

Sentencias DDL

Las **sentencias DDL** (Data Definition Language), permiten manejar la estructura de la base de datos. Se utiliza para crear, modificar o borrar objetos de la estructura de la base de datos (tablas, vistas, índices).

Tenemos los siguientes tipos de sentencias DDL:

- **CREATE** - permite crear objetos en la base de datos.
- **ALTER** - modifica la estructura de la base de datos.
- **DROP** - borra objetos de la base de datos.
- **RENAME** - renombra objetos.

1. Sentencia CREATE

La sentencia **CREATE** permite crear objetos de base de datos. Los tipos más habituales son los siguientes:

- **CREATE DATABASE/SCHEMA** - permite crear el contenedor (base de datos o esquema).
- **CREATE TABLE** - permite crear tablas en la base de datos.
- **CREATE VIEW** - permite crear vistas de base de datos.
- **CREATE INDEX** - permite crear índices de base de datos.
- **CREATE PARTITION** - permite crear particiones.

Una parte fundamental de la creación de la tabla es la asignación de los tipos de datos de los campos. Tenemos los siguientes tipos de datos en MySQL:

BINARY - Se puede almacenar cualquier tipo de datos en un campo de este tipo. Los datos no se traducen (por ejemplo, a texto). La forma en que se introducen los datos en un campo binario indica cómo aparecerán al mostrarlos.

BIT - Valores Sí y No, y campos que contienen solamente uno de dos valores.

TINYINT - Un número entero entre 0 y 255.

COUNTER - Se utiliza para campos contadores cuyo valor se incrementa automáticamente al crear un nuevo registro.

MONEY - Un número entero comprendido entre - 922.337.203.685.477,5808 y 922.337.203.685.477,5807.

DATETIME - Un valor de fecha u hora entre los años 100 y 9999.

UNIQUEIDENTIFIER - Un número de identificación único utilizado con llamadas a procedimientos remotos.

DECIMAL - Un tipo de datos numérico exacto con valores comprendidos entre 1028 - 1 y - 1028 - 1. Puede definir la precisión (1 - 28) y la escala (0 - precisión definida). La precisión y la escala predeterminadas son 18 y 0, respectivamente.

REAL - Un valor de coma flotante de precisión simple con un intervalo comprendido entre - 3,402823E38 y - 1,401298E45 para valores negativos, y desde 1,401298E-45 a 3,402823E38 para valores positivos, y 0.

FLOAT - Un valor de coma flotante de precisión doble con un intervalo comprendido entre - 1,79769313486232E308 y - 4,94065645841247E-324 para valores negativos, y desde 4,94065645841247E-324 a 1,79769313486232E308 para valores positivos, y 0.

SMALLINT - Un entero corto entre - 32.768 y 32.767.

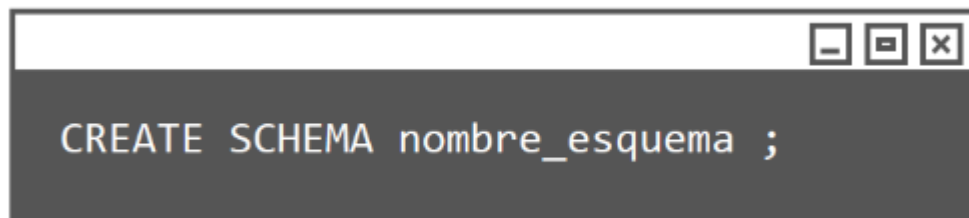
INTEGER - Un entero largo entre - 2.147.483.648 y 2.147.483.647.

IMAGE - Desde cero hasta un máximo de 2.14 gigabytes. Se utiliza para objetos OLE.

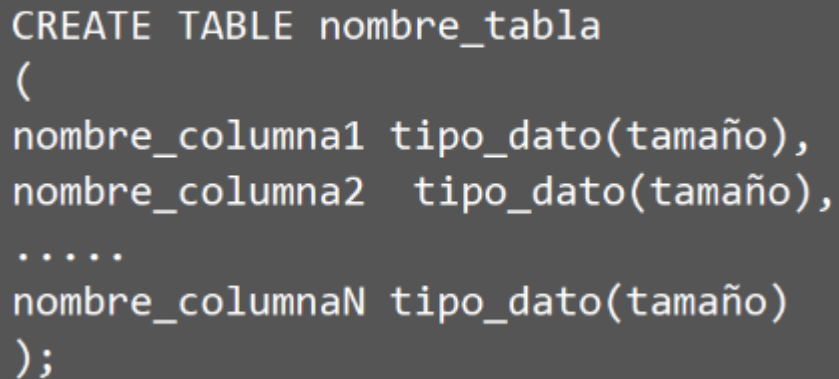
TEXT - Desde cero hasta un máximo de 2.14 gigabytes.

CHAR - Desde cero a 255 caracteres.

Sintaxis para la creación de un esquema:

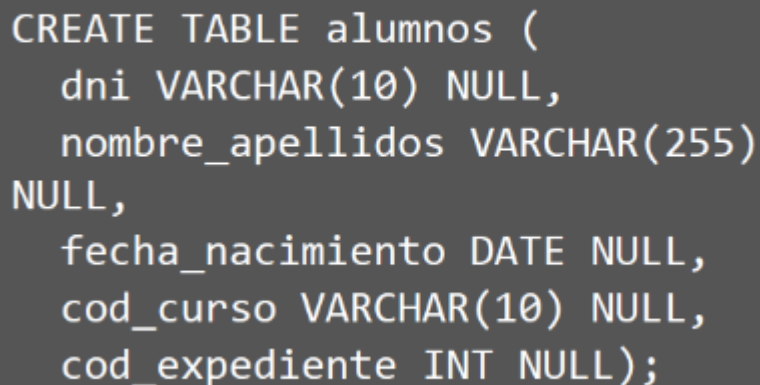
A screenshot of a SQL command window. The window has a title bar with standard minimize, maximize, and close buttons. The main area is dark gray with white text displaying the SQL command: `CREATE SCHEMA nombre_esquema ;`

Sintaxis para la creación de una tabla:



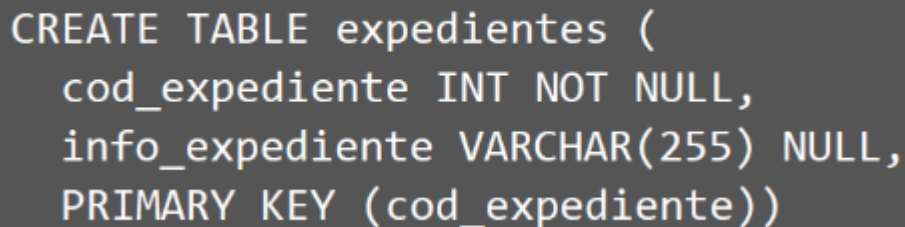
```
CREATE TABLE nombre_tabla
(
nombre_columna1 tipo_dato(tamaño),
nombre_columna2  tipo_dato(tamaño),
.....
nombre_columnaN tipo_dato(tamaño)
);
```

Ejemplo de creación de tabla alumnos en MySQL:



```
CREATE TABLE alumnos (
  dni VARCHAR(10) NULL,
  nombre_apellidos VARCHAR(255)
NULL,
  fecha_nacimiento DATE NULL,
  cod_curso VARCHAR(10) NULL,
  cod_expediente INT NULL);
```

Ejemplo de creación de tabla expedientes incluyendo clave primaria:



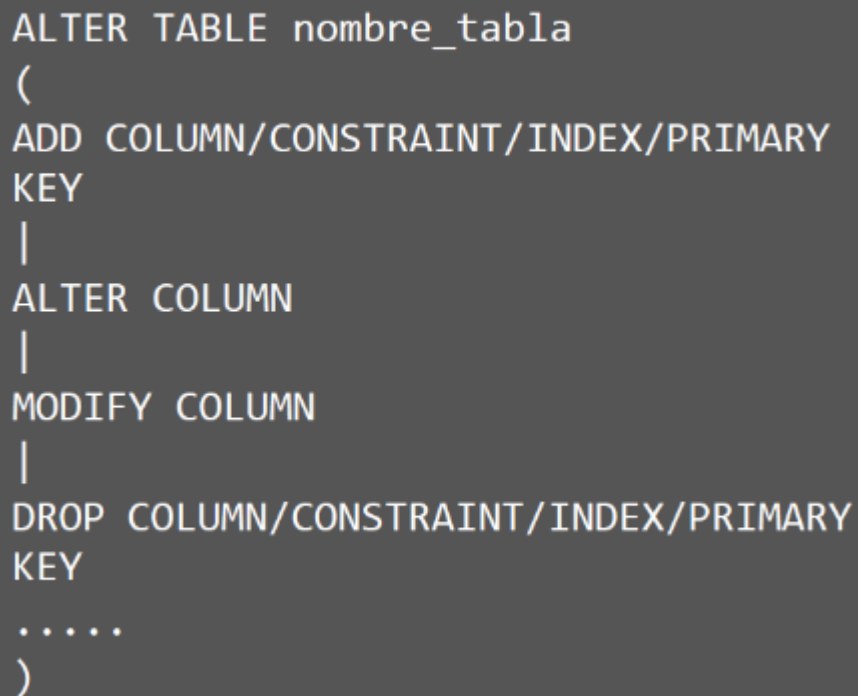
```
CREATE TABLE expedientes (
  cod_expediente INT NOT NULL,
  info_expediente VARCHAR(255) NULL,
  PRIMARY KEY (cod_expediente))
```

2. Sentencias ALTER

La sentencia **ALTER** permite realizar modificaciones sobre los objetos de base de datos. En una sentencia alter table podemos realizar las siguientes acciones:

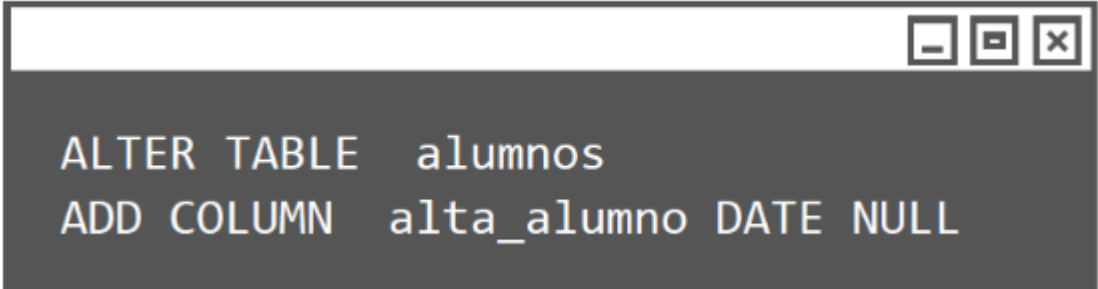
- **ADD COLUMN**: añadir columna a la tabla.
- **MODIFY COLUMN**: modificar una columna de la tabla.
- **DROP COLUMN**: borrar una columna de la tabla.
- **ADD CONSTRAINT**: añadir una constraint a la tabla (clave primaria o clave foránea).
- **DROP CONSTRAINT**: eliminar una constraint de la tabla.

Sintaxis para la modificación de una tabla:



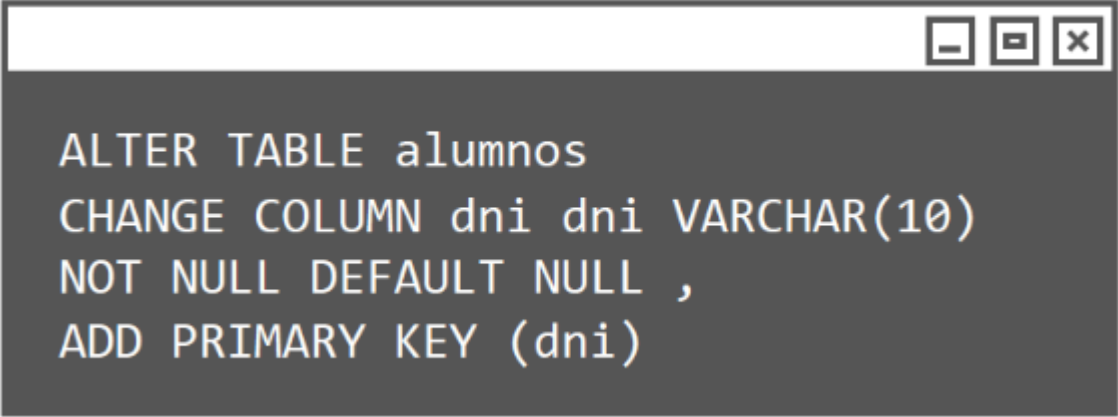
```
ALTER TABLE nombre_tabla
(
ADD COLUMN/CONSTRAINT/INDEX/PRIMARY
KEY
|
ALTER COLUMN
|
MODIFY COLUMN
|
DROP COLUMN/CONSTRAINT/INDEX/PRIMARY
KEY
.....
)
```

Creación de un campo nuevo en la tabla alumnos:



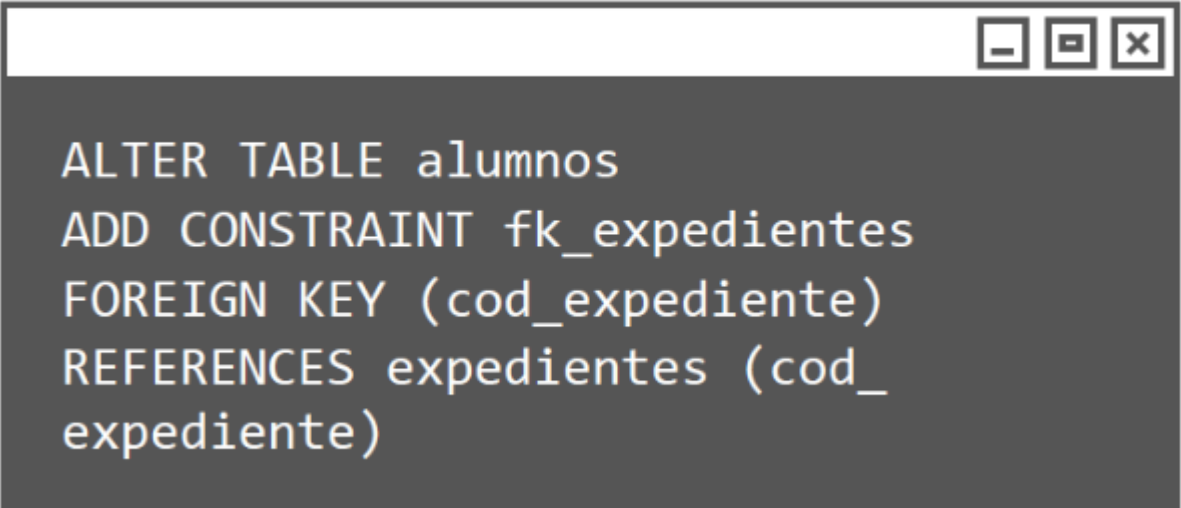
```
ALTER TABLE  alumnos
ADD COLUMN  alta_alumno DATE NULL
```

Creación de clave primaria sobre tabla alumnos:



```
ALTER TABLE alumnos
CHANGE COLUMN dni dni VARCHAR(10)
NOT NULL DEFAULT NULL ,
ADD PRIMARY KEY (dni)
```

Creación de clave foránea de tabla alumnos a tabla expedientes:



```
ALTER TABLE alumnos
ADD CONSTRAINT fk_expedientes
FOREIGN KEY (cod_expediente)
REFERENCES expedientes (cod_
expediente)
```

3. Sentencias DROP

La sentencia **DROP** permite borrar objetos de base de datos.

Ejemplos:

- **DROP TABLE**: borra la tabla.
- **DROP VIEW**: borra la vista.
- **DROP INDEX**: borra el índice.

La sintaxis para el borrado:



A terminal window with a dark gray background and a light gray title bar. The title bar contains three window control icons (minimize, maximize, close) on the right. The main area of the terminal displays the SQL command `DROP TABLE/VIEW/INDEX nombre_objeto` in a light blue monospace font.

```
DROP TABLE/VIEW/INDEX nombre_objeto
```

Borrado de columna de la tabla alumnos:



A terminal window with a dark gray background and a light gray title bar. The title bar contains three window control icons (minimize, maximize, close) on the right. The main area of the terminal displays the SQL command `DROP TABLE alumnos` in a light blue monospace font.

```
DROP TABLE alumnos
```