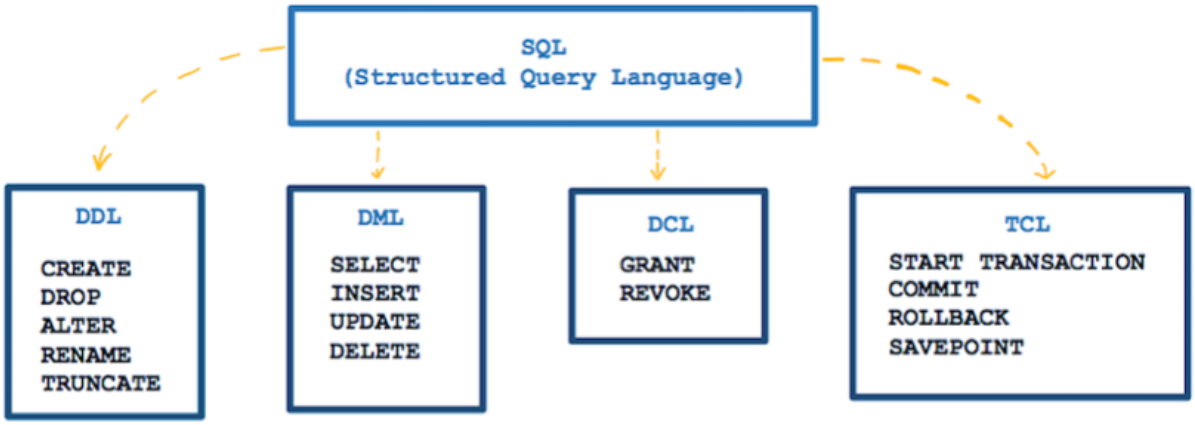
TEMA 04 – CREACIÓN DE BASES DE DATOS EN MYSQL

El lenguaje DDL de SQL



- DDL (Data Definition Language): lenguaje de definición de datos
    - CREATE: crear objetos (BBDD, tablas, vistas, índices, triggers, procedi-s)
    - DROP: eliminar objetos
    - ALTER: modificar objetos
    - SHOW: consultar objetos
    - Otras sentencias de utilidad:
        - USE: indicar la BBDD con la que queremos trabajar
        - DESCRIBE: mostrar info sobre la estructura de una tabla

Manipulación de bases de datos

- Crear bases de datos

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

  - DATABASE y SCHEMA son sinónimos
  - {} es obligatorio, [] es opcional
  - Si no especificamos el set de caracteres: latin1
    - Usar utf8 o utf8mb4 (admite emojis)

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

  - Cotejamiento
    - Criterio para ordenar las cadenas de caracteres de la BBDD

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

- Conceptos básicos sobre la codificación de caracteres
  - SHOW CHARACTER SET;: para ver cuáles son los sets de caracteres que tenemos disponibles
  - SHOW COLLATION;: para ver qué tipos de cotejamiento tenemos disponibles
  - SHOW COLLATION LIKE 'utf8%';: si queremos hacer una consulta más específica sobre los tijos de cotejamiento que Podemos usar con uft8
  - 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 cotejamiento está utilizando una determinada base de datos podemos consultar el valor de las variables:
    - 'character_set_database'
    - 'collation_database'
    - Primero la seleccionamos, luego consultamos:

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

- Ejemplo de cómo afecta el cotejamiento al ordenar una tabla
  - Se crea la tabla:

```
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  |
+------+
```

- Ejemplos 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  |
+------+
```

- 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;
```

- Modificar una base de datos

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

Ejemplo:

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

- Consultar el listado de bases de datos disponibles

```
SHOW DATABASES;
```

- Seleccionar una base de datos

```
USE nombre_base_datos;
```

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

```
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.

Manipulación de tablas

- Crear una tabla

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

  - Restricciones sobre las columnas de la tabla
    - NOT NULL o NULL: si puede almacenar o no valores nulos
    - DEFAULT: permite indicar un valor inicial por defecto SI NO especificamos ninguno en la inserción
    - AUTO_INCREMENT: su valor se incrementará automática- en cada inserción (solo tipo INT)
    - UNIQUE KEY: el valor de la columna es único
    - PRIMARY KEY
    - CHECK: nos permite realizar restricciones sobre una columna

```
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)
);
```

  - Opciones en la declaración de claves ajenas (FK)
    - ON DELETE y ON UPDATE: efecto que provoca el borrado o actualización
      - RESTRICT: impide que se puedan actualizar o eliminar las filas que tienen valores referenciados por FK (opción por defecto)
      - CASCADE: permite actualizar o eliminar las filas que tienen valores referenciados por FK
      - SET NULL: asigna NULL a las filas que tienen valores referenciados por FK
      - NO ACTION: equivalente a RESTRICT
      - SET DEFAULT: no es posible con InnoDB
  - Opciones a tener en cuenta en la creación de tablas

- AUTO_INCREMENT: podemos indicar el valor inicial a usar en el campo definido como AUTO_INCREMENT
- CHARACTER SET: set de caracteres en la tabla
- COLLATE: especifica el tipo de cotejamiento que vamos a utilizar en la tbla
- ENGINE: por defecto InnoDB

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

- Eliminar una tabla

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

- Modificar una tabla
  - 'alter table [table] modify'
    - Permite modificar el tipo de dato de una columna y sus atributos

```
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:

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

  - 'alter table [table] change'
    - Permite renombrar una columna, modificar el tipo de dato de una columna y sus atributos

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:

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

Después de ejecutar esta sentencia la tabla quedaría así:

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

- 'alter table [table] alter'
  - Permite asignar un valor por defecto a una columna o eliminar el valor por defecto que tenga establecido

```
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`. En este caso usaríamos la sentencia:

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

Después de ejecutar esta sentencia la tabla quedaría así:

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

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

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

- 'alter table [table] add'
  - Permite añadir nuevas columnas a una tabla
  - Con FIRST y AFTER podemos elegir el lugar de la tabla donde queremos insertar la nueva columna
    - FIRST: la primera
    - AFTER: detrás de la columna que se especifique
    - No especificar nada: la última de la 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;
```

En este caso la nueva columna se ha añadido detrás de la última columna, `sexo`. La tabla quedaría 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
);
```

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),
  sexo ENUM('H', 'M') NOT NULL DEFAULT 'M',
  fecha_nacimiento DATE NOT NULL
);
```

- ‘alter table [table] drop’
  - Elimina una columna de la tabla

```sql
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`. En este caso usaríamos la sentencia:

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

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,
  apellido1 VARCHAR(50) NOT NULL,
  apellido2 VARCHAR(50),
  sexo ENUM('H', 'M') NOT NULL DEFAULT 'M'
);
```

    - Si no tiene datos, se puede eliminar y volver a crear
- Consultar el listado de tablas disponibles

```sql
SHOW TABLES;
```

- Mostrar información sobre la estructura de una tabla

```sql
DESCRIBE nombre_tabla;
```

    - También DESC
- Mostrar la sentencia SQL de creación de una tabla

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

Tipos de datos

- 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

  - Zerofill
    - Todos admiten el atributo ZEROFILL
    - También se le añade automáticamente el atributo UNSIGNED
    - El campo quedaría como UNSIGNED ZEROFILL
  - Nota importante sobre INT(11)
    - Indica el ancho de la columa que ocupará dicho valor y tiene utilidad cuando asignamos el atributo UNSIGNED ZEROFILL
    - En una columna declarada como INT(4) ZEROFILL, el valor 5 será representado como 0005
  - Bit, bool, boolean, serial

Tipo	Descripción
BIT(M)	M puede ser un valor de 1 a 64.
	Indica el número de bits que vamos a utilizar para este campo.
	Si se omite el valor de M se utiliza 1 bit por defecto.
BOOL, BOOLEAN	Son equivalentes a TINYINT(1).
	El valor 0 se considera como FALSE.
	Cualquier valor distinto de 0 será TRUE.
SERIAL	Es un alias para: BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE.

- Número en punto flotante (valores aproximados)

Tipo	Bytes	Mínimo	Máximo
FLOAT	4		
FLOAT(M,D)	4	±1.175494351E-38	±3.402823466E+38
FLOAT(M,D)UNSIGNED	4	1.175494351E-38	3.402823466E+38
DOUBLE	8		
DOUBLE(M,D)	8	±1.7976931348623157E+308	±2.2250738585072014E-308
DOUBLE(M,D)UNSIGNED	8	1.7976931348623157E+308	2.2250738585072014E-308

    - M indica el número de dígitos en total (la precisión)
    - D es el número de cifras decimales
- Números en punto fijo (valores exactos)
    - Se utilizan cuando es necesario guardar los valores exactos sin redondeos
    - Se suelen utilizar cuando trabajamos con datos monetarios

Tipo	Bytes
DECIMAL	
DECIMAL(M,D)	M + 2 bytes si D > 0
DECIMAL(M,D)UNSIGNED	M + 1 bytes si D = 0

    - En MySQL DECIMAL y NUMERIC son equivalentes
    - M: dígitos en total (precisión) en un rango 1-65
    - D: número de cifras decimales en un rango 0-30
- Fechas y tiempo

Tipo	Bytes	Descripción	Rango	Máximo
DATE	3	YYYY-MM-DD	1000-01-01	9999-12-31
DATETIME	8	YYYY-MM-DD HH:MM:SS	1000-01-01 00:00:00	9999-12-31 23:59:59
TIMESTAMP	4	YYYY-MM-DD HH:MM:SS	1970-01-01 00:00:00	2038-01-19 03:14:07
TIME	3	HH:MM:SS	-838:59:59	838:59:59
YEAR[(2|4)]	1	YY o YYYY	YY: 70 (1970)	YY: 69 (2069)
			YYYY: 1901	YYYY: 2155

- Cadenas de caracteres

Tipo	Descripción
`CHAR(M)`	0 <= M <= 255
`VARCHAR(M)`	0 <= M <= 65535
`TEXT[(M)]`	L + 2 bytes, donde L < 216 = 65536
`MEDIUMTEXT`	L + 3 bytes, donde L < 224 = 16 MB
`LONGTEXT`	L + 4 bytes, donde L < 232 = 4 GB

- Datos binarios

Tipo	Descripción
`BINARY`	0 <= M <= 255
`VARBINARY`	0 <= M <= 65535
`BLOB`	L + 2 bytes, donde L < 216 = 65536
`MEDIUMBLOB`	L + 3 bytes, donde L < 224 = 16 MB
`LONGBLOB`	L + 4 bytes, donde L < 232 = 4 GB

- Enum y set

Tipo	Descripción
`ENUM('valor1', 'valor2', ...)`	Puede tener 65535 valores. Sólo permite seleccionar un valor de la lista
`SET('valor1', 'valor2', ...)`	Puede tener 64 valores. Permite seleccionar varios valores de la lista

- Valores fuera de rango y desbordamientos
  - 'SET sq_mode = 'TRADITIONAL';': da un mensaje de error y no se guarda
  - 'SET sq_mode = '';': se guarda el valor máximo
    - 400 en TINYINT UNSIGNED sería 255

```
SET sql_mode = '';

CREATE TABLE test (data TINYINT UNSIGNED);

INSERT INTO test VALUES(400);

1 row(s) affected, 1 warning(s): 1264 Out of range value for column 'data' at row
    1
```

```
Si consultamos el contenido de la tabla veremos que en lugar de almacenar el
valor `400` se ha almacenado el valor `255`.
```

```
SELECT * FROM test;
+------+
| data |
+------+
| 255  |
+------+
```