LENGUAJE DDL

CREATE DATABASE:

```
CREATE DATABASE mi_base_de_datos;
```

Opciones: conjunto de caracteres y la ordenación.

Al crear una base de datos o una tabla en MySQL, es posible especificar opciones adicionales que definen ciertos aspectos del comportamiento y las características de la base de datos o la tabla. Dos de estas opciones comunes son el conjunto de caracteres (character set) y la ordenación (collation).

Conjunto de caracteres (Character Set): Un conjunto de caracteres es un conjunto de símbolos y reglas utilizadas para representar datos en una base de datos. Define qué caracteres se pueden almacenar y cómo se almacenan. Por ejemplo, un conjunto de caracteres puede ser "UTF-8" que permite representar una amplia gama de caracteres de diferentes idiomas.

Ordenación (Collation): La ordenación determina cómo se comparan y ordenan los datos almacenados en una base de datos. Cada conjunto de caracteres puede tener varias ordenaciones disponibles. Por ejemplo, una ordenación "utf8_general_ci" se utiliza para comparaciones de texto insensibles a mayúsculas y minúsculas en UTF-8.

Cómo especificar estas opciones al crear una base de datos:

```
CREATE DATABASE mi_base_de_datos
    CHARACTER SET utf8mb4
    COLLATE utf8mb4_unicode_ci;
```

CREATE TABLE:

```
CREATE TABLE estudiantes (
    id INT PRIMARY KEY,
    nombre VARCHAR(50),
    edad INT,
    carrera VARCHAR(50)
);
```

Definición de columnas con nombres, tipos de datos y restricciones:

```
CREATE TABLE empleados (
    id INT PRIMARY KEY,
```

```
    nombre VARCHAR(50) NOT NULL,
    edad INT,
    salario DECIMAL(10, 2) CHECK (salario > 0),
    fecha_contratacion DATE,
    departamento_id INT,
      CONSTRAINT fk_departamento FOREIGN KEY (departamento_id) REFERENCES
departamentos(id)
);
```

Uso de cláusulas como PRIMARY KEY, FOREIGN KEY y UNIQUE:

```
CREATE TABLE departamentos (
    id INT PRIMARY KEY,
    nombre VARCHAR(50) UNIQUE
);

CREATE TABLE empleados (
    id INT PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    departamento_id INT,
    FOREIGN KEY (departamento_id) REFERENCES departamentos(id)
);
```

ALTER TABLE:

Agregar

```
ALTER TABLE empleados
ADD COLUMN email VARCHAR(100);
```

modificar

```

ALTER TABLE empleados
MODIFY COLUMN salario DECIMAL(12, 2);
```

eliminar columnas.

```
ALTER TABLE empleados
DROP COLUMN email;
```

Cambiar el nombre de una tabla.

```
ALTER TABLE empleados
```

```
RENAME TO personal;
```

Agregar y eliminar restricciones.

```
ALTER TABLE departamentos
ADD CONSTRAINT pk_departamentos PRIMARY KEY (id);
```

DROP TABLE

```
DROP TABLE nombre_de_tabla;
```

Eliminación segura de tablas con la cláusula IF EXISTS.

```
DROP TABLE IF EXISTS nombre_de_tabla;
```

CREATE INDEX

Porque usar índices

Los índices en una base de datos son como índices en un libro: permiten acceder rápidamente a datos específicos en lugar de tener que buscar a través de toda la tabla. Algunas razones para usar índices son:

- Mejora del rendimiento: Los índices aceleran las búsquedas y consultas, lo que puede ser crucial en tablas grandes.
- Optimización de JOINs: Los índices mejoran el rendimiento de las operaciones JOIN, que involucran la combinación de datos de múltiples tablas.
- Ordenamiento: Los índices también pueden acelerar el ordenamiento de resultados en consultas.
- Restricciones de unicidad: Los índices únicos garantizan que los valores en una columna no se repitan.

Sintaxis básica de CREATE INDEX.

```
CREATE INDEX nombre_del_indice
ON nombre_de_tabla (columna);
```

Creación de índices en una o varias columnas.

```
CREATE INDEX idx_apellidos
ON empleados (apellido);

CREATE INDEX idx_nombre_apellido
ON empleados (nombre, apellido);
```

Uso de índices para acelerar búsquedas

Supongamos que desea buscar todos los empleados con un apellido específico:

```
SELECT * FROM empleados
WHERE apellido = 'Smith';
```

Si tiene un índice en la columna "apellido", esta búsqueda será mucho más rápida, ya que el índice actúa como un mapa que lleva directamente a las filas relevantes.

En resumen, los índices son herramientas esenciales para mejorar el rendimiento de las consultas y la eficiencia en la recuperación de datos. Sin embargo, es importante utilizar índices de manera estratégica y equilibrada, ya que demasiados índices pueden tener un impacto negativo en las operaciones de escritura y el almacenamiento.

Tipos de datos

Tipos de datos numéricos:

- INT: Números enteros. Ejemplo: 10, -5.
- DECIMAL(p, s): Número decimal con precisión p y escala s. Ejemplo: 123.45.

Tipos de datos de cadena:

- VARCHAR(n): Cadena de longitud variable con hasta n caracteres. Ejemplo: 'Hola'.
- CHAR(n): Cadena de longitud fija con exactamente n caracteres. Ejemplo: 'ABC'.

Tipos de datos de fecha y hora:

- DATE: Fecha en formato 'YYYY-MM-DD'. Ejemplo: '2023-08-04'.
- TIME: Hora en formato 'HH:MM:SS'. Ejemplo: '14:30:00'.
- DATETIME: Fecha y hora en formato 'YYYY-MM-DD HH:MM:SS'. Ejemplo: '2023-08-04 14:30:00'.

Tipos de datos binarios:

- BLOB: Datos binarios grandes. Ejemplo: Imágenes, archivos.
- BINARY: Cadena binaria de longitud fija. Ejemplo: '\x01\x02\x03'.

Tamaño y formato de tipos de datos:

- Los tipos numéricos tienen diferentes tamaños según la precisión.
- Los tipos de cadena tienen longitudes máximas que pueden variar.
- Los tipos de fecha y hora almacenan información de fecha y/o hora.
- Los tipos binarios pueden almacenar datos binarios de diversos tamaños.

Uso adecuado de tipos de datos en la definición de columnas:

- Elijan el tipo de datos más apropiado según el tipo de información que vas a almacenar.
- Utilicen INT para identificadores numéricos, VARCHAR o CHAR para texto y DATE o DATETIME para fechas y horas.
- Ajusten el tamaño de las columnas de cadena y los parámetros de los tipos numéricos según las necesidades reales de los datos que almacenarás.

Ejemplo de uso adecuado de tipos de datos en la definición de columnas:

```
CREATE TABLE productos (
    id INT PRIMARY KEY,
    nombre VARCHAR(100),
    precio DECIMAL(10, 2),
    fecha_vencimiento DATE
);
```

En este ejemplo, se utilizan tipos de datos adecuados para almacenar información sobre productos, como identificadores, nombres, precios y fechas de vencimiento.

Constraints

Una restricción (constraint, en inglés) en una base de datos es una regla o condición que se aplica a una o más columnas en una tabla para mantener la integridad y coherencia de los datos almacenados en esa tabla

Restricciones para columnas:

- PRIMARY KEY: Identifica una columna o conjunto de columnas como clave primaria. Garantiza que los valores sean únicos y no nulos.
- FOREIGN KEY: Crea una relación entre tablas, donde los valores en la columna referenciante deben existir en la columna referenciada de otra tabla.
- UNIQUE: Asegura que los valores en la columna sean únicos en la tabla.
- NOT NULL: Requiere que los valores en la columna no sean nulos.

Uso de restricciones para garantizar la integridad de los datos:

Las restricciones son cruciales para mantener la integridad de los datos. Por ejemplo, una restricción PRIMARY KEY asegura que cada fila tenga una identificación única. Una restricción FOREIGN KEY garantiza que no se puedan introducir datos inconsistentes en una relación.

Mantenimiento de la integridad de los datos:

Mantenimiento de la integridad de los datos:

PRIMARY KEY:

Se usa para establecer una columna o conjunto de columnas como clave primaria en una tabla. Garantiza que los valores sean únicos y no nulos. Ejemplo:

```
CREATE TABLE estudiantes (
    id INT PRIMARY KEY,
    nombre VARCHAR(50),
    edad INT
);
```

FOREIGN KEY: Se utiliza para establecer relaciones entre tablas. Asegura que los valores en una columna coincidan con los valores en otra columna de otra tabla. Ejemplo:

```
CREATE TABLE cursos (
    id INT PRIMARY KEY,
    nombre VARCHAR(100)
);

CREATE TABLE inscripciones (
    id INT PRIMARY KEY,
    estudiante_id INT,
    curso_id INT,
    FOREIGN KEY (estudiante_id) REFERENCES estudiantes(id),
    FOREIGN KEY (curso_id) REFERENCES cursos(id)
);
```

En este ejemplo, la tabla "inscripciones" tiene dos columnas que son claves foráneas que se relacionan con las tablas "estudiantes" y "cursos".

El uso de restricciones como PRIMARY KEY y FOREIGN KEY garantiza que los datos se mantengan consistentes y evita la introducción de información errónea o incoherente en la base de datos. Esto es fundamental para asegurar la integridad de los datos a lo largo del tiempo.

Interfaz línea de comandos

Abra la ventana de línea de comandos (cmd) en Windows o la terminal en sistemas basados en Unix (como Linux o macOS).

Navegue hasta la ubicación donde está instalado XAMPP. Por lo general, se encuentra en la raíz del disco (por ejemplo, C:\xampp en Windows).

Dentro de la carpeta XAMPP, navega a la subcarpeta mysql\bin.

Ejecute el comando mysql -u root -p para iniciar la interfaz de línea de comandos de MySQL. Esto le pedirá la contraseña del usuario "root" de MySQL en XAMPP. Inicialmente el campo contraseña esta vacío.

A partir de este punto, podrá ingresar y ejecutar comandos SQL directamente en la línea de comandos de MySQL.