2 Manipulación de Bases de Datos

2.1 Crear una base de datos

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] nombre_base_datos;
```

- `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.

Ejemplos:

Si no especificamos el set de caracteres en la creación de la base de datos, se usará `latin1` por defecto.

```
CREATE DATABASE nombre_base_datos;
```

Las bases de datos que vamos a crear durante el curso usarán el set de caracteres `utf8` o `utf8mb4`.

```
CREATE DATABASE nombre_base_datos CHARACTER SET utf8;
```

El cotejamiento, es el criterio que vamos a utilizar para ordenar las cadenas de caracteres de la base de datos. Si no especificamos ninguno se usará el que tenga asignado por defecto el set de caracteres escogido. Por ejemplo, para el set de caracteres `utf8` se usa `utf8_general_ci`.

El siguiente ejemplo muestra cómo podemos especificar el cotejamiento queremos de forma explícita:

```
CREATE DATABASE nombre_base_datos CHARACTER SET utf8 COLLATE utf8_general_ci;
```

2.1.1 Conceptos básicos sobre la codificación de caracteres

Unicode es un set de caracteres universal, un estándar en el que se definen todos los caracteres necesarios para la escritura de la mayoría de los idiomas hablados en la actualidad. El estándar Unicode describe las propiedades y algoritmos necesarios para trabajar con los caracteres Unicode y este estándar es gestionado por el consorcio Unicode.

Los formatos de codificación que se pueden usar con Unicode se denominan **UTF-8**, **UTF-16** y **UTF-32**.

- **UTF-8** utiliza 1 byte para representar caracteres en el set ASCII, 2 bytes para caracteres en otros bloques alfabéticos y 3 bytes para el resto del BMP (Basic Multilingual Plane), que incluye la mayoría de los caracteres utilizados frecuentemente. Para los caracteres complementarios se utilizan 4 bytes.
- **UTF-16** utiliza 2 bytes para cualquier carácter en el BMP y 4 bytes para los caracteres complementarios.
- **UTF-32** emplea 4 bytes para todos los caracteres.

Se recomienda la lectura del artículo Codificación de caracteres: conceptos básicos publicado por la W3C.org.

2.1.2 `utf8` y `utf8mb4` en MySQL

En MySQL el set de caracteres `utf8` utiliza **un máximo de 3 bytes por carácter** y contiene sólo los caracteres del BMP (Basic Multilingual Plane). Según el estándar Unicode, el formato de codificación `utf8` permite representar caracteres desde 1 hasta 4 bytes, esto quiere decir que **el set de caracteres `utf8` de MySQL no permite almacenar caracteres Unicode con 4 bytes**.

Este problema se solucionó a partir de MySQL 5.5.3, cuando se añadió el set de caracteres `utf8mb4` que permite utilizar hasta 4 bytes por carácter.

Por ejemplo, en MySQL los caracteres Emoji Unicode no se podrían representar con `utf8`, habría que utilizar `utf8mb4`:

Emoji	Unicode	Bytes (UTF-8)
🙃	U+1F603	`\xF0\x9F\x98\x83`
🙈	U+1F648	`\xF0\x9F\x99\x88`
🙾	U+1F47E	`\xF0\x9F\x91\xBE`

En la documentación oficial de MySQL informan que en las próximas versione de de MySQL se espera solucionar este problema haciendo que `utf8` sea un alias de `utf8mb4`. Hasta que esto no ocurra, se recomienda utilizar `utf8mb4`.

2.1.3 `CHARACTER SET` y `COLLATE`

- `CHARACTER SET`: Especifica el set de caracteres que vamos a utilizar en la base de datos.
- `COLLATE`: Especifica el tipo de cotejamiento que vamos a utilizar en la base de datos. Indica el criterio que vamos a seguir para ordenar las cadenas de caracteres.

Para ver cuáles son los sets de caracteres que tenemos disponibles podemos usar la siguiente sentencia:

```
1  SHOW CHARACTER SET;
```

Para consultar qué tipos de cotejamiento hay disponibles podemos usar:

```
1  SHOW COLLATION;
```

Si queremos hacer una consulta más específica sobre los tipos de cotejamiento que podemos usar con `utf8`:

```
1  SHOW COLLATION LIKE 'utf8%';
```

El cotejamiento puede ser:

- case-sensitive (_cs): Los caracteres `a` y `A` son diferentes.

- case-insensitive (_ci): Los caracteres a y A son iguales.
- binary (_bin): Dos caracteres son iguales si los valores de su representación numérica son iguales.

Para consultar qué set de caracteres y qué cotejamiento está utilizando una determinada base de datos podemos consultar el valor de las variables `character_set_database` y `collation_database`.

En primer lugar seleccionamos la base de datos con la que vamos a trabajar.

```
USE database;
```

Y una vez seleccionada, consultamos el valor de las variables `character_set_database` y `collation_database` .

```
SELECT @@character_set_database, @@collation_database;
```

2.1.4 Ejemplo de cómo afecta el cotejamiento al ordenar una tabla

Suponemos que tenemos una tabla que contiene cadenas de caracteres codificadas en `latin1`.

```
mysql> CREATE TABLE t (c CHAR(3) CHARACTER SET latin1);

mysql> INSERT INTO t (c) VALUES('AAA'),('bbb'),('aaa'),('BBB');

mysql> SELECT c FROM t;
+------+
| c    |
+------+
| AAA  |
| bbb  |
| aaa  |
| BBB  |
+------+
```

Ahora vamos a obtener los registros de la tabla aplicando diferentes tipos de cotejamiento:

- Cotejamiento **case-sensitive** (los caracteres a y A son diferentes).

```
mysql> SELECT c FROM t ORDER BY c COLLATE latin1_general_cs;
+------+
| c    |
+------+
| AAA  |
| aaa  |
| BBB  |
| bbb  |
+------+
```

- Cotejamiento **case-insensitive** (los caracteres a y A son iguales).

```
mysql> SELECT c FROM t ORDER BY c COLLATE latin1_swedish_ci;
+------+
| c    |
+------+
```

```
| AAA  |
| aaa  |
| bbb  |
| BBB  |
+------+
```

- Cotejamiento **binary** (dos caracteres son iguales si los valores de su representación numérica son iguales).

```
mysql> SELECT c FROM t ORDER BY c COLLATE latin1_bin;
+------+
| c    |
+------+
| AAA  |
| BBB  |
| aaa  |
| bbb  |
+------+
```

2.2 Eliminar una base de datos

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

- `DATABASE` y `SCHEMA` son sinónimos.
- `IF EXISTS` elimina la la base de datos sólo si ya existe.

Ejemplo:

```
DROP DATABASE nombre_base_datos;
```

2.3 Modificar una base de datos

```
ALTER {DATABASE | SCHEMA} [nombre_base_datos]
  alter_specification [, alter_especification] ...
```

Ejemplo:

```
ALTER DATABASE nombre_base_datos CHARACTER SET utf8;
```

2.4 Consultar el listado de bases de datos disponibles

```
SHOW DATABASES;
```

Muestra un listado con todas las bases de datos a las que tiene acceso el usuario con el que hemos conectado a MySQL.

2.5 Seleccionar una base de datos

```
1 USE nombre_base_datos;
```

Se utiliza para indicar la base de datos con la que queremos trabajar.

2.6 Mostrar la sentencia SQL de creación de una base de datos

```
1 SHOW CREATE DATABASE nombre_base_datos;
```

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

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] ...
```