

Tema 6

Lenguaje de descripción de datos (DDL).

Lenguaje de descripción (o definición) de datos¹ (DDL) sirve para crear, modificar y eliminar bases de datos, tablas, vistas, índices, triggers y procedimientos almacenados.

Nota:

Todo este tema está muy basado en el de [José Juan Sánchez](#) al que agradezco que comparta su trabajo.

No obstante se puede consultar la bibliografía completa al final del tema.

Sentencias DDL:

- CREATE: crear objetos de la BdD.
- DROP: eliminar objetos de la BdD.
- ALTER: modificar objetos de la BdD.
- SHOW: consultar objetos de la BdD.

También nos serán útiles:

- USE: seleccionar BdD sobre la que se aplicarán el resto de sentencias.
- DESCRIBE: mostrar información sobre la estructura de una tabla.

¹Data Definition Language.

6.1. Operaciones sobre bases de datos.

6.1.1. Crear una base de datos.

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
[create_option] ...

create_option: [DEFAULT] {
    CHARACTER SET [=] charset_name
    | COLLATE [=] collation_name
    | ENCRYPTION [=] {'Y' | 'N'}
}
```

Notas:

- DATABASE y SCHEMA son sinónimos.
- IF NOT EXISTS crea la base de datos sólo si no existe una base de datos con el mismo nombre.
- CHARACTER SET especifica la codificación que usaremos.
- COLLATE (cotejamiento) indica qué reglas se aplicarán a la hora de comparar cadenas de caracteres (ordenar).
- ENCRYPTION (a partir de MySQL 8.0.16) permite encriptar los datos.

```
CREATE database empleados;
```

```
CREATE database IF NOT EXISTS empleados CHARACTER SET utf8;
```

6.1.1.1. Operaciones asociadas.

```
SHOW DATABASES; -- muestra las bdd disponibles.

USE empleados; -- selecciona la bdd para el resto de sentencias.

SELECT DATABASE(); -- muestra la bdd seleccionada.

SHOW CREATE DATABASE nombre_base_datos; /*muestra la sentencia que
                                         necesitaríamos para crear la bdd indicada. */

SHOW CHARACTER SET;

SHOW COLLATION;
```

6.1.2. Eliminar una base de datos.

```
DROP {DATABASE | SCHEMA} [IF EXISTS] nombre_base_datos;
```

6.1.3. Modificar una base de datos.

```
ALTER {DATABASE | SCHEMA} [db_name]
alter_option ...

alter_option: {
    [DEFAULT] CHARACTER SET [=] charset_name
    | [DEFAULT] COLLATE [=] collation_name
    | [DEFAULT] ENCRYPTION [=] {'Y' | 'N'}
    | READ ONLY [=] {DEFAULT | 0 | 1}
}
```

READ ONLY (a partir de MySQL 8.0.22) especifica si se permite la modificación de la base de datos y los objetos que contiene.

```
ALTER DATABASE empleados CHARACTER SET utf8;
```

6.2. Operaciones sobre tablas.

6.2.1. Crear una tabla.

Versión simplificada de la sintaxis. La versión completa (demasiado compleja para este curso) se puede consultar [aquí](#).

```
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
```

El nombre de las tablas debe cumplir las siguientes reglas:

- Comenzar con una letra.

- Longitud máxima: 30 caracteres.
- Solo letras del alfabeto (inglés), números o el guion bajo (también el signo \$ y #, pero tienen un significado especial por lo que se recomienda no usarlos).
- Debe ser único en el mismo esquema.
- No puede ser ninguna palabra reservada de SQL.

6.2.1.1. Restricciones sobre las columnas de la tabla.

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

- `NULL` o `NOT NULL`: si puede almacenar valores nulos o no.
- `DEFAULT valor_por_defecto`: permite indicar un valor inicial por defecto si no especificamos ninguno en la inserción.
- `AUTO_INCREMENT`: columna autonumérica; su valor se incrementa automáticamente en cada inserción de una fila. Solo se utiliza en campos de tipo entero.
- `UNIQUE [KEY]`: indica que el valor de la columna debe ser único.
- `[PRIMARY] KEY`: Para indicar que una columna o varias son clave primaria. Implica las restricciones `UNIQUE` y `NOT NULL` implícitamente.
- `CHECK expresión`: permite establecer condiciones que deben cumplir los valores de la tabla que se introduzcan en dicha columna.²
- `FOREIGN KEY`: añade una clave ajena.

²En las versiones previas a MySQL 8.0 estas restricciones no se aplicaban, solo se cotejaba 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`.

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

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

CREATE TABLE pieza (
    codigo 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),
    codigo_categoria INT UNSIGNED NOT NULL,
    FOREIGN KEY (codigo_categoria) REFERENCES categoria(codigo)
);
```

```
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
);

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)
);
```

En el script de creación de la BdD *empleados* tienes más ejemplos.

Es importante destacar que se pueden asignar nombres a las restricciones (se considera más adecuado) y permite usarlas en operaciones posteriores.

```
CREATE TABLE empleados (  
    id_empleado INT NOT NULL,  
    nombre VARCHAR(20),  
    apellido VARCHAR(20) NOT NULL,  
    email VARCHAR(25) NOT NULL,  
    telefono VARCHAR(20),  
    fecha_contratacion date NOT NULL,  
    id_trabajo VARCHAR(10) NOT NULL,  
    salario DECIMAL(8,2),  
    comision DECIMAL(2,2),  
    id_director INT,  
    id_departamento INT,  
    CONSTRAINT empleados_pk PRIMARY KEY (id_empleado),  
    CONSTRAINT empleados_fk1 FOREIGN KEY (id_director)  
        REFERENCES empleados(id_empleado),  
    CONSTRAINT empleados_fk2 FOREIGN KEY (id_departamento)  
        REFERENCES departamentos(id_departamento),  
    CONSTRAINT empleados_fk3 FOREIGN KEY (id_trabajo)  
        REFERENCES trabajos(id_trabajo)  
);
```

IMPORTANTE: si una clave primaria está compuesta por más de un atributo, la clave ajena se creará referenciándolos a todos a la vez:

```
CONSTRAINT num_cuenta_fk1 FOREIGN KEY (codigo_entidad, codigo_oficina)  
    REFERENCES oficina (codigo_entidad, codigo_oficina)
```

Ejercicios 6.1

Escribe los scripts de creación correspondientes a los modelos relaciones obtenidos en los ejercicios:

1. Ejercicio 4.4.1.
2. Ejercicio 4.4.2.
3. Ejercicio 4.4.3.
4. Ejercicio 4.5.1.
5. Ejercicio 4.5.2.
6. Ejercicio 4.5.3.

7. Ejercicio 4.5.4.

6.2.1.2. Opciones en la declaración de claves ajenas.

ON DELETE y ON UPDATE: permiten indicar qué sucederá al borrar o la actualizar 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: si se elimina o actualiza una fila de la tabla, se eliminan o actualizan todas las filas coincidentes en las tablas que tengan valores referenciados por claves ajenas.
- SET NULL: si se elimina o actualiza una fila de la tabla, todas las columnas de las filas con valores referenciados por claves ajenas se establecen a NULL.
- NO ACTION: es una palabra clave del estándar SQL. En MySQL es equivalente a RESTRICT.
- SET DEFAULT: si se elimina o actualiza una fila de la tabla, todas las columnas de las filas con valores referenciados por claves ajenas se establecen al valor predeterminado de la columna. No es posible utilizar esta opción cuando trabajamos con los motores actuales de MySQL. Puedes encontrar más información en la [documentación](#) oficial.

Podemos actualizar la creación de *Proveedores*.

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

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

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

E insertar algunos datos (lo veremos en el tema siguiente):

```
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);
```

6.2.1.3. Opciones al crear la tabla.

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

- AUTO_INCREMENT: indicar el valor inicial.
- CHARSET: set de caracteres a utilizar.
- COLLATE: tipo de colación a utilizar.
- ENGINE: motor de almacenamiento que vamos a utilizar. Los más habituales en

MySQL son *InnoDB* y *MyISAM*. Por defecto las tablas se crean con el motor *InnoDB*

Nota:

No entraremos en detalles pero puedes obtener más información sobre los diferentes motores en la [documentación](#) oficial.

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

CREATE TABLE categoria (
    codigo INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1000;

CREATE TABLE pieza (
    codigo INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    color VARCHAR(50) NOT NULL,
    precio FLOAT NOT NULL,
    codigo_categoria INT UNSIGNED NOT NULL,
    FOREIGN KEY (codigo_categoria) REFERENCES categoria(codigo)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1000;
```

Ejercicios 6.2

1. En la base de datos *Proveedores*:
 - a) ¿Podríamos borrar la Categoría A de la tabla *categoria*?
 - b) ¿Y la Categoría C?
 - c) ¿Podríamos actualizar la Categoría A de la tabla *categoria*?
2. En la base de datos *Proveedores*, suponiendo que hemos cambiado ambas opciones a *CASCADE*:
 - a) ¿Podríamos borrar la Categoría A de la tabla *categoria*?
 - b) ¿Qué le ocurre a las piezas que pertenecen la Categoría A después de borrarla?
 - c) ¿Podríamos actualizar la Categoría A de la tabla *categoria*?

- d) ¿Qué le ocurre a las piezas que pertenecen la Categoría A después de actualizarla?
3. En la base de datos *Proveedores*, suponiendo que hemos cambiado ambas opciones a SET NULL:
- a) ¿Podríamos borrar la Categoría A de la tabla *categoria*?
- b) ¿Qué le ocurre a las piezas que pertenecen la Categoría A después de borrarla?
- c) ¿Podríamos actualizar la Categoría A de la tabla *categoria*?
- d) ¿Qué le ocurre a las piezas que pertenecen la Categoría A después de actualizarla?
4. Añade al script de *Proveedores* la opción para que el campo autoincrementado empiece en 100:
- a) Ejecuta las inserciones y consulta el contenido de las tablas
- p.ej: `select * from categoria;`
- ¿qué valores se asignan a los identificadores de cada elemento? ¿te parece coherente con lo que hemos escrito en el script de creación?
- Esto sucede porque para que se use el autoincremento hay que realizar la inserción con ese valor a NULL.
- p.ej: `INSERT INTO pieza VALUES (NULL, 'Pieza 1', 'Blanco', 25.90, 1);`
- b) Modifica el script de inserción para que la primera columna de *pieza* lleve los valores a NULL. Tras ejecutarlo, ¿qué sucede?
- c) Ahora pon a NULL también los valores de *categoria*. ¿Puedes ejecutar el script correctamente? Lee el mensaje de error. ¿Te acuerdas de la integridad referencial?
- d) ¿cómo podríamos solucionar el problema del punto anterior?
5. En la base de datos *Agencia*, ¿qué restricciones de modificación y borrado pondrías a cada clave ajena? ¿Por qué?
6. Añade restricciones de modificación y borrado al script del ejercicio 6.1.1.
7. Añade restricciones de modificación y borrado al script del ejercicio 6.1.2.
8. Añade restricciones de modificación y borrado al script del ejercicio 6.1.3.
9. Añade restricciones de modificación y borrado al script del ejercicio 6.1.4.
10. Añade restricciones de modificación y borrado al script del ejercicio 6.1.5.
11. Añade restricciones de modificación y borrado al script del ejercicio 6.1.6.
12. Añade restricciones de modificación y borrado al script del ejercicio 6.1.7.

6.2.2. Borrar una tabla.

```
DROP [TEMPORARY] TABLE [IF EXISTS] nombre_tabla [, nombre_tabla] ...  
[RESTRICT | CASCADE];
```

El borrado de una tabla es irreversible ya que ejecuta un COMMIT implícito al finalizar.

IMPORTANTE:

Las opciones RESTRICT y CASCADE. No hacen nada. Existen para facilitar el traslado desde otros SGBD.

```
DROP TABLE pieza;  
  
DROP TABLE IF EXISTS pieza;  
  
DROP TABLE pieza, categoria;
```

El orden de borrado de las tablas puede ser importante si hay claves ajenas, dependiendo de las restricciones de borrado:

- Hay que borrar antes las tablas con referencia (clave ajena) que las referenciadas.
- Si el borrado de las tablas es en la misma instrucción, el orden no afecta.

Ejercicios 6.3

1. ¿En qué orden deberías borrar las tablas de la base de datos *Agencia*? Escribe las sentencias necesarias para ello.
2. En el script de creación de la BdD *Empleados* del Apéndice B, ¿en qué orden deberías borrar las tablas?

6.2.3. Modificar una tabla.

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.

Versión **MUY** simplificada de la sintaxis. La versión completa (demasiado compleja para este curso) se puede consultar [aquí](#).

```

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 [CONSTRAINT [symbol]]
      FOREIGN KEY [index_name] (index_col_name,...)
      reference_definition
    | 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
    | DROP [COLUMN] col_name
    | DROP {INDEX|KEY} index_name
    | DROP PRIMARY KEY
    | DROP FOREIGN KEY fk_symbol
    | 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

```

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

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

table_option:
    AUTO_INCREMENT [=] value
    | [DEFAULT] CHARACTER SET [=] charset_name
    | [DEFAULT] COLLATE [=] collation_name
    | ENCRYPTION [=] {'Y' | 'N'}
    | ENGINE [=] engine_name
```

Las opciones son muchísimas. Vamos a ver algunas de las más comunes.

6.2.3.1. Modificar el tipo de dato de una columna y sus atributos: MODIFY.

Tenemos la siguiente tabla:

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

Y queremos modificar la columna nombre para que pueda almacenar 50 caracteres y además que sea NOT NULL:

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

Sería equivalente a haber creado la tabla así:

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

6.2.3.2. Renombrar una columna, modificar el tipo de dato de una columna y sus atributos: CHANGE.

Similar al anterior pero permite renombrar el nombre de la columna también.

Tenemos la siguiente tabla:

```
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:

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

Sería equivalente a haber creado la tabla así:

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

6.2.3.3. Asignar un valor por defecto a una columna o eliminar el valor por defecto que tenga establecido: ALTER TABLE nombre tabla ALTER.

Tenemos la siguiente tabla:

```
CREATE TABLE usuario (  
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
  nombre VARCHAR(50) NOT NULL,  
  sexo ENUM('H', 'M') NOT NULL  
);
```

Y queremos que el valor por defecto de la columna sexo sea M:


```
ALTER TABLE usuario ALTER sexo SET DEFAULT 'M';
```

Sería equivalente a haber creado la tabla así:


```
CREATE TABLE usuario (  
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
  nombre VARCHAR(50) NOT NULL,  
  sexo ENUM('H', 'M') NOT NULL DEFAULT 'M'  
);
```

Eliminar el valor por defecto de la columna sexo:

```
ALTER TABLE usuario ALTER sexo DROP DEFAULT;
```

6.2.3.4. Añadir nuevas columnas a una tabla: ALTER TABLE nombre tabla ADD.

Elegir posición:

- FIRST: la primera.
- AFTER nombre_columna: detrás de la indicada.
- AFTER  no poner nada: al final.

Tenemos la siguiente tabla:

```
CREATE TABLE usuario (  
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
  nombre VARCHAR(50) NOT NULL,  
  sexo ENUM('H', 'M') NOT NULL  
);
```

Y queremos añadir la columna fecha_nacimiento de tipo DATE:

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

Sería equivalente a haber creado la tabla así:

```
CREATE TABLE usuario (  
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(50) NOT NULL,  
    sexo ENUM('H', 'M') NOT NULL,  
    fecha_nacimiento DATE NOT NULL  
);
```

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;
```

Sería equivalente a haber creado la tabla así:

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

6.2.3.5. Eliminar una columna de una tabla: ALTER TABLE nombre tabla DROP.

Tenemos la siguiente tabla:

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

Y queremos eliminar la columna fecha_nacimiento:

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

Sería equivalente a haber creado la tabla así:

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

6.2.3.6. Añadir o modificar restricciones: ALTER TABLE nombre tabla {ADD | MODIFY} CONSTRAINT.

Tenemos la siguiente tabla:

```
CREATE TABLE departamentos (  
  id_departamento INT NOT NULL,  
  nombre_departamento VARCHAR(30) NOT NULL,  
  id_director INT,  
  id_localizacion INT,  
  CONSTRAINT departamentos_pk PRIMARY KEY (id_departamento),  
  CONSTRAINT departamentos_fk1 FOREIGN KEY (id_localizacion)  
    REFERENCES localizaciones(id_localizacion)  
);
```

Y queremos añadir la restricción de clave ajena sobre la columna `id_director`:

```
ALTER TABLE departamentos ADD CONSTRAINT departamentos_fk2  
  FOREIGN KEY (id_director) REFERENCES empleados(id_empleado);
```

Este caso, sacado del script del Apéndice B, es común. Las tablas `empleados` y `departamentos`, tienen claves ajenas cada una que referencian a la otra. No se puede crear la clave ajena antes que la columna a la que referencia. Una de las dos deberá ser añadida después.

6.2.3.7. Borrar restricciones: ALTER TABLE nombre tabla DROP CONSTRAINT.

No entraremos en detalles porque la sintaxis depende del tipo de restricción a eliminar.

Algunos ejemplos:

```
ALTER TABLE empleados DROP FOREIGN KEY departamentos_fk2;
```

```
ALTER TABLE empleados DROP INDEX departamentos_ind1;
```

```
ALTER TABLE empleados DROP PRIMARY KEY;
```

Eliminar una clave primaria es delicado por lo que podemos encontrarnos con muchos problemas, especialmente si es referenciada por claves ajenas.

6.2.3.8. Desactivar o activar restricciones: ALTER TABLE nombre tabla {DISABLE | ENABLE} KEYS.

```
ALTER TABLE empleados DISABLE KEYS;
```

```
ALTER TABLE empleados ENABLE KEYS;
```

ATENCIÓN: no funciona en *InnoDB*. Podemos usar:

```
SET FOREIGN_KEY_CHECKS=0;
```

```
SET FOREIGN_KEY_CHECKS=1;
```

6.2.3.9. Cambiar el nombre de una tabla: RENAME.

Cambiar nombre:

```
RENAME empleados TO personal;
```

6.2.3.10. Eliminar el contenido de una tabla: TRUNCATE TABLE.

Borrar el contenido de una tabla:

```
TRUNCATE TABLE empleados;
```

6.2.4. Operaciones asociadas.

Mostrar información sobre la estructura de una tabla:

```
DESCRIBE empleados;
```

```
DESC Empleados;
```

Mostrar la sentencia SQL de creación de una tabla:

```
SHOW CREATE TABLE empleados;
```

Ejercicios 6.4

Tras cada modificación, realiza una captura en la que se vea que la tabla ha cambiado.

1. Escribe la sentencia para modificar el *nombre* de la tabla *empleados* para que admita 25 caracteres.
2. Escribe la sentencia para modificar el *salario* de la tabla *empleados* para que admita solo un número decimal.
3. Escribe la sentencia para modificar el *apellido* de la tabla *empleados* para que admita 25 caracteres y se llama *apellido1*.
4. Modifica la *comision* de la tabla *empleados* para que su valor por defecto sea 0.
5. En la tabla *empleados*, añade una columna *apellido2* detrás de *apellido1* con sus mismas características.
6. Elimina la columna creada en el punto anterior.
7. Elimina la restricción *empleados_fk2* de la tabla *empleados*.
8. Escribe la sentencia para añadir la restricción eliminada en el punto anterior.

6.3. Bibliografía.

- José Juan Sánchez Hernández (2021/2022). [Bases de datos / Gestión de Bases de datos](#)
- Departamento de Informática, IES Luis Vélez de Guevara (2021). [Gestión de Bases de Datos](#).
- [Structured Query Language](#). Wikibook. Wikipedia.
- [Curso de MySQL](#). Con Clase.
- [MySQL 8.0 Reference Manual](#).

- Iván López Montalbán, Manuel de Castro Vázquez, John Ospino Rivas (2022). Bases de Datos (2ª Edición) . Garceta.