Material descargable

Lenguaje DDL

El Lenguaje de Definición de Datos (DDL) desempeña un papel crítico en la administración de bases de datos. DDL es un subconjunto de SQL que se utiliza para definir y gestionar la estructura de una base de datos, lo que incluye la creación, modificación y eliminación de objetos de base de datos.

Es importante distinguir entre DDL y DML. DDL se enfoca en la definición de la base de datos en sí, mientras que DML se utiliza para manipular los datos dentro de la base de datos.

Comandos DDL

CREATE SCHEMA o CREATE DATABASE

El comando **CREATE SCHEMA o CREATE DATABASE** se utiliza para crear una nueva base de datos. La base de datos recién creada actuará como un contenedor para organizar tablas, vistas, procedimientos almacenados y otros objetos relacionados.

```
CREATE DATABASE nombre_de_la_base_de_datos;

CREATE SCHEMA nombre_de_la_base_de_datos;
```

CREATE TABLE

El comando **CREATE TABLE** se utiliza para crear una nueva tabla dentro de una base de datos existente. Define la estructura de la tabla, especificando el nombre de la tabla, las columnas y sus tipos de datos, restricciones, claves primarias y otros detalles importantes.

```
CREATE TABLE nombre_de_la_tabla (
```

```
    columna1 tipo_de_dato1 restricciones,
    columna2 tipo_de_dato2 restricciones,
    ...
    [restricciones]
);


CREATE TABLE empleados (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    edad INT,
    salario DECIMAL(10, 2)
);
```

- **Filas y Columnas:** Las tablas constan de filas y columnas. Cada fila representa una entrada de datos única, mientras que cada columna describe un atributo específico o campo de datos.
- **Estructura Predefinida:** Antes de insertar datos en una tabla, debes definir su estructura, especificando qué columnas contendrá y qué tipo de datos pueden almacenar.
- **Relaciones:** Las tablas pueden relacionarse entre sí mediante claves primarias y foráneas, lo que permite representar datos interconectados y facilitar la consulta.
- **Integridad de los Datos:** Las bases de datos aplican restricciones y reglas para garantizar la integridad de los datos almacenados en las tablas. Esto ayuda a mantener la precisión y consistencia de la información.

Tipos de Datos en MySQL

A continuación, se presenta una tabla con algunos de los tipos de datos más comunes admitidos por MySQL, junto con una breve descripción de cada uno:

Tipo de Datos	Descripción	Tipo General
BIT(M)	Almacena valores de bits. M puede variar de 1 a 64 bits.	Numeric
INT	Número entero (con signo o sin signo).	Numeric
SMALLINT	Número entero pequeño (con signo o sin signo).	Numeric

TINYINT	Número entero muy pequeño (con signo o sin signo).	Numeric
MEDIUMINT	Número entero mediano (con signo o sin signo).	Numeric
BIGINT	Número entero grande (con signo o sin signo).	Numeric
FLOAT	Número decimal de precisión simple (comúnmente utilizado para números con decimales).	Numeric
DOUBLE	Número decimal de precisión doble (comúnmente utilizado para números con decimales).	Numeric
DECIMAL(p, s)	Número decimal con precisión p y escala s (ideal para valores monetarios y otros números precisos).	Numeric
VARCHAR(n)	Cadena de caracteres de longitud variable (hasta n caracteres).	String
CHAR(n)	Cadena de caracteres de longitud fija (exactamente n caracteres).	String
TEXT	Texto largo (hasta 65,535 caracteres).	String
BLOB	Datos binarios largos (hasta 65,535 bytes).	Binary
DATE	Fecha (formato "YYYY-MM-DD").	Date and Time
TIME	Hora (formato "HH:MM:SS").	Date and Time
DATETIME	Fecha y hora (formato "YYYY-MM-DD HH:MM:SS").	Date and Time
TIMESTAMP	Fecha y hora, con soporte para zonas horarias (formato "YYYY-MM-DD HH:MM:SS").	Date and Time
BOOLEAN	Valor booleano (VERDADERO o FALSO).	Boolean

💡 También *se utiliza el BIT(1) para valores de tipo boolean, representando el valor 0 false y el valor 1 true.*

💡 *DATETIME es adecuado cuando necesitas mantener la fecha y hora exactas sin ajustes de zona horaria, mientras que TIMESTAMP es útil cuando deseas que las fechas y horas se guarden en UTC y se ajusten a la zona horaria del servidor al recuperarlas. En la mayoría de los casos siempre es preferible usar DATETIME.*

Rango de los tipos Numeric

Tipo de Dato	Almacenamiento (Bytes)	Valor Mínimo con Signo	Valor Mínimo sin Signo	Valor Máximo con Signo	Valor Máximo sin Signo
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-9223372036854775808	0	9223372036854775807	18446744073709551615
DECIMAL	Variable	Variable	Variable	Variable	Variable
FLOAT	4	-3.402823466E+38	0	3.402823466E+38	=
DOUBLE	8	-1.7976931348623157E+308	0	1.7976931348623157E+308	=

Rango de los tipos String

Tipo de Datos	Descripción

CHAR(M)	Cadena de longitud fija que se rellena con espacios a la derecha hasta alcanzar la longitud (M) de hasta 255 caracteres.
VARCHAR(M)	Cadena de longitud variable con una longitud máxima (M) de hasta 21,844 caracteres.
TINYTEXT	Cadena de texto con una longitud máxima de 255 caracteres.
TEXT	Cadena de texto con una longitud máxima de 65,535 caracteres.
MEDIUMTEXT	Cadena de texto con una longitud máxima de 16,777,215 caracteres.
LONGTEXT	Cadena de texto con una longitud máxima de 4,294,967,295 caracteres.
BINARY(M)	Cadena de bytes de longitud fija de hasta 255 bytes.
VARBINARY(M)	Cadena de bytes de longitud variable con una longitud máxima especificada (M) de hasta 65,535 bytes.
TINYBLOB	Almacena datos binarios con una longitud máxima de 255 bytes.
BLOB	Almacena datos binarios con una longitud máxima de 65,535 bytes.
MEDIUMBLOB	Almacena datos binarios con una longitud máxima de 16,777,215 bytes.
LONGBLOB	Almacena datos binarios con una longitud máxima de 4,294,967,295 bytes.
ENUM('valor1', 'valor2', ...)	Columna que puede contener un valor de una lista de valores predefinidos (hasta 65535).
SET('valor1', 'valor2', ...)	Columna que puede contener cero o más valores de una lista de valores predefinidos (hasta 64).

Los tipos de datos más comunes que utilizarás son el, BIT(1) INT, el VARCHAR, el DATE y el DATETIME

Restricciones

Las restricciones en una base de datos son reglas y condiciones que se aplican a las columnas o tablas para garantizar la integridad de los datos y las relaciones entre ellas.

Tabla de restricciones

Restricción	Descripción
PRIMARY KEY	Identifica de manera única una fila en una tabla. Puede ser aplicada a una o más columnas.
UNIQUE	Asegura que los valores en la columna o conjunto de columnas especificados sean únicos en todas las filas de la tabla.
NOT NULL	Requiere que un campo no tenga valores nulos (vacíos).
DEFAULT valor	Asigna un valor predeterminado a una columna si no se proporciona un valor al insertar un nuevo registro.
CHECK	Permite definir una condición que los valores en una columna deben cumplir.
FOREIGN KEY	Establece una relación entre dos tablas, asegurando que los valores en una columna coincidan con los valores en otra tabla.
AUTO_INCREMENT	Se utiliza generalmente en columnas numéricas para que el DBMS genere automáticamente un valor único cada vez que se inserta una fila.
INDEX	Crea un índice en una o más columnas para mejorar el rendimiento de las consultas.
FULLTEXT	Habilita búsquedas de texto completo en columnas de texto, permitiendo búsquedas avanzadas en texto.
ENUM	Limita los valores que se pueden insertar en una columna a un conjunto predefinido de valores.
SET	Similar a ENUM, pero permite múltiples valores seleccionados de un conjunto predefinido.

ON DELETE	Especifica qué debe suceder cuando se elimina una fila en la tabla relacionada. Puede tener valores como CASCADE, SET NULL, RESTRICT, etc.
ON UPDATE	Especifica qué debe suceder cuando se actualiza una fila en la tabla relacionada. Puede tener valores como CASCADE, SET NULL, RESTRICT, etc.
REFERENCES	Utilizado con FOREIGN KEY, especifica la tabla y columna a la que se hace referencia.

PRIMARY KEY

La restricción PRIMARY KEY se utiliza para garantizar que una columna o conjunto de columnas tenga valores únicos en una tabla. Se recomienda, crear un índice en la columna o conjunto de columnas correspondiente para optimizar las búsquedas y agilizar la recuperación de datos, como así, la unicidad de datos.

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

UNIQUE

La restricción UNIQUE se utiliza para asegurarse de que los valores en una columna o conjunto de columnas sean únicos en una tabla. A diferencia de PRIMARY KEY, una tabla puede tener varias columnas con restricciones UNIQUE, pero cada valor individual en esas columnas debe ser único.

```
CREATE TABLE empleados (
    id INT PRIMARY KEY,
    codigo_empleado INT UNIQUE,
    nombre VARCHAR(50),
    correo VARCHAR(100) UNIQUE
);
```

NOT NULL

La restricción NOT NULL se utiliza para asegurarse de que una columna no contenga valores nulos (sin información). Esto significa que cada fila debe tener un valor válido en esa columna.

```
CREATE TABLE pedidos (
    id INT PRIMARY KEY,
    fecha_pedido DATE NOT NULL,
    cliente_id INT
);
```

DEFAULT valor

La restricción DEFAULT se utiliza para establecer un valor predeterminado para una columna si no se proporciona ningún valor en la inserción de datos.

```
CREATE TABLE cuentas (
    id INT PRIMARY KEY,
    saldo DECIMAL(10, 2) DEFAULT 0.00
);
```

CHECK

La restricción CHECK se utiliza para garantizar que los valores en una columna cumplan con una condición especificada. Esta condición puede ser una expresión que limite los valores válidos en la columna.

```
CREATE TABLE productos (
    id INT PRIMARY KEY,
    precio DECIMAL(10, 2),
    cantidad INT,
    CHECK (precio > 0),
    CHECK (cantidad >= 0)
);
```

AUTO_INCREMENT

La restricción AUTO_INCREMENT se utiliza para generar automáticamente valores numéricos únicos al insertar registros en una tabla. Esto es comúnmente utilizado para generar valores de clave primaria únicos, identificadores o contadores.

```
CREATE TABLE empleados (
    empleado_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50),
    salario DECIMAL(10, 2)
);
```

INDEX

La restricción INDEX se utiliza para crear un índice en una o más columnas de una tabla para mejorar la velocidad de búsqueda y recuperación de datos. Aunque no es una restricción en el sentido tradicional, es fundamental para mejorar el rendimiento de las consultas en bases de datos grandes.

```
CREATE TABLE productos (
    producto_id INT PRIMARY KEY,
    nombre VARCHAR(100),
    categoria VARCHAR(50),
    INDEX idx_categoria (categoria)
);
```

FULLTEXT

La restricción FULLTEXT se utiliza en columnas de tipo TEXT para habilitar la búsqueda de texto completo en texto largo. Esto permite realizar búsquedas más avanzadas y precisas en grandes cantidades de texto.

```
CREATE TABLE articulos (
    id INT PRIMARY KEY,
    titulo VARCHAR(100),
    contenido TEXT,
    FULLTEXT (contenido)
);
```

💡*La restricción FULLTEXT se usa principalmente cuando se necesita realizar búsquedas de texto completo en campos con una gran cantidad de contenido, como artículos, descripciones o documentos extensos. Esta característica está diseñada para mejorar la eficiencia y precisión de las consultas en dichos campos, y no es necesario habilitarla en todas las columnas de texto.*

ENUM

La restricción ENUM se utiliza para definir un conjunto de valores permitidos para una columna. Esto garantiza que los valores ingresados sean uno de los valores especificados en la lista.

```
CREATE TABLE tallas (
    id INT PRIMARY KEY,
    talla ENUM('XS', 'S', 'M', 'L', 'XL')
);
```

SET

La restricción SET se utiliza para definir un conjunto de valores permitidos para una columna, similar a ENUM. Sin embargo, a diferencia de ENUM, SET permite múltiples valores en una sola columna.

```
CREATE TABLE opciones (
    id INT PRIMARY KEY,
    preferencias SET('Notificaciones', 'Correo', 'SMS', 'Teléfono')
);
```

FOREIGN KEY

La restricción FOREIGN KEY se utiliza para establecer y mantener relaciones entre tablas en una base de datos. Esta restricción garantiza que los valores en una columna (denominada clave foránea) coincidan con los valores en una columna de otra tabla (denominada clave principal). La restricción FOREIGN KEY es esencial para mantener la integridad referencial en una base de datos.

Supongamos que tenemos dos tablas: estudiantes y cursos. Para establecer una relación entre ellas, podríamos utilizar la columna curso_id en la tabla estudiantes como clave foránea que hace referencia a la columna id en la tabla curso. Esto asegura que cada registro en la tabla estudiantes esté asociado a un curso válido en la tabla cursos.

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

```
);

CREATE TABLE estudiantes (
    id INT PRIMARY KEY,
    nombre VARCHAR(50),
    curso_id INT,
    FOREIGN KEY (curso_id)
    REFERENCES cursos(id)
);
```

ON DELETE y ON UPDATE

Las restricciones ON DELETE y ON UPDATE se utilizan en las restricciones FOREIGN KEY para especificar qué debe suceder cuando se elimina o actualiza una fila en la tabla relacionada. Pueden tener valores como CASCADE, SET NULL, RESTRICT, entre otros, y desempeñan un papel crucial en el mantenimiento de la integridad referencial entre tablas.

CASCADE: elimina automáticamente los registros relacionados en la tabla hija cuando se elimina un registro en la tabla principal.

SET NULL: establece a NULL los registros relacionados en la tabla hija cuando se actualiza el valor en la tabla principal.

RESTRICT: impide que se realice una acción de eliminación si hay registros relacionados en la tabla hija que violen la restricción.

Supongamos que deseamos que cuando se elimine un estudiante de la tabla estudiantes, todos los cursos asociados a ese estudiante también se eliminen. Además, cuando se actualice el ID de un estudiante en la tabla estudiantes, se actualice automáticamente el ID del estudiante en la tabla cursos. Esto se puede lograr de la siguiente manera:

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

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

```
    FOREIGN KEY (curso_id)
    REFERENCES cursos(id)
     ON DELETE CASCADE # Si se elimina un curso, también elimina a los estudiantes asociados
     ON UPDATE SET NULL # Cuando se actualiza el ID del curso, establece NULL en el campo curso_id del estudiante (no es buena práctica actualizar el ID de ningún registro)
);
```

💡*el simbolo "#" se utiliza para realizar comentarios en el lenguaje SQL, no forma parte del script, es decir, no se ejecuta.*

Estas restricciones aseguran que cuando se elimina o actualiza un curso en la tabla cursos, se refleje de manera adecuada en los registros relacionados en la tabla estudiantes.

ALTER TABLE

El comando ALTER TABLE se utiliza para realizar cambios en una tabla existente sin tener que eliminarla y volver a crearla. Puedes utilizar ALTER TABLE para agregar, modificar o eliminar columnas, así como aplicar restricciones adicionales a la tabla.

Agregar una columna

```
ALTER TABLE nombre_de_tabla
ADD nombre_de_columna tipo_de_dato restricciones;

ALTER TABLE empleados
ADD fecha_de_nacimiento DATE;
```

Modificar una columna

```
ALTER TABLE nombre_de_tabla
MODIFY COLUMN nombre_de_columna nuevo_tipo_de_dato restricciones;

ALTER TABLE empleados
MODIFY COLUMN salario DECIMAL(10, 2) NOT NULL;
```

Eliminar una columna

```
ALTER TABLE nombre_de_tabla
DROP COLUMN nombre_de_columna;

ALTER TABLE empleados
DROP COLUMN telefono;
```

DROP TABLE

El comando **DROP TABL**E se utiliza para eliminar una tabla y todos los datos que contiene de forma permanente. Debes tener cuidado al usar este comando, ya que no hay forma de recuperar los datos una vez que se ha ejecutado.

```
DROP TABLE nombre_de_tabla;

DROP TABLE empleados;
```

📍*Es importante tener precaución al usar DROP TABLE, ya que puede provocar la pérdida irrecuperable de datos.*