Módulo SQL – Base de datos

Contenidos:

- Introducción
- Diseño de bases de datos:
  - Requerimientos
  - Modelo Entidad-Relación
  - Paso a tablas
  - Creación de tablas
- Tratamiento de datos y consultas
  - Modificar Información
  - Consulta de registros
  - Consultas con varias tablas

Creación de tablas

Diseño de base de datos:

Creación de tablas:

El lenguaje SQL (Structured Query Language) permite la comunicación con el SGBD. Actualmente es el lenguaje estándar para la gestión de Bases de Datos relacionales para ANSI (American National Standard Institute) e ISO (International Standarization Organization).

Entre las principales características de este lenguaje destaca que es un lenguaje para todo tipo de usuarios ya que quién lo utiliza especifica qué quiere, pero no dónde ni cómo, de manera que permite realizar cualquier consulta de datos por muy complicada que parezca.

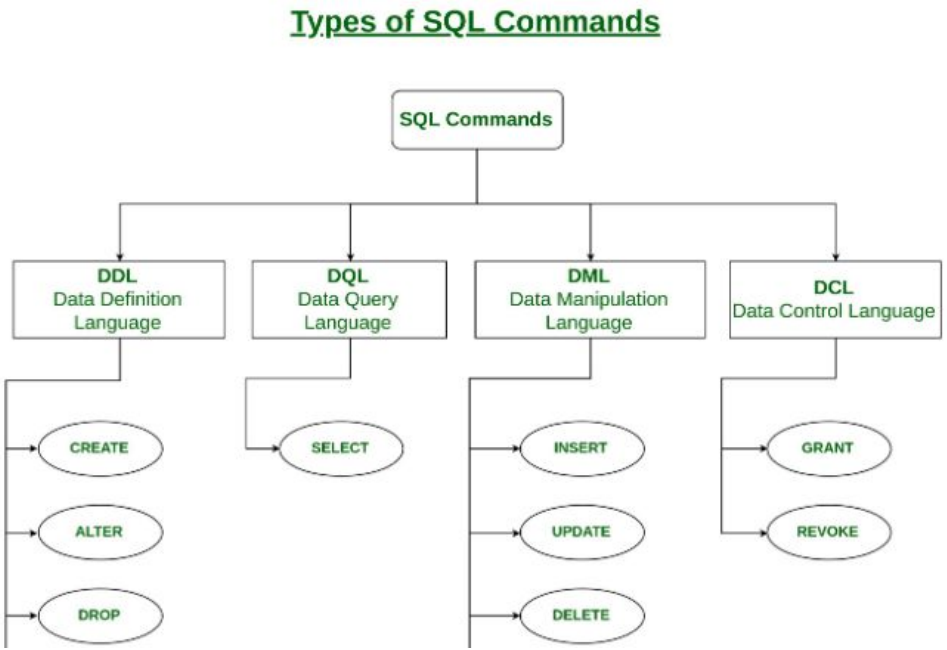
Diseño de base de datos:

Manipulación de tablas (Sentencias DDL):

En el lenguaje SQL, dependiendo de las tareas que se quieran realizar, se distinguen dos tipos de sentencias. Sentencias DDL (Data Definition Language) que sirven para especificar y gestionar estructuras de datos, y las sentencias DML (Data Manipulation Language) que sirven para trabajar con los datos almacenados en dichas estructuras.

Puesto que por ahora abordaremos aquellas sentencias que nos van a permitir crear nuestras Bases de Datos en un SGBD relacional, comenzaremos por ver el grupo de sentencias DDL, que son las que se citan a continuación:

Diseño de base de datos:



Diseño de base de datos:

Crear un objeto: CREATE

Es la sentencia utilizada para la creación de un objeto (base de datos, tabla, usuario, vista, procedimiento, . . .) en una Base de Datos.

La sintaxis para la creación de una tabla es la siguiente:

```
CREATE TABLE [IF NOT EXISTS] <nombre_tabla>
(
   <nombre_columna1>  <tipo_dato>  <restricciones>,
   <nombre_columna2>  <tipo_dato>  <restricciones>,
   ..............................
)
```

Y para crear una Base de Datos:

```
CREATE DATABASE [IF NOT EXISTS] <nombre_base_de_datos>
```

Diseño de base de datos:

Conectar con una Base de Datos

Permite realizar la conexión con una Base de Datos de MySQL.

```
USE <nombre_base_de_datos>
```

Diseño de base de datos:

Eliminar un objeto: DROP

Es la sentencia utilizada para eliminar objetos (tabla, usuario, vista, procedimiento, . . .) en una Base de Datos.

La sintaxis para la eliminación de tablas es la siguiente:

```
DROP TABLE [IF EXISTS] <nombre_tabla>
```

Y para eliminar una Base de Datos:

```
DROP DATABASE [IF EXISTS] <nombre_base_de_datos>
```

Diseño de base de datos:

Modificar un objeto: ALTER

Es la sentencia utilizada para modificar objetos (tabla, usuario, vista, procedimiento, . . .) en una Base de Datos.

La sintaxis para modificar una tabla es la siguiente:

```
ALTER TABLE <nombre_tabla>
    [ ADD <definicion_columna> ]
    [ CHANGE <nombre_columna> <definicion_columna> ]
    [ DROP COLUMN <nombre_columna> ]
    [ ADD CONSTRAINT <restriccion> ]
```

Diseño de base de datos:

Revocar privilegios sobre un objeto: REVOKE

Permite eliminar el privilegio sobre un objeto a un usuario.

```
REVOKE <privilegio>
ON <objeto>
FROM <usuario>
```

Diseño de base de datos: Tipos de datos

Usuarios y privilegios

Supongamos que somos los administradores de un SGBD MySQL y tenemos que proporcionar acceso a una Base de Datos para una aplicación biblioteca a un desarrollador de mi compañía:

```
-- Si no hemos creado la base de datos, podemos hacerlo ahora
CREATE DATABASE biblioteca;
-- Crea el usuario asignándole contraseña
CREATE USER 'desarrollador' IDENTIFIED BY 'micontraseña';
-- Asigna todos los privilegios al usuario sobre la base de datos
GRANT ALL PRIVILEGES ON biblioteca.* TO desarrollador;
```

Diseño de base de datos: Tipos de datos

Cadenas de caracteres:

Tipo CHAR, VARCHAR

Este tipo de datos permite almacenar cadenas de texto fijas (CHAR) o variables (VARCHAR).

El tipo CHAR permite almacenar cadenas de caracteres de longitud fija entre 1 y 255 caracteres. La longitud de la cadena se debe especificar entre paréntesis en el momento de la declaración (cadena CHAR(25)).

Por otro lado, el tipo VARCHAR permite almacenar cadenas de caracteres variables de hasta 4.000 caracteres. La declaración del tipo VARCHAR es similar a la de un tipo CHAR (cadena VARCHAR(25)). La principal y única diferencia entre estos dos tipos, es que el tipo CHAR declara una cadena fija de la longitud que se especifica mientras que en la declaración de un tipo VARCHAR lo que se especifica es un tamaño máximo, la cadena sólo ocupará el tamaño necesario para almacenar el dato que contenga (hasta llegar al máximo). En cualquier caso, no es posible almacenar cadenas de mayor tamaño al especificado en su declaración, puesto que el SGBD truncará el valor almacenándose sólo hasta la longitud establecida.

Diseño de base de datos: Tipos de datos

Tipo TEXT

El tipo TEXT permite almacenar cadenas de caracteres de hasta varios GB de longitud. Sólo se recomienda su uso para almacenar textos realmente grandes, puesto que presenta ciertas restricciones, aunque algunas pueden variar dependiendo del SGBD que se utiliza:

- Sólo se puede definir una columna TEXT por tabla
- No se pueden establecer restricciones en columnas de este tipo
- No se permite su utilización en ciertas cláusulas

Diseño de base de datos: Tipos de datos

Cadenas de caracteres:

Tipo TEXT

Tipo	Tamaño (bytes)
TINYTEXT	0 - 255
TEXT	0 - 65535
MEDIUMTEXT	0 - 16777215
LONGTEXT	0 - 4294967295

Diseño de base de datos: Tipos de datos

Tipos numéricos

Para la representación de tipos de datos numéricos. Los tipos más utilizados son TINYINT, INT, SMALLINT, BIGINT, FLOAT y DECIMAL, para la representación de números enteros de menor o mayor tamaño, y para números en coma flotante de menor o mayor precisión, respectivamente.

En ocasiones el rango de los valores negativos resultará prescindible (claves numéricas, valores de dinero, cantidades, . . .) por lo que será posible ampliar el rango positivo de un tipo numérico añadiendo la restricción UNSIGNED tras definir el tipo de éste.

```
id INT UNSIGNED
```

Diseño de base de datos: Tipos de datos

Tipos numéricos (enteros con signo):

Data type	Bytes	Mínimo valor	Máximo valor
TINYINT	1	-128	127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INTEGER	4	-2147483648	2147483647
BIGINT	8	-9223372036854775808	9223372036854775807

Diseño de base de datos: Tipos de datos

Tipos numéricos (enteros sin signo):

Data type	Bytes	Mínimo valor	Máximo valor
TINYINT	1	0	255
SMALLINT	2	0	65535
MEDIUMINT	3	0	16777215
INTEGER	4	0	4294967295
BIGINT	8	0	18446744073709551615

Diseño de base de datos: Tipos de datos

Tipos numéricos: (números de punto flotante)

En DECIMAL se debe especificar su alcance: [M, D]

- M es el número máximo de dígitos (la precisión)
- D es el número de dígitos a la derecha del punto decimal (la escala)

Diseño de base de datos: Tipos de datos

¡Ojo con los floats!

```
mysql> CREATE TABLE Numbers (Id TINYINT, Floats FLOAT, Decimals DECIMAL(3, 2));
mysql> INSERT INTO Numbers VALUES (1, 1.1, 1.1), (2, 1.1, 1.1), (3, 1.1, 1.1);
```

```
mysql> SELECT * FROM Numbers;
+------+--------+----------+
| Id   | Floats | Decimals |
+------+--------+----------+
|    1 |    1.1 |     1.10 |
|    2 |    1.1 |     1.10 |
|    3 |    1.1 |     1.10 |
+------+--------+----------+
3 rows in set (0,00 sec)

mysql> SELECT SUM(Floats), SUM(Decimals) FROM Numbers;
+--------------------+---------------+
| SUM(Floats)        | SUM(Decimals) |
+--------------------+---------------+
| 3.3000000715255737 |          3.30 |
+--------------------+---------------+
1 row in set (0,08 sec)
```

Diseño de base de datos: Tipos de datos

Tipos para fechas:

Los tipos más utilizado para almacenar valores de fechas (DATE) o fechas con hora (DATETIME). Por defecto el formato más utilizado es DD/MM/YY ó DD/MM/YYYY.

También se puede usar el tipo TIMESTAMP para almacenar una marca de tiempo (fecha y hora). Además, permite el uso de la constante CURRENT_TIMESTAMP en la definición de la columna al definirle un valor por defecto cuando se crea la tabla.

Diseño de base de datos: Tipos de datos

Tipo booleano

Permite almacenar valores lógicos Verdadero/Falso o Sí/No. Realmente se define la columna como del tipo TINYINT, que simplemente almacena los valores 0 y 1 para indicar los valores lógicos Verdadero y Falso, respectivamente. Así, podremos utilizar los valores TRUE ó FALSE o directamente asignar 1 ó 0 para asignar valor.

Diseño de base de datos: Tipos de datos

Clave primaria

Una clave primaria dentro de una tabla, es una columna o conjunto de columnas cuyo valor identifica unívocamente a cada fila. Debe ser única, no nula y es obligatoria. Como máximo podremos definir una clave primaria por tabla y es muy recomendable definirla.

Para definir una clave primaria utilizamos la restricción PRIMARY KEY.

```
dni VARCHAR(9) PRIMARY KEY
```

Y si lo hacemos al final de la definición de las columnas, quedaría así:

```
PRIMARY KEY (dni)
```

Hay que tener en cuenta que a la hora de definir claves primarias compuestas (la componen 2 ó más campos de la tabla), ésta deberá ser definida forzosamente tras la definición de los campos involucrados, siguiendo esta sintaxis

```
PRIMARY KEY (nombre, apellidos)
```

Diseño de base de datos: Tipos de datos

Autonumérico

Especialmente útil en el caso de aquellas columnas que se definan como claves primarias de cada tabla, resulta añadir la restricción de campo autonumérico, siempre y cuando ésta sea una columna de tipo entero. De esa manera será el SGBD el encargado de asignarle valor de forma automática, siempre asignando un valor entero de forma secuencial a medida que se van insertando las filas en dicha tabla.

La forma de definirlo es añadiendo la restricción AUTO_INCREMENT en la definición de la columna que se ha definido como clave primaria:

```
id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT
```

Diseño de base de datos: Tipos de datos

Clave ajena

Una clave ajena está formada por una o varias columnas que hacen referencia a una clave primaria de otra o de la misma tabla. Se pueden definir tantas claves ajenas como sea necesario (no hay límite). El valor de la columna o columnas que son clave ajena será NULL, o bien el valor de la clave primaria de la tabla a la que hacen referencia (integridad referencial). Así, a la hora de definir una clave ajena, deberemos indicar con la cláusula REFERENCES la tabla a que ésta hace referencia (se tomará automáticamente la clave primaria de ésta como campo con el que mantener la integridad referencial).

Diseño de base de datos: Tipos de datos

Clave ajena

Habrá que tener en cuenta que mientras que un campo definido como clave ajena haga referencia a un campo definido como clave primaria, la columna de la segunda tabla no podrá ser eliminada hasta que no lo haga la columna que le hace referencia (integridad referencial). Para evitar estos problemas (aunque no siempre es un problema) es posible definir la restricción de clave ajena añadiendo la cláusula ON DELETE o bien ON UPDATE para el caso de una actualización. De esa manera, cuando se vaya a eliminar o actualizar una fila a cuya clave primaria se haga referencia, podremos indicar a MySQL que operación queremos realizar con las filas que le hacen referencia:

- RESTRICT: Se rechaza la operación de eliminación/actualización
- CASCADE: Realiza la operación y se elimina o actualiza en cascada en las filas que hacen referencia
- SET NULL: Realiza la operación y fija a NULL el valor en las filas que hacen referencia
- NO ACTION: Se rechaza la operación de eliminación/actualización, como ocurre con la opción RESTRICT

Diseño de base de datos: Tipos de datos

Clave ajena

A continuación, un par de ejemplos que definen claves ajenas con diferentes cláusulas:

```
FOREIGN KEY (id_curso) REFERENCES cursos (id)
FOREIGN KEY (id_curso) REFERENCES cursos (id) ON DELETE CASCADE
```

Como ocurre con las claves primarias, si las claves ajenas son compuestas, se definen forzosamente al final de las definiciones de las columnas de la tabla, de la siguiente forma:

```
FOREIGN KEY (id_curso, id_aula) REFERENCES cursos
FOREIGN KEY (id_curso, id_aula) REFERENCES cursos ON DELETE CASCADE
```

En cualquiera de los casos hay que tener en cuenta que habrá que definir primero el campo con el tipo de dato correcto (el mismo que dicho campo en la tabla donde aparece como clave principal) y luego la propia definición de dicho campo como clave ajena.

Diseño de base de datos: Tipos de datos

Definir claves ajenas en MySQL

Para definir claves ajenas en MySQL habrá que tener en cuenta algunas consideraciones:

- La columna deberá ser del mismo tipo (y atributos) que la columna de la que es clave ajena
- La columna deberá ser un índice
- Si la columna se define como obligatoria (NOT NULL) no podrá contener la cláusula (SET NULL) para los casos de borrado (ON DELETE) o actualización (ON UPDATE)

Diseño de base de datos: Tipos de datos

Definir claves ajenas en MySQL

```
CREATE TABLE Clientes (
    id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
    . . .
);
```

Actualmente, a partir de MySQL 8.0 (y versiones equivalentes de MariaDB) es suficiente definir el campo y la restricción de clave ajena para que el gestor haga el resto (definir dicho campo como índice). El caso anterior se podría reducir a:

```
id_cliente INT UNSIGNED,
FOREIGN KEY (id_cliente)
  REFERENCES Clientes (id)
  ON DELETE SET NULL ON UPDATE SET NULL,
```

Diseño de base de datos: Tipos de datos

Campos obligatorios

Esta restricción obliga a que se le tenga que dar valor obligatoriamente a una columna. Por tanto, no podrá tener el valor NULL. Se utiliza la palabra reservada NOT NULL.

```
    apellidos VARCHAR(250) NOT NULL
```

Valores por defecto

Se puede definir el valor que una columna tomará por defecto, es decir, si al introducir una fila no se especifica valor para dicha columna. Se utiliza la palabra reservada DEFAULT.

```
fecha  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
nombre VARCHAR(250) DEFAULT 'Sin nombre'
```

Diseño de base de datos: Tipos de datos

Condiciones

De forma más genérica, podemos forzar a que los valores de determinados campos de la tabla cumplan una ciertas condiciones. En caso contrario no se permitirá la inserción de esa fila en dicha tabla.

```
nombre VARCHAR(250) CHECK (nombre = UPPER(nombre))
edad   INT CHECK (edad > 0)
curso  INT CHECK (curso BETWEEN 1 AND 2)
```

Hay que tener en cuenta que en MySQL está restricción no tiene ningún efecto y lo habitual es definir una columna como de tipo enumeración (ENUM en MySQL) si queremos indicar que solamente una serie de valores (definidos) son válidos:

```
curso   ENUM ('0', '1', '2'),
horario ENUM ('mañana', 'tarde', 'noche'),
```

Diseño de base de datos: Tipos de datos

Valores únicos

La restricción UNIQUE evita valores repetidos en una misma columna. Al contrario que ocurre con la restricción PRIMARY KEY, la restricción de valor único se puede aplicar a varias columnas de una misma tabla y admite el valor NULL. Con respecto a esta última consideración, conviene saber que si una columna se define como UNIQUE, sólo una de sus filas podrá contener el valor NULL

```
email VARCHAR(100) UNIQUE
```

Diseño de base de datos:

Creación de scripts en MySQL

La forma más habitual de trabajo a la hora de lanzar órdenes en SQL sobre un SGBD relacional como MySQL es crear ficheros por lotes de órdenes SQL, lo que se conoce como scripts SQL, donde podemos escribir todas las sentencias SQL que queremos ejecutar una detrás de otra separadas por el carácter ;.

Existe la posibilidad de añadir comentarios al código según la siguiente sintaxis:

```
-- Esto es un comentario y MySQL no lo ejecuta
/* Esto también es un comentario y
   tampoco se ejecuta */
```

Diseño de base de datos:

Ejemplo de script en MySQL

Por ejemplo, para la creación de una nueva Base de Datos y sus tablas podríamos preparar un script SQL como el siguiente:

```
CREATE DATABASE IF NOT EXISTS pagina_web;
USE pagina_web;

CREATE TABLE IF NOT EXISTS usuarios (
    id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
    . . .
    . . .
);
CREATE TABLE IF NOT EXISTS productos (
    . . .
    . . .
);
```

Diseño de base de datos:

*truqui

A veces resulta útil desactivar las claves ajenas, realizar algunas operaciones sobre tablas que puedan tener relaciones con otras y volver a activarlas. De esa manera es posible realizar ciertas operaciones sin que las reglas de validación de la integridad referencial lancen ningún error.

```
-- Desactivar claves ajenas
SET FOREIGN_KEY_CHECKS = 0;
. . .
// Realizar algunos cambios en la estructura y datos de la Base de Datos
. . .
-- Activar claves ajenas
SET FOREIGN_KEY_CHECKS = 1;
```

Diseño de base de datos:

Comprobaciones sobre el script SQL

- Utilizar notación snake_case para todos los identificadores (nombre de la base de datos, nombres de tablas, nombres de columnas, . . .). Y siempre en minúscula
- No utilizar acentos, el carácter ñ ni otros caracteres extraños (|@#…) para nombres de bases de datos, tablas, columnas o cualquier otro elemento
- Escribir las palabras reservadas del lenguaje SQL en mayúsculas
- Todas las tablas tendrán un campo clave primaria cuyo nombre será id (definir como id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT)

Diseño de base de datos:

Comprobaciones sobre el script SQL

- Las claves ajenas indicarán la tabla a la que hacen referencia (en singular) como parte de su nombre. Por ejemplo: id_usuario si es una clave ajena de una tabla usuarios. Si en una tabla hay dos claves ajenas que apuntan a la misma tabla, añadiremos algo al nombre para distinguirla (id_usuario_emisor e id_usuario_receptor, por ejemplo)
- Se recomienda que los nombres de las tablas sean en plural (users mejor que user, orders mejor que order)
- Antes de definir un tipo de dato como numérico, comprobar si realmente voy a operar con él como tal
- Cuidado con los campos contraseña. Realmente nunca se guarda tal cual sino como un hash utilizando algún algoritmo, por lo que la longitud real es mayor (la longitud de un hash creado con SHA1 es de 40 caracteres y con SHA2 hasta 128)

Diseño de base de datos:

Conceptos básicos:

- **Campo o atributo:** Área de almacenamiento para almacenar datos de un tipo específico. No pueden almacenarse tipos diferentes una vez definido.
- **Registro o tupla:** Colección de datos relacionados. Dichos datos pueden ser de diferentes tipos.
- **Tabla o archivo:** Colección de registros.

Diseño de base de datos:

Conceptos básicos:

Tabla o archivo

Campo o atributo

Id	Marca	Modelo	Matricula	Color
1	Seat	Ibiza	7777 HYX	Rojo
2	Honda	Accord	2567 FGH	Verde
3	Peugeot	208	1946 LJC	Amarillo
4	Ford	Mustang	9951 KTR	Azul

Registro o tupla