3 Manipulación de tablas

3.1 Crear una tabla

A continuación se muestra una **versión simplificada** de la sintaxis necesaria para la creación de una tabla en MySQL.

Para una definición más exhaustiva, puede consultar la sintaxis de creación de tablas en la documentación oficial de MySQL.

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...)
    [table_options]

create_definition:
    col_name column_definition
  | [CONSTRAINT [symbol]] PRIMARY KEY (index_col_name,...)
  | [CONSTRAINT [symbol]] FOREIGN KEY (index_col_name,...) reference_definition
  | CHECK (expr)

column_definition:
    data_type [NOT NULL | NULL] [DEFAULT default_value]
      [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]

reference_definition:
    REFERENCES tbl_name (index_col_name,...)
      [ON DELETE reference_option]
      [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

table_options:
    table_option [[,] table_option] ...

table_option:
    AUTO_INCREMENT [=] value
  | [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
  | ENGINE [=] engine_name
```

3.1.1 Restricciones sobre las columnas de la tabla

Podemos aplicar las siguientes restricciones sobre las columnas de la tabla:

- `NOT NULL` o `NULL`: Indica si la columna permite almacenar valores nulos o no.
- `DEFAULT`: Permite indicar un valor inicial por defecto si no especificamos ninguno en la inserción.
- `AUTO_INCREMENT`: Sirve para indicar que es una columna autonumérica. Su valor se incrementa automáticamente en cada inserción de una fila. Sólo se utiliza en campos de tipo entero.
- `UNIQUE KEY`: Indica que el valor de la columna es único y no pueden aparecer dos valores iguales en la misma columna.
- `PRIMARY KEY`: Para indicar que una columna o varias son clave primaria.
- `CHECK`: Nos permite realizar restricciones sobre una columna. En las versiones previas a MySQL 8.0 estas restricciones no se aplicaban, sólo se parseaba la sintaxis pero eran ignoradas por el sistema gestor de base de datos. A partir de la versión de MySQL 8.0 ya sí se aplican las restricciones definidas con `CHECK`.

Ejemplo 1:

```sql
DROP DATABASE IF EXISTS proveedores;
CREATE DATABASE proveedores CHARSET utf8mb4;
USE proveedores;

CREATE TABLE categoria (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL
);

CREATE TABLE pieza (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  color VARCHAR(50) NOT NULL,
  precio DECIMAL(7,2) NOT NULL CHECK (precio > 0),
  id_categoria INT UNSIGNED NOT NULL,
  FOREIGN KEY (id_categoria) REFERENCES categoria(id)
);
```

Ejemplo 2:

```sql
DROP DATABASE IF EXISTS agencia;
CREATE DATABASE agencia CHARSET utf8mb4;
USE agencia;

CREATE TABLE turista (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  apellidos VARCHAR(100) NOT NULL,
  direccion VARCHAR(100) NOT NULL,
  telefono VARCHAR(9) NOT NULL
);

CREATE TABLE hotel (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  direccion VARCHAR(100) NOT NULL,
  ciudad VARCHAR(25) NOT NULL,
  plazas INTEGER NOT NULL,
  telefono VARCHAR(9) NOT NULL
);

```

```sql
CREATE TABLE reserva (
  id_turista INT UNSIGNED NOT NULL,
  id_hotel INT UNSIGNED NOT NULL,
  fecha_entrada DATETIME NOT NULL,
  fecha_salida DATETIME NOT NULL,
  regimen ENUM('MP', 'PC'),
  PRIMARY KEY (id_turista,id_hotel),
  FOREIGN KEY (id_turista) REFERENCES turista(id),
  FOREIGN KEY (id_hotel) REFERENCES hotel(id)
);
```

3.1.2 Opciones en la declaración de claves ajenas (`FOREIGN KEY`)

- `ON DELETE` y `ON UPDATE`: Nos permiten indicar el efecto que provoca el borrado o la actualización de los datos que están referenciados por claves ajenas. Las opciones que podemos especificar son las siguientes:
  - `RESTRICT`: Impide que se puedan actualizar o eliminar las filas que tienen valores referenciados por claves ajenas. Es la opción por defecto en MySQL.
  - `CASCADE`: Permite actualizar o eliminar las filas que tienen valores referenciados por claves ajenas.
  - `SET NULL`: Asigna el valor `NULL` a las filas que tienen valores referenciados por claves ajenas.
  - `NO ACTION`: Es una palabra clave del estándar SQL. En MySQL es equivalente a `RESTRICT`.
  - `SET DEFAULT`: No es posible utilizar esta opción cuando trabajamos con el motor de almacenamiento **InnoDB**. Puedes encontrar más información en la documentación oficial de MySQL.

Ejemplo 1:

```sql
DROP DATABASE IF EXISTS proveedores;
CREATE DATABASE proveedores CHARSET utf8mb4;
USE proveedores;

CREATE TABLE categoria (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL
);

CREATE TABLE pieza (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  color VARCHAR(50) NOT NULL,
  precio DECIMAL(7,2) NOT NULL,
  id_categoria INT UNSIGNED NOT NULL,
  FOREIGN KEY (id_categoria) REFERENCES categoria(id)
  ON DELETE RESTRICT
  ON UPDATE RESTRICT
);

INSERT INTO categoria VALUES (1, 'Categoria A');
INSERT INTO categoria VALUES (2, 'Categoria B');
INSERT INTO categoria VALUES (3, 'Categoria C');

INSERT INTO pieza VALUES (1, 'Pieza 1', 'Blanco', 25.90, 1);
```

```
INSERT INTO pieza VALUES (2, 'Pieza 2', 'Verde', 32.75, 1);
INSERT INTO pieza VALUES (3, 'Pieza 3', 'Rojo', 12.00, 2);
INSERT INTO pieza VALUES (4, 'Pieza 4', 'Azul', 24.50, 2);
```

- ¿Podríamos borrar la **Categoría A** de la tabla **categoria**?
- ¿Y la **Categoría C**?
- ¿Podríamos actualizar la **Categoría A** de la tabla **categoria**?

Ejemplo 2:

```
DROP DATABASE IF EXISTS proveedores;
CREATE DATABASE proveedores CHARSET utf8mb4;
USE proveedores;

CREATE TABLE categoria (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL
);

CREATE TABLE pieza (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  color VARCHAR(50) NOT NULL,
  precio DECIMAL(7,2) NOT NULL,
  id_categoria INT UNSIGNED NOT NULL,
  FOREIGN KEY (id_categoria) REFERENCES categoria(id)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);

INSERT INTO categoria VALUES (1, 'Categoria A');
INSERT INTO categoria VALUES (2, 'Categoria B');
INSERT INTO categoria VALUES (3, 'Categoria C');

INSERT INTO pieza VALUES (1, 'Pieza 1', 'Blanco', 25.90, 1);
INSERT INTO pieza VALUES (2, 'Pieza 2', 'Verde', 32.75, 1);
INSERT INTO pieza VALUES (3, 'Pieza 3', 'Rojo', 12.00, 2);
INSERT INTO pieza VALUES (4, 'Pieza 4', 'Azul', 24.50, 2);
```

- ¿Podríamos borrar la **Categoría A** de la tabla **categoria**?
- ¿Qué le ocurre a las piezas que pertenecen la **Categoría A** después de borrarla?
- ¿Podríamos actualizar la **Categoría A** de la tabla **categoria**?
- ¿Qué le ocurre a las piezas que pertenecen la **Categoría A** después de actualizarla?

Ejemplo 3:

```
DROP DATABASE IF EXISTS proveedores;
CREATE DATABASE proveedores CHARSET utf8mb4;
USE proveedores;

CREATE TABLE categoria (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL
);

```

```sql
CREATE TABLE pieza (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  color VARCHAR(50) NOT NULL,
  precio DECIMAL(7,2) NOT NULL,
  id_categoria INT UNSIGNED,
  FOREIGN KEY (id_categoria) REFERENCES categoria(id)
  ON DELETE SET NULL
  ON UPDATE SET NULL
);

INSERT INTO categoria VALUES (1, 'Categoria A');
INSERT INTO categoria VALUES (2, 'Categoria B');
INSERT INTO categoria VALUES (3, 'Categoria C');

INSERT INTO pieza VALUES (1, 'Pieza 1', 'Blanco', 25.90, 1);
INSERT INTO pieza VALUES (2, 'Pieza 2', 'Verde', 32.75, 1);
INSERT INTO pieza VALUES (3, 'Pieza 3', 'Rojo', 12.00, 2);
INSERT INTO pieza VALUES (4, 'Pieza 4', 'Azul', 24.50, 2);
```

- ¿Podríamos borrar la **Categoría A** de la tabla **categoria**?
- ¿Qué le ocurre a las piezas que pertenecen la **Categoría A** después de borrarla?
- ¿Podríamos actualizar la **Categoría A** de la tabla **categoria**?
- ¿Qué le ocurre a las piezas que pertenecen la **Categoría A** después de actualizarla?

3.1.3 Opciones a tener en cuenta en la creación de las tablas

Algunas de las opciones que podemos indicar durante la creación de las tablas son las siguientes:

- `AUTO_INCREMENT`: Aquí podemos indicar el valor inicial que vamos a usar en el campo definido como `AUTO_INCREMENT`.
- `CHARACTER SET`: Especifica el set de caracteres que vamos a utilizar en la tabla.
- `COLLATE`: Especifica el tipo de cotejamiento que vamos a utilizar en la tabla.
- `ENGINE`: Especifica el motor de almacenamiento que vamos a utilizar para la tabla. Los más habituales en MySQL son **InnoDB** y **MyISAM**. Por defecto las tablas se crean con el motor **InnoDB**

Para conocer todas las opciones posibles podemos consultar la sintaxis de creación de tablas en la documentación oficial de MySQL. Con el objetivo de simplificar la creación de tablas solamente hemos enumerado las opciones con las que vamos a trabajar durante el curso.

En la documentación oficial de MySQL podemos encontrar más información sobre los diferentes motores de almacenamiento disponibles en MySQL.

Ejemplo:

```sql
DROP DATABASE IF EXISTS proveedores;
CREATE DATABASE proveedores CHARSET utf8mb4;
USE proveedores;

CREATE TABLE categoria (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
```

```
  nombre VARCHAR(100) NOT NULL
)  ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1000;

CREATE TABLE pieza (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  color VARCHAR(50) NOT NULL,
  precio DECIMAL(7,2) NOT NULL,
  id_categoria INT UNSIGNED NOT NULL,
  FOREIGN KEY (id_categoria) REFERENCES categoria(id)
)  ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1000;
```

En este ejemplo se ha seleccionado para cada una de las tablas las siguientes opciones de configuración: **InnoDB** como motor de base de datos, **utf8** como el set de caracteres y el valor **1000** como valor inicial para las columnas de tipo **AUTO_INCREMENT**.

3.2 Eliminar una tabla

```
DROP [TEMPORARY] TABLE [IF EXISTS] nombre_tabla [, nombre_tabla];
```

Ejemplos:

```
DROP TABLE nombre_tabla;
```

```
DROP TABLE IF EXISTS nombre_tabla;
```

```
DROP TABLE nombre_tabla_1, nombre_tabla_2;
```

3.3 Modificar una tabla

En muchas ocasiones es necesario modificar los atributos de una tabla, añadir nuevos campos o eliminar otros. Si la tabla no tiene datos podemos eliminar la tabla y volver a crearla, pero si se trata de una tabla que ya contiene datos tenemos que hacer uso de la sentencia `ALTER TABLE`.

A continuación se muestra la sintaxis necesaria para la modificación de una tabla en MySQL.

Puede consultar la sintaxis de modificación de tablas en la documentación oficial de MySQL.

```
ALTER TABLE tbl_name
    [alter_specification [, alter_specification] ...]
    [partition_options]

alter_specification:
    table_options
  | ADD [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
  | ADD [COLUMN] (col_name column_definition,...)
  | ADD {INDEX|KEY} [index_name]
        [index_type] (index_col_name,...) [index_option] ...
```

```
    | ADD [CONSTRAINT [symbol]] PRIMARY KEY
          [index_type] (index_col_name,...) [index_option] ...
    | ADD [CONSTRAINT [symbol]]
          UNIQUE [INDEX|KEY] [index_name]
          [index_type] (index_col_name,...) [index_option] ...
    | ADD FULLTEXT [INDEX|KEY] [index_name]
          (index_col_name,...) [index_option] ...
    | ADD SPATIAL [INDEX|KEY] [index_name]
          (index_col_name,...) [index_option] ...
    | ADD [CONSTRAINT [symbol]]
          FOREIGN KEY [index_name] (index_col_name,...)
          reference_definition
    | ALGORITHM [=] {DEFAULT|INPLACE|COPY}
    | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
    | CHANGE [COLUMN] old_col_name new_col_name column_definition
          [FIRST|AFTER col_name]
    | [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]
    | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
    | {DISABLE|ENABLE} KEYS
    | {DISCARD|IMPORT} TABLESPACE
    | DROP [COLUMN] col_name
    | DROP {INDEX|KEY} index_name
    | DROP PRIMARY KEY
    | DROP FOREIGN KEY fk_symbol
    | FORCE
    | LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
    | MODIFY [COLUMN] col_name column_definition
          [FIRST | AFTER col_name]
    | ORDER BY col_name [, col_name] ...
    | RENAME {INDEX|KEY} old_index_name TO new_index_name
    | RENAME [TO|AS] new_tbl_name
    | {WITHOUT|WITH} VALIDATION
    | ADD PARTITION (partition_definition)
    | DROP PARTITION partition_names
    | DISCARD PARTITION {partition_names | ALL} TABLESPACE
    | IMPORT PARTITION {partition_names | ALL} TABLESPACE
    | TRUNCATE PARTITION {partition_names | ALL}
    | COALESCE PARTITION number
    | REORGANIZE PARTITION partition_names INTO (partition_definitions)
    | EXCHANGE PARTITION partition_name WITH TABLE tbl_name [{WITH|WITHOUT}
      VALIDATION]
    | ANALYZE PARTITION {partition_names | ALL}
    | CHECK PARTITION {partition_names | ALL}
    | OPTIMIZE PARTITION {partition_names | ALL}
    | REBUILD PARTITION {partition_names | ALL}
    | REPAIR PARTITION {partition_names | ALL}
    | REMOVE PARTITIONING
    | UPGRADE PARTITIONING

index_col_name:
    col_name [(length)] [ASC | DESC]

index_type:
    USING {BTREE | HASH}

index_option:
    KEY_BLOCK_SIZE [=] value
```

```
    | index_type
    | WITH PARSER parser_name
    | COMMENT 'string'

table_options:
    table_option [[,] table_option] ...

table_option:
    AUTO_INCREMENT [=] value
  | AVG_ROW_LENGTH [=] value
  | [DEFAULT] CHARACTER SET [=] charset_name
  | CHECKSUM [=] {0 | 1}
  | [DEFAULT] COLLATE [=] collation_name
  | COMMENT [=] 'string'
  | COMPRESSION [=] {'ZLIB'|'LZ4'|'NONE'}
  | CONNECTION [=] 'connect_string'
  | {DATA|INDEX} DIRECTORY [=] 'absolute path to directory'
  | DELAY_KEY_WRITE [=] {0 | 1}
  | ENCRYPTION [=] {'Y' | 'N'}
  | ENGINE [=] engine_name
  | INSERT_METHOD [=] { NO | FIRST | LAST }
  | KEY_BLOCK_SIZE [=] value
  | MAX_ROWS [=] value
  | MIN_ROWS [=] value
  | PACK_KEYS [=] {0 | 1 | DEFAULT}
  | PASSWORD [=] 'string'
  | ROW_FORMAT [=] {DEFAULT|DYNAMIC|FIXED|COMPRESSED|REDUNDANT|COMPACT}
  | STATS_AUTO_RECALC [=] {DEFAULT|0|1}
  | STATS_PERSISTENT [=] {DEFAULT|0|1}
  | STATS_SAMPLE_PAGES [=] value
  | TABLESPACE tablespace_name [STORAGE {DISK|MEMORY|DEFAULT}]
  | UNION [=] (tbl_name[,tbl_name]...)

partition_options:
    (see CREATE TABLE options)
```

3.3.1 Ejemplo de ALTER TABLE <tbl_name> MODIFY

`MODIFY` nos permite modificar el tipo de dato de una columna y sus atributos.

Suponemos que tenemos la siguiente tabla creada

```
CREATE TABLE usuario (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(25)
);
```

Y queremos modificar las columna `nombre` para que pueda almacenar 50 caracteres y además que sea `NOT NULL`. En este caso usaríamos la sentencia:

```
ALTER TABLE usuario MODIFY nombre VARCHAR(50) NOT NULL;
```

Después de ejecutar esta sentencia la tabla quedaría así:

```sql
CREATE TABLE usuario (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL
);
```

3.3.2 Ejemplo de ALTER TABLE <tbl_name> CHANGE

`CHANGE` nos permite renombrar una columna, modificar el tipo de dato de una columna y sus atributos.

Suponemos que tenemos la siguiente tabla creada

```sql
CREATE TABLE usuario (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre_de_usuario VARCHAR(25)
);
```

Y queremos renombrar el nombre de la columna `nombre_de_usuario` como `nombre`, que pueda almacenar 50 caracteres y además que sea `NOT NULL`. En este caso usaríamos la sentencia:

```sql
ALTER TABLE usuario CHANGE nombre_de_usuario nombre VARCHAR(50) NOT NULL;
```

Después de ejecutar esta sentencia la tabla quedaría así:

```sql
CREATE TABLE usuario (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL
);
```

3.3.3 Ejemplo de ALTER TABLE <tbl_name> ALTER

`ALTER` nos permite asignar un valor por defecto a una columna o eliminar el valor por defecto que tenga establecido.

Suponemos que tenemos la siguiente tabla creada

```sql
CREATE TABLE usuario (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  rol ENUM('Estudiante', 'Profesor') NOT NULL
);
```

Y queremos que el valor por defecto de la columna `rol` sea `Estudiante`. En este caso usaríamos la sentencia:

```sql
ALTER TABLE usuario ALTER rol SET DEFAULT 'Estudiante';
```

Después de ejecutar esta sentencia la tabla quedaría así:

```sql
CREATE TABLE usuario (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
```

```
  rol ENUM('Estudiante', 'Profesor') NOT NULL DEFAULT 'Estudiante'
);
```

Si ahora quisiéramos eliminar el valor por defecto de la columna `rol`, usaríamos la siguiente sentencia:

```
ALTER TABLE usuario ALTER rol DROP DEFAULT;
```

3.3.4 Ejemplo de ALTER TABLE <tbl_name> ADD

`ADD` nos permite añadir nuevas columnas a una tabla. Con los modificadores `FIRST` y `AFTER` podemos elegir el lugar de la tabla donde queremos insertar la nueva columna. `FIRST` coloca la nueva columna en primer lugar y `AFTER` la colocaría detrás de la columna que se especifique. Si no se especifica nada la nueva columna se añadiría detrás de la última columna de la tabla.

Suponemos que tenemos la siguiente tabla creada

```
CREATE TABLE usuario (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  rol ENUM('Estudiante', 'Profesor') NOT NULL
);
```

Y queremos añadir la columna `fecha_nacimiento` de tipo `DATE`:

```
ALTER TABLE usuario ADD fecha_nacimiento DATE NOT NULL;
```

En este caso la nueva columna se ha añadido detrás de la última columna, `rol`. La tabla quedaría así:

```
CREATE TABLE usuario (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  rol ENUM('Estudiante', 'Profesor') NOT NULL,
  fecha_nacimiento DATE NOT NULL
);
```

Suponemos que ahora queremos añadir las columnas `apellido1` y `apellido2` detrás de la columna `nombre`.

```
ALTER TABLE usuario ADD apellido1 VARCHAR(50) NOT NULL AFTER nombre;

ALTER TABLE usuario ADD apellido2 VARCHAR(50) AFTER apellido1;
```

Después de ejecutar todas las sentencias la tabla quedaría así:

```
CREATE TABLE usuario (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  apellido1 VARCHAR(50) NOT NULL,
  apellido2 VARCHAR(50),
  rol ENUM('Estudiante', 'Profesor') NOT NULL DEFAULT 'Estudiante',
  fecha_nacimiento DATE NOT NULL
);
```

3.3.5 Ejemplo de ALTER TABLE <tbl_name> DROP

DROP nos permite eliminar una columna de la tabla.

Suponemos que tenemos la siguiente tabla creada

```
CREATE TABLE usuario (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  apellido1 VARCHAR(50) NOT NULL,
  apellido2 VARCHAR(50),
  rol ENUM('Estudiante', 'Profesor') NOT NULL DEFAULT 'Estudiante',
  fecha_nacimiento DATE NOT NULL
);
```

Y queremos eliminar la columna `fecha_nacimiento`. En este caso usaríamos la sentencia:

```
ALTER TABLE usuario DROP fecha_nacimiento;
```

Después de ejecutar esta sentencia la tabla quedaría así:

```
CREATE TABLE usuario (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  apellido1 VARCHAR(50) NOT NULL,
  apellido2 VARCHAR(50),
  rol ENUM('Estudiante', 'Profesor') NOT NULL DEFAULT 'Estudiante'
);
```

3.4 Consultar el listado de tablas disponibles

```
SHOW TABLES;
```

Muestra un listado de todas las tablas que existen en la base de datos con la que estamos trabajando.

3.5 Mostrar información sobre la estructura de una tabla

```
DESCRIBE nombre_tabla;
```

También podemos utilizar:

```
DESC nombre_tabla;
```

Esta sentencia se utiliza para mostrar información sobre la estructura de una tabla.

3.6 Mostrar la sentencia SQL de creación de una tabla

```sql
SHOW CREATE TABLE nombre_tabla;
```

Se puede utilizar para visualizar la sentencia SQL que sería necesaria ejecutar para crear la tabla que le estamos indicando como parámetro.

4 Tipos de datos

4.1 Números enteros

Tipo	Bytes	Mínimo	Máximo
`TINYINT`	1	-128	127
`TINYINT UNSIGNED`	1	0	255
`SMALLINT`	2	-32768	32767
`SMALLINT UNSIGNED`	2	0	65535
`MEDIUMINT`	3	-8388608	8388607
`MEDIUMINT UNSIGNED`	3	0	16777215
`INT`	4	-2147483648	2147483647
`INT UNSIGNED`	4	0	4294967295
`INTEGER`	4	-2147483648	2147483647
`INTEGER UNSIGNED`	4	0	4294967295
`BIGINT`	8	-9223372036854775808	9223372036854775807
`BIGINT UNSIGNED`	8	0	18446744073709551615

Puedes encontrar más información sobre números enteros en la documentación oficial de MySQL.

4.1.1 `ZEROFILL`

Todos los tipos de datos numéricos admiten el atributo `ZEROFILL`. Cuando asignamos este atributo a una columna también se le añade de forma automática el atributo `UNSIGNED`, de modo que el campo quedaría como `UNSIGNED ZEROFILL`.

4.1.2 Nota importante sobre INT(11)

`INT(11)` **no** quiere decir que queremos guardar un número entero de 11 cifras. El número indicado entre paréntesis **indica el ancho de la columna que ocupará dicho valor** y tiene utilidad cuando asignamos el atri-