BBDD

Bases de Datos – Manipulación de Datos

IES Ciudad Escolar
Desarrollo de Aplicaciones Multiplataforma

Objetivos

- Utilizar las sentencias facilitadas por el sistema gestor para la inserción, borrado, y actualización de la información.
- Incluir en una tabla la información resultante de la ejecución de una consulta.

MySQL®

Lenguaje SQL

El lenguaje de programación **SQL es el lenguaje fundamental de los SGBD** relacionales y los elementos que lo componen son:

a) **DML (*Data Manipulation Language*)**: es el lenguaje que consulta o manipula los datos ya existentes de nuestra BD.

b) **DDL (*Data Definition Language*)**: permite la **definición, modificación y eliminación de las estructuras básicas** (BD, tablas, etc.) en un SGBD.

c) **DCL (*Data Control Language*)**: administra a los usuarios de la BD, concediendo o denegando los permisos oportunos.

d) **TCL (*Transaction Control Language*)**: lenguaje que controla el procesamiento de las transacciones de la BD.

DML (Data Manipulation Language)

DML (Data Manipulation Language) o Lenguaje de Manipulación de Datos es la parte de SQL dedicada a la manipulación de los datos.

Las sentencias DML son las siguientes:

- **SELECT**: se utiliza para realizar consultas y extraer información de la base de datos.
- **INSERT**: se utiliza para insertar registros en las tablas.
- **UPDATE**: se utiliza para actualizar los registros de una tabla.
- **DELETE**: se utiliza para eliminar registros de una tabla.

INSERT (inserciones de registros)

- De forma introductoria, podemos decir que la sentencia **INSERT** nos permite dar de alta (insertar) registros en las tablas de las bases de datos.

- Debemos prestar atención a los posibles errores que pueden surgir en el alta de registros:
  - Que no coincidan los tipos.
  - Que dejemos vacío un campo con restricción **NOT NULL**.
  - Que tratemos de insertar un registro en una tabla cuyo valor de clave esté ya presente en la tabla, **PRIMARY KEY**.
  - Que incumplamos alguna condición con restricción **CHECK**.
  - Que tratemos de insertar un registro en una tabla que tiene una restricción de clave extranjera en un campo, y el valor introducido en ese campo no se corresponda con ninguno de los que existen en la tabla de referencia, **FOREIGN KEY**.
  - Que tratemos de insertar un registro en una tabla que tiene un campo definido como **UNIQUE** y nuestro valor ya esté presente en la tabla.

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [(col_name [, col_name] ...)]
    { {VALUES | VALUE} (value_list) [, (value_list)] ... }
    [AS row_alias[(col_alias [, col_alias] ...)]]
    [ON DUPLICATE KEY UPDATE assignment_list]

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    SET assignment_list
    [AS row_alias[(col_alias [, col_alias] ...)]]
    [ON DUPLICATE KEY UPDATE assignment_list]

INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [(col_name [, col_name] ...)]
    { SELECT ...
      | TABLE table_name
      | VALUES row_constructor_list
    }
    [ON DUPLICATE KEY UPDATE assignment_list]

value:
    {expr | DEFAULT}

value_list:
    value [, value] ...

row_constructor_list:
    ROW(value_list)[, ROW(value_list)][, ...]

assignment:
    col_name =
          value
        | [row_alias.]col_name
        | [tbl_name.]col_name
        | [row_alias.]col_alias

assignment_list:
    assignment [, assignment] ...
```

https://dev.mysql.com/doc/refman/8.0/en/insert.html

INSERT (inserciones de registros)

- Disponemos de distintas sintaxis a la hora de insertar registros mediante la sentencia **INSERT**:

```
INSERT INTO nombre_tabla (col1,col2, colx) VALUES (val1, val2, valx);

INSERT INTO nombre_tabla VALUES (val1, val2, val3, val4);

INSERT INTO nombre_tabla VALUES (val1, val2, val3, val4), (val5, val6, val7 , val8)... ;

INSERT INTO nombre_tabla (col1, col2, col3, col4)
                    SELECT valor1, col2, col3, valor4
                    FROM …
                    WHERE … ;
```

- Si se detecta un error durante la inserción de muchos registros (ej. insert into … select…), se cancelarán el resto de inserciones pendientes. Para forzar a que se inserten todos los registros válidos, podemos utilizar la clausula **IGNORE**: INSERT IGNORE INTO nombre_tabla ...

- Si queremos forzar a que todas las inserciones tengan lugar, podemos utilizar la sentencia **REPLACE:** REPLACE INTO nombre_tabla .... De esa forma, si encuentra un registro con la misma PK o hay conflicto con una restricción UNIQUE, el SGBD realizará el borrado del registro existente para habilitar la inserción del nuevo.

UPDATE (modificaciones de registros)

- De forma introductoria, podemos decir que la sentencia **UPDATE** nos permite modificar registros (uno o varias columnas) en las tablas de las bases de datos.

- Debemos prestar atención a las siguientes circunstancias que se presentan en la modificación de registros:

  - Al actualizar se comprueban TODAS las condiciones de integridad de datos (PKs, FKs, UNIQUE, NOT NULL, etc)

  - Los registros mantienen los valores originales de las columnas no actualizadas.

  - Si no especificamos al menos una condición para la modificación y la base de datos permite UPDATE sin condiciones, se modificarán TODOS los registros de la tabla.

  - Si especificamos la clausula ORDER BY, las actualizaciones se llevarán a cabo en el orden indicado.

  - Si especificamos la clausula LIMIT, un número de registros serán procesados como máx. para ser actualizados, lo que no significa que se actualicen ese número concreto de registros.

  - Podemos realizar actualizaciones en múltiples tablas de una sola vez, añadiendo las tablas en la clausula UPDATE y las columnas en la clausula SET. (no permite LIMIT ni ORDER BY)

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
    SET assignment_list
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]

value:
    {expr | DEFAULT}

assignment:
    col_name = value

assignment_list:
    assignment [, assignment] ...
```

https://dev.mysql.com/doc/refman/8.0/en/update.html

```
UPDATE nom_tabla
SET col1 = val1, col2 = val2, …;

UPDATE nom_tabla
SET col1 = val1, col2 = val2, …
WHERE …;

UPDATE nom_tabla1, nom_tabla2, …
SET nom_tabla1.col1 = val1,
    nom_tabla2.col2 = val2, …
WHERE …;
```

DELETE (borrado de registros)

- De forma introductoria, podemos decir que la sentencia **DELETE** nos permite borrar o eliminar registros (completos) en las tablas de las bases de datos.

- Debemos prestar atención a las siguientes circunstancias que se presentan en la modificación de registros:

  - Al borrar registros se comprueban TODAS las condiciones de integridad de datos también.

  - Si no especificamos al menos una condición para el borrado, y la base de datos permite DELETE sin condiciones, se borrarán TODOS los registros de la tabla (similar a un TRUNCATE).

  - Debemos por tanto especificar en la clausula WHERE las condiciones de los registros a borrar.

  - Si especificamos la clausula ORDER BY, los borrados se llevarán a cabo en el orden indicado.

  - Si especificamos la clausula LIMIT, un número de registros serán procesados como máx. para ser borrados, lo que no significa que se borren ese número concreto de registros.

  - Podemos realizar borrados en múltiples tablas de una sola vez, añadiendo las tablas en la clausula DELETE y las condiciones en el WHERE. (no permite LIMIT ni ORDER BY)

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name [[AS] tbl_alias]
    [PARTITION (partition_name [, partition_name] ...)]
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
    tbl_name[.*] [, tbl_name[.*]] ...
    FROM table_references
    [WHERE where_condition]

DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
    FROM tbl_name[.*] [, tbl_name[.*]]
    USING table_references
    [WHERE where_condition]
```

https://dev.mysql.com/doc/refman/8.0/en/delete.html

```
DELETE FROM nom_tabla
WHERE …;

DELETE nom_tabla FROM
nom_tabla1, nom_tabla2, …
WHERE …;

DELETE nom_tabla1
FROM nom_tabla1
        xxxx JOIN nom_tabla2
        ON (condición_cruce)
WHERE (condición_filtrado);
```