**SQL (Structured Query Language)**

**Poniendo en práctica la teoría en MySQL: DML**

El lenguaje SQL tiene dos grandes rubros. Ya vimos el corte DDL, ahora nos toca DML.

**DML (Data Manipulation Language)**

A diferencia de **DDL,** que consistía en formar la estructura de la base de datos; es decir, lo que concierne a lo primero: formar nuestras tablas, nuestras relaciones, las restricciones (constrains),... y toda esa parte que tiene que ver con la *estructura* propiamente y que se utiliza muy al inicio de un proyecto, está **DML.** Es DML, a diferencia DDL, quien sí se utiliza mucho a lo largo del proyecto, DDL no (más que todo al inicio). DML trata, no de la estructura de la base de datos; si no, del contenido propio de la base de datos (sus datos). Lo dicho, se trata más bien de la manipulación directa sobre los datos (meterlos, actualizarlos, extraerlos y/o borrarlos); mas no sobre las tablas o, incluso, sobre las bases de datos.

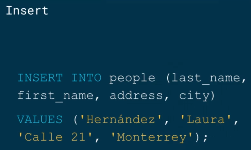
Dicho lo anterior, para ejercer cualquier tipo de acción directa sobre los datos; quiere decir que con anterioridad, además de que ya hay evidencia de una base de datos, la hay de una tabla (al menos); pues, para trabajar con DML y, en consecuencia, con los datos (ya sea meterlos, actualizarlos, extraerlos y/o borrarlos), ya deben existir bases de datos y, dentro de la base de datos en cuestión, tablas (o con una solamente): lo que es propio a las funciones del lenguaje DDL. Quiere decir entonces que, para trabajar con el lenguaje DML, anteriormente ya debiera haber trabajado con el lenguaje DDL.

**Cuatro comandos principales de DML**



Estos son los comandos que vas a usar, a diferencia de los comandos o sentencias de DDL, a lo largo de todo un proyecto (como por ejemplo *platziblog*): cada vez que quieras meter *(Insert),* actualizar *(update),* borrar *(delete)* o extraer *(select)* una información en concreto (datos). Vamos a ver, una a una, cómo se estructura cada sentencia y, de paso, cómo se operan y se ejecutan ya propiamente dentro de consola MySQL.

**Insert**

, *expliquemos la sentencia* ***Insert*** *parte por parte.*

**1ra parte:**

***Insert into:*** Es la sentencia que tiene como función *insertar* (insertar un dato, o varios) *into,* dentro, de una tabla de una base de datos (la que le pase, estando situados sobre una base de datos*,* y que ya debe estar creada).

***people:*** Es la tabla pasada, de una base de datos, en cuestión; en esta ocasión, la tabla ha sido creada con el nombre de *people;* lo dicho, la tabla ya debe haber sido creada con anterioridad y, dentro de esta sentencia, significa que es la tabla que ha sido seleccionada, *people,* para *insertarle* un nuevo (o nuevos) datos.

***(last\_name, first\_name, address, city):*** son los campos, también atributos (o columnas de una tabla), ya existentes dentro de una tabla (la tabla *people* en este caso), a los que les pasará los nuevos datos que piensa insertar para cada uno de ellos, respectivamente. La cantidad de datos a *insertar* va a depender de la cantidad de parametros (campos), separadados por coma, que usted le pase a la sentencia *insert* dentro de sus parentesis.

Dato que no pase para una columna o campo en cuestión es rellenada por *default,*... se recuerda este constraint? Es decir, si resulta que no llamó a un campo de su tabla, en la sentencia *insert,...* pasa que dentro de su celda, del nuevo renglón (registro) que se rellene con datos para dicha tabla, se va a poner (recordamos, para la celda de dicho campo que no llamó) un valor por *default* (ya sea el que se pase por defecto, que es NULL, o el que usted le configure con la restricción *DEFAULT*).

**2da parte:**

***values:***Siguiendo el mismo orden respectivo con el que fueron convocados los campos, columnas o atributos,... así mismo, *values,* pondrá justamente los datos que usted desea pasar para dichos campos en cuestión; es decir, en *values*, se evidencia propiamente los valores que usted desea insertar a cada uno de los campos llamados o convocados en la *1ra parte.* Por ejemplo, si teníamos que los campos llamados fueron: ***(last\_name, first\_name, address, city)*** y los valores, o datos, pasados serán: ***(‘Hernandez’, ‘Laura’, ‘Calle 21’, ‘Monterrey’).***

*Quiere decir que,* ***el nuevo dato a insertar en una nueva regla o registro,...***

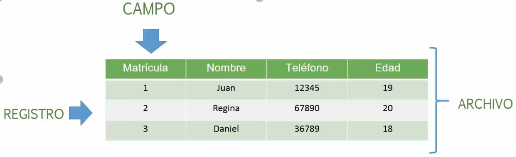
*para* ***last\_name*** *será:* ***‘Hernandez’;***

*para* ***first\_name*** *será:* ***‘Laura’****;*

*para* ***address*** *será:* ***‘Calle 21’****;*

*& para* ***city*** *será:* ***‘Monterrey’...*** *si se da cuenta es muy importante mantener la equivalencia en el orden.*

Tenga en cuenta lo siguiente: **se hace un registro, por renglón, con cada sentencia *value* que hagamos (luego de declarar un *insert*). Es decir, supongamos que los campos, atributos, hacen parte de las columnas de la tabla (la cabecera de ellas); entonces, lo que se hace con *value* es que se pase sólo un nuevo registro, una nueva fila, para insertar sólo un dato para cada una de ellas (para cada una de las columnas o campos).** Recuerde esto:



**Vamos a la consolaahora para crear un nuevo registro (un nuevo renglón con datos para cada campo de una tabla) en nuestra tabla *people* de la base de datos *platziblog.***

Acá toca, por lo general, *codificar* directamente el lenguaje de programación SQL para declarar nuestras sentencias de manipulación directa sobre cada celda o dato de cada campo o columna de una tabla. Esa *codificación* se hace en algo que se llama *terminal* o *cliente gráfico,* tal que así:

Abrimos una nueva pestaña ; ya estando situados y habiendo seleccionado por *default* la base de datos en cuestión, obvio, y la tabla de su interés,... para correr justamente nuestro mismo caso de estudio. Luego, declaramos en la *terminal* o *cliente* *gráfico,* de una nueva pestaña, lo siguiente:

INSERT INTO people (last\_name, first\_name, address, city)

VALUES ('Hernandez', 'Laura', 'Calle 21', 'Monterrey');

***Debe verse así:***



Luego de constatar que, en efecto, queremos ejecutar ese código; que significa, recordemos, agregar un nuevo registro o fila con un nuevo dato para cada una de las columnas o campos de la tabla *people* de la base de datos *platziblog,...* luego de constatar eso, *corremos* .

Y, ***en los mensajes del sistema,*** podrá develar que ya se ejecutó nuestra línea de código y sentencia para afectar nuestra tabla *people. Veamos:*



Dice que se afectó*, agregó,* un nuevo registroo fila *(row).* Si todo salio bien, verá un *check* *.* En todo caso, *select* su tabla *people* nuevamente para que visualice que, en efecto, se produjeron los cambios deseados: *insertar* un registro para ella con los datos pasados y en los campos deseados.

En efecto, todo salió bien:



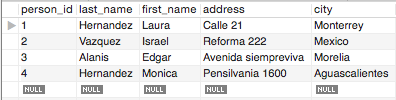
Ahora, puede agregar cuantas ***values*** desee, en una misma sentencia ***value,*** de esta manera: cada conjunto de datos (valores) pasados entre parentesis ***()*** va a representar un nuevo registro o renglón a la tabla; esto sin necesidad de declarar nuevamente, para cada conjunto de datos por registro o renglón, un ***insert*** o un ***value.* Sólo es necesario separar, por medio de comas, cada conjunto de datos pasados entre parentesis para una misma sentencia *value.*** Mejor veamos esto en consola:



Si se da cuenta, hemos creado ahora 3 filas más (o registros) a la tabla *people* de la base de datos *platziblog,...*



*y exitosamente.* En efecto, todo sigue saliendo bien:

 Esto es todo en cuanto a ***Insert.***

**Update**