***Clase 11. Sublenguaje DML 1***

SENTENCIAS INSERT, UPDATE, DELETE

# SUBLENGUAJE DML 1

**DML: Data Manipulation Language.**

Inicialmente aprendimos que DML posee una serie de sentencias que habilitan la manipulación de la información presente en las tablas de una DB, pudiendo agregar, actualizar y/o eliminar parte o toda está información.

## ¿DML o DQL?

Si bien mencionamos oportunamente la sentencia SELECT como parte de DML, en algunos casos podemos encontrar publicaciones que la agrupan bajo el paradigma DQL (Data Query Language).

Esto tiene lógica dado que, el término DQL, hace foco en “queries” o “consultas” en sí, que es lo que realmente hacemos utilizando la sentencia SELECT para obtener datos.

Ejemplo:

**SELECT** **\***

**FROM** **system\_user**

**WHERE** **last\_name like**

**‘B%’;**

## Componentes de DML

### INSERT

INSERT: se utiliza para insertar o agregar registros en una tabla.

La información que agregamos puede ser :

👉De forma individual (1 registro).

👉De forma plural (varios registros a la vez).

#### ¿Cómo se utiliza INSERT?

Su sintaxis se elabora mediante la cláusula

**INSERT INTO**, seguida del nombre de la tabla, la cláusula **VALUES**, y los datos que se insertarán en la misma.

Ejemplo:

**INSERT INTO** **nombre\_de\_la\_table** **(campo1, campo2, campo3,....)**

**VALUES** **(dato1, dato2, dato3, ...);**

Cada “dato” hace referencia al dato individual que conformará el registro que deseamos insertar.

Es conveniente detallar los nombres de los campos que componen la tabla, caso contrario tenemos que dar los valores a insertar en el orden que aparecen en la tabla (según orden de creación de la tabla).

Si mi tabla tiene 4 campos, entonces debería insertar 4 datos, uno por cada campo.

Ejemplo:

**MiTabla** **VALUES**

campo1 dato1

campo2 ‘dato2’

campo3 ‘dato3’

campo4 dato4

Si el tipo de datos de un campo es number o boolean, el dato a insertar no lleva comillas.

Si es text o varchar, sí lleva comillas.

Ejemplo práctico de INSERT:

**INSERT INTO** **class (id\_level,id\_class,description)**

**VALUES** **(1, 999, ‘Spain comedy’);**

#### ¿Y los campos ‘AUTOINCREMENT’?

En el caso de contar con campos AUTOINCREMENT, tenemos la posibilidad de permitir que el DBMS resuelva el valor correlativo para éste, pasando como parámetro el valor NULL o no colocandolo en la lista de campos.

Tomemos como ejemplo la tabla PAY que creamos en la clase 6.

**INSERT INTO** **pay** **VALUES** **(NULL, 250, ‘U$S’, ‘2021-07-22’, ‘Paypal’, 850, 77);**

#### INSERTAR DATOS PARCIALES

Podemos insertar datos en determinados campos de una tabla.

Para ello, debemos especificar en la sentencia **INSERT INTO,** cuáles son los campos donde insertamos información.

Ejemplo:

**INSERT INTO pay (id\_pay, amount, currency, date\_pay, id\_system\_user, id\_game)**

**VALUES** **(** **NULL, 300, ‘U$S’, ‘2021-07-22’, 501, 13);**

El total de datos a insertar debe condecir con el total de campos mencionados, además de tener que coincidir el tipo de dato.

Si la tabla posee un campo AUTOINCREMENT, debes referenciarlo, asignándole el valor NULL.

Si los campos que obviamos poseen un valor definido por defecto, estos aparecerán en el registro insertado.

Caso contrario, el dato quedará como NULL

#### INSERTAR MÚLTIPLES DATOS

Podemos aprovechar la sentencia INSERT para agregar múltiples registros en una misma ejecución. **Cada nuevo registro debe encerrarse entre (...) y separarse con una coma.**

Ejemplo:

**INSERT INTO pay VALUES**

**(NULL, 250, 'U$S', '2021-07-22', 'Paypal', 850, 77),**

**(NULL, 3700, 'Pesos Arg', '2021-07-22', 'Visa', 38, 31),**

**(NULL, 180, 'Libras', '2021-07-22', 'Transfer', 175, 16);**

De igual forma procederemos en el caso de tener que agregar múltiples registros sin especificar todos los campos de la tabla.

### UPDATE

#### ACTUALIZAR REGISTROS

La sentencia SQL UPDATE actualiza registros almacenados en uno o más campos de una tabla.

Su sintaxis es simple, y permite hacer una actualización masiva de datos, o aquellos registros que cumplan con una determinada condición.

Ésta se elabora mediante la cláusula **UPDATE** tabla, seguida la palabra **SET** y luego la(s) columna(s) o campo(s) con su(s) nuevo(s) valor(es).

Ejemplo:

**UPDATE** **nombre\_de\_la\_tabla** **SET** **campo2 = ‘dato2’;**

Para actualizar el valor de más de un campo, debemos separar cada uno de éstos con una coma.

Especificamos **campo = valor, otroCampo = otroValor**, y así con cada campo que deseamos actualizar.

Ejemplo:

**UPDATE** **nombre\_de\_la\_tabla SET** **campo1 = ‘dato1’, campo2 = ‘dato2’, campo3 = ‘dato3’;**

#### ACTUALIZAR DATOS QUE CUMPLAN DETERMINADA CONDICIÓN

Podemos limitar la actualización de datos que cumplan una determinada condición.

Por ejemplo, cambiar la fecha de pago de aquellos realizados en el día. Para esto debemos integrar la cláusula WHERE, especificando la o las condiciones que deben cumplirse.

Ejemplo:

**UPDATE** **pay**  
**SET date\_pay = CURRENT\_DATE - 1**

**WHERE** **date\_pay = CURRENT\_DATE;**

**Nota:** es importante conocer los modos de seteos del servidor y de la sesión:

SELECT @@GLOBAL.sql\_mode;

SELECT @@SESSION.sql\_mode;

Sobre todo para el caso del update anterior o de los delete que vendrán luego que la variable safe\_mode esté en 0. Si se realizan actualizaciones que no involucren la clave de la tabla puede dar error. Para setearla utilizar:

**SET SQL\_SAFE\_UPDATES=0;**

### DELETE

#### ELIMINAR REGISTROS

Para eliminar registros de una tabla, debemos utilizar la sentencia **DELETE.**

Se ocupa de eliminar todos los registros, o sólo aquellos que coincidan con determinados parámetros que le indiquemos en la condición del **WHERE.**

#### ¿Para qué se utiliza DELETE?

Su sintaxis se elabora mediante la cláusula **DELETE FROM** seguido del nombre de la tabla. Además debemos agregar la cláusula **WHERE**, para indicarle cuál o cuáles registros se deben eliminar.

Ejemplo:

**DELETE FROM nombre\_de\_la\_tabla** **WHERE (campo = ‘dato’);**

##### ¡ATENCIÓN CON ESTA CLÁUSULA!

Seamos cuidadosos al utilizar DELETE.

Siempre comencemos escribiendo WHERE, junto a la o las condiciones que deben cumplir los registros a eliminar.

Un error involuntario, como el olvidar el condicional, hará que perdamos toda la información de la tabla

#### POSIBLES ERRORES AL INTENTAR ELIMINAR REGISTROS

Si intentamos eliminar registros de una tabla cuya PRIMARY KEY es FOREIGN KEY en otra u otras tablas, SQL no realizará la operación y nos advertirá dicho impedimento a través de la consola.

Por ejemplo: **delete from** **level\_game** **where** **id\_level = 5;**

Saldrá el siguiente error:

Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`gamers`.`level\_game`, CONSTRAINT `fk\_class\_level` FOREIGN KEY (`id\_level`) REFERENCES `le` (`id`))

Ante esta situación, debemos eliminar primero el registro asociado mediante la FOREIGN KEY en la otra tabla, para luego proceder con la eliminación de este registro.

#### ELIMINACIÓN TOTAL DE REGISTROS DE UNA TABLA

Recordemos que para eliminar todos los registros de una tabla podemos utilizar **TRUNCATE** en lugar de **DELETE** sin where.

Internamente, TRUNCATE borra todos los registros mientras que DELETE, recorre uno a uno y los va eliminando.

Ejemplo:

**TRUNCATE** **nombre\_de\_la\_tabla;**