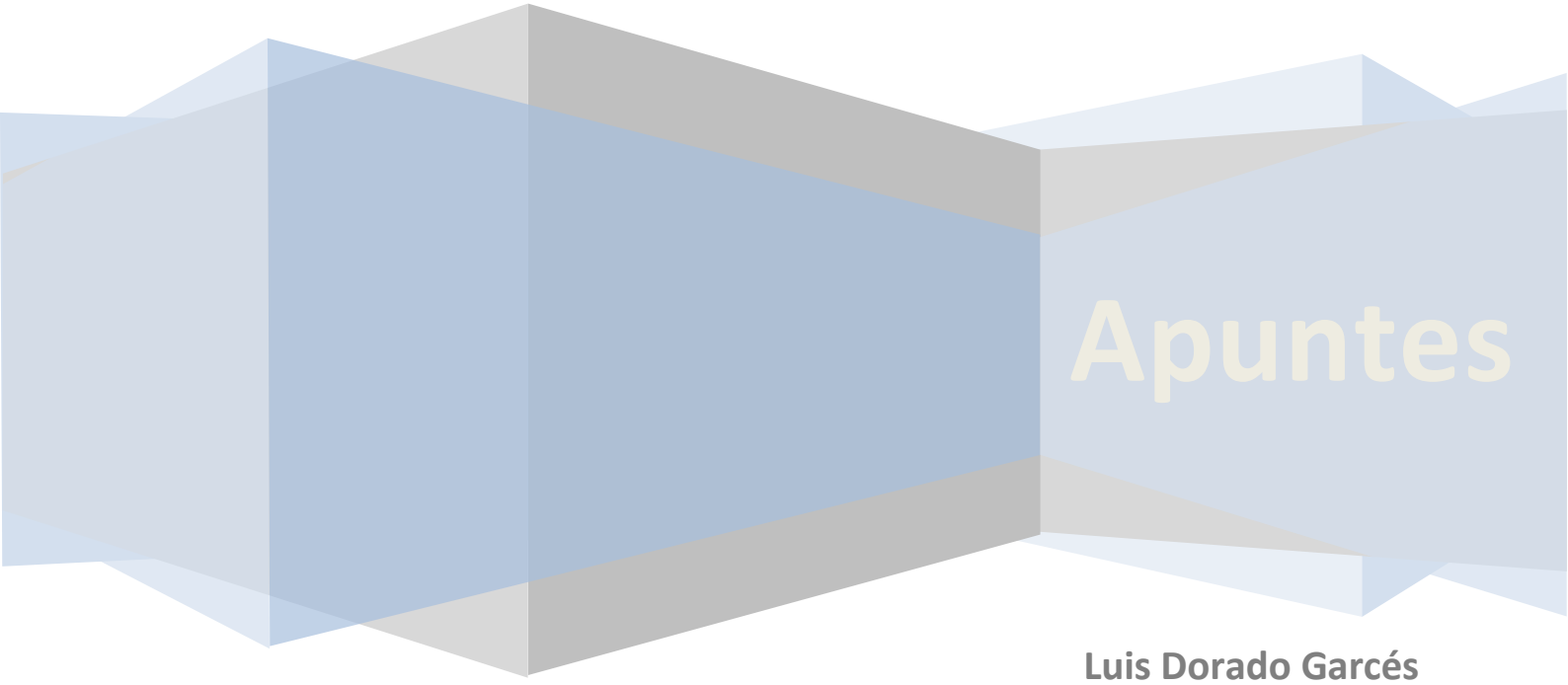
BASES DE DATOS

Desarrollo de Aplicaciones Web

GESTIÓN DE BASES DE DATOS

Administración de Sistemas Informáticos en Red

MODELO FÍSICO
LENGUAJE DE MANIPULACIÓN DE DATOS - EDICIÓN



Luis Dorado Garcés
Basado en el trabajo de Alba Tortosa López

Contenido

1 Introducción

En esta unidad veremos 3 sentencias SQL DML para la modificación de datos.

Una vez que se ha creado de forma conveniente las tablas, el siguiente paso consiste en insertar datos en ellas, es decir, añadir registros. Durante la vida de la base de datos será necesario, además, borrar determinados registros o modificar los valores que contienen. Las cláusulas SQL que se van a estudiar en este apartado son **INSERT, UPDATE y DELETE**. Estos comandos pertenecen al **DML**.

2 Edición de datos con SQL I

2.1 Inserción de datos

El comando **INSERT** de SQL permite introducir datos en una tabla o en una vista de la base de datos. La sintaxis del comando es la siguiente:

```
INSERTINTOnombre_tabla[(columna1[,columna2]...)]
VALUES(valor1[,valor2]...);
```

Indicando la tabla se añaden los datos que se indiquen en VALUES en un nuevo registro. Los valores deben corresponderse con el orden de las columnas y debe darse valor a todas las columnas.

Si no es así se pueden indicar las columnas tras el nombre de la tabla y entre paréntesis.

Ejemplos:

Supongamos que tenemos el siguiente diseño físico de una tabla:

```
CREATETABLEEMPLEADOS(
CODINTEGER,
NOMBREVARCHAR(50)NOTNULL,
LOCALIDADVARCHAR(50)DEFAULT'Pamplona',
FECHANACDATE,
PRIMARYKEY (COD)
);
```

La forma más habitual de introducir datos es la siguiente:

```
INSERTINTOEMPLEADOSVALUES(1,'Pepe','Tudela','01/01/1990');
INSERTINTOEMPLEADOSVALUES(2,'Juan',DEFAULT,NULL);
INSERTINTOEMPLEADOSVALUES(3,'Sara',NULL,NULL);
```

Es obligatorio introducir valores para los campos COD y NOMBRE. Dichos campos no pueden tener valor NULL.

Podemos insertar sólo el valor de ciertos campos. En este caso hay que indicar los campos a insertar y el orden en el que los introducimos. El resto de los campos del registro tendrá valor NULL.

```
INSERTINTOEMPLEADOS (NOMBRE,COD)VALUES('Ana',5);
```

2.2 Modificación de datos

Para la modificación de registros dentro de una tabla o vista se utiliza elcomando **UPDATE**. La sintaxis del comando es la siguiente:

```
UPDATEnombre_tabla
SETcolumna1=valor1[,columna2=valor2]...
[WHEREcondición];
```

Se modifican los valores de loscampos indicados en el apartado **SET** con los valores indicados. La cláusula **WHERE** permite especificar qué registros serán modificados.

Ejemplos

```
-- Retrasar en tres días la fecha de comienzo de aquellos
-- cursillos que comiencen por H.
UPDATE CURSILLOS SET FECHA = FECHA +3
WHERE NOMCUR LIKE'H%';
```

```
-- Ponemos todos los nombres a mayúsculas
-- y todas las localidades a Estella
UPDATEEMPLEADOS
SETNOMBRE=UPPER(NOMBRE),LOCALIDAD='Estella';


-- Para los empleados que nacieron a partir de 1970
-- ponemos nombres en mayúsculas y localidades a Estella
UPDATEEMPLEADOS
SETNOMBRE=UPPER(NOMBRE),LOCALIDAD='Estella'
WHEREFECHANAC>='1970-01-01';
```

Atención

Por defecto, MySQL Workbench tiene activado el modo seguro que evita que se realicen borrados o modificaciones en las tablas si no se utiliza una clave primaria en el WHERE.

2.3 Eliminación de datos

Esta cláusula elimina los registros de la tabla que cumplan la condición indicada. Se realiza mediante la instrucción **DELETE**:

```
DELETEFROMnombre_tabla
[WHEREcondición];
```

Ejemplos:

```
-- Borramos empleados de Estella
DELETE FROMEMPLEADOS
WHERELOCALIDAD='Estella';

-- Borramos empleados cuya fecha de nacimiento sea anterior
-- a 1970y localidad sea Estella
DELETE FROMEMPLEADOS
WHEREFECHANAC<'1970-01-01'ANDLOCALIDAD='Estella';

-- Borramos TODOS los empleados;
DELETE FROMEMPLEADOS;
```

Atención

Por defecto, MySQL Workbench tiene activado el modo seguro que evita que se realicen borrados o modificaciones en las tablas si no se utiliza una clave primaria en el WHERE.

2.4 Anexos

2.4.1 Desactivar modo seguro

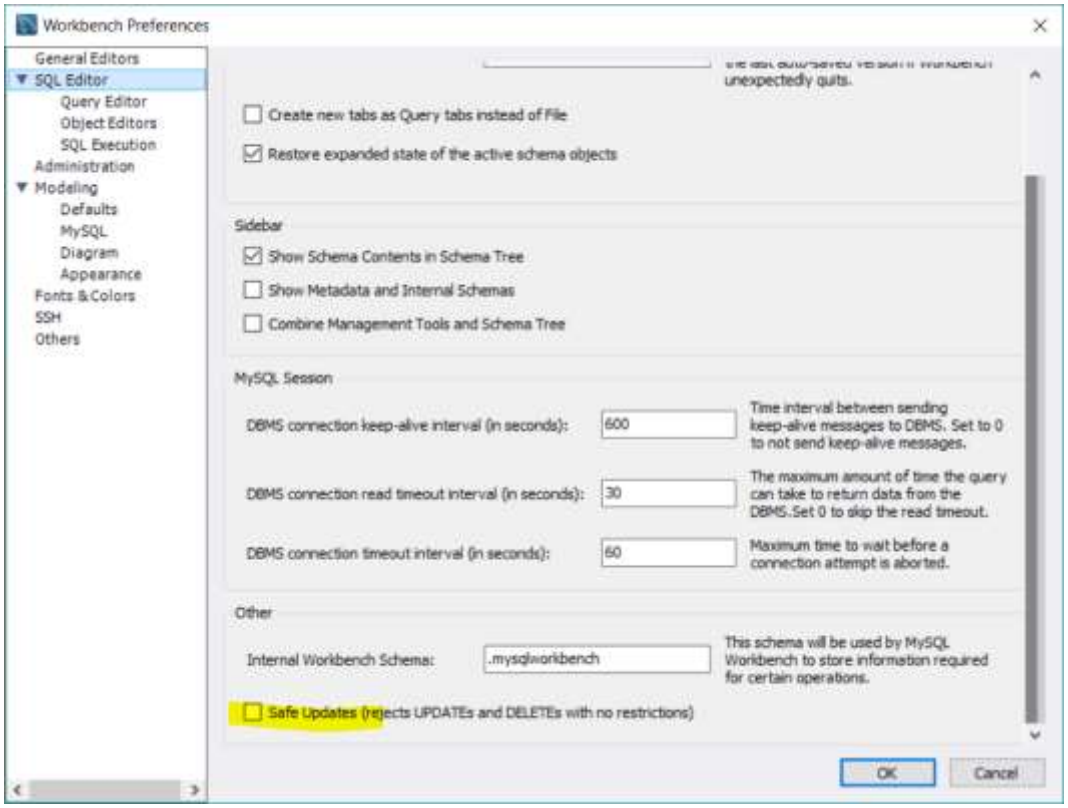
Por defecto, MySQL Workbench tiene activado el modo seguro que evita que se realicen borrados o modificaciones en las tablas si no se utiliza una clave primaria en el WHERE. Para desactivar este modo tenemos dos opciones:

Usar un comando

```
SET SQL_SAFE_UPDATES = 0;
```

Modificar las preferencias de MySQL Workbench

1. Ve al menú "Edit → Preferences"
2. Haz click en la opción "SQL Editor" y deselecciona "Safe Updates"



3. Ve al menú "Query" y escoge "Reconnect to Server"

3 Edición de datos con SQL II

3.1 Inserción de datos obtenidos de una consulta

También es posible insertar datos en una tabla que hayan sido obtenidos de una consulta realizada a otra tabla u otras tablas. Su forma es:

```
INSERTINTOtabla
SELECT...
```

Debe respetarse lo dicho anteriormente respecto a los campos. La consulta SELECT debe devolver la misma cantidad y tipo de campos que los definidos en la tabla.

Por ejemplo, suponiendo que disponemos de una tabla SOLICITANTES con el siguiente diseño:

```
CREATETABLECANDIDATOS_ASCENSO(
NUMINTEGER,
NOMBREVARCHAR(50),
  OFICINAVARCHAR(50),
PRIMARYKEY(NUM)
);
```

Podemos usar el resultado de una consulta sobre la tabla EMPLEADOS para insertar registros en CANDIDATOS_ASCENSO.

```
INSERTINTOCANDIDATOS_ASCENSO
SELECTE.NUM,E.NOMBRE,NOMBRE_OFICINA
FROMEMPLEADOS E INNER JOIN OFICINAS O ON E.OFICINA=O.NUM
WHEREO.ZONA='NORTE';
```

En este caso, quiero insertar en la tabla de candidatos a todos los candidatos de las oficinas de la zona norte. Guardaré, por tanto, para cada empleado su clave primaria (en la tabla EMPLEADOS), su nombre (en la tabla EMPLEADOS) y el nombre de su oficina ((en la tabla OFICINAS).

También podemos indicar los campos a insertar, teniendo en cuenta que, en este caso los campos COD y NOMBRE de la tabla EMPLEADO no aceptan valores NULL.Por tanto, es obligatorio introducir valores para ellos:

```
INSERTINTOEMPLEADOS (FECHANAC,NOMBRE,COD)
SELECTNACIMIENTO,NOMBRE,NUM
FROMSOLICITANTES_EMPLEO
WHEREESTUDIOS='ASIR';
```

3.2 Creación de tabla e inserción de datos con CREATE TABLE… INSERT

Puede crear una tabla a partir de otra agregando una cláusula **SELECT**a continuación de la cláusula **CREATE TABLE**:

```
CREATE TABLEnuevaTabla [AS]
      SELECTcol1[, col2…]
      FROM tablaExistente [JOIN otraTablaExistente…];
```

MySQL crea nuevas columnas para todos los elementos en **SELECT**. Por ejemplo:

```
CREATE TABLECANDIDATOS_ASCENSOAS
SELECTE.NUM,E.NOMBRE,NOMBRE_OFICINA
FROMEMPLEADOS E INNER JOIN OFICINAS O ON E.OFICINA=O.NUM
WHEREO.ZONA='NORTE';
```

En este caso, quiero crear una tabla de candidatos con ciertas columnas de la combinación de las tablas empleados y oficinas a todos los candidatos de las oficinas de la zona norte.

3.3 Actualización de datos usando varias tablas

A menudo usamos combinaciones para consultar filas de una tabla que tienen filas coincidentes en otra tabla (JOINS). En MySQL, puede utilizar JOIN en la cláusula UPDATE para realizar la actualización usandocombinación de tablas.

La sintaxis de MySQL UPDATE JOIN es la siguiente:

```
UPDATEtabla1INNER JOINtabla2
            ONTABLA1.columnaX=TABLA2.columnaY
SETtabla1=valor1[,tabla2=valor2]...
```

```
[WHERE condición];
```

Examinemos la sintaxis deUPDATE JOIN con mayor detalle:

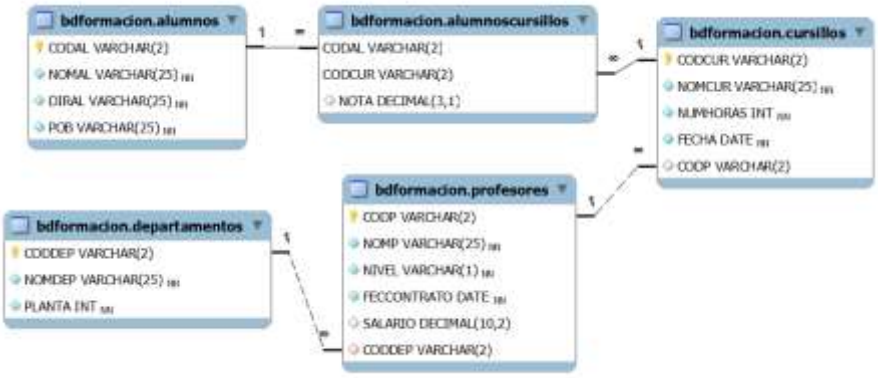
Después de la cláusula UPDATE. Primero, especificamos la combinación de tablas (T1 y T2). Tengamos en cuenta que debemos especificar al menos una tabla después de la cláusula UPDATE.

A continuación, especificamos un tipo de combinación que deseemos utilizar, es decir, INNER JOIN o LEFT JOIN con una cláusula ON que especifique las columnas relacionadas. La cláusula JOIN debe aparecer inmediatamente después de la cláusula UPDATE.

Luego, asignamos nuevos valores a las columnas de las tablas T1y/o T2que deseamos actualizar.

Después de eso, especificamos una condición en la cláusula WHERE para limitar filas a filas para actualizar.

Echemos un vistazo a un ejemplo del uso de la UPDATE JOIN para tener una mejor comprensión:




```
-- Para todos los cursillos que empiecen por 'H' trasformar a
-- minúsculas el nombre del cursillo y subir 0.5 puntos la
-- notas de todos los alumnos matriculados en dichos cursos
UPDATE ALUMNOSCURSILLOS AC INNERJOIN CURSILLOS C
ON AC.CODCUR = C.CODCUR
SET NOTA=NOTA+0.5, C.NOMCUR=LOWER(NOMCUR)
WHERE NOMCUR ='H%';
```

4 Edición de datos con SQL III

4.1 Actualización de datos usando una subconsulta

También se admiten subconsultas. Por ejemplo:

```
UPDATEempleados
SETsueldo=sueldo*1.10
WHEREid_seccion=(SELECTid_seccionFROMsecciones
WHEREnom_seccion='Producción');
```

Esta instrucción aumenta un 10% el sueldo de los empleados que están dados de alta en la sección llamada Producción.

4.2 Eliminación de datos usando una subconsulta

Al igual que en el caso de las instrucciones INSERT o SELECT, **DELETE** dispone de la cláusula WHERE y en dicha cláusula podemos utilizar subconsultas. Por ejemplo:

```
DELETEempleados
WHEREid_empleadoIN(SELECTid_empleadoFROMoperarios);
```

En este caso se trata de una subconsulta creada con el operador IN, se eliminarán los empleados cuyo identificador esté dentro de la tabla operarios.