Fundamento Bases de Datos
Facultad de Ciencias UNAM
M.I. Gerardo Avilés Rosas
gar@ciencias.unam.mx
Laboratorio:
Efraín Hipólito Chamú <*chamugauss@gmail.com* >

PRACTICA 09
INSERT-DELETE-UPDATE

1. Introducción

Aun cuando ya se creó la base de datos dentro del SMBD, es preciso agregar los datos. Esta tarea es sumamente importante dado que un proceso de carga de los datos de entrada con errores puede ocasionar múltiples fallas como pueden ser errores de integridad o duplicidad de información que dejen en un estado inconsistente a la base de datos.

- Inserción de datos:

Antes de iniciar el proceso de inserción de datos, es recomendable verificar que se cuenta por un lado con los datos y por otro con la tabla ya creada, es importante aclarar que si bien ya se encuentran cargados los datos, la tabla puede ser modificada (alterada) como se presentó anteriormente (PRACTICA07).

Por otro lado, la sentencia INSERT se encarga de agregar los datos a las tablas.

```
INSERT INTO nombre_tabla (atributo1, atributo2,..,atributon ) VALUES (valor1,valor2,..valorn);
```

Una sentencia INSERT típica es la siguiente:

```
INSERT INTO Socio VALUES (3, 'pedro', 'garcía', 'hernández','01/01/12','20');
```

Hay que notar dos aspectos, primero el orden en que se colocan los datos después de la palabra VALUES es importante ya que existe una correspondencia uno a uno entre los datos insertados y los atributos definidos de la tabla en cuestión. Si se desea modificar el orden de los datos por insertar, los atributos deben aparecer explícitos, en nuestro ejemplo la sentencia que se presenta a continuación es equivalente a la anterior:

```
INSERT INTO Socio(id_socio, nombre, app_, apm_, fecha_nacimiento, edad) VALUES (3, 'pedro', 'garcía', 'hernández','01/01/12','20');
```

Al igual que la siguiente (notar que se cambió de posición el atributo edad):

```
INSERT INTO Socio(edad,id_socio, nombre, app_, apm_, fecha_nacimiento) VALUES ('20', 3, 'pedro', 'garcía', 'hernández','01/01/12');
```

- Eliminación de datos:

A efecto de eliminar información de nuestra base de datos, desde una tupla hasta el total de una tabla, se cuenta con el comando DELETE. Su sintaxis es la siguiente:

```
DELETE FROM nombre_tabla [WHERE condición_booleana];
```

Como podemos observar, la sentencia es relativamente sencilla sin embargo es importante resaltar tres aspectos. El primero y más importante radica en que la condición booleana no es más que el valor de una expresión que regresa valores del tipo boolean y determina las tuplas que serán eliminadas; segundo, que aun cuando aparece la condición booleana como un parámetro optativo, es necesario especificarlo siempre pues el omitirlo provoca un borrado total de los datos contenidos en la tabla. El tercer aspecto importante es que las condiciones booleanas pueden generarse a partir de una simple comparación o hasta del resultado de una consulta sobre otras tablas.

Como ejemplo tenemos el caso donde se vacía en su totalidad la tabla Socio ya que no hay condición alguna.

```
DELETE FROM Socio;
```

Ahora, solamente se está eliminando el socio al cual le corresponde el 'id_socio' con valor de 2.

```
DELETE FROM Socio WHERE id_socio=2;
```

En este caso se eliminan los socios cuya edad no sean de 29 años.

```
DELETE FROM tomar_clase WHERE ID_SOCIO NOT IN (SELECT ID_SOCIO FROM Socio WHERE EDAD=29);
```

La eliminación de datos es un proceso importante dado que en caso de falla durante el mismo, siempre tendrá consecuencias en la nueva captura de los datos. Aun así, se ha visto que también es un proceso muy sencillo.

Por último, hay que notar que aunque existe cierta semejanza entre los comandos DELETE y DROP, existe un diferencia importante, DROP elimina por completo el esquema que define a la tabla en cuestión, mientras que DELETE solamente elimina los elementos que se encuentran dentro de esta, por ello DELETE resulta una operación muy costosa cuando la base de datos ha crecido considerablemente.

- Actualización de datos

El hecho de que tengamos la información dentro de la base de datos no es, ni por mucho, el fin del proceso de explotación de la misma. Generalmente las aplicaciones que interactúan con la base de datos se encargan de actualizarla, así por ejemplo, en una base de datos de alumnos su promedio se actualiza semestralmente o en una base de datos de cuentas bancarias, el saldo de cada socio comercial se puede actualizar incluso cada segundo.

En SQL el comando que proporciona esta funcionalidad es UPDATE. Su sintaxis básica es la siguiente:

```
UPDATE nombre_tabla SET columna = expresión [WHERE condición_booleana];
```

La utilización de UPDATE es similar a DELETE y por esto, se deben tener en cuenta las mismas precauciones que se nombraron anteriormente.

Como ejemplo se tiene la actualización de la edad cuyo nombre es pepe de la tabla Socio.

```
UPDATE Socio SET edad=31 WHERE nombre='pepe';
```

Entregables

1. Usando el comando INSERT, ingresa 100 tuplas a cada tabla de la base de datos "Gimnasio Hércules". Guarda tus sentencias en un archivo inserta.sql.
2. Elimina 10 tuplas de cada una de las tablas de la base de datos utilizando la sentencia DELETE. Guarda tus sentencias en un archivo elimina.sql.
3. Genera un script que actualice al menos 10 tuplas de 3 entidades diferentes utilizando la sentencia UPDATE. Guarda tus sentencias en un archivo actualiza.sql.
4. Crea 2 registros de inserción que prueben que las restricciones que generaste en la Práctica08, un registro deberá lograr la inserción y el otro no. Explica en forma de comentarios porqué una inserción se logra y la otra no. Guarda tus sentencias en un archivo prueba.sql.
5. Entregar: inserta.sql, elimina.sql, actualiza.sql y prueba.sql
6. **Fecha de entrega: 22 de Octubre del 2017, hora máxima: 23:59:59.**

Nota: Puedes apoyarte de la página http://www.generatedata.com/ para generar tus sentencias (Opción SQL en EXPORT TYPES).