La primera BD

Dentro de MySQL

Creación de una tabla en SQL

Primero crearemos una base de datos llamada “empresa”, y después, la tabla de empleados, con las siguientes columnas:

ID: Almacena el identificador único de cada trabajador. Este campo se generará automáticamente.
Nombre: Almacena el nombre de los empleados.
Apellido: Almacena el apellido de los empleados.
Edad: Almacena la edad de los empleados.
Salario: Almacena el salario de los empleados.

Esta tabla nos permitirá ejemplificar cómo crear, visualizar y manipular datos en SQL.

Comando CREATE

El comando CREATE DATABASE se utiliza para crear una nueva base de datos. Aquí tienes un ejemplo de cómo se ve la sintaxis para crear una base de datos:

```
CREATE DATABASE nombre_de_la_base_de_datos;
```

El comando CREATE se utiliza en SQL para crear una nueva tabla en una base de datos. Es fundamental comprender cómo utilizar este comando correctamente, ya que nos permite definir la estructura de la tabla y sus columnas.

Comando USE

El comando USE se utiliza en SQL para seleccionar una base de datos específica en la que se realizarán las consultas. Nos permite establecer el contexto de trabajo y asegurarnos de que nuestras operaciones se realicen en la base de datos deseada.

```
USE nombre_de_la_base_de_datos;
```

Al ejecutar este comando, nos moveremos al contexto de la base de datos especificada. A partir de ese momento, todas las consultas y comandos que ejecutemos se aplicarán a esa base de datos en particular.

Comando USE

Aquí tienes como se vería la creación de la base de datos "empresa" y la selección de la misma:

```
CREATE DATABASE empresa;
USE empresa;
```

Una vez que la base de datos "empresa" se ha creado y se ha seleccionado utilizando el comando USE, puedes proceder a crear tablas, realizar consultas y ejecutar otros comandos en el contexto de esa base de datos.

Comando CREATE

Y ahora procedemos a la creación de la tabla “empleados” dentro de la BD “empresa”. Aquí tienes un ejemplo de cómo se vería la sintaxis del comando CREATE para nuestra tabla:

```
CREATE TABLE empleados  (
  ID INT AUTO_INCREMENT PRIMARY KEY,
  Nombre VARCHAR(255),
  Apellido VARCHAR(255),
  Edad INT,
  Salario DECIMAL(10, 2)
);
```

Comando CREATE

En este ejemplo, especificamos el nombre de la tabla seguido de la lista de columnas y sus tipos de datos.
Hemos agregado la columna "**ID**" como un identificador único para cada trabajador utilizando el tipo de dato **INT** y la cláusula **AUTO_INCREMENT** para generar automáticamente valores incrementales. Además, hemos definido el campo "**ID**" como **PRIMARY KEY** para asegurar su unicidad.
Utilizamos **VARCHAR** para los campos de nombre y apellido, ya que son cadenas de texto de longitud variable.
El campo de edad se define como **INT** para almacenar valores numéricos enteros.
Por último, el campo de salario se define como **DECIMAL**(10, 2) para almacenar valores numéricos decimales con un máximo de 10 dígitos y 2 decimales.

Comando SHOW

El comando SHOW se utiliza en SQL para mostrar información sobre las bases de datos, tablas o columnas. Nos permite obtener detalles sobre la estructura y los objetos de nuestra base de datos.
En nuestro ejemplo de la tabla de empleados, utilizaremos el comando SHOW para mostrar información sobre la tabla "empleados" que hemos creado. Aquí tienes un ejemplo de cómo se vería la sintaxis del comando SHOW para nuestra tabla:

```
SHOW TABLES;
```

Comando SHOW

Al ejecutar este comando, obtendremos una lista de las tablas existentes en la base de datos. En nuestro caso, deberíamos ver la tabla "empleados" en la lista resultante.

El comando SHOW también puede utilizarse para obtener información sobre las columnas de una tabla específica. Por ejemplo:

```
SHOW COLUMNS FROM empleados;
```

Comando INSERT INTO

En la tabla "Trabajadores" que creamos previamente, podemos ingresar registros para almacenar la información de los empleados. A continuación, se muestra un ejemplo de inserción de cinco registros en la tabla:

```
INSERT INTO empleados (Nombre, Apellido, Edad, Salario)
VALUES ('Juan', 'Pérez', 35, 2500.00),
       ('María', 'Gómez', 28, 2000.00),
       ('Pedro', 'López', 42, 3000.00),
       ('Laura', 'García', 31, 2800.00),
       ('Carlos', 'Martínez', 39, 3200.00);
```

Comando INSERT INTO

En este ejemplo, hemos utilizado el comando INSERT INTO para agregar registros a la tabla “empleados".
Cada registro se especifica utilizando la sintaxis VALUES y se proporcionan los valores correspondientes para cada columna en el mismo orden en que se definen las columnas en la tabla.

Ejercicios SQL 01

```
-- Crear la tabla Trabajadores
CREATE TABLE Trabajadores (
  ID INT PRIMARY KEY AUTO_INCREMENT,
  Nombre VARCHAR(255),
  Apellidos VARCHAR(255),
  Cargo VARCHAR(255),
  Antiguedad INT
);
```

```
-- Insertar datos en la tabla Trabajadores
INSERT INTO Trabajadores (Nombre, Apellidos
Cargo, Antiguedad) VALUES
  ('John', 'Doe', 'Gerente', 3),
  ('Jane', 'Smith', 'Analista', 1),
  ('Mike', 'Johnson', 'Desarrollador', 2),
  ('Sarah', 'Williams', 'Diseñadora', 4),
  ('David', 'Brown', 'Asistente', 1);
```

Ejercicios SQL 02 | Consultas

1. Obtener todos los trabajadores:

```
SELECT * FROM Trabajadores;
```

2. Obtener los nombres y apellidos de los trabajadores que tienen una antigüedad mayor a 2 años:

```
SELECT Nombre, Apellidos FROM Trabajadores WHERE Antiguedad > 2;
```

3. Obtener los cargos únicos de los trabajadores:

```
SELECT DISTINCT Cargo FROM Trabajadores;
```

4. Obtener el número total de trabajadores:

```
SELECT COUNT(*) AS Total FROM Trabajadores;
```

Ejercicios SQL 02 | Consultas

5. Obtener los trabajadores ordenados por antigüedad de forma descendente:

```
SELECT * FROM Trabajadores ORDER BY Antiguedad DESC;
```

6. Obtener el nombre completo de los trabajadores
concatenando el nombre y los apellidos:

```
SELECT CONCAT(Nombre, ' ', Apellidos) AS NombreCompleto FROM Trabajadores;
```