**SQL (Structured Query Language)**

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

El lenguaje SQL tiene dos grandes rubros o dos, digamos, sublenguajes. Son DDL &

**DDL (Data Definition Language) y sus sentencias/comandos**

Es un lenguaje que nos ayuda a crear la estructura de una base de datos. Más precisamente, nos ayuda a crear los cimientos: las relaciones, las entidades, los atributos de dichas entidades; todo eso que ya hemos venido hablando en la *teoría* con respecto a los diagramas; entonces, es este el lenguaje que nos va a permitir crear todas esas estructuras ya de facto en una base de datos real. **En DDL se utilizan tres grandes comandos: Create, Alter & Drop.**

**Create**

Como su nombre lo dice, nos ayuda a crear, ¿crear qué cosa? Pues, podría ser una base de datos (schema), también podría crear una tabla (la representación de una entidad) o una vista; al igual, podría ser un índice, etc.

**Alter**

Como su nombre también lo dice, nos va a permitir *alterar* o modificar algunas de esas entidades que hemos creado con *create* (representadas singularmente por *tablas*);al igual,podemos modificar la tabla en cuestión, por ejemplo, agrengandole una nueva columna (o quitandosela); o, también, podría considerar cambiar un tipo de dato de alguna columna, por ejemplo. Las columnas o campos de una tabla *(entidad)* son sus atributos.

**Drop**

*Soltar.* Tiene que ver con *dejar ir.* Es básicamente la sentencia que nos va a ayudar a **borrar;** por lo que, toca tener mucho cuidado con ella. Al hacer *drop,* por ejemplo, tú puedes borrar una columna (atributo de una entidad), borrar una tabla completa (una entidad) o, incluso, toda tu base de datos.

**Los tres grandes grupos de objetos que vamos a manipular con el lenguaje DDL son: Database, Table & View.**

**Database:** Se entiende como el repositorio de datos que va a usar algún proyecto, generalmente grande, que se encuentre desarrollando.

**Table:** Son la traducción a SQL de cómo crear las entidades, la puesta en marcha de lo que ya hemos visto en la parte teórica de las bases de datos relacionales; es decir, con *table* es que se crean ya las entidades, pero haciendo uso de un lenguaje de programación como SQL. Es como se pone en práctica toda la teoría relacionada a la creación de entidades, tanto la parte de *entidad-relación* como los *diagramas físicos*; también, lo que tiene que ver con sus atributos, tipos de datos y restricciones, etc: la información que envuelve a toda *table* en sí. Cada tabla representa una entidad por separada; mas, en una base de datos, puede tener varias tablas.

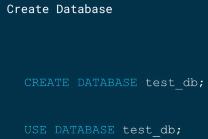
**View:** Hace referencia a la manera en cómo proyectamos toda la información de una base de datos de tal manera que se entienda muy bien, casi que para cualquier persona, lo que se transmite. Usted se recuerda que, como ya tendríamos *normalizada* toda la información; tendríamos, entonces, un pedacito de la información en una tabla, otro pedacito en otra tabla *,* y otro en otra *(table)* y así sucesivamente. Como la idea es que toda esa información pueda ser presentada de una manera clara y conjunta, de una manera coherente para cualquier persona y que sirva para comunicar una información concreta, interpretable y con sentido propio, se recurre al recurso de los *views* por medio de los *selects* (más adelante profundizamos este nuevo concepto).



***Entonces, tenemos que, los comandos Create, Alter & Drop son los que manipulan a los objetos Database, Table & View,... Echemos un vistazo a las primeras sentencias que podemos declarar al conjugar ambos factores: comandos & objetos.***

***Un vistazo a las primeras sentencias,... Creación de una base de datos (database) en MySQL***

***Ejemplo 1.***



Lo que estamos haciendo es, por medio de la sentencia ***create,***crear una base de datos. La base de datos, recuerdese, es un **objeto:** *Database (*o lo que es lo mismo: *Schema).* Posteriormente le hemos pasado un nombre a la reciente base de datos creada, el cual es: “test\_db”. Esto en cuanto a nuestra primera **sentencia.**

Luego, en la segunda sentencia, hemos dejado claro que quiero usar la base de datos “test\_db” al declarar la expresión *use.* Es preciso hacer uso del recurso *use* para que la consola interprete que tienes definido sobre qué objeto vas a ejercer una acción y/o manipular (hacer consultas o algún cambio).

Apliquemos la primera sentencia, del ejemplo 1, en una base de datos real por medio de MySQL Workbench; esto para que veas, ahora que ya conoces, cuáles son las sentencias (comandos) y cuáles son los objetos y, en consecuencia, cuál sería la estructura que nos va a quedar al final en una sentencia SQL.

Si recuerda esto: , (Si no lo recuerda, lea el word que está en la carpeta: ***2. Presentación del RDBMS,...***) pues bien, es exactamente lo que hemos venido explicando en estos últimos párrafos. Debe reconocer en la declaración cuál es la sentencia (comando) y sobre qué objeto se está aplicando. Ahora, como resultado, tendríamos una nueva base de datos con el nombre de: **platziblog:** 

En caso que queramos aplicarle la sentencia ***use database*** a ***platziblog,*** viendose entre líneas de código algo así: **use schema ‘platziblog’**,... debemos, primero que todo, presionar click sobre la base de datos *platziblog.*

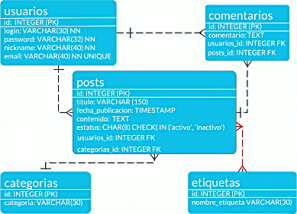
Luego, debemos seleccionar la opción:  (recuerde, esta opción se habilita al presionar click derecho sobre una base de datos ya creada; en nuestro caso, sobre **platziblog**), y listo!

Sin hacer uso de códigos extras le hemos dicho a la consola que queremos centrarnos solamente en la base de datos ***platziblog;*** es decir, *platziblog* sería la base de datos por default para cualquier consulta (o manipulación) que se haga de ahora en adelante en MySQL Workbench. Posteriormente verá algo así:



***Luego de tener nuestra base de datos creada, ¿Qué prosigue?*** El siguiente paso lógico es: **Crear una tabla;** es decir, **la** **primera tabla o entidad.** Ya la tabla, cada una por separada, comprende un conjunto de elementos propios que requieren de una mayor atención y complejidad; tales como: ***atributos, tipos de datos, restricciones y relaciones (todo esto en función de una entidad, cada una con su propia identidad, por tabla). Veamos cómo creamos estos objetos, tablas, en nuestra consola (MySQL Workbench).***

*Tenga esta imagen presente:*

 ***Para este caso, por ej., se tendrían que hacer 5 tablas.***

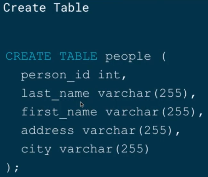
***Creación de tablas (tables) en MySQL* (create)**

Continuando con el proyecto *Platziblog* dentro de la consola, tenemos que el proceso para crear una primera tabla para dicha base de datos es el siguiente:

 *Estando situados en este lugar...*

Debemos, ahora, presionar click derecho sobre la opción que dice *tables.* Ahí se nos mostrará un pliego de más opciones. La opción que debe seleccionar es: 

Antes de empezar a desarrollar propiamente las tablas del proyecto en sí, vamos a intentar modelar el siguiente ejemplo/caso de *primera entidad:*

 *código en lenguaje SQL*

De entrada, al presionar click sobre *create table,* se nos mostrará una pestaña nueva que se llama: *new\_table - Table;* inmediatamente abajo, usted podrá asignarle un nombre a su primera tabla; es decir, la tabla de la *primera entidad* que desee trabajar. Según nuestro ejemplo, el nombre que va a recibir sería: ***people.***

Posteriormente debajo, por defecto, al presionar click sobre el siguiente recuadro, **se nos asignará un primer atributo en nuestro primer campo: el correspondiente al *id*** de la entidad a la que le está creando una tabla; en este caso, sería el ***id*** para la entidad ***people.*** Además, se marcarán las restricciones ***PK*** y ***NN.*** Tal que así:



La consola entiende bien que, generalmente, los *id* reciben dos (2) restricciones como clave principal: *PK (Primary Key)* y *NN (Not Null),* esto ya lo explicamos en la teoría. Sin embargo, más allá de las restricciones, la consola también nos ofrece un recurso aparte, adicional, que es ***AI*** *(Auto Increment).* Nosotros podríamos determinar, si quisieramos, que el campo correspondiente al ***id*** sea manejado, de forma automatica, por nuestra propia base de datos; eso se logra activando la casilla del: ***AI. AI***, en efecto,tiene la facultad de decirle a la base de datos que gestione, por sí sola, el campo del ***id.* ¿Cómo podríamos interpretar esta “auto-gestión”?** Pues bien, significa que, cada que se agregue un valor nuevo al campo, columna o atributo ***id***; en este caso, cada que agreguemos una nueva persona como valor en la columna *person\_id* (y dentro de la entidad people); a esa persona se le asignará, automaticamente, una enumeración única y propia en término absoluto; entonces, en la medida que se incremente el número de personas se incrementará,y se le dará también, de forma automatica, la asignación de un numero propio e identificativo a cada una de esas personas registradas o ad portas de ser registradas. Es decir, por ejemplo, la primera persona añadida va a ser referenciada con el valor númerico uno (1), la segunda persona con el número dos (2), la tercera con el tres (3), y así sucesivamente lo hará de forma automatica con las *personas* siguientes para la columna *person\_id*. Su tabla, por ahora, debe verse así:



Ya no habrá motivos para preocuparse de que el ID sea único porque, como va en secuencia, no se repetirá nunca: siempre estará en constante crecimiento, eso es lo que hace ***AI*** *(Auto Increment).*

*Si se da cuenta hemos decidido cambiar un poco el nombre que, por defecto, la consola le asignó al atributo id de la entidad en cuestión.*

Siguiente atributo: **last\_name varchar (255)**

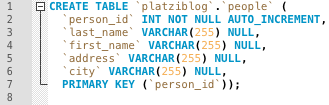
En la siguiente fila, presionando click sobre *<click to edit>,* podrá asignar un nuevo atributo (campo, columna) a la entidad (o tabla) *people*. Dejese guiar y asigne en *Column:* **last\_name** yen *Datatype:* **varchar (255).** Por defecto, *varchar* se asigna con un límite máximo de 45 caracteres posibles; sin embargo, esto lo puede modificar.

**Ahora prosiga usted con los demás.**

3er atributo: **first\_name varchar (255),** 4to atributo: **address varchar (255)** & 5to atributo: **city varchar (255).**

**Luego de registrar exitosamente las propiedades de su entidad *people,* presione en *Apply*** *(al final, debajo).*

***Lo que viene, como por arte de magia, es que la consola nos muestre lo que piensa ejecutar, ya en código SQL, antes de que se ejecute.*** Debe verse algo así:

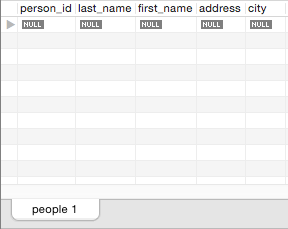


*Si se fija todas las demás propiedades, a excepción del id, tienen como restricción NULL; esto es así, por defecto, cuando no se le pasa ninguna restricción manualmente al atributo.*

Si estamos de acuerdo con todo, presionamos *Apply* nuevamente. Luego, efectivamente, se verá reflejada nuestra *table.* Sólo presione de nuevo sobre *tables* para verla: 

Para hacer una consulta sobre la tabla *people* debemos presionar sobre su celda y ver la opción , nuestro *1er select*, esto nos mostrará los primeros 1000 registros de la tabla recién creada: ***people*** *(la tabla de nuestra primera entidad).*

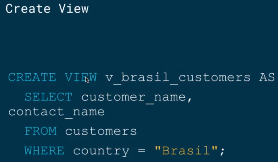
Aunque de momento realmente no tengamos datos ahí, la tabla se vería así:



***Creación de vistas (views) en MySQL* (create)**

Las *Views* o vistas tienen que ver un poco con un concepto que abordaremos más adelante: los *Selects* o proyecciones.

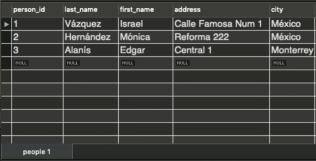
Las *vistas* lo que hacen es tomar datos de una base de datos, organizan dicha información de tal manera que se vea de forma *presentable,* y la convierten en algo que se pueda consultar de manera recurrente. Una declaración de la sentencia *View,* en código,más precisamente para crear una vista, se puede ver más o menos así:



La creación de una vista, con *View,* se compone principalmente de dos (2) partes. **1.** La sentencia *Create View*, que es la que crea justamente la *vista* y **2.** El nombre que se le asigna a la *vista* que se está, o se pretende, crear; que, por lo general, ese nombre recibe una “v\_” de entrada al inicio del nombre (por convención se coloca para dar a entender a cualquier usuario de que el objeto tratado es, justamente, una *vista*). Por el nombre que recibe nuestra *view* de ejemplo se puede entender que la *vista* ha sacado datos, de una base de datos, relacionados a los datos de unos clientes de brasil,... para su posterior consulta *(v\_brasil\_customers)*. Ahora, si ha prestado atención, lo que se sugiere es que trate siempre de que el nombre que reciban sus objetos sean muy descriptivos a la función del objeto en sí mismo. Posteriormente, se ve que se declara la instrucción *AS,* lo cual significa *“como”;* es decir, con esa instrucción se pretende establecer ***cómo usted va*** *a consultar los datos de los clentes de brasil,* qué filtros o qué parametros va a tener en consideración. Déspues hay un sin número de instrucciones que se declaran en conjunto con la sentencia *Select*, mas más adelante profundizaremos acerca de eso. **Vamos a la consola primero para *crear una vista.***

Supongamos que ya contabamos con información dentro de la *table* **people;** para hacer el ejercicio más llevadero y, pues, como no podía ser de otra manera, poder reflejar la creación de una *vista* (una *vista* no funciona si no hay datos ya dentro de una base de datos).

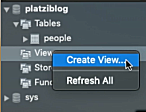
Entonces, vamos a nuestra tabla de interés ***people****,* como lo hicimos antes, ; y, estando ahí, ya estamos suponiendo que hay información dentro de la tabla para la entidad ***people.***Algo más o menos así:

 ***Lo que se querrá hacer ahora es proyectar dicha información en una vista; es decir, la información de la tabla de la entidad “people” (sólo esa).* Del resto, aquí tenemos una sola entidad, *people,* con 5 atributos (campos, columnas) definidas donde cada una puede recibir valores, *datos,* de manera indefinida.**

**1ro.** Se debe seleccionar toda la sentencia ***Select*** que está sobre el recuadro de arriba, por fuera de la tabla *people,* lo que parece una terminal de comandos; y, posteriormente, hacer click derecho sobre ella (sobre la sentencia) y presionar sobre la opción ***Copy****.* La sentencia, para nuestro caso puntual, se debe ver algo más o menos así:



Luego, en el panel de la izquierda, nos vamos a la sección que dice *views* o *vistas*; ahíhacemos click derecho, sobre *views,* y presionamos la opción que dice ***Create view,*** tal que así:



Al seleccionar la opción ***Create view,*** por default, ya nos pone la sentencia *Create View* que se acompaña con un “nombre” por default también y, por último, la instrucción *AS,* tal como lo explicamos en un principio. Veamos:

 (El nombre por defecto *“new\_view”* debe ser sustituido; recuerde, debe tratar siempre de que el nombre que reciban sus objetos, y eso incluye a los *views,* sean muy descriptivos a la función del objeto en sí mismo). El nombre que le pasaremos será: *‘platzi\_people’.*

En la línea de código inmediatamente después de la primera, tendríamos que pegar justamente lo que copiamos con anterioridad; es decir, esto: , con esto estaríamos diciendo que queremos una *vista;* sí, pero de los datos, de la base de datos *platziblog,* relacionados a la entidad *people* (nada más). Al final, quedará algo así:



Damos click en *Apply.* ***Lo que viene, de nuevo, como por arte de magia, es que la consola nos muestre lo que piensa ejecutar, ya en código SQL, antes de que se ejecute.*** Debe verse algo así:



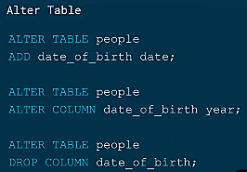
Si estamos de acuerdo con todo, presionamos *Apply* nuevamente. Siempre asegurese de que al final de cada sentencia sólo se vea un punto y coma (;). Luego, efectivamente, se verá reflejada nuestra *view*. Sólo presione de nuevo sobre *view* para verla: 

Para hacer una consulta sobre la vista *platzi\_people* debemos presionar sobre su celda y ver la opción , nuestro *1er select*, esto nos mostrará los primeros 1000 registros (como ya sabe) de la vista recién creada: ***platzi\_people*** *(la vista de nuestra primera entidad).*

***Con una vista podemos, por ejemplo, unir varias tablas; o donde seleccionemos sólo, por ejemplo, la gente que viene de Mexico (según nuestro caso de estudio actual).* Entonces, es eso lo que hace la vista: dejarnos ver exactamente un conjunto de datos que necesitamos, los que deseemos filtrar.** Nos da la información ya digerida para ser presentada al usuario que la necesite. Lo más básico, ante todo, en cuanto a las *vistas,* es lo que acabamos de hacer: coger los datos de una única tabla y ponerlas en una *vista.*

***Alteración de tablas (tables) en MySQL* (alter)**

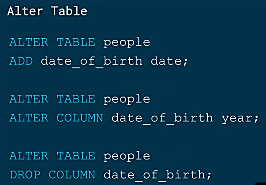
Alter, como ya se imagina, es la sentencia que nos va a permitir modificar, ¿qué cosas? Una tabla y, en concreto, sus columnas. Ojo! Es una sentencia para modificar la tabla, mas no para eliminarla *(drop).* Sin embargo, la eliminación de columnas, de una tabla, sí hace parte de las funciones de la sentencia *alter.* Veamos su estructura.



***Alter table***nos va a permitir modificar nuestra *tabla,* ¿cuál? Pues, la tabla de la entidad que le pasemos; en este caso, tabla *people* *.*

Ahora, en la tabla lo que usted ve como **ADD**,...significa ***agregar una nueva columna a una tabla ya existente***; luego, posteriormente, se pasan dos parametros a dicha sentencia; en su respectivo orden son: **1.** **el nombre que recibirá la nueva columna que piensa *agregar*; es decir, el nuevo *campo o atributo***y, por último, **2. el tipo de dato del nuevo campo, columna o atributo que se está creando en dicha tabla ya existente.** En nuestro caso tenemos que el nombre del nuevo campo será: **“date\_of\_birth”** y que el tipo de dato con el que se va a crear dicho campo será **“date”**.

Ahora, lo anterior que usted vio fue lo concerniete a *Agregar una nueva columna a una tabla ya existente;* mas, ésta no es la única función que tiene la sentencia *Alterar,* esa es la primera. Con *Alterar usted también puede*, o puede ejecutar acciones como: ***Modificar una columna a una tabla ya existente*** **(ALTER COLUMN) y/o *Eliminar una columna a una tabla ya existente* (DROP COLUMN)**

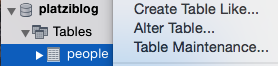
Siempre que queramos ejecutar alguna de las tres acciones posibles de la sentencia *alter,* debemos ejecutar primero la sentencia *alter table* (+ el nombre de la tabla a manipular en cuestión). Tal que así: 

En *Alter Column*, el 1er parametro pasado (date\_of\_birth) se refiere al nombre de la columna, o nombre del campo, que desea *alterar* (modificar);y, el 2do parametro (year), se refiere al tipo de dato que reemplazaría al dato que ya existía con anterioridad (date); es decir, con **ALTERN COLUMN** podemos modificar, por ejemplo, el tipo de dato de una columna ya existente; en este caso, hemos alterado o modificado el tipo de dato del campo o columna (atributo) “date\_of\_birth”, paso de tener un tipo de dato “date” a un tipo de dato “year” (lo dicho).

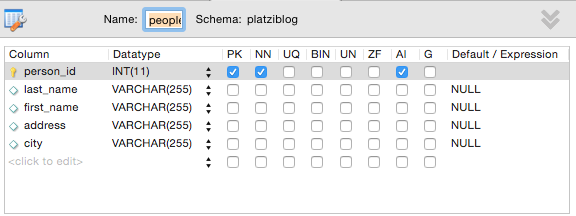
Por último, en *Drop Column*, sólo se pasa un 1er parametro; en nuestro caso ese parametro sería: “date\_of\_birth”; es decir, lo que haría *Drop Column* sería eliminar un campo o columna (atributo) que reciba por nombre *“date\_of\_birth”.* Palabras más palabras menos, con **DROP COLUMN** podemos eliminar, por ejemplo, un campo o columna dentro de una tabla ya existente.

***Operando las acciones de Alter en Consola...***

Retomando nuestra *tabla* ya creada: *people,...* presionamos click derecho sobre ella y nos vamos a la opción que dice: ***Alter table***, tal que así:

, posteriormente, 

*Alter table* nos vuelve a llevar al principio de todo, casi como si fueramos a reconstruir de nuevo la tabla; en efecto, estando sobre la presentación de la tabla en su modo edición, podría hacerle todos los cambios que queramos. Debe verse algo más o menos así:



1. Ejecutando **ADD** en consola: En “click to edit” es que, por medio de la consola propia del MySQL, podemos crear un nuevo campo o columna. Haremos click ahí y le pasaremos el nombre al nuevo campo/columna de: “date\_of\_birth” y luego, justamente al lado, le vamos a definir su correspondiente tipo de datos: “date”. Veamos.



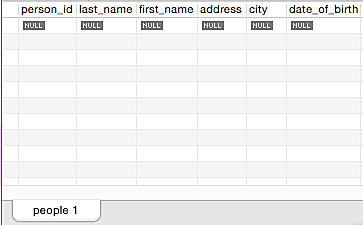
Damos click en *Apply.* ***Lo que viene, de nuevo, como por arte de magia, es que la consola nos muestre lo que piensa ejecutar, ya en código SQL, antes de que se ejecute.*** Debe verse algo así:



Si estamos de acuerdo con todo, presionamos *Apply* nuevamente. Siempre asegurese de que al final de cada línea de código sólo se vea un punto y coma (;). Luego, efectivamente, se verá reflejada nuestra nueva *columna* o *campo.* Sólo presione de nuevo sobre *people*, la tabla donde fue agregada la columna en cuestión,para verla:



Para hacer una consulta sobre la tabla *people* debemos presionar sobre su celda y ver la opción , nuestro *1er select*, esto nos mostrará los primeros 1000 registros (como ya sabe) de la tabla *people* recién manipulada y alterada; pues, ahora conserva una nueva columna o campo dentro de ella que se llama: “date\_of\_birth”. Veamos.



Ahora, si queremos alterar nuevamente nuestra tabla, para 2. Ejecutar **ALTER COLUMN** en consola, debemos de nuevo presionar click sobre *Alter table* (ya estando situados antes sobre la tabla *people,* obvio). Al ya visualizar de nuevo su tabla *people* podrá modificar cualquier campo existente; en este caso, queremos hacer cambios sobre el recién creado “date\_of\_birth”; en concreto, queremos cambiar el tipo de datos de dicho campo (que pase de “datetime” a “year”): sólo es posicionarse sobre él (sobre la celda del tipo de dato) y elegir otra opción. Acá se verá el cambio ya efectuado listo para *aplicar.*



Damos click en *Apply.* ***Lo que viene, de nuevo, como por arte de magia, es que la consola nos muestre lo que piensa ejecutar, ya en código SQL, antes de que se ejecute.*** Debe verse algo así:



Si estamos de acuerdo con todo, presionamos *Apply* nuevamente. Siempre asegurese de que al final de cada línea de código sólo se vea un punto y coma (;).

Ahora, vamos a 3. Ejecutar **DROP COLUMN** en consola; ahí mismo, sin salerse del panel para modificar su *tabla,* debemos seleccionar qué *campo* deseamos eliminar: sólo se debe hacer click derecho sobre el campo en cuestión y se mostrará un pliego de opciones, debe seleccionar el que dice: 

Damos click en *Apply.* ***Lo que viene, de nuevo, como por arte de magia, es que la consola nos muestre lo que piensa ejecutar, ya en código SQL, antes de que se ejecute.*** Debe verse algo así:



Si estamos de acuerdo con todo, presionamos *Apply* nuevamente. Siempre asegurese de que al final de cada línea de código sólo se vea un punto y coma (;). Luego, efectivamente, si revisa su tabla *people,* verá que el campo o la columna “date\_of\_birth” ya no existe. Su tabla, de nuevo, se verá como en un principio. Así:



***Eliminación de tablas (tables) en MySQL* (drop)**

Ahora sí vamos a hablar propiamente sobre cómo se eliminan las tablas -y hasta las bases de datos (y no precisamente las columnas de una tabla, eso hace la sentencia *alter*). Veamos su estructura en lenguaje SQL.



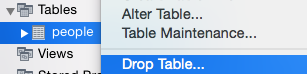
***Drop table***nos va a permitir eliminar nuestra *tabla,* ¿cuál? Pues, la tabla de la entidad que le pasemos; en este caso, tabla *people* *.* Ahora, también podríamos eliminar, incluso, hasta la *base de datos* (lo dicho), ¿cuál? Pues, el nombre de base de datos y, por extensión, la base de datos que le pasemos; en este caso, sería la base de datos *test\_db* *.*

*Tenga mucho cuidado con la sentencia* ***drop****; pues, se estará encargando de eliminar, sin vuelta atrás, información muy sensible.* Veamos esto en consola.

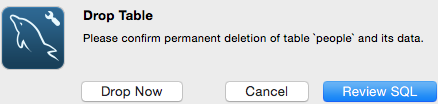
***Operando las acciones de Drop en Consola...***

***Drop table***

Retomando nuestra *tabla* ya creada: *people,...* presionamos click derecho sobre ella y nos vamos a la opción que dice: ***Drop table***, tal que así:



Como justamente el manejador de la base de datos, MySQL, sabe sobre la dimensión de la acción que está considerando tomar, *le emite un mensaje de* ***Advertencia.*** Tal que así:



Le pide una 2da confirmación para saber si en realidad su deseo es eliminar, permanentemente, una *tabla; es decir, una entidad*. Ahora, lo ideal es que, incluso aunque usted esté seguro de la acción que pretende tomar, revise el código SQL que llevará a cabo la eliminación de su tabla. Veamos.

Damos click en *Review SQL.* ***Lo que viene, de nuevo, como por arte de magia, es que la consola nos muestre lo que piensa ejecutar, ya en código SQL, antes de que se ejecute.*** Debe verse algo así:



Si definitivamente estamos de acuerdo con *eliminar nuestra tabla*, presionamos click en *Execute*.

De esta manera, dejo de existir definitivamente nuestra tabla *people* de la base de datos *platziblog;* de hecho, si intenta ejecutar o correr algún código que se relacione con ella, la consola le dejará claro, por medio de un mensaje de error, que la tabla en cuestión ya no existe. Tal que así:



***tip:*** para poner a correr un código, cualquiera que sea, dentro del editor de código de la consola, en muchas ocasiones le tocará presionar el siguiente botón: 

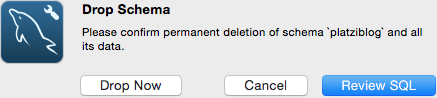
***Drop database (schema)***

Ahora, para poder eliminar, ya no una tabla, sino toda una *base de datos*,... el proceso realmente es el mismo; sólo que ahora se va a posicionar sobre la base de datos *(o schema)* en cuestión.

Retomando nuestra *base de datos* ya creada: *platziblog,...* presionamos click derecho sobre ella y nos vamos a la opción que dice: ***Drop schema***, tal que así:



De nuevo, como justamente el manejador de la base de datos, MySQL, sabe sobre la dimensión de la acción que está considerando tomar, *le emite un mensaje de* ***Advertencia.*** Tal que así:



Damos click en *Review SQL.* ***Lo que viene, de nuevo, como por arte de magia, es que la consola nos muestre lo que piensa ejecutar, ya en código SQL, antes de que se ejecute.*** Debe verse algo así:



Si definitivamente estamos de acuerdo con *eliminar nuestra base de datos*, presionamos click en *Execute*.

De esta manera, dejo de existir definitivamente, ahora, nuestra base de datos *platziblog* :(

**Esto es lo concerniente a DDL (Data Definition Language) y sus sentencias/comandos (muy usados al inicio de un proyecto: lo que tiene que ver con la conformación de la estructura de la base de datos); ahora, en su próximo documento Word, va a estudiar lo que es el DML.**