

13

Actualización de bases de datos

Introducción

Hasta ahora nos hemos dedicado a consultar datos de la base de datos mediante la sentencia SELECT. Hemos trabajado con tablas que ya tenían datos y hemos estado seleccionando los datos de esas tablas. Ha llegado el momento de cambiar los datos de las tablas de la base de datos. En este capítulo aprenderemos a insertar nuevas filas en una tabla, a actualizar los valores de las columnas en las filas y a borrar filas enteras.

Contenido

- 13.1. Inserción de datos. Orden INSERT
- 13.2. Modificación. UPDATE
- 13.3. Borrado de filas. DELETE
- 13.4. Transacciones. ROLLBACK y COMMIT
- Ejercicios propuestos

Objetivos

- ▲ Manejar con fluidez las órdenes para insertar, modificar y eliminar filas de una tabla
- ▲ Utilizar la orden INSERT
- ▲ Usar la orden UPDATE
- ▲ Manejar la orden DELETE
- ▲ Entender los conceptos de transacción, COMMIT y ROLLBACK
- ▲ Utilizar manuales en línea para obtener información adicional

13.1 Inserción de datos. Orden INSERT

Para insertar filas en una tabla utilizamos la orden **INSERT**. El formato básico de esta orden es el siguiente:

```
INSERT INTO NombreTabla [(columna1 [, columna2] ...)]  
VALUES (valor1 [, valor2] ...);
```

Dónde:

- **NombreTabla** es la tabla en la que se van a insertar las filas.
- **[(columna1 [, columna2] ...)]** representa la columna o columnas donde se van a introducir valores. Si las columnas no se especifican en la cláusula **INSERT**, se consideran, por defecto, todas las columnas de la tabla.

• **(valor1 [, valor2] ...)** representan los valores que se van a dar a las columnas. Los valores se deben corresponder con cada una de las columnas que aparecen; además, deben coincidir con el tipo de dato definido para cada columna. Cualquier columna que no se encuentre en la lista de columnas recibirá el valor NULL, siempre y cuando no esté definida como NOT NULL, en cuyo caso INSERT fallará. Si no se da la lista de columnas, se han de introducir valores en todas las columnas.

Es posible introducir los valores directamente en la sentencia u obtenerlos a partir de la información existente en la base de datos mediante la inclusión de una consulta utilizando la sentencia **SELECT**.

Actividad de Aplicación 13.1

Abre MySQL Query Browser y crea un nuevo esquema de base de datos de nombre UNIDAD13 para cargar el script que crea las tablas con las que trabajaremos en esta unidad.

A continuación carga el script TABLAS_UNIDAD13.SQL que está en la página web del libro en el Editor de Scripts y ejecútalo.



Ejercicio Resuelto 13.1

Partimos de la tabla **EMPLEADOS** utilizada en la unidad anterior. A la hora de insertar filas en una tabla es muy importante consultar la descripción de ésta con el fin de comprobar los tipos de datos y restricciones de cada columna. La descripción de la tabla se muestra en la Figura 13.1.



1. Insertamos un empleado dando valores a todas las columnas. Los datos son los siguientes: número de empleado 1000, apellido: 'QUEVEDO', oficio: 'ANALISTA', su director es el empleado de número 7782, la fecha de alta es la fecha actual (CURDATE()), el salario es 3000, la comisión es 0 y el departamento es el 40:

```
INSERT INTO empleados  
(emp_no, apellido, oficio, dir, fecha_alt,  
salario, comision, dept_no)  
VALUES  
(1000, 'QUEVEDO', 'ANALISTA', 7782, CURDATE(), 3000, 0, 40);
```

Si la fila se inserta correctamente, MySQL responderá con el mensaje: *I row affected by the last command, no results set returned* (véase la Figura 13.2), que indica que la inserción se ha hecho correctamente.

I row affected by the last command, no results set returned.

Figura 13.2. Mensaje al insertar 1 fila

Observamos en esta sentencia que:

- Las columnas a las que damos valores se identifican por su nombre.
- La asociación columna=valor es opcional.
- Los valores que se dan a las columnas deben coincidir con el tipo de dato definido en la columna.
- Los valores constantes de tipo carácter han de ir encerrados entre comillas simples o dobles (los de tipo fecha también).

Ahora, la tabla **EMPLEADOS** tendrá una fila más.

Cuando se dan valores a todas las columnas de la tabla, no es preciso especificar su nombre en la orden **INSERT**; pero los valores de las columnas han de ir en el mismo orden en que están definidas en la tabla. Por tanto, la orden **INSERT** anterior puede sustituirse por ésta:

```
INSERT INTO empleados VALUES  
(1000, 'QUEVEDO', 7782, CURDATE(), 3000, 0, 40);
```

2. Insertamos un empleado dando valores a algunas de las columnas: número de empleado 1001, apellido: 'RODRIGUEZ', oficio: 'ANALISTA', el salario es 3000 y el departamento es el 40:

```
INSERT INTO empleados  
(emp_no, apellido, oficio, salario, dept_no)  
VALUES (1001, 'RODRIGUEZ', 'ANALISTA', 3000, 40);
```

En este caso es necesario especificar el nombre de las columnas a las que se dan valores en la orden **INSERT**, ya que no se dan valores a todas. Las filas de la tabla **EMPLEADOS** que acabamos de insertar se muestran en la Figura 13.3.

emp_no	apellido	oficio	salario	dept_no
1000	QUEVEDO	ANALISTA	3000	40
1001	RODRIGUEZ	ANALISTA	3000	40

Figura 13.3. Filas insertadas en la tabla EMPLEADOS

En la figura se puede observar que las columnas a las que no dimos valores aparecen con la palabra NULL; es el caso de las columnas DIR, FECHA_ALT y COMISION del empleado 1001.

- Insertamos un empleado dando valores a algunas de las columnas: número de empleado 1002 y apellido: 'PÉREZ'.

```
INSERT INTO empleados (emp_no, apellido, oficio, dir, fecha_alt, salario, comision, dept_no)
VALUES (1002, 'PÉREZ');
```

Observamos que aparece un mensaje de error, véase la Figura 13.4. Este error indica que no se puede insertar una columna con valor NULO en la tabla si en su definición se ha especificado NOT NULL (la columna DEPT_NO está definida como NOT NULL).



Figura 13.4. Error al insertar fila

```
INSERT INTO empleados (emp_no, apellido) VALUES (1002, 'PÉREZ');
```

Como las tablas EMPLEADOS y EMPLE30 tienen la misma descripción, no es preciso especificar los nombres de las columnas, siempre y cuando queramos dar valores a todas las columnas. Esta sentencia daría el mismo resultado:

```
INSERT INTO empleados * FROM empleados WHERE dept_no = 30;
```

- Insertamos un empleado en la tabla EMPLEADOS con número de empleado 1003 y apellido 'LOPEZ' en el departamento que tiene 3 empleados. En esta inserción damos valores a las columnas EMP_NO, APELLIDO Y DEPT_NO. Para calcular el departamento que tiene 3 empleados necesitamos ejecutar la siguiente consulta:

```
SELECT dept_no FROM empleados GROUP BY dept_no HAVING COUNT(*) = 3;
```

La cláusula INSERT con la sentencia SELECT que calcula el departamento quedaría así:

```
INSERT INTO empleados (emp_no, apellido, dept_no)
SELECT 1003, 'LOPEZ', dept_no
FROM empleados GROUP BY dept_no HAVING COUNT(*) = 3;
```

Actividad de Aplicación 13.3

Inserta en la tabla EMPLE30 los datos de los empleados del departamento de 'PRODUCCIÓN'.

- ### 13.1.1. Inserción con SELECT
- Hasta el momento sólo hemos insertado una fila, pero si añadimos a la orden INSERT una consulta, es decir, una sentencia SELECT, se añaden tantas filas como devuelva la consulta. Los valores que se dan a las columnas se especifican en la cláusula SELECT; pueden ser columnas de una tabla o valores constantes. El formato de INSERT con SELECT es el siguiente:
- ```
INSERT INTO NombreTabla1 [(columna1 [, columna2] ...)]
SELECT {columna1 [, columna2] ... | *}
FROM NombreTabla2 [CLÁUSULAS DE SELECT];
```
- Si las columnas no se especifican en la cláusula INSERT, por defecto se consideran todas las columnas de la tabla.

### Ejercicio Resuelto 13.2

Disponemos de la tabla EMPLE30, cuya descripción es la misma que la de la tabla EMPLEADOS.

- Insertamos los datos de los empleados del departamento 30:

```
UPDATE NombreTabla
SET columna1 = valor1, ..., columnan = valorn
WHERE condicion;
```

Donde:

- NombreTabla es la tabla cuyas columnas se van a actualizar.
- SET indica las columnas que se van a actualizar y sus valores.
- WHERE selecciona las filas que se van a actualizar. Si se omite, la actualización afectará a todas las filas de la tabla.



## 13.4 Transacciones, ROLLBACK y COMMIT

Una transacción es una secuencia de una o más sentencias SQL que juntas forman una unidad de trabajo. Supongamos que queremos borrar una fila de una tabla, pero, al teclear la orden **SQL**, se nos olvida la cláusula **WHERE** y... ¡borramos todas las filas de la tabla! Esto no es problema, pues MySQL permite dar marcha atrás a un trabajo realizado mediante la orden **ROLLBACK**, siempre y cuando no hayamos validado los cambios en la base de datos mediante la orden **COMMIT**.

Por defecto, MySQL funciona en modo **autocommit**. Esto implica que, tan pronto como ejecutemos una actualización, MySQL la almacenará en disco. Los pasos para usar transacciones en MySQL son los siguientes:

- Iniciar una transacción con el uso de la sentencia **BEGIN** o **START TRANSACTION**.
- Actualizar, insertar o eliminar filas de la base de datos.
- Si se quieren validar los cambios en la base de datos, completar la transacción con el uso de la sentencia **COMMIT**. Si sucede algún problema, podemos utilizar la sentencia **ROLLBACK** para cancelar los cambios realizados hasta el momento.
- La transacción termina con la orden **COMMIT**. Para iniciar una nueva hemos de iniciarla con la orden **BEGIN** o **START TRANSACTION**.

Los pasos anteriores son válidos cuando ejecutamos MySQL en modo línea de comandos. Para utilizar transacciones utilizaremos tablas **InnoDB**.

### 3. Iniciamos la transacción con la orden BEGIN o START TRANSACTION:

```
mysql> BEGIN;
Query OK, 0 rows affected (0.31 sec)

4. Realizamos operaciones sobre la tabla INNOPRUEBA y vemos su contenido:
```

```
mysql> INSERT INTO innoprueba VALUES (10);
Query OK, 1 rows affected (0.00 sec)
mysql> INSERT INTO innoprueba VALUES (20);
Query OK, 1 rows affected (0.00 sec)
mysql> UPDATE INTO innoprueba SET campo = campo + 11;
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2 Changed: 2 Warnings: 0
mysql> SELECT * FROM innopruueba;
+-----+
| campo |
+-----+
| 21 |
| 31 |
+-----+
2 rows in set (0.00 sec)
```

- 5. Deshacemos el trabajo porque nos hemos confundido al insertar datos y visualizamos de nuevo el contenido de la tabla. Vemos que las filas insertadas ya no están:

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.07 sec)
mysql> SELECT * FROM innoprueba;
Empty set (0.00 sec)
```

### 6. Insertamos de nuevo una fila y ejecutamos la orden COMMIT para validar los cambios:

```
mysql> INSERT INTO innopruueba VALUES (100);
Query OK, 0 rows affected (0.06 sec)
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

7. Ejecutamos de nuevo la orden ROLLBACK y visualizamos el contenido de la tabla. Vemos que el último INSERT que se hizo sobre la tabla permanece, debido a que anteriormente se validó la transacción:
```

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)
mysql> SELECT * FROM innoprueba;
+-----+
| campo |
+-----+
| 100 |
+-----+
1 row in set (0.00 sec)
```

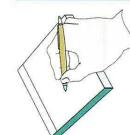
Figura 13.5. Línea de comandos de MySQL

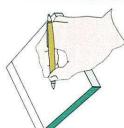
2. Nos conectamos a la base de datos UNIDAD13 escribiendo la orden: **USE** unidad13

```
mysql> USE unidad13
Database changed
```

8. Escribimos **EXIT** o cerramos la ventana para salir de la línea de comandos de MySQL:

```
mysql> EXIT;
```





## Ejercicio Resuelto 13.7

Utilizamos la tabla INNOPRUEBA.

1. Desde *MySQL Query Browser* pulsamos la tecla F11 o hacemos clic en la opción de menú *View/Maximize Query Edit*. Se visualiza la barra de herramientas Avanzada con el grupo de botones de transacciones: *Transaction, Commit y Rollback*.
2. Iniciamos una transacción haciendo clic en el botón *Transaction*, véase la Figura 13.6. Los botones *Commit* (situated a la derecha de *Transaction*) y *Rollback* se activarán; el botón *Transaction* se desactivará.



Figura 13.6. Paso 2. Ejercicio Resuelto 13.7

3. Escribimos y ejecutamos las siguientes órdenes en el Área de Consulta SQL:

```
INSERT INTO innoprueba VALUES (200);
INSERT INTO innoprueba VALUES (202);
SELECT * FROM innoprueba;
```

Vemos que la tabla tiene 3 filas, véase la Figura 13.7.

| campo |
|-------|
| 100   |
| 200   |
| 202   |

Figura 13.7. Paso 3. Ejercicio Resuelto 13.7

4. Deshacemos el trabajo haciendo clic en el botón *Rollback*. El botón *Transaction* se activa de nuevo indicando que la transacción anterior ha terminado; para crear una nueva transacción será necesario activarlo. Los botones *Commit* y *Rollback* se desactivarán.



Figura 13.8. Paso 4. Ejercicio Resuelto 13.7

5. Si de nuevo consultamos el contenido de la tabla, veremos que sólo contiene una fila.

Desde el navegador de transacciones se puede ver la lista de las sentencias ejecutadas en la transacción actual haciendo clic en la pestaña *Trx*.

## Actividad de Aplicación 13.7

Partimos del ejemplo anterior.

- Inicia una nueva transacción.
- Ejecuta las siguientes órdenes:
 

```
INSERT INTO innoprueba VALUES (300);
INSERT INTO innoprueba VALUES (303);
```
- Valida la transacción usando el botón *Commit*.
- Consulta el contenido de la tabla.

Las siguientes órdenes SQL llevan implícita la finalización de una transacción:

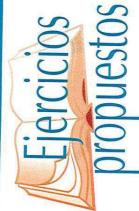
|                 |                   |
|-----------------|-------------------|
| ALTER FUNCTION  | BEGIN             |
| CREATE DATABASE | CREATE INDEX      |
| CREATE TABLE    | DROP FUNCTION     |
| DROP PROCEDURE  | DROP INDEX        |
| DROP TABLE      | LOCK TABLES       |
| RENAME TABLE    | START TRANSACTION |
| UNLOCK TABLES   | TRUNCATE TABLE    |

Usar cualquiera de estas órdenes es como usar *COMMIT*.



### Direcciones Web de interés:

- Zona de descarga de documentación sobre MySQL:  
<http://dev.mysql.com/doc/>
- Manual de referencia de MySQL en castellano:  
<http://dev.mysql.com/doc/refman/5.0/es/index.html>
- Fuente de consulta y referencia para el aprendizaje de MySQL: artículos, noticias, eventos, etc.  
<http://www.mysql-hispano.org/>
- Tutorial básico de MySQL:  
[http://www.programacion.com/bbdd/tutorial/mysql\\_basico/](http://www.programacion.com/bbdd/tutorial/mysql_basico/)



### Sean las tablas PERSONAL, PROFESORES Y CENTROS:

**PERSONAL:** Contiene datos de todas las personas que trabajan en los centros menos de los profesores.

**PROFESORES:** Contiene sólo los datos de los profesores.

**CENTROS:** Contiene los datos de los centros. El tipo de centro puede ser 'P' (Primaria) o 'S' (Secundaria).

| Field      | Type             | Null | Key | Default |
|------------|------------------|------|-----|---------|
| cod_centro | smallint(6)      | NO   |     |         |
| dni        | int(10) unsigned | YES  |     |         |
| apellidos  | varchar(30)      | YES  |     |         |
| funcion    | varchar(15)      | YES  |     |         |
| salario    | float(7,2)       | YES  |     |         |

Figura 13.9. Tabla PERSONAL

| Field        | Type             | Null | Key | Default |
|--------------|------------------|------|-----|---------|
| cod_centro   | smallint(6)      | NO   |     |         |
| dni          | int(10) unsigned | YES  |     |         |
| apellidos    | varchar(30)      | YES  |     |         |
| especialidad | varchar(15)      | YES  |     |         |

Figura 13.10. Tabla PROFESORES

| Field       | Type                 | Null | Key | Default |
|-------------|----------------------|------|-----|---------|
| cod_centro  | smallint(6)          | NO   |     |         |
| tipo_centro | char(1)              | YES  |     |         |
| nombre      | varchar(30)          | YES  |     |         |
| direccion   | varchar(25)          | YES  |     |         |
| telefono    | varchar(10)          | YES  |     |         |
| num_plazas  | smallint(5) unsigned | YES  |     |         |

Figura 13.11. Tabla CENTROS

**13.1** Insertar en la tabla de PROFESORES los datos de una profesora con estos apellidos y nombre: 'Quiroga Martín, A. Isabel', de la especialidad 'INFORMATICA' y con el código de centro 45.

**13.2** Insertar un profesor en la tabla PROFESORES de apellidos y nombre 'González Sevilla, Miguel A.' en el código de centro 22, con DNI 23444800 y de la especialidad de 'HISTORIA'.

**13.3** Inserta en la tabla PERSONAL los profesores de la tabla PROFESORES. El cod\_centro, el dni y los apellidos son los mismos que los de la tabla PROFESORES, la función es 'PROFESOR' y el salario, 1900.

**13.4** Inserta en la tabla PROFESORES un profesor de apellidos y nombre 'Guijarro Alia, Manuela', con DNI 28848110, de la especialidad de 'INFORMATICA' en el código de centro que tenga dos trabajadores cuya función sea 'CONSERJE'.

**13.5** Sea la tabla CENTROS. Cambiar la dirección del cod\_centro 22 a 'C/Pilón 13' y el número de plazas a 295.

**13.6** Modificar en la tabla PERSONAL el código de centro de aquellos cuya función es 'CONSERJE' igualándolo al código de centro donde hay dos profesores de la

especialidad 'INFORMATICA' (obtener los datos de los profesores de la tabla PROFESORES).

**13.7** Sumar 50 al número de plazas (columna NUM\_PLAZAS) de la tabla CENTROS, para aquellos centros que tengan profesores de la especialidad de 'INFORMATICA'.

**13.8** Eliminar de la tabla CENTROS aquellos centros que no tengan profesores de la especialidad de 'INFORMATICA'.

**13.9** Eliminar de la tabla PERSONAL aquellas filas cuyo código de centro no exista en la tabla CENTROS.

**13.10** Eliminar de la tabla PROFESORES aquellas filas cuyo dni sea nulo.