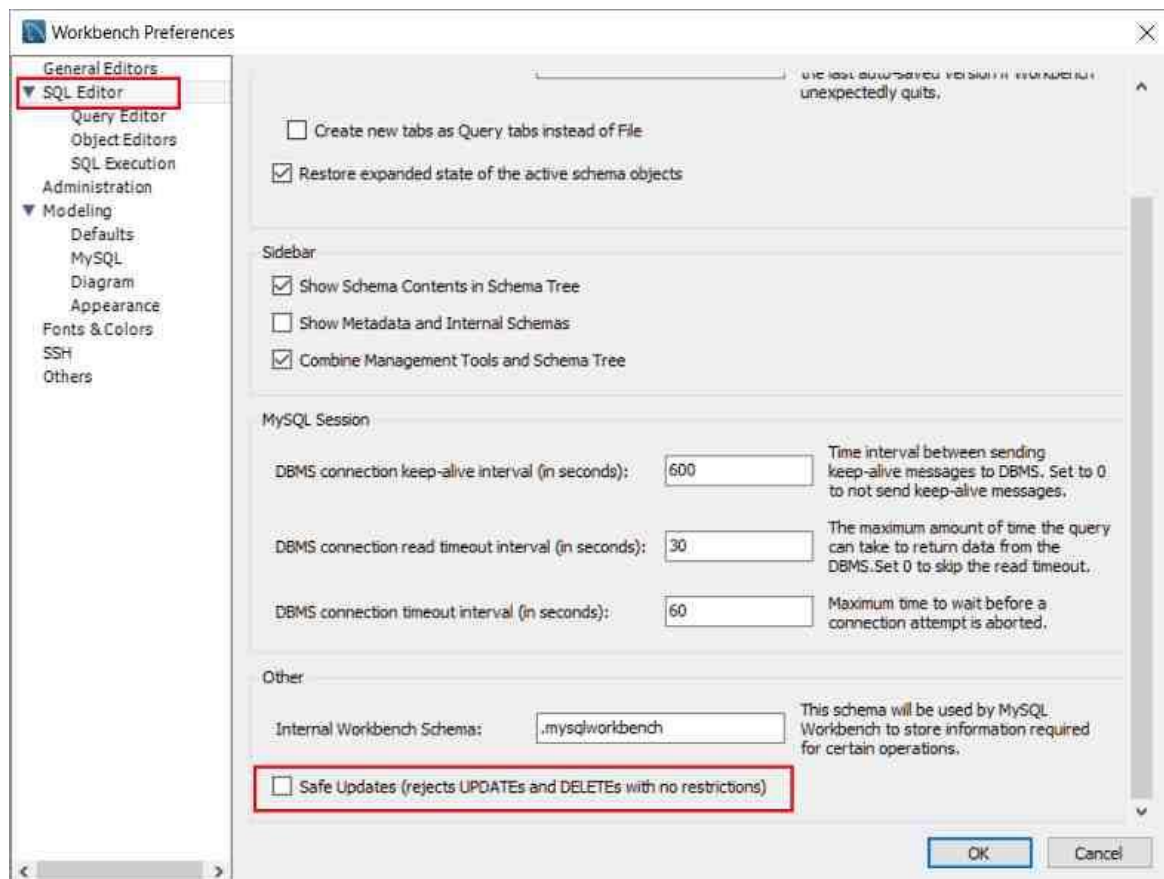
```
select @@sql_safe_updates;
```

1. El segundo método es cambiar el estado de la variable SQL_SAFE_UPDATES a nivel general, para ello desde el programa "Workbench" ingresamos a la opción Edit -> Preferences...:



En la pestaña "SQL Editor" debemos desmarcar la opción "Safe Updates (ejecuta Updates and Deletes with not restrictions)", con esto se permite ejecutar cualquier comando SQL delete, sin importar la cantidad de filas que se borran:

MySQL_ Bases de datos y tablas



Debemos cerrar y volver a entrar al programa "Workbench" para que el cambio se haga efectivo.

Si prueba en el servidor web2020 el comando SQL 'delete' verá que no hay restricciones en el borrado de filas, es decir que la variable SQL_SAFE_UPDATES se encuentra en '0'.

Seguramente cambiará el estado de la variable 'SQL_SAFE_UPDATES' a '0' en su servidor, pero es bueno tener cuidado cada vez que ejecutamos un comando SQL 'delete':

9. Modificación de registros de una tabla (update)

Para modificar uno o varios datos de uno o varios registros utilizamos "update" (actualizar).

Por ejemplo, en nuestra tabla "usuarios", queremos cambiar los valores de todas las claves, por "RealMadrid":

```
update usuarios set clave='RealMadrid';
```

Utilizamos "update" junto al nombre de la tabla y "set" junto con el campo a modificar y su nuevo valor.

El cambio afectará a todos los registros.

Podemos modificar algunos registros, para ello debemos establecer condiciones de selección con "where".

Por ejemplo, queremos cambiar el valor correspondiente a la clave de nuestro usuario llamado 'MarioPerez', queremos como nueva clave 'Boca', necesitamos una condición "where" que afecte solamente a este registro:

```
update usuarios set clave='Boca'  
where nombre='MarioPerez';
```

Si no encuentra registros que cumplan con la condición del "where", ningún registro es afectado.

Las condiciones no son obligatorias, pero si omitimos la cláusula "where", la actualización afectará a todos los registros.

También se puede actualizar varios campos en una sola instrucción:

```
update usuarios set nombre='MarceloDuarte', clave='Marce'  
where nombre='Marcelo';
```

Para ello colocamos "update", el nombre de la tabla, "set" junto al nombre del campo y el nuevo valor y separado por coma, el otro nombre del campo con su nuevo valor.

MySQL_ Bases de datos y tablas

Servidor de MySQL instalado en forma local.

Igual al concepto anterior cuando utilizamos el comando 'update' si la variable 'SQL_SAFE_UPDATES' se encuentra con un 1 (activa) luego solo se pueden ejecutar actualizaciones de una única fila disponiendo en el where la clave primaria (tema que no hemos visto)

Por el momento es aconsejable cambiar 'SQL_SAFE_UPDATES' al valor cero si no lo hizo en el concepto anterior.

Luego de cambiar 'SQL_SAFE_UPDATES' a cero puede ejecutar este conjunto de sentencias SQL en el "Workbench":

```
drop table if exists usuarios;

create table usuarios (
  nombre varchar(30),
  clave varchar(10)
);

insert into usuarios (nombre, clave) values ('Leonardo','payaso');
insert into usuarios (nombre, clave) values ('MarioPerez','Mario');
insert into usuarios (nombre, clave) values ('Marcelo','River');
insert into usuarios (nombre, clave) values ('Gustavo','River');

select * from usuarios;

update usuarios set clave='RealMadrid';

select nombre,clave from usuarios;

update usuarios set nombre='GustavoGarcia'
where nombre='Gustavo';

update usuarios set nombre='MarceloDuarte', clave='Marce'
where nombre='Marcelo';

select nombre,clave from usuarios;
```

MySQL_ Bases de datos y tablas

The screenshot shows a MySQL SQL File editor with the following SQL queries:

```
5 | clave varchar(10)
6 | );
7 |
8 | insert into usuarios (nombre, clave) values ('Leonardo', 'payaso');
9 | insert into usuarios (nombre, clave) values ('MarioPerez', 'Marito');
10 | insert into usuarios (nombre, clave) values ('Marcelo', 'River');
11 | insert into usuarios (nombre, clave) values ('Gustavo', 'River');
12 |
13 | select * from usuarios;
14 |
15 | update usuarios set clave='RealMadrid';
16 |
17 | select nombre,clave from usuarios;
18 |
19 | update usuarios set nombre='GustavoGarcia'
20 |   where nombre='Gustavo';
21 |
22 | update usuarios set nombre='MarceloDuarte', clave='Marce'
23 |   where nombre='Marcelo';
24 |
25 | select nombre,clave from usuarios;
```

The results of the last query are displayed in the Result Grid:

| nombre | clave |
|------------|------------|
| Leonardo | RealMadrid |
| MarioPerez | RealMadrid |
| Marcelo | RealMadrid |
| Gustavo | RealMadrid |

The bottom of the window shows the tab bar with three tabs: usuarios 1, usuarios 2 (selected), and usuarios 3.

MySQL_ Bases de datos y tablas

10. Operadores Relacionales = <> < <= > >=

Hemos aprendido a especificar condiciones de igualdad para seleccionar registros de una tabla; por ejemplo:

```
select titulo,autor,editorial from libros where autor='Borges';
```

Utilizamos el operador relacional de igualdad.

Los operadores relacionales vinculan un campo con un valor para que MySQL compare cada registro (el campo especificado) con el valor dado.

Los operadores relacionales son los siguientes:

| | |
|----|---------------|
| = | igual |
| <> | distinto |
| > | mayor |
| < | menor |
| >= | mayor o igual |
| <= | menor o igual |

Podemos seleccionar los registros cuyo autor sea diferente de 'Borges', para ello usamos la condición:

```
select titulo,autor,editorial from libros where autor<>'Borges';
```

Podemos comparar valores numéricos. Por ejemplo, queremos mostrar los libros cuyos precios sean mayores a 20 pesos:

```
select titulo,autor,editorial,precio from libros where precio>20;
```

También, los libros cuyo precio sea menor o igual a 30:

```
select titulo,autor,editorial,precio from libros where precio<=30;
```

MySQL_ Bases de datos y tablas

Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutamos los siguientes comandos SQL donde utilizamos los operadores relacionales:

```
drop table if exists libros;

create table libros(
  titulo varchar(20),
  autor varchar(30),
  editorial varchar(15),
  precio float
);

insert into libros (titulo,autor,editorial,precio) values ('El aleph','Borges','Planeta',12.50);
insert into libros (titulo,autor,editorial,precio) values ('Martin Fierro','Jose Hernandez','Emece',16.00);
insert into libros (titulo,autor,editorial,precio) values ('Aprenda PHP','Mario Molina','Emece',35.40);
insert into libros (titulo,autor,editorial,precio) values ('Cervantes','Borges','Paidos',50.90);

select titulo, autor,editorial,precio from libros;

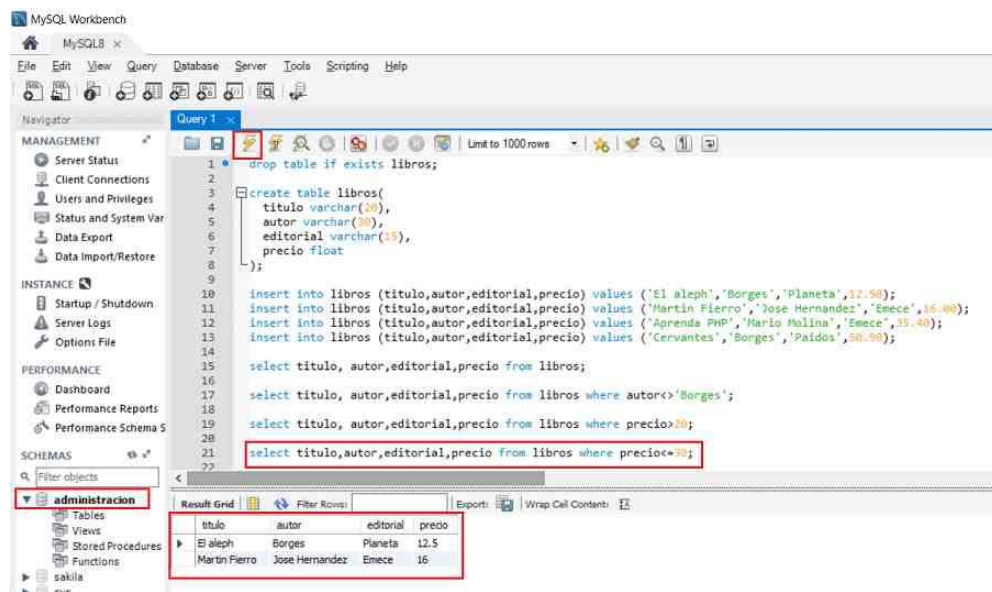
select titulo, autor,editorial,precio from libros where autor<>'Borges';

select titulo, autor,editorial,precio from libros where precio>20;

select titulo,autor,editorial,precio from libros where precio<=30;
```

MySQL_ Bases de datos y tablas

Tenemos como resultado:



The screenshot shows the MySQL Workbench interface. The left sidebar has the 'Schemas' tab selected, with 'administracion' highlighted. The main editor displays a SQL script for creating a table 'libros' and inserting data. The 'Query 1' tab is active, showing the script. The 'Result Grid' at the bottom displays the results of the last query, which is a SELECT statement filtering books by price.

```
1 drop table if exists libros;
2
3 create table libros(
4     titulo varchar(20),
5     autor varchar(30),
6     editorial varchar(15),
7     precio float
8 );
9
10 insert into libros (titulo,autor,editorial,precio) values ('El aleph','Borges','Planeta',12.50);
11 insert into libros (titulo,autor,editorial,precio) values ('Martin Fierro','Jose Hernandez','Emece',16.00);
12 insert into libros (titulo,autor,editorial,precio) values ('Aprenda PHP','Mario Molina','Emece',15.40);
13 insert into libros (titulo,autor,editorial,precio) values ('Cervantes','Borges','Paidós',50.90);
14
15 select titulo, autor,editorial,precio from libros;
16
17 select titulo, autor,editorial,precio from libros where autor<>'Borges';
18
19 select titulo, autor,editorial,precio from libros where precio>30;
20
21 select titulo,autor,editorial,precio from libros where precio<=30;
```

| titulo | autor | editorial | precio |
|---------------|----------------|-----------|--------|
| El aleph | Borges | Planeta | 12.5 |
| Martin Fierro | Jose Hernandez | Emece | 16 |

MySQL_ Bases de datos y tablas

11. Truncar Tabla

Aprendimos que para borrar todos los registro de una tabla se usa "delete" sin condición "where".

También podemos eliminar todos los registros de una tabla con "truncate table". Por ejemplo, queremos vaciar la tabla "libros", usamos:

```
truncate table libros;
```

La sentencia "truncate table" vacía la tabla (elimina todos los registros) y vuelve a crear la tabla con la misma estructura.

La diferencia con "drop table" es que esta sentencia borra la tabla, "truncate table" la vacía.

La diferencia con "delete" es la velocidad, es más rápido "truncate table" que "delete" (se nota cuando la cantidad de registros es muy grande) ya que éste borra los registros uno a uno.

Otra diferencia es la siguiente: cuando la tabla tiene un campo "auto_increment", si borramos todos los registros con "delete" y luego ingresamos un registro, al cargarse el valor en el campo autoincrementable, continúa con la secuencia teniendo en cuenta el valor mayor que se había guardado; si usamos "truncate table" para borrar todos los registros, al ingresar otra vez un registro, la secuencia del campo autoincrementable vuelve a iniciarse en 1.

Por ejemplo, tenemos la tabla "libros" con el campo "codigo" definido "auto_increment", y el valor más alto de ese campo es "5", si borramos todos los registros con "delete" y luego ingresamos un registro sin valor de código, se guardará el valor "6"; si en cambio, vaciamos la tabla con "truncate table", al ingresar un nuevo registro sin valor para el código, iniciará la secuencia en 1 nuevamente.

MySQL_ Bases de datos y tablas

Servidor de MySQL instalado en forma local.

Probemos ejecutar este conjunto de comandos SQL en nuestro servidor local de MySQL accediendo al mismo desde el programa "Workbench":

```
drop table if exists libros;

create table libros(
  codigo integer auto_increment,
  titulo varchar(50),
  autor varchar(50),
  editorial varchar(25),
  primary key (codigo)
);

insert into libros (titulo,autor,editorial)
  values('Martin Fierro','Jose Hernandez','Planeta');
insert into libros (titulo,autor,editorial)
  values('Aprenda PHP','Mario Molina','Emece');
insert into libros (titulo,autor,editorial)
  values('Cervantes y el quijote','Borges','Paidos');
insert into libros (titulo,autor,editorial)
  values('Matematica estas ahi', 'Paenza', 'Paidos');
insert into libros (titulo,autor,editorial)
  values('El aleph', 'Borges', 'Emece');

delete from libros;

select * from libros;

insert into libros (titulo,autor,editorial)
  values('Antología poetica', 'Borges', 'Emece');

select * from libros;

truncate table libros;

insert into libros (titulo,autor,editorial)
  values('Antología poetica', 'Borges', 'Emece');

select * from libros;
```

MySQL_ Bases de datos y tablas

El resultado es:

The screenshot shows a MySQL query editor window titled "Query 1". The SQL code in the editor is as follows:

```
20 values('El aleph', 'Borges', 'Emece');
21
22 delete from libros;
23
24 select * from libros;
25
26 insert into libros (titulo,autor,editorial)
27 values('Antología poetica', 'Borges', 'Emece');
28
29 select * from libros;
30
31 truncate table libros;
32
33 insert into libros (titulo,autor,editorial)
34 values('Antología poetica', 'Borges', 'Emece');
35
36 select * from libros;
37
```

The bottom part of the window shows the "Result Grid" with the following data:

| codigo | titulo | autor | editorial |
|--------|-------------------|--------|-----------|
| 1 | Antología poetica | Borges | Emece |

The interface includes a toolbar at the top with icons for file operations, a "Limit to 1000 rows" dropdown, and a "Result Grid" button on the right. The bottom status bar shows "libros 1", "libros 2", and "libros 3" with "Apply" and "Revert" buttons.