NOMBRES: **GRADO**: **FECHA**: dd/mm/aaaa

INTRODUCCIÓN

En esta guía aprenderás a crear una base de datos, a crear tablas e insertar datos en ellas.

CREACIÓN DE UNA BASES DE DATOS

El lenguaje para crear y manipular las bases de datos se llama SQL (por sus siglas en inglés **Structured Query Language**; en español **lenguaje de consulta estructurada**) (SQL, 2020).

El SQL está dividido en el **DDL**,El lenguaje de definición de datos (en inglés Data Definition Language, o DDL) (SQL, 2020); y éste se encarga de la modificación de la estructura de los objetos de la base de datos. Las cuatro operaciones que utilizan con el DDL son: CREATE, ALTER, DROP Y TRUNCATE.

La otra división del SQL es el **DML**, lenguaje de manipulación de datos (Data Manipulation Language, o DML en inglés) (SQL, 2020). Este lenguaje permite la gestión de los datos que están en la base de datos. Las operaciones más comúnmente usadas con DML son: SELECT, INSERT, DELETE.

Para crear una base de datos y las tablas en la misma usamos la función CREATE del DDL. La estructura del comando para crear una base de datos es (maridadb, 2020):

```
CREATE [OR REPLACE] {DATABASE | SCHEMA} [IF NOT EXISTS] db_name

    [create_specification] ...
```

Por ejemplo, si quiero crear una base de datos para un colegio escribiría lo siguiente (maridadb, 2020):

```
CREATE DATABASE colegio;
```

Todos los comandos que usemos deben terminan con punto y coma (;), este indica que el comando termina allí y que puede empezar otro comando.

Puede ocurrir que intentes crear una base de datos y esta ya exista, por lo que te marcará un error al intentar ejecutar el comando de crear una tabla. Para evitar esto, podríamos escribir el siguiente comando (maridadb, 2020):

```
CREATE DATABASE IF NOT EXISTS colegio;
```

El cuál intenta crear la crear la base de datos si esta no existe. De esta manera, si ya existe no tendremos error.

Por otro lado si ejecutamos el comando (maridadb, 2020):

```
DROP DATABASE IF EXISTS colegio;

CREATE DATABASE colegio;
```

NOMBRES: **GRADO**: **FECHA**: dd/mm/aaaa

El comando **DROP** se usa para borrar, por lo tanto **DROP DATABASE IF EXISTS** borrará la base de datos, y todo el contenido, si esta existe y luego el comando **CREATE DATABASE** la creará.

Información:
En su ambiente de base de datos, ya cuentan con una ya creada. El usuario que se les asignó, no tiene permiso para crear una base de datos.
La base de datos con que cuentan para trabajar es **bd{cod_estudiante}**

Cuando existen muchas bases de datos creadas en el motor de base de datos usamos el comando USE para seleccionar la base de datos con la que queremos trabajar. Por ejemplo:

```
USE colegio;

USE bd807;
```

Con los anteriores comandos estamos seleccionando primero la base de datos `colegio` y luego la base de datos que se llama `bd807`. Los comando que lleguemos a usar después de esto, afectaran a la base de datos con el nombre `bd807`, por que fue la última en seleccionarse.

CREACIÓN DE UNA TABLA

Una vez tengamos la base de datos creada y seleccionada, podremos crear la primera tabla usando el comando create table la sintaxis del comando es la siguiente:

```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name

    (create_definition,...) [table_options    ]... [partition_options]
```

El comando **IF NOT EXISTS** nos ayuda a crear la tabla si esta no existe. Si ya existe, el comando no se ejecuta.

Veamos con un ejemplo la creación de la tabla de *estudiante*. Un estudiantes del colegio, tiene, por lo menos, codigo, dni, primer nombre, segundo nombre, primer apellido, segundo apellido, telefono fijo, telefono celular, sexo.

```
CREATE TABLE IF NOT EXISTS estudiante (
codigo          INT(4) PRIMARY KEY,
dni             INT(11)  NOT NULL,
nombre1         VARCHAR(30) NOT NULL,
nombre2         VARCHAR(30),
apellido1       VARCHAR(30) NOT NULL,
apellido2       VARCHAR(30),
telfijo         INT(7),
telcelular      INTEGER UNSIGNED,
sexo            CHAR(1) NOT NULL
);
```

En este ejemplo, con el comando:

```
CREATE TABLE IF NOT EXISTS estudiante
```

estamos creando una tabla estudiantes, si esta ya no existe.

NOMBRES: **GRADO**: **FECHA**: dd/mm/aaaa

Con el comando:

```
codigo          INT(4) PRIMARY KEY,
```

estamos creando un campo de la tabla que va a almacenar los códigos de los estudiantes. Que este campo es entero de 4 dígitos y que el campo es de tipo llave primaria. Con llave primaria queremos decir que este campo es único en toda la tabla y es el campo por el cual identificaremos a un estudiante de otro.

Con el comando:

```
dni             INT(11)  NOT NULL
```

estamos creando un campo donde almacenar el número de identificación nacional. Que este campo es numérico de 11 dígitos y que no puede ser nulo, esto quiere decir, que no se puede dejar vacío.

Con los comandos:

```
nombre1         VARCHAR(30) NOT NULL,
apellido1       VARCHAR(30) NOT NULL,
```

Estamos creando campos donde se pueden almacenar el primer nombre y primer apellido. Estos campos son de tipo texto varchar y de 30 caracteres máximo. Que el campo es obligatorio y que por lo tanto no se puede dejar vació o nulos.

Con el comando:

```
nombre2         VARCHAR(30),
apellido2       VARCHAR(30),
```

Estamos los campos donde se pueden almacenar el segundo nombre y segundo apellido. Estos campos son texto de tipo varchar con 30 caracteres máximos, y que estos campos pueden ser vacios o nulos.

Con los comando:

```
telfijo         INT(7),
telcelular      INTEGER UNSIGNED,
sexo            CHAR(1) NOT NULL
```

Estamos creando campos para almacenar el telefono fijo, el telefono celular y sexo del estudiante. Estos campos son numéricos de 7 digítos, numérico entero sin signo y texto de tipo carácter de 1 dígito respectivamente. Que los telefonos pueden se pueden dejar sin llenar, pero el sexo es obligatorio llenarlo. El telefono celular se crea como entero sin signo, porque el máximo de un entero con signo es 2.147.483.647 y el número de un celular actual supera este valor.

NOMBRES: **GRADO**: **FECHA**: dd/mm/aaaa

INSERTAR DATOS EN LA TABLA

Una vez creada la tabla estudiantes, nos pondremos a llenarla con datos. Para esto usaremos el comando DML **INSERT**. Este comando tiene la siguiente sintaxis:

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

Por ejemplo, insertaremos 3 datos de estudiantes en la tabla *estudiante* que acabamos de crear.

```
INSERT INTO estudiante (codigo, dni, nombre1, nombre2, apellido1,
apellido2, telfijo, telcelular, sexo)
VALUES (123, 10123, "Carlos", "", "Gonzalez", "", 6485634, 3178568671,
"F");

INSERT INTO estudiante (codigo, dni, nombre1, nombre2, apellido1,
apellido2, sexo)
VALUES (321, 91478214, "Daniel", "Alejandro", "Rojas", "Duarte", "F");

INSERT INTO estudiante (codigo, dni, nombre1, apellido1, sexo)
VALUES (221, 63121421, "Gabriela", "Garcia", "F");
```

En la primera inserción,

```
INSERT INTO estudiante (codigo, dni, nombre1, nombre2, apellido1,
apellido2, telfijo, telcelular, sexo)
VALUES (123, 10123, "Carlos", "", "Gonzalez", "", 6485634, 3178568671,
"F");
```

Se inserta todos los datos. Los datos nombre2 y apellido2 -que son texto-, se insertan vacios (`""`).

En el segundo **`insert`**, se llenan todos los campos exceptos los telefonos que no son obligatorios.

En el tercer **`insert`**, solo se llenan los campos que se marcaron como obligatorios cuando se creo la base de datos.

Si quisiera insertar varios campos a la vez puedo usar el insert de la siguiente forma:

```
INSERT INTO estudiante (codigo, dni, nombre1, nombre2, apellido1,
apellido2, telfijo, telcelular, sexo)
VALUES (123, 10123, "Carlos", "", "Gonzalez", "", 6485634,
3178568671, "F"),
(321, 91478214, "Daniel", "Alejandro", "Rojas", "Duarte", null, null,
"F"),
(221, 63121421, "Gabriela", null, "Garcia", null, null, null, "F");
```

Los campos que no voy a llenar, por que no tengo información, los dejo vacios o **`null`**. Por que al crear la tabla los definimos para que pudieran estar vacios o nulos.

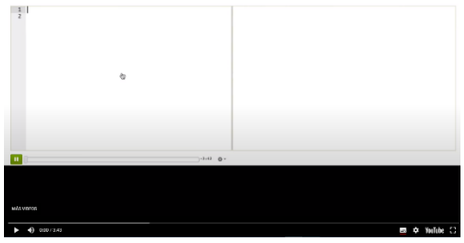
NOMBRES: **GRADO**: **FECHA**: dd/mm/aaaa

RECURSOS

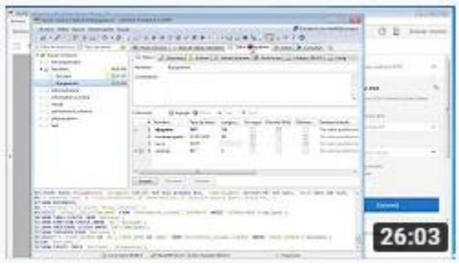
Para aprender más, puedes consultar los siguientes recursos:

1. Crear una tabla e insertar datos.

Link. Subtitulos en español



2. Crear BD e Insertar Datos en MySQL con HeidiSQL



Link

3. Cómo crear una base de datos MySQL. Link.

4. SQL CREATE DATABASE Statement. Link.

5. SQL INSERT INTO SELECT Statement. Link.

REFERENCIAS

GCF Aprende Libre. (25 de 03 de 2020). Obtenido de Excel 2016: https://edu.gcfglobal.org/es/excel-2016/

maridadb. (29 de 06 de 2020). Obtenido de create database: https://mariadb.com/kb/en/create-database/

SQL. (29 de 06 de 2020). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/SQL#:~:text=SQL%20(por%20sus%20siglas%20en, de%20bases%20de%20datos%20relacionales.

Wikipedia - Hoja de cálculo. (25 de 3 de 2020). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Hoja_de_c%C3%A1lculo