Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

SQL. MANIPULACIÓN DE DATOS

INSERTAR, MODIFICAR Y BORRAR DATOS

Índice

Presentación

Estamos estudiando el lenguaje del SGBD más utilizado en la actualidad: SQL. Anteriormente vimos un **sublenguaje del SQL utilizado para la manipulación de datos (LMD)** que permite consultar y modificar la instancia de la base de datos. Con posterioridad estudiaremos el **sublenguaje del SQL para la definición de datos (LDD)** que permite crear y modificar el esquema de la BD, chequear la integridad y definir autorizaciones.

En los temas 1 y 2 hemos estudiado cómo consultar los datos que hay almacenados a través de la consulta **SELECT.** En el tema 1, hemos estudiado las consultas más sencillas incluyendo **selección de columnas, selección de filas y ordenación de resultados.** En el 2 hemos visto las más complejas, incluyendo **combinación de tablas resultado, subconsultas, funciones de agregación y agrupación de resultados.**

Al finalizar este tema, el estudiante será capaz de:

- Conocer el conjunto de sentencias del **LMD** de SQL que permiten **modificar la instancia actual de la BD**.
- Aplicar correctamente las sentencias: **INSERT, UPDATE y DELETE** (insertar, modificar y borrar datos).

Insertar datos

Cada vez que llega un cliente a una sucursal de nuestro banco, tanto para abrir una nueva cuenta ó solicitar un nuevo préstamo, como para cancelar una cuenta ó un préstamo que ya tenía, tendremos que actualizar los datos almacenados para que reflejen esta nueva situación.

En el primer caso el cajero tendrá que dar de alta una nueva cuenta o un nuevo préstamo a nombre de este cliente que, además, si no estaba previamente en la base de datos, también tendrá que darse de alta como tal.

Para dar de alta nuevas instancias en una BD se utiliza la sentencia **INSERT** de SQL. Dar de alta una instancia se corresponde con insertar una nueva fila en la tabla de la BD correspondiente.

La sintaxis de esta sentencia nos permite introducir datos de dos formas:

- **Indicando los valores concretos para las columnas de la fila que estamos insertando** y que representan los valores de los atributos para esta instancia.
- **Utilizando el resultado de una consulta.** De esta forma insertamos varias filas a la vez, aquellas que aparecen en la tabla resultado de la consulta.

Insertar datos proporcionando una lista de valores

Para insertar datos proporcionando los valores de una fila se utiliza la siguiente sintaxis:

```
INSERT INTO <tabla> <listacolumnas> VALUES (<listavalores>);
```

Donde:

- **<tabla>** indica el nombre de la tabla donde queremos realizar la inserción de la fila.
- **<listacolumnas>**es una lista de nombres de columna (atributos) separados por comas. Si no se indica, se tomarán las columnas en el orden en que se introdujeron al crear la tabla.
- **<listavalores>** es la lista con los valores que queremos insertar. Se debe corresponder con <listacolumnas> de modo que:
    - Ambas listas deben tener el mismo número de elementos.
    - Las posiciones de las columnas y de los valores correspondientes deben coincidir.
    - Los tipos de los datos de ambas listas deben coincidir.

Supongamos que la **Sra. García** que vive en la calle **Catalanes**, **4** de **Motril**, entra en la sucursal **Nueva** dispuesta a abrir una nueva cuenta con un saldo inicial de **1.000 euros**. Para llevar a cabo esta operación, el cajero tendrá que dar de alta a la Sra. García en la **tabla de clientes** y dar de alta la nueva cuenta en la **tabla de depósitos**. Las sentencias en SQL serían:

INSERT INTO clientes VALUES (GARCIA, CATALANES, 4,MOTRIL);

INSERT INTO depositos VALUES (NUEVA, 13,GARCIA,1000);

Como resultado de estas consultas en las tablas de clientes y depósitos habrá una fila más correspondiente a la operación que el cajero acaba de realizar:

Tabla de clientes

Tabla de depósitos

En detalle

Tabla de clientes

nombre_cli	calle	ciudad
alvarez	castilla 12	madrid
garcia	catalanes 4	motril
gaya	franco	madrid
gonzalez	rioverde	malaga
gutierrez	alcala 12	granada
lopez	constitucion	granada
perez	alcala 12	madrid
pineda	alcala 1	malaga

En detalle

Tabla de depósitos

sucursal	num_cta	nombre_cli	saldo
neptuno	1	perez	1500
caleta	2	perez	11500
caleta	3	gaya	500
nueva	4	alvarez	100
nueva	5	pineda	15000
zaidin	10	perez	200
zaidin	11	alvarez	200
nueva	13	garcia	1000

Ejemplo

A continuación vamos a ver con un ejemplo cómo introducir datos con una lista de valores: Imaginemos que tenemos que dar de alta en la sucursal **Caleta** al **Sr. Cervera** del que desconocemos la calle y la ciudad en que vive. En este caso necesitamos utilizar la **lista de columnas** que vamos a especificar puesto que los valores correspondientes a **calle** y **ciudad** no los vamos a introducir. La consulta sería:

```
INSERT INTO cliente (nombre_cli) VALUES (CERVERA);
```

Tras realizar la inserción y comprobar el contenido de la **tabla clientes** observamos que aparece una nueva fila para el cliente Cervera que tiene el resto de las columnas a valor **NULL**. Este es el valor que utiliza el SGBD para indicar que no disponemos de esta información. Esta sentencia **INSERT** hemos podido llevarla a cabo porque en el esquema de la tabla clientes no se ha definido calle y ciudad como atributos obligatorios. En caso contrario habríamos obtenido un error y no se habría podido realizar la inserción. La tabla de clientes tras la inserción anterior quedaría:

nombre_cli	calle	ciudad
alvarez	castilla 12	madrid
cervera	(null)	(null)
garcia	catalanes 4	motril
gaya	franco	madrid
gonzalez	rioverde	malaga
gutierrez	alcala 12	granada
lopez	constitucion	granada
perez	alcala 12	madrid
pineda	alcala 1	malaga

Insertar datos de una subconsulta

La sintaxis para insertar datos en una tabla utilizando el resultado de una subconsulta es:

```
INSERT INTO <tabla> (<listacolumnas>) SELECT…
```

Donde:

- **<tabla>** indica el nombre de la tabla donde queremos realizar la inserción de las filas de la tabla resultado de la consulta SELECT.
- **<listacolumnas>**es una lista de nombres de columna (atributos) separados por comas y deben coincidir con las columnas de la tabla resultado de la consulta. En este caso también es opcional. Si no se establece, la tabla resultado del SELECT tiene que tener el mismo número de columnas, en el mismo orden y con el mismo dominio que el esquema de <tabla>.

Por ejemplo, supongamos que nuestra base de datos tiene una tabla de **clientes_andaluces** que contiene los clientes del banco cuya ciudad es alguna de la comunidad andaluza. En este caso la consulta que deberíamos realizar es:

```
INSERT INTO clientes_andaluces SELECT * FROM cliente WHERE ciudad IN
(GRANADA,MALAGA,CADIZ,HUELVA,SEVILLA,CORDOBA,JAEN,ALMERIA);
```

Si tras la ejecución de esta sentencia consultamos los valores de la tabla **clientes_andaluces** que previamente estaba vacía, aparecerán los siguientes datos:

nombre_cli	calle	ciudad
gonzalez	rioverde	malaga
gutierrez	alcala 12	granada
lopez	constitucion	granada
pineda	alcala 1	malaga

Ejemplo

Ahora vamos a ver con un ejemplo cómo introducir datos de una subconsulta: insertar en la tabla de clientes todos aquellos que tengan algún préstamo en la sucursal **Zaidin**. Tenemos que utilizar una consulta **SELECT** para este caso. El resultado de ésta tiene que devolver las columnas de la tabla clientes: **el nombre del cliente**, **la calle** y **la ciudad**. El nombre del cliente aparece en la tabla préstamos, la calle no aparece en ninguna tabla y vamos a suponer que la ciudad del cliente será la misma de la sucursal donde tiene el préstamo concedido. La consulta sería:

```
INSERT INTO cliente (nombre_cli, ciudad) SELECT p.nombre_cli, s.ciudad FROM prestamo p, sucursal s WHERE p.sucursal=s.sucursal AND s.sucursal=ZAIDIN;
```

El resultado del **SELECT** es **este**.

Y la **tabla de clientes** tras realizar la sentencia:

nombre_cli	calle	ciudad
alvarez	(null)	granada
cervera	(null)	(null)
garcia	catalanes 4	motril
gaya	franco	madrid
gonzalez	rioverde	malaga
gutierrez	alcala 12	granada
lopez	constitucion	granada
perez	alcala 12	madrid
pineda	alcala 1	malaga

SELECT

Nombre_cli	ciudad
Álvarez	Granada

Modificar datos

La instrucción **UPDATE** permite modificar datos ya existentes en una tabla de una base de datos.

```
UPDATE <tabla> SET <columna1> = <valor1>, <columna2> = <valor2>.
. . WHERE <condición>;
```

Donde:

- **<tabla>** es la tabla que contiene los datos que queremos modificar.
- **SET** indica el nombre de una o más columnas a actualizar.
- **WHERE** seleccionará las filas que cumplan la condición, y sólo se harán los cambios en las columnas pertenecientes a ellas.
- Los valores deben ser compatibles con los tipos de los atributos (columnas).

Pongamos que la **Sra. García** entra en la sucursal para indicar que se ha cambiado de domicilio y proporcionar la nueva dirección, calle **Ancha 30**, donde desea recibir el correo del banco. La operación que tendría que realizar el cajero en este caso es cambiar la calle de este cliente para reflejar la nueva dirección.

Siguiendo con este ejemplo, la sentencia que permitirá al cajero actualizar la base de datos es:

```
UPDATE cliente SET calle = ANCHA, 30 WHERE nombre_cli=GARCIA;
```

Los datos se ven modificados en la tabla cliente así:

Tabla de clientes

En detalle

Tabla de clientes

nombre_cli	calle	ciudad
alvarez	(null)	granada
cervera	(null)	(null)
garcia	ancha 30	motril
gaya	franco	madrid
gonzalez	rioverde	malaga
gutierrez	alcala 12	granada
lopez	constitucion	granada
perez	alcala 12	madrid
pineda	alcala 1	malaga

Ejemplos

Ahora vamos a ver con unos ejemplos cómo **modificar datos de una tabla**.

- El banco ha cumplido su 100 aniversario. Para celebrarlo ha decidido premiar a los clientes **incrementando en un 5%** los saldos de las cuentas.

```
UPDATE depositos SET saldo = saldo*1.05;
```

Tabla de clientes con incremento de saldo

- Supongamos que disponemos en la base de datos de una **tabla con los empleados del banco**. El contenido de la tabla es:

sucursal	salario	nombre_emp	puesto
nueva	158	alvarez	cajero
caleta	525	gaya	cajero
neptuno	6930	perez	cajero
nueva	15750	pineda	jefe

Queremos ascender a **PEREZ** al puesto de **jefe** incrementándole el salario en un **50%**. La sentencia sería:

```
UPDATE empleados SET puesto=JEFE, salario=salario*1.5 WHERE nombre_emp=PEREZ;
```

Tabla de empleados con ascenso

En detalle

Tabla de clientes con incremento de saldo

sucursal	num_cta	nombre_cli	saldo
neptuno	1	perez	1575
caleta	2	perez	12075
caleta	3	gaya	525
nueva	4	alvarez	105
nueva	5	pineda	15750
zaidin	10	perez	210
zaidin	11	alvarez	210

En detalle

Tabla de empleados con ascenso

sucursal	salario	nombre_emp	puesto
nueva	158	alvarez	cajero
caleta	525	gaya	cajero
neptuno	4620	perez	jefe
nueva	15750	pineda	jefe

Borrar datos

La instrucción **DELETE** permite borrar datos de una tabla indicada:

```
DELETE FROM <tabla> WHERE <condición>;
```

Donde:

- **<tabla>** es la tabla que contiene los datos que queremos eliminar.
- Si se **indica una condición**, se eliminan las filas de la tabla que la satisfagan.
- Si se **omite la condición de búsqueda**, se eliminan TODAS las filas.
- Si se quiere eliminar tanto el contenido de la tabla como su definición (esquema), debe usarse el comando **DROP TABLE**.

Ahora la **Sra. García** quiere cerrar la cuenta bancaria que abrió en la sucursal **Nueva**. El cajero debería darla de baja eliminando la fila correspondiente de la tabla de depósitos. Esto se hace con la sentencia **DELETE** del LMD de SQL.

La sentencia que permite eliminar la cuenta de la tabla de depósitos de la Sra. García es:

```
DELETE FROM depositos WHERE nombre_cli='GARCIA';
```

La **tabla de depósitos** quedaría:

sucursal	num_cta	nombre_cli	saldo
neptuno	1	perez	1500
caleta	2	perez	11500
caleta	3	gaya	500
nueva	4	alvarez	100
nueva	5	pineda	15000
zaidin	10	perez	200
zaidin	11	alvarez	200

Ejemplo

Veamos ahora dos ejemplos sobre cómo **borrar datos** de una tabla.

- El banco ha decidido renovar su plantilla de trabajadores. Para ello va a empezar por los jefes. Necesitamos eliminar de la **tabla de empleados** aquellos que sean **jefe**.

```
DELETE FROM empleados WHERE puesto='JEFE';
```

El resultado sería esta tabla:

sucursal	salario	nombre_emp	puesto
nueva	158	alvarez	cajero
caleta	525	gaya	cajero

- El banco sigue con su expediente de regulación de empleo, quiere **eliminar a todos los empleados del banco**. La sentencia sería:

```
DELETE FROM empleados;
```

El resultado es la **tabla en blanco**:

sucursal	salario	nombre_emp	puesto

El comando **DELETE** solo borra las instancias de la tabla. Si queremos **eliminar la tabla de empleados** de nuestra base de datos, tendremos que utilizar la sentencia **DROP TABLE**. La sentencia sería:

```
DROP TABLE empleados;
```

Resumen

SQL permite a través de su LMD modificar la instancia de la BD. Para ello dispone de las operaciones de insertar, modificar y eliminar.

- **Insertar**. La operación de insertar introduce nuevas instancias en una relación, lo que es equivalente a decir que introduce filas en una tabla. Existen dos formas de realizar la operación de inserción:
  - Indicando los valores que queremos que formen la fila a insertar. En este caso se utiliza la sintaxis:
  - INSERT INTO tabla (listacolumnas) VALUES (lista valores)
  - Estableciendo una consulta cuya tabla resultado estará compuesta por las filas que queremos insertar:
    INSERT INTO tabla (listacolumnas) SELECT…

- **Modificar**. La operación de modificar permite cambiar los valores que determinadas instancias tienen en la relación. La sintaxis es:
  UPDATE tabla SET columna = valor, … WHERE condición
- **Borrar**. La operación de borrar permite eliminar una instancia ó conjunto de instancias de la BD. La sintaxis que se utiliza es:
  DELETE FROM tabla WHERE condición..