Índice.

1. Lenguaje de definición de datos.

El DDL es el sublenguaje de SQL que permite la definición de los datos a través de las siguientes funciones:

- Creación de bases de datos, tablas, índices y otros objetos (vistas, …).
- Definición de estructuras físicas que contendrán los objetos de las bases de datos.

El término DDL fu introducido por primera vez con el modelo de base de datos CODASYL, en el que el esquema de la base de datos fue escrito en un lenguaje de descripción de datos con descripciones de registros, campos y conjuntos que conformaban el modelo de datos del usuario.

Posteriormente fue usado para referirse a un subconjunto de SQL, aunque ahora hace referencia en un sentido genérico a cualquier lenguaje formal que describe datos o estructuras de información (como los esquemas XML).

El DDL tiene tres instrucciones básicas:

- Create → crea la base de datos y los objetos de ella misma.
- Drop → borra todos los objetos de base de datos (continente y contenido).
- Alter → modifica la estructura de la base de datos.
- Truncate → borra todos los registros de una tabla (sólo contenido).
- Comment → agrega un comentario al diccionario de datos.
- Rename → cambia el nombre de cualquier objeto de la base de datos.

```
Create Database Teatro;

Use Teatro;

Create table Actores(
        #Actor INTEGER PRIMARY KEY,
        Nombre VARCHAR(40),
        Fecha DATE,
        Nacionalidad VARCHAR(20)
);
```

2. Operaciones con Bases de Datos.

Las operaciones que se pueden realizar desde el DDL con las bases de datos son las siguientes:

- **Creación** → crea una tabla: CREATE DATABASE.

```
CREATE DATABASE   [IF NOT EXISTS]  nombre
            [especificación_create [, especificación_create] ...]
especificación_create:
            [DEFAULT] CHARACTER SET juego_caracteres
      |     [DEFAULT] COLLATE nombre_colación
```

```
CREATE DATABASE ejemplo;
CREATE DATABASE ejemplo CHARACTER SET latin1 COLLATE latin1_spanish_ci;
```

- **Modificación** → cambia algún aspecto de una base de datos: ALTER DATABASE. En MySQL sólo se puede cambiar el juego de caracteres y su colación:

```
ALTER DATABASE      nombre
            [DEFAULT] CHARACTER SET juego_caracteres
      |     [DEFAULT] COLLATE nombre_colación
```

```
ALTER DATABASE ejemplo CHARACTER SET latin1 COLLATE latin1_german1_ci;
```

- **Borrado** → elimina una base de datos: DROP DATABASE.

```
DROP DATABASE [IF EXISTS] nombre_base_datos;
```

```
DROP DATABASE ejemplo;
```

- **Consulta** → muestra las base de datos: SHOW DATABASES.

```
SHOW DATABASES;
```

- **Uso** → se carga la base de datos para trabajar: USE.

```
USE nombre_base_datos;
```

```
USE ejemplo;
```

El juego de caracteres de la base de datos puede ser utf8, latin1, latin2, …

La colación especifica cómo se va a tratar el alfabeto del juego de caracteres (ordenación y cómo se compararán los caracteres): si la ñ va después de la n.

3. Operaciones con Tablas.

Las operaciones que se pueden realizar desde el DDL con las tablas son las siguientes:

- **Creación** → crea una tabla: CREATE TABLE.

```
CREATE TABLE  nombre_tabla
            [definición_create [, definición_create] ...]
            [opciones_tabla]
definición_create:
            definición_columna
      |     [CONSTRAINT [símbolo]] PRIMARY KEY (nombre_columna, ...)
      |     [CONSTRAINT [símbolo]] FOREIGN KEY (nombre_columna, ...)
            [definición_referencia]
definición_columna:
            nombre_columna tipo_datos [NOT NULL | NULL]  [DEFAULT valor]  [UNIQUE [KEY] | [PRIMARY] KEY]
            [definición_referencia ]
definición_referencia:
            REFERENCES nombre_tabla [(nombre_columna, ...)]
                        [ON DELETE {CASCADE | SET NULL | NO ACTION} ]
                        [ON UPDATE {CASCADE | SET NULL | NO ACTION} ]
opciones_tabla:
            opción_tabla [opción_tabla] ...
opción_tabla:
            ENGINE = nombre_motor
      |     AUTO_INCREMENT = valor
      |     [DEFAULT] CHARACTER SET juego caracteres [collate colación]
      |     CHECKSUM = {0 | 1}
      |     COMMENT = 'string'
      |     MAX_ROWS = valor
      |     MIN_ROWS = valor
```

```
CREATE TABLE ejem(
      id  int  PRIMARY KEY,
      nombre varchar(20),
      direccion varchar(40)
);
```

- **Modificación** → cambia algún aspecto del contenido de una tabla: ALTER TABLE.

```
ALTER TABLE  nombre_tabla
            especificación_alter [, especificación_alter] ...
especificación_alter:
            ADD definición_columna [FIRST|AFTER nombre_columna]
      |     ADD (definición_columna, ...)
      |     ADD [CONSTRAINT [símbolo]]  PRIMARY KEY (nombre_columna, ...)
      |     ADD [CONSTRAINT [símbolo]]  UNIQUE (nombre_columna, ...)
      |     ADD [CONSTRAINT [símbolo]]  FOREIGN KEY (nombre_columna, ...)  [definición_referencia]
      |     CHANGE  [column]  nombre_anterior  definición_columna  [FIRST|AFTER nombre_columna]
      |     RENAME COLUMN  nombre_anterior  TO nuevo_nombre
      |     MODIFY definición_columna  [FIRST|AFTER nombre_columna]
      |     DROP COLUMN nombre_columna
      |     DROP PRIMARY KEY
      |     DROP FOREIGN KEY fk_símbolo
      |     opciones_tabla
```

```
ALTER TABLE nombre_tabla
            add fechaNacimiento DATE
;
```

- **Borrado** → elimina una tabla: DROP TABLE.

```
DROP TABLE nombre_tabla;
```

```
DROP TABLE ejem;
```

- **Renombrado** → reasigna un nombre a una tabla creada: RENAME TABLE.

```
RENAME TABLE nombre_tabla TO nuevo_nombre_tabla;
```

```
RENAME TABLE ejem TO ejemplo;
```

- **Consulta** → mostrar las tablas de una base de datos: SHOW TABLES.

```
SHOW TABLES;
```

- **Consultar la estructura** → mostrar la estructura de una tabla: DESCRIBE.

```
DESCRIBE nombre_tabla;
```

```
DESCRIBE ejem;
```

4. Tipos de datos.

Los tipos de datos que pueden usarse en MySQL son los siguientes:

- Numéricos.

Tipo	Naturaleza	Tamaño
tinyint [unsigned]	Entero	1 byte
smallint [unsigned]	Entero	2 bytes
mediumint [unsigned]	Entero	3 bytes
int [unsigned]	Entero	4 bytes
integer [unsigned]	Entero	4 bytes
bigint [unsigned]	Entero	8 bytes
float [unsigned]	Real aproximado	4 bytes
double [unsigned]	Real aproximado	8 bytes
decimal(longitud, decimales)	Real exacto	Variable
numeric(longitud, decimales)	Real exacto	Variable

- String.

Tipo	Naturaleza	Tamaño
char(longitud)	Caracteres	Longitud fija
varchar(longitud)	Caracteres	Longitud variable
tinyblob	Objetos binarios	Hasta 255 caracteres
blob	Objetos binarios	Hasta 65.535 caracteres
mediumblob	Objetos binarios	Hasta 16.777.215 caracteres
longblob	Objetos binarios	Hasta 4.294.967.298 caracteres
tinytext	Texto plano	Hasta 255 caracteres
text	Texto plano	Hasta 65.535 caracteres
mediumtext	Texto plano	Hasta 16.777.215 caracteres
longtext	Texto plano	Hasta 4.294.967.298 caracteres
set(valor1,valor2,...)	Conjuntos	Conjuntos de valores
enum(valor1, valor2, ...)	Enumeraciones	Lista de valores

- De fecha.

Tipo	Naturaleza	Tamaño
date	Fecha	‘aaaa-mm-dd’
time	Hora	‘hh:mm:ss’
timestamp	Fecha y hora	‘aaaa-mm-dd hh:mm:ss’
datetime	Fecha y hora	‘aaaa-mm-dd hh:mm:ss’
year	Año	‘aaaa’

5. Restricciones.

Las restricciones (o constraints) en SQL definen condiciones o reglas que han de cumplir los elementos de las tablas, evitando la inserción de datos incorrectos y garantizando que el tipo de datos sea el correcto.

Las restricciones se pueden aplicar a una columna o a toda una tabla.

Las restricciones son las siguientes:

- NOT NULL → para que la columna tenga valores nulos.
- UNIQUE → todos los valores de la columna son distintos.
- PRIMARY KEY → las restricciones NOT NULL y UNIQUE permiten identificar de forma unívoca cada fila de la tabla.
- FOREIGN KEY → permite detectar o identificar de forma unívoca una fila o registro de otra tabla.
- CHECK → permite asegurar que todos los valores cumplen una condición.
- DEFAULT → valor concreto que tendrá una columna si no se especifica otro.
- INDEX → útil para recuperar y crear datos de forma rápida en la base de datos.

5.1. Restricción NOT NULL

La restricción NOT NULL asegura que todos los valores de una columna NO pueden tomar valor nulo.

Si queremos reflejar que el nombre de los socios NO puede ser nulo:

Si queremos reflejar que el nombre de los socios NO se puede repetir:

```
Create Table Socio(
    ID int,
    Nombre varchar(20) NOT NULL,
    Apellido varchar(30)
);
```

5.2. Restricción UNIQUE

La restricción UNIQUE asegura que todos los valores de una columna son diferentes.

Si queremos reflejar que el nombre de los socios NO se puede repetir:

```
Create Table Socio(
    ID int,
    Nombre varchar(20) UNIQUE,
    Apellido varchar(30)
);
```

```
Create Table Socio(
    ID int,
    Nombre varchar(20),
    Apellido varchar(30),
    Constraint Unique( Nombre )
);
```

5.3. Restricción PRIMARY KEY

La restricción de clave primaria (**Primary Key Constraint**) identifica de forma unívoca cada tupla de una tabla. Esta restricción IMPLICA AUTOMÁTICAMENTE una restricción UNIQUE

Los valores de las claves primarias deben contener **valores únicos** y no pueden tener valor **NULL**.

Si queremos reflejar que el ID de los socios es la clave primaria:

```
Create Table Socio(
    ID int PRIMARY KEY,
    Nombre varchar(20),
    Apellido varchar(30)
);
```

```
Create Table Socio(
    ID int,
    Nombre varchar(20),
    Apellido varchar(30),
    Primary Key( ID )
);
```

5.4. Restricción FOREIGN KEY

La restricción de clave externa (Foreign Key) es para establecer un vínculo entre dos tablas, siendo un campo (o colección de campos) en la tabla que hace referencia a la clave primaria de la otra tabla:

- La tabla con la clave externa se denomina **tabla secundaria**.
- La tabla con la clave candidata se denomina **tabla principal o referenciada**.

Si queremos reflejar que el ID de los socios es la clave primaria:

```
Create Table Socio(
    ID int PRIMARY KEY,
    Nombre varchar(20) FOREIGN KEY
                References Datos(Nombre),
    Apellido varchar(30)
);
```

```
Create Table Socio(
    ID int,
    Nombre varchar(20),
    Apellido varchar(30),
    Primary Key( ID ),
    Foreign Key( Nombre) references Datos( Nombre )
);
```

5.5. Restricción CHECK

La restricción CHECK se usa para limitar el rango de valor de una columna:

- Si se define en una única columna → restringe los valores de esa columna.
- Si se define en una tabla → restringe los valores de todas las columnas.

Si queremos reflejar que el nombre de los socios no puede quedar vacío:

```
Create Table Socio(
    ID int,
    Nombre varchar(20),
    Apellido varchar(30),
    check ( Nombre != '')
);
```

5.6. Restricción DEFAULT

La restricción Default se usa para proporcionar un valor por defecto.

Si queremos reflejar que el nombre de los socios va a tomar AA ese valor por defecto:

```
Create Table Socio(
    ID int DEFAULT 'AA',
    Nombre varchar(20),
    Apellido varchar(30)
);
```

5.7. Restricción INDEX

La restricción INDEX sirve para crear índices en las tablas.

Si queremos reflejar que el nombre de los socios va a tomar AA ese valor por defecto:

```
Create Table Socio(
      ID int,
      Nombre varchar(20),
      Apellido varchar(30)
);

Create Index indice ON Socio(firstName);
```