SQL

DDL y DML

Las sentencias SQL se dividen en dos categorías; **Lenguaje de definición de datos**; data definition language **(DDL)** y **Lenguaje de manipulación de datos** ;data manipulation language **(DML).**

Lenguaje de definición de datos (DDL)

Las sentencias DDL se utilizan para crear y modificar la estructura de las tablas así como otros objetos de la base de datos.

- CREATE - para crear objetos en la base de datos.
- ALTER - modifica la estructura de la base de datos.
- DROP - borra objetos de la base de datos.
- TRUNCATE - elimina todos los registros de la tabla, incluyendo todos los espacios asignados a los registros.

Lenguaje de manipulación de datos (DML)

Las sentencias de lenguaje de manipulación de datos (DML) son utilizadas para gestionar datos dentro de los schemas. Algunos ejemplos:

- SELECT - para obtener datos de una base de datos.
- INSERT - para insertar datos a una tabla.
- UPDATE - para modificar datos existentes dentro de una tabla.
- DELETE - elimina todos los registros de la tabla; no borra los espacios asignados a los registros.

Tablas (instrucciones CREATE, ALTER, DROP) con mysql workbench

SENTENCIA CREATE TABLE

La sentencia CREATE TABLE es utilizada para crear una tabla en una base de datos.

Las tablas se organizan en filas y columnas; y cada tabla debe tener un nombre.

SINTAXIS PARA LA SENTENCIA CREATE TABLE

```
CREATE TABLE table_name
(
column_name1 data_type(size),
column_name2 data_type(size),
column_name3 data_type(size),
....
);
```

Los parámetros "column_name" especifican los nombres de las columnas de las tablas.

Los parámetros "data_type" especifican que tipo de datos puede haber en esa columna (ej. varchar, integer, decimal, date, etc.).

El parámetro "size" especifica la longitud máxima de la columna de la tabla.

SENTENCIA ALTER TABLE

La sentencia ALTER TABLE se utiliza para añadir, borrar o modificar columnas de una tabla existente.

SINTAXIS PARA LA SENTENCIA ALTER TABLE

Para borrar una columna de una tabla, hay que utilizar la siguiente sintaxis (algunos sistemas de bases de datos no permiten borrar una columna):

```
ALTER TABLE table_name
DROP COLUMN column_name
```

Para añadir una columna a una tabla, hay que utilizar la siguiente sintaxis;

```
ALTER TABLE table_name
ADD column_name datatype
```

Sentencia SELECT

La sentencia SELECT se utiliza para seleccionar datos de una base de datos.

Se guarda el resultado en una tabla llamada "result-set".

Sintaxis de la Sentencia SELECT 1

```
SELECT column_name,column_name
FROM table_name;
```

y

Sintaxis de la Sentencia SELECT 2

```
SELECT * FROM table_name;
```

EL asterisco * significa que queremos todas las columnas de la tabla.

Sentencia SQL WHERE

La sentencia WHERE se usa para extraer sólo los registros que cumplan con una condición. Funciona como un filtro.

Sintaxis de la sentencia SQL WHERE

```
SELECT column_name,column_name
FROM table_name
WHERE column_name operator value;
```

Claúsula ORDER BY

La claúsula ORDER BY se utiliza para ordenar los resultados a través de una o más columnas.

La claúsula ORDER BY ordena los registros de manera ascendente por defecto. Para hacerlo de manera descendente, se puede utilizar la claúsula DESC.

Sintaxis de la claúsula SQL ORDER BY

```
SELECT column_name,column_name
FROM table_name
ORDER BY column_name,column_name ASC|DESC;
```

Sentencia SQL INSERT INTO

La sentencia INSERT INTO se utiliza para insertar nuevos registros a una tabla

Sintaxis SQL INSERT INTO

Se puede escribir la sentencia INSERT INTO de dos maneras.

La primera forma no especifica los nombres de las columnas en las que se inserta los datos, sólo se especifican los valores:

```
INSERT INTO table_name
VALUES (value1,value2,value3,...);
```

La segunda forma especifica tanto los nombres de las columnas como los valores a insertar;

```
INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

Con SQL, podemos copiar información de una tabla a otra.

Sentencia SQL INSERT INTO SELECT

La sentencia INSERT INTO SELECT selecciona datos de una tabla y los inserta en otra tabla ya existente. Los registros ya existentes de la tabla destino, no se ven afectadas.

Sintaxis SQL INSERT INTO SELECT

Podemos copiar todas los registros de una tabla a otra ya existente:

```
INSERT INTO table2
SELECT * FROM table1;
```

O podemos copiar de una tabla a otra, solamente las columnas que queremos:

```
INSERT INTO table2
(column_name(s))
SELECT column_name(s)
FROM table1;
```

Nota: Tanto en el caso de la sentencia INSERT INTO como de la sentencia INSERT INTO SELECT, no es necesario escribir las columnas en la consulta SQL, cuando los valores coincidan en tipo y tamaño con el perfil de la columna donde van a ser insertados;

Sentencia SQL UPDATE

La sentencia UPDATE se utiliza para actualizar registros ya existentes de una tabla.

Nos permite elegir los campos a actualizar y los datos con que actualizarlos.

Sintaxis SQL UPDATE

La sintaxis básica de la claúsula UPDATE es la siguiente:

```
UPDATE table_name
SET column_name = value
WHERE condition
```

Sentencia SQL DELETE

La sentencia DELETE se utiliza para borrar registros de una tabla.

Se especifica de que tabla se quieren borrar los registros y si se necesita, se puede añadir una claúsula WHERE para especificar qué registros borrar.

Hay que tener en cuenta que si se omite la claúsula WHERE, se borrarán todos los registros!

Sintaxis SQL DELETE

```
DELETE FROM table_name
WHERE some_column=some_value;
```

Sentencia SQL TRUNCATE

Elimina todas las filas de una tabla. Es una instrucción DDL; internamente hace un DROP de la tabla y después hace un CREATE de la misma tabla.

TRUNCATE TABLE es similar a la sentencia DELETE pero sin la claúsula WHERE. Tiene la desventaja de que no se puede borrar selectivamente toda la tabla y la ventaja de que TRUNCATE TABLE es más rápido y utiliza menos recursos.

Sintaxis SQL TRUNCATE

```
TRUNCATE table_name;
```

Operadores lógicos SQL AND ,OR con MySql Workbench

Los operadores lógicos AND y OR, permiten que usando la sentencia WHERE, se haga filtro con más de una condición.

OPERADOR LÓGICO SQL AND

El operador AND muestra un registro cuando la primera condición y la segunda se cumplen.

Cuando usamos AND, es conveniente poner la condición con menor probabilidad de que se cumpla, en primer lugar. El sistema de la base de datos evalúa las condiciones de izquierda a derecha. Si tenemos dos o más operadores AND en una condición, el que está a la izquierda, es el primero en ser evaluado, y sólo si es verdadera, se evalua la siguiente condición. Si, esa condición también es verdadera, se evaluará la tercera condición. Si ponemos la condición menos probable en primer lugar, se ahorra trabajo al sistema de la base de datos, aumentando así la velocidad.

OPERADOR LÓGICO SQL OR

El operador OR, muestra los registros cuando se cumple la primera condición Ó la segunda.

SQL SELECT DISTINCT

La claúsula DISTINCT nos devuelve valores únicos. En una tabla, una columna puede contener valores duplicados; y algunas veces sólo se necesita un listado de los valores diferentes.

SINTAXIS SELECT DISTINCT SQL

La claúsula DISTINCT se añade a las sentencias SELECT , justo después de la palabra clave SELECT.

```
SELECT DISTINCT column_name,column_name
FROM table_name;
```

OPERADOR SQL LIKE

El operador LIKE permite utilizar caracteres comodín en la búsqueda de un patrón dentro de una columna. Un caracter comodín es aquel que no coincide con un caracter específico si no con cualquier caracter o caracteres.

El operador LIKE selecciona valores alfanuméricos con un determinado patrón.

SINTAXIS DEL OPERADOR SQL LIKE

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

El símbolo "%" se utiliza para definir los comodines en el patrón, tanto delante como detrás de otros caracteres;

En algunas bases de datos, el operador LIKE es sensible a las mayúsculas y las minúsculas; en otras no. Oracle, por ejemplo si que lo es y SQL no;

OPERADOR SQL BETWEEN

BETWEEN es un operador muy útil a utilizar dentro de la claúsula WHERE, para especificar un rango de valores inclusivos. Se utiliza normalmente con fechas pero también se puede usar con strings y con números.

SINTAXIS DEL OPERADOR SQL BETWEEN

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Utilizando el operador BETWEEN, podemos ver que nos devuelve los mismos resultados que los operadores "mayor o igual" (>=) y "menor o igual" (<=). Es importante recordar que el operador BETWEEN es inclusivo.

Se puede utilizar BETWEEN con más tipos de datos además de fechas, como por ejemplo texto;

Uso del operador NOT con el operador BETWEEN:

Podemos utilizar el operador BETWEEN en conjunción con el operador NOT. En este caso SQL selecciona el valor que no esté en el rango especificado.