## **BASE DE DATOS**

## **REFLEXIÓN**

## Había una vez un pequeño pueblo llamado Databaseville. Los habitantes de Databaseville tenían una gran pasión por organizar y almacenar información de manera eficiente. Decidieron que era hora de crear una base de datos para gestionar todos los datos importantes del pueblo. En primer lugar, crearon una tabla llamada "Personas" para almacenar información sobre los habitantes del pueblo. Esta tabla tenía varios campos, como "ID" (un número único que identificaba a cada persona), "Nombre" (una cadena de texto que contenía el nombre de la persona), "Edad" (un número entero que indicaba la edad de la persona) y "Altura" (un número decimal que representaba la altura de la persona en metros). A medida que la base de datos se iba llenando con información, los habitantes de Databaseville comenzaron a hacer diferentes tipos de consultas para obtener información específica. Por ejemplo, utilizaron la consulta SELECT para obtener una lista de todas las personas en la tabla "Personas". También utilizaron la cláusula WHERE para filtrar los resultados y obtener solo las personas que tenían una edad mayor a 30 años. Además de la tabla "Personas", decidieron crear otra tabla llamada "Eventos" para registrar todos los eventos importantes que ocurrían en el pueblo. Esta tabla tenía campos como "ID" (un número único para identificar cada evento), "Nombre" (una cadena de texto con el nombre del evento), "Fecha" (una fecha para indicar cuándo ocurrió el evento) y "Descripción" (una cadena de texto que describía el evento en detalle). A medida que los eventos se iban registrando en la base de datos, los habitantes de Databaseville comenzaron a hacer consultas más complejas. Por ejemplo, utilizaron la cláusula ORDER BY para ordenar los eventos por fecha, de manera que pudieran ver los eventos más recientes primero. También utilizaron la cláusula LIKE para buscar eventos que tuvieran una palabra clave específica en su descripción. A medida que Databaseville crecía y se desarrollaba, los habitantes se dieron cuenta de la importancia de tener diferentes tipos de datos en su base de datos. Utilizaron el tipo de dato INT para almacenar los IDs y las edades de las personas. Utilizaron el tipo de dato FLOAT para almacenar las alturas de las personas, ya que era necesario tener mayor precisión decimal. Utilizaron el tipo de dato VARCHAR para almacenar los nombres y las descripciones de los eventos, ya que eran cadenas de texto variables en longitud. Gracias a su base de datos bien organizada, los habitantes de Databaseville pudieron acceder y gestionar la información de manera eficiente. Pudieron obtener listas de personas según diferentes criterios, como la edad o la altura. Pudieron buscar eventos específicos utilizando palabras clave en la descripción. Y lo más importante, pudieron tomar decisiones informadas basadas en los datos almacenados en su base de datos. Y así, Databaseville se convirtió en un ejemplo de cómo una base de datos bien diseñada y utilizada puede facilitar la gestión de información y ayudar en la toma de decisiones.

## ¿Qué es una tabla en base de datos?

Una base de datos es un sistema organizado de almacenamiento de información que permite gestionar y acceder a datos de manera eficiente. Consiste en una colección de tablas relacionadas entre sí, donde se almacena la información en forma de registros.  
  
Para entender mejor qué es una base de datos, veamos un ejemplo:  
  
Imagina que tienes una tienda en línea y necesitas almacenar información sobre tus productos y clientes. Podrías crear una base de datos con dos tablas: "Productos" y "Clientes".  
  
La tabla "Productos" podría tener los siguientes campos: ID, Nombre, Descripción, Precio, Cantidad. Cada registro en esta tabla representaría un producto en tu tienda.  
  
La tabla "Clientes" podría tener los siguientes campos: ID, Nombre, Dirección, Email, Teléfono. Cada registro en esta tabla representaría un cliente de tu tienda.

**Una Tabla en base de datos es** un tipo de modelado de datos donde se guarda una información recogida por un sistema. Es decir, son objetos o estructuras que contienen los datos organizados en **filas y columnas**. Cada fila representa un registro único, y cada columna un campo dentro del registro.

## Estructura de una tabla en base de datos

Ahora que sabemos **que es una tabla en base de datos**, toca preguntarnos

¿Cómo se componen?

Toda tabla tiene dos componentes básicos, el **nombre de campo** y de registro:

* Campos: El campo es el **nombre de la columna**. Es un dato único y se le otorga un único tipo de datos.
* Registros: Los registros son cada una de las tablas. Aquí se almacenan o cargan los datos que guardamos. Algunos de estos datos pueden ser nulos

# Tipos de Datos de Mysql

Después de la fase de diseño de una base de datos, y una vez se ha realizado el paso a tablas del mismo, en necesario crear las tablas correspondientes dentro de la base de datos. Para cada campo de cada una de las tablas, es necesario determinar el tipo de datos que contiene, para de esa forma ajustar el diseño de la base de datos, y conseguir un almacenamiento óptimo con la menor utilización de espacio.

Los tipos de datos que puede haber en un campo, se pueden agrupar en tres grandes grupos:

1. Tipos Numericos

Existen tipos de datos numéricos, que se pueden dividir en dos grandes grupos, los que están en coma flotante (con decimales) y los que no.

ejemplo:

### Integer, Int:

Número entero con o sin signo. Con signo el rango de valores va desde -2147483648 a 2147483647. Sin signo el rango va desde 0 a 429.4967.295

### Float:

Número pequeño en coma flotante de precisión simple. Los valores válidos van desde -3.402823466E+38 a -1.175494351E-38, 0 y desde 1.175494351E-38 a 3.402823466E+38.

1. Tipo de Fechas

A la hora de almacenar fechas, hay que tener en cuenta que Mysql no comprueba de una manera estricta si una fecha es válida o no. Simplemente comprueba que el mes esta comprendido entre 0 y 12 y que el día esta comprendido entre 0 y 31.

### Date:

Tipo fecha, almacena una fecha. El rango de valores va desde el 1 de enero del 1001 al 31 de diciembre de 9999. El formato de almacenamiento es de año-mes-dia

### DateTime:

Combinación de fecha y hora. El rango de valores va desde el 1 de enero del 1001 a las 0 horas, 0 minutos y 0 segundos al 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos. El formato de almacenamiento es de año-mes-dia horas:minutos:segundos

1. [Tipo de Cadenas](https://desarrolloweb.com/articulos/1054.php" \l "cadena)

### Char(n):

Almacena una cadena de longitud fija. La cadena podrá contener desde 0 a 255 caracteres.

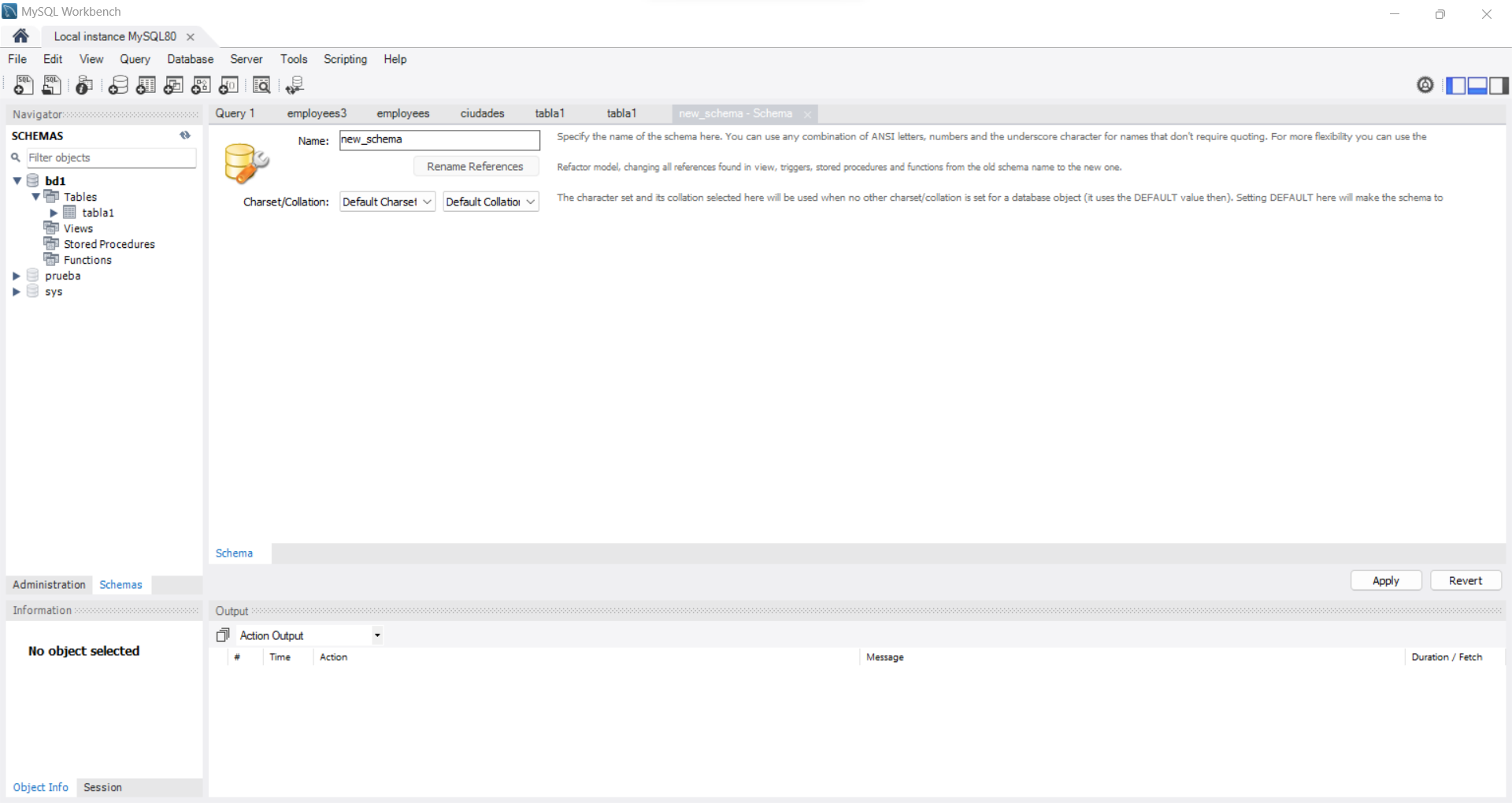
### VarChar(n):

Almacena una cadena de longitud variable. La cadena podrá contener desde 0 a 255 caracteres.

**CREAR UNA BASE DE DATOS DE FORMA GRAFICA**

Mirar si tiene conexión con Mysqlserver, en caso contrario crear la conexión con una nueva contraseña.

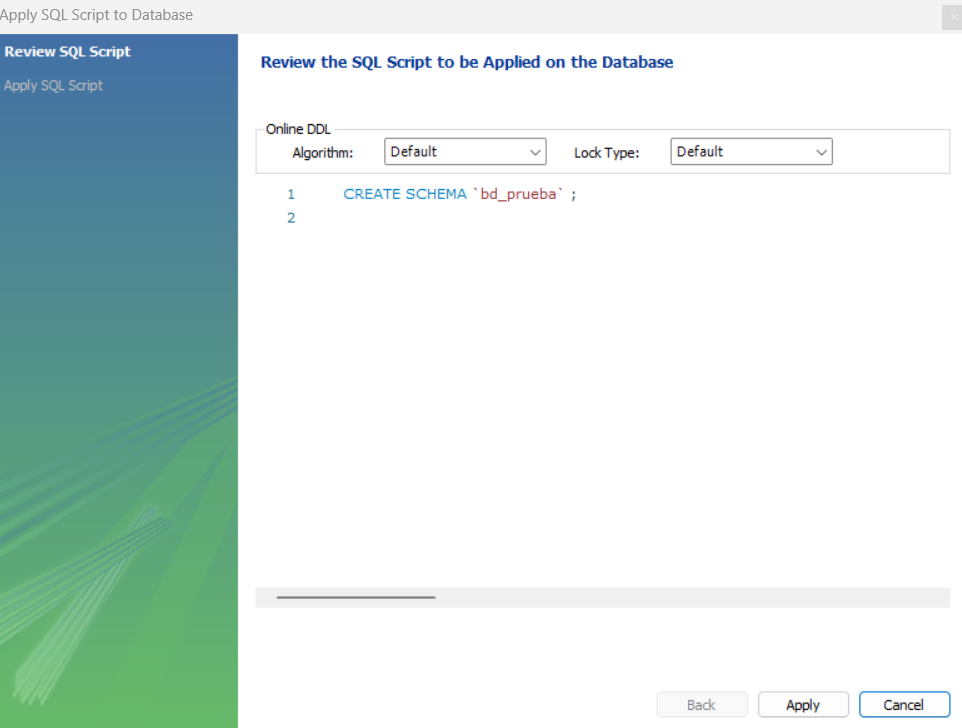
Click en la opción Create a new schema in the connected server (tiene icono db)

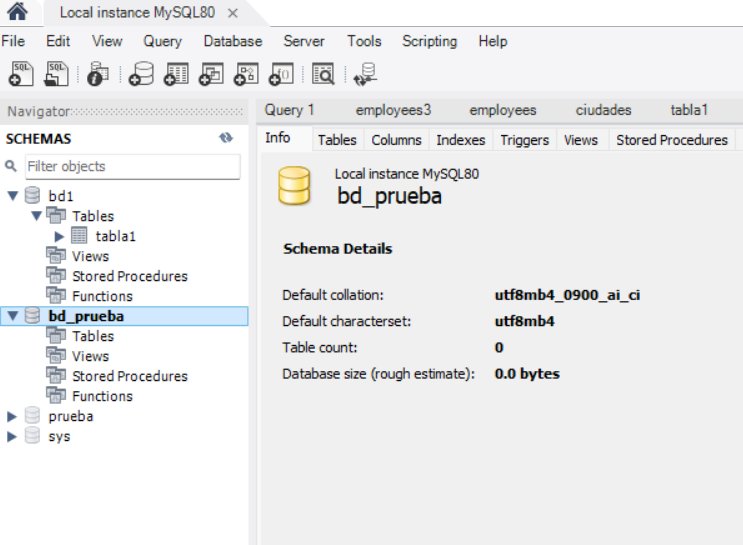


Ingresamos el nombre a nuestra db en la opción **name**

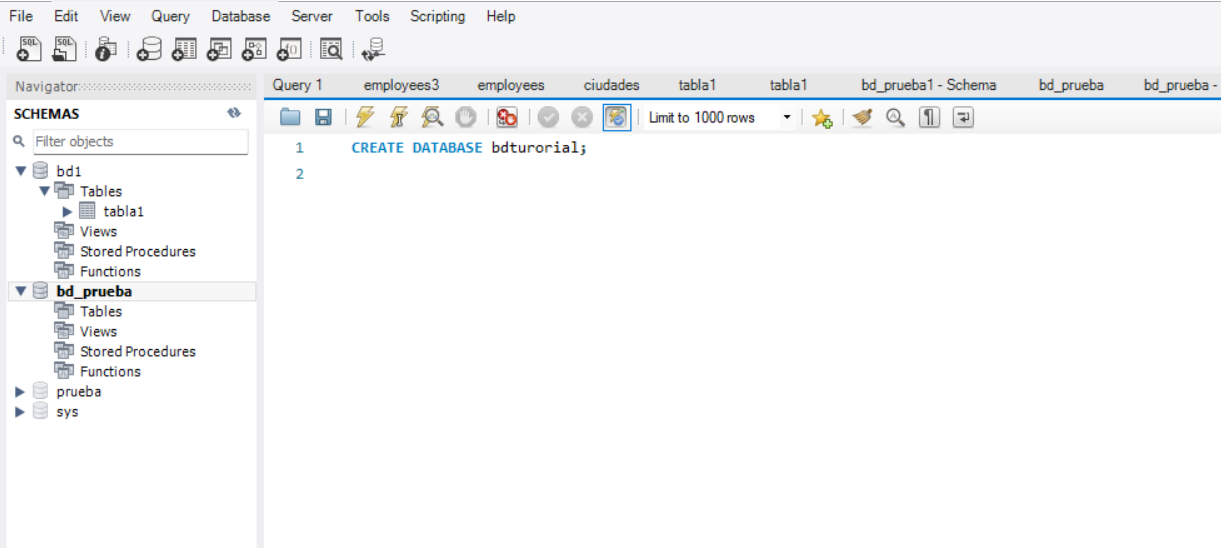
## ****Ejemplo:****

**bdprueba – apply**

**Escoger la opción Apply – finnish**

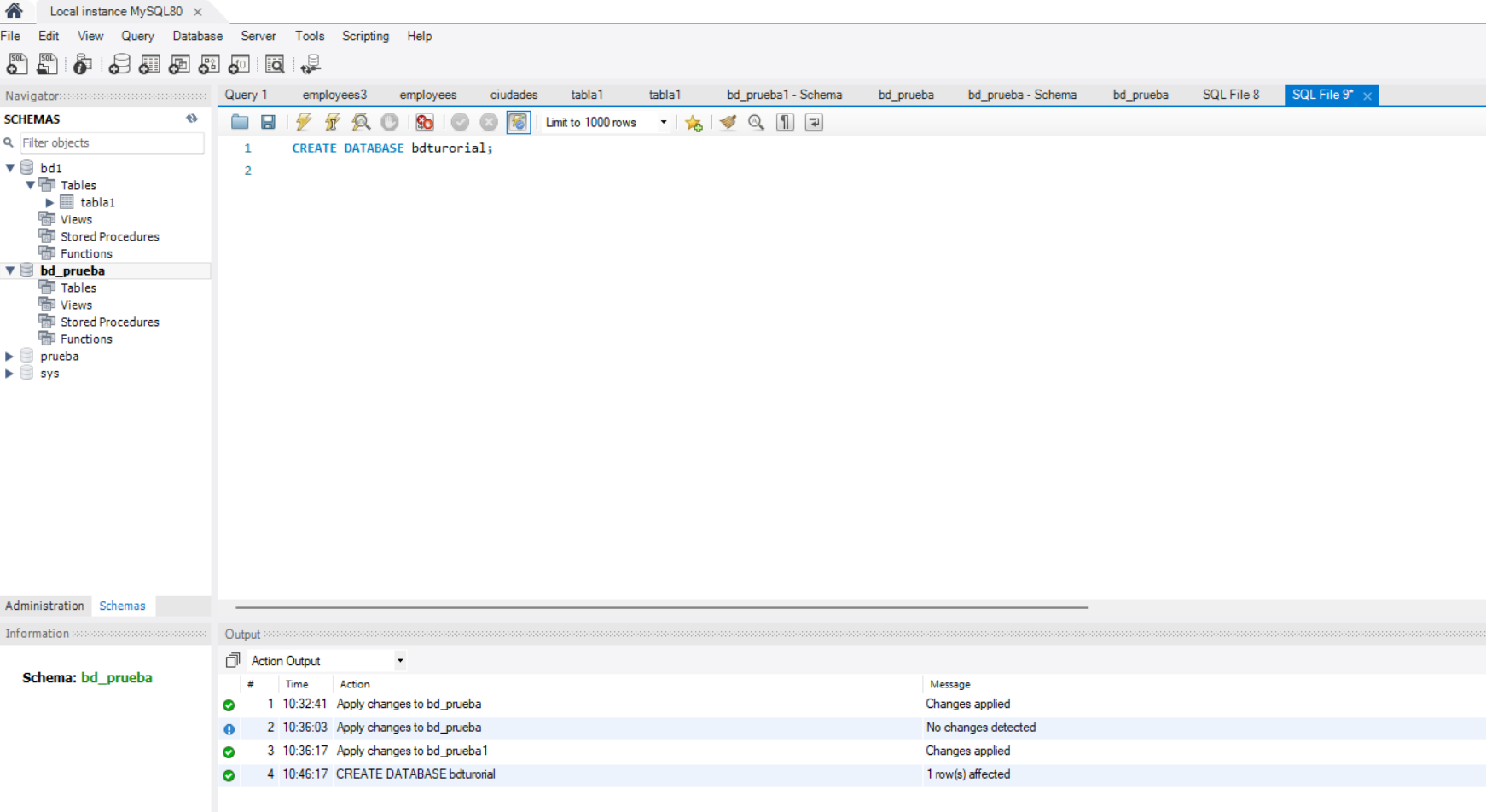
**se ha creado la base de datos bd\_prueba**

**clic en la opción create new sql tab for executing queries**



## ****crear la base de datos dbtutorial;****

**Ejecutamos en la forma o imagen del rayo.**



## ****Como crear una tabla en MySql workbench****

La sentencia básica de MySQL que se utiliza para crear una tabla es **create table** su función es crear una tabla con las características que le indiquemos seguidamente del comando en referencia.

Uno de los aspectos importantes a tomar en cuenta es que nuestra tabla estará compuesta por columnas y a cada columna debemos especificarle un tipo de dato para identificar si son datos numéricos, cadenas (alfanuméricos) o fechas y horas los que estaremos guardando en dicha columna.

## ****Sintaxis para crear tablas en Mysql****

La sintaxis de como crear una tabla en MySQL es la siguiente:

CREATE TABLE Nombre\_Tabla  
(  
Nombre\_Columna1 Tipo\_de\_Dato (longitud),  
Nombre\_Columna2 Tipo\_de\_Dato (longitud),  
Nombre\_Columna3 Tipo\_de\_Dato (longitud),  
….  
);

Los parámetros «**Nombre\_Columna**» especifican los nombres de las columnas que integran las tablas.

Los parámetros «**Tipo\_de\_Dato**» especifican que tipo de datos admitirá esa columna (ej. varchar, integer, decimal, date, etc.).

El parámetro «**Longitud**» especifica la longitud máxima de caracteres que admitirá la columna de la tabla.

Cabe aclarar que hay varias propiedades que pueden ser aplicadas a las columnas de la tabla, por ejemplo una de las más comunes es **primary key**, la cual nos permite indicar que columna será llave primaria, también podemos establecer que alguna columna no acepte valores nulos, a través de la propiedad **not null**.

## ****Codigo para crear tablas en MySql****

A continuación presentare un ejemplo de cómo crear una tabla llamada “**Alumnos**” con cinco columnas, el código es el siguiente:

CREATE DATABASE Tutorial;

use Tutorial;

CREATE TABLE Alumnos

(

IdAlumno int primary key not null,

Nombre varchar (45),

Apellido Varchar (45),

Edad int,

Direccion\_Residencia varchar (50)

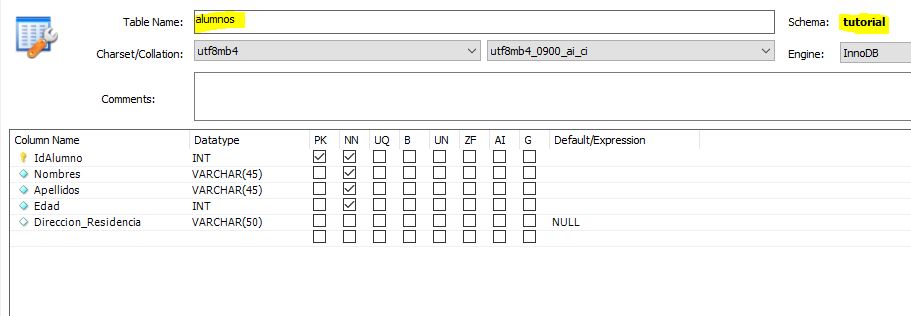
);

En el código anterior estamos creando la tabla “**Alumnos**” que tiene 5 columnas, podemos darnos cuenta que una de las novedades es en la columna IdAlumno, ya que tenemos la propiedad **PRIMARY KEY**, la cual indica que es la clave primaria que identifica de manera única cada registro/fila de la tabla, así mismo con la propiedad **not null** estamos indicando que esa columna es indispensable, es decir que siempre debe ser ingresado un IdAlumno.

# Insertar Datos en MySQL

En este artículo aprenderemos a **insertar datos** en una tabla de Mysql a través de código y de forma gráfica con la herramienta **MySQL Worbench**, obviamente lo que necesitamos es haber creado la tabla a la cual le vamos a insertar registros.

La estructura de la tabla a la cual le vamos a insertar datos, tiene 5 columnas (**IdAlumno, Nombres, Apellidos, Edad, Direccion\_Residencia**), podemos observarla a continuación:



## Insert en MySQL

El comando INSERT INTO va seguido del **nombre de la tabla** y su **encabezado** (lista de campos) entre paréntesis, luego se coloca la palabra reservada VALUES y, finalmente, la lista de **valores** a insertar encerrada entre paréntesis.

En MySQL, «INSERT» es una declaración utilizada para agregar nuevos registros o filas a una tabla en una base de datos. Esta declaración permite ingresar datos en una tabla específica, especificando los valores que deseas insertar en las columnas correspondientes. Básicamente, «INSERT» te permite crear nuevos registros en una tabla y llenar esos registros con datos específicos sin modificar los registros existentes.

## Sintaxis de Insert into en MySQL

La sintaxis para usar la instrucción **insert into** en una tabla de MySQL es la siguiente:

**NombreTabla**:

Es el nombre de la tabla en la que insertaremos registros.

**PrimeraColumna, SegundaColumna,..**:

Son las columnas de la tabla en la que vamos a insertar registros.

**“Dato1”, “Dato2” ,..**:

Son los valores que vamos a guardar en cada columna especificada.

Es importante mencionar que la sintaxis anterior se puede reducir en los casos que vamos agregar datos a todas las columnas, ya que podríamos plantearlo de la forma siguiente:

INSERT INTO “NombreTabla” VALUES (“Dato1”, “Dato2”, etc);

## Insertar datos en una tabla mysql

Ya habiendo conocido la sintaxis ahora nos interesa poder agregar los registros, para ello lo haremos con la tabla Alumnos, vamos a tomar en cuenta que tiene los siguientes campos:  
IdAlumno  
Nombres  
Apellidos  
Edad  
Direccion\_Residencia

**INSERT INTO** Alumnos (IdAlumno, Nombres, Apellidos, Edad, Direccion\_Residencia) VALUES

('0101', 'Franklin1', 'Garcia', '25', 'avenida 01');

Si queremos agregar varios registros a través de un mismo insert, basta con agregar una coma en los valores que le enviamos en values, y especificar los datos a insertar.

**INSERT INTO** Alumnos (IdAlumno, Nombres, Apellidos, Edad, Direccion\_Residencia)

**VALUES**

('0102', 'Franklin1', 'Garcia', '25', 'avenida 01'),

('0103', 'Franklin2', 'Garcia', '25', 'avenida 02'),

('0104', 'Franklin3', 'Garcia', '25', 'avenida 03');

**CONSULTA DE DATOS MYSQL**

Las bases de datos relacionales son una poderosa herramienta que nos permite almacenar datos dentro de sus tablas de forma ordenada y permanente, a los cuales podemos acceder posteriormente por medio de consultas.

### Cláusula SELECT FROM

La funció **SELECT** \* **FROM**es una instrucción básica en SQL que se utiliza para recuperar todos los datos de todas las columnas de una tabla en una base de datos. En otras palabras, esta instrucción se utiliza para seleccionar todas las filas y todas las columnas de una tabla.

La sintaxis básica de la instrucción **SELECT** \* **FROM**es la siguiente:

**SELECT** \* **FROM** NOMBRE\_DE\_LA\_TABLA;

EJEMPLO

**SELECT** ListaColumnas **FROM** NombreTabla;

**SELECT**: Es el comando que se utiliza para obtener registros de las tablas.

**SELECT \* FROM** Alumnos;

Cuando se hace referencia a dos columnas de la misma tabla.

**SELECT** Nombre,Apellidos **FROM** Alumnos;

**CLAUSULA WHERE EN SQL**

**Dónde cláusula** en MySQL es una palabra clave que se utiliza para especificar los criterios exactos de datos o filas que se verán afectados por la declaración SQL especificada.

hay ocasiones en las que queremos restringir los resultados de la consulta a una condición específica. La cláusula WHERE de SQL resulta útil en tales situaciones.

**Operadores Relacionales:**



### **Sintaxis de la cláusula WHERE**

La sintaxis básica de la cláusula WHERE cuando se usa en una declaración SELECT WHERE de MySQL es la siguiente.

**SELECT \* FROM** tableName WHERE condition;

**ejemplo:**

SELECT \* FROM Alumnos WHERE Nombres =’Raul’;

SELECT \* FROM Alumnos WHERE Edad > 28;

**Sintaxis Eliminar Registros**

DELETE FROM Alumnos WHERE Id = 3;

para verificar se ha eliminado

SELECT \* FROM Alumnos;

**Sintaxis para Actualizar**

**\*/actualiza datos\*/**  
UPDATE alumnos set nombres = 'abraham' WHERE Idalumnos = '01';  
 **\*/ordenar tablas \*/**

SELECT nombres, apellido FROM alumnos ORDER BY nombres, apellido;

Con esta estructura de base de datos, podrías realizar diferentes operaciones, como:  
  
1. **Insertar datos:** Puedes agregar nuevos productos o clientes a las tablas utilizando comandos SQL como **INSERT INTO.**  
2. **Consultar datos:** Puedes hacer consultas para obtener información específica de la base de datos. Por ejemplo, podrías buscar todos los productos con un precio mayor a $50 o encontrar todos los clientes que viven en una determinada ciudad.  
  
3. **Actualizar datos:** Puedes modificar la información existente en la base de datos. Por ejemplo, podrías actualizar el precio de un producto o cambiar la dirección de un cliente.  
  
4. **Eliminar datos:** Puedes eliminar registros de la base de datos que ya no sean necesarios. Por ejemplo, podrías eliminar un producto que ya no esté disponible o un cliente que haya cerrado su cuenta.

Taller N°1

**Ejercicio 1: Crear una tabla de empleados  
  
1**. Abre MySQL Workbench y crea una nueva conexión a tu servidor MySQL.  
2. Crea una nueva base de datos llamada "empresa" utilizando el siguiente comando SQL:  
  
  
CREATE DATABASE empresa;  
  
  
3. Utiliza la base de datos "empresa" ejecutando el siguiente comando:  
  
USE empresa;  
 **4. Crea una tabla llamada "empleados" con los siguientes campos:  
  
-** id: INT, clave primaria, autoincremental  
- nombre: VARCHAR(50)  
- apellido: VARCHAR(50)  
- edad: INT  
- salario: DECIMAL(10, 2) **Utiliza el siguiente comando SQL para crear la tabla:**CREATE TABLE empleados (  
 id INT PRIMARY KEY AUTO\_INCREMENT,  
 nombre VARCHAR(50),  
 apellido VARCHAR(50),  
 edad INT,  
 salario DECIMAL (10, 2)  
); **Ejercicio 2: Insertar datos en la tabla de empleados  
  
1. Inserta los siguientes datos en la tabla "empleados":**- nombre: "Juan", apellido: "Pérez", edad: 30, salario: 2500.00  
- nombre: "María", apellido: "González", edad: 28, salario: 2800.50  
- nombre: "Pedro", apellido: "López", edad: 35, salario: 3000.75 **Utiliza el siguiente comando SQL para insertar los datos:**INSERT INTO empleados (nombre, apellido, edad, salario)  
VALUES  
 ('Juan', 'Pérez', 30, 2500.00),  
 ('María', 'González', 28, 2800.50),  
 ('Pedro', 'López', 35, 3000.75); **Ejercicio 3: Consultar datos de la tabla de empleados  
  
1.** Consulta todos los empleados de la tabla "empleados":  
Utiliza el siguiente comando SQL para consultar los datos:  
  
  
SELECT \* FROM empleados;  
 **2. Consulta el nombre y salario de los empleados que tienen un salario mayor a 2800.00:**Utiliza el siguiente comando SQL para consultar los datos:SELECT nombre, salario FROM empleados WHERE salario > 2800.00;

**Referencias Bibliograficas:**

https://codigosql.top/mysql/crear-tablas/

https://desarrolloweb.com/articulos/1054.php

[https://www.guru99.com/es/where-clause.html#:~:text=%C2%BFQu%C3%A9%20es%20la%20cl%C3%A1usula%20WHERE,por%20la%20declaraci%C3%B3n%20SQL%20especificada](https://www.guru99.com/es/where-clause.html" \l ":~:text=¿Qué es la cláusula WHERE,por la declaración SQL especificada).