

Unidad 5

Data Definition Language

DDL

2

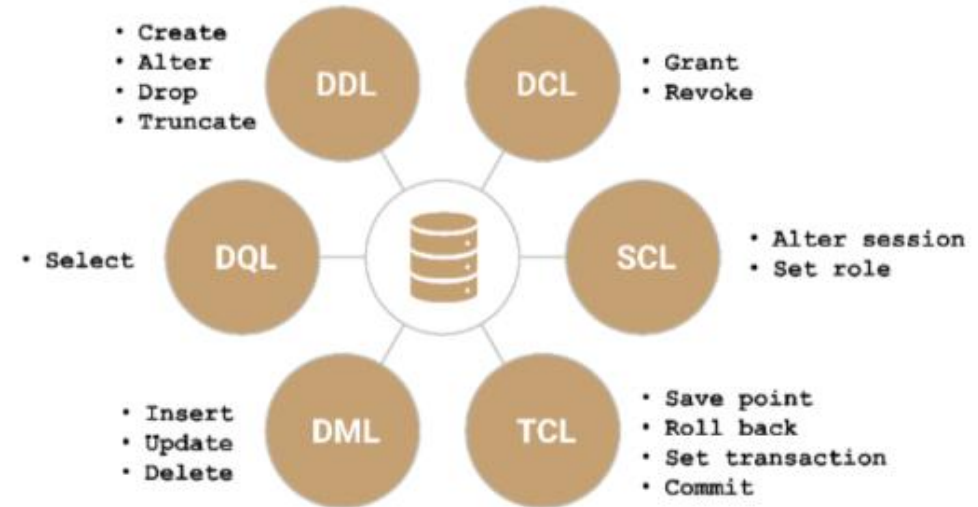
Contenido:

- ▶ Lenguaje SQL
- ▶ SENTENCIAS DDL
- ▶ INTRO MYSQL

Lenguaje SQL

3

- El lenguaje **SQL** es un lenguaje específico de dominio que permite la comunicación con la Base de Datos de forma directa o a través de una aplicación.
- Las sentencias SQL se pueden agrupar en 6 grandes bloques:



- Tipos básicos de **dominios SQL**: El lenguaje SQL debe integrarse con otros lenguajes de programación, por lo que debe soportar los mismos dominios y tipos de datos básicos que dichos lenguajes.
- **Numéricos**

| | |
|-------------------------------|---|
| <i>integer o int</i> | Permite trabajar con números enteros, será el HW o el SGBDR quien limitará su rango de valores. Existe tipos llamados <i>tinyint</i> , <i>smallint</i> y <i>mediumint</i> para rangos más pequeños, o tipos llamados <i>bigint</i> para rangos más grandes. |
| <i>numeric (p, d)</i> | Permite trabajar con números decimales. El usuario definirá: <ul style="list-style-type: none">- con el parámetro p: el número de cifras enteras- con el parámetro d: el número de cifras decimales. |
| <i>real, double precision</i> | Permite trabajar con números en coma flotante con <i>real</i> de precisión simple o con <i>double</i> de doble precisión utilizando el estándar IEEE 754. |
| <i>float (n)</i> | Permite trabajar con números en coma flotante; en este caso es el usuario quien decide con el parámetro <i>n</i> el número de cifras para la precisión de este. |

► Fechas y horas:

| | |
|------------------|---|
| datetime | Permite trabajar con valores que contengan partes de fecha y/o hora. |
| timestamp | Permite trabajar con valores que contengan partes de fecha y/o hora. La diferencia con datetime es que hace referencia al huso horario UTC, por lo que dependiendo del país en el que nos encontremos podemos hacer el cálculo adecuado para mostrar el valor según nuestro huso horario. |
| date | Permite trabajar con valores que contengan solo la fecha. Puede que alguna vez nos interese tener estos valores de las fechas y las horas por separado; por eso, disponemos de los siguientes tipos de campos. |
| time | Permite trabajar con valores que contengan solo la hora. |
| year | Permite trabajar con valores que contengan solo las 4 cifras del año dentro del rango de 1901 a 2155. Si queremos guardar el año actual lo podemos obtener de la función NOW(). |

Lenguaje SQL

6

► Valores alfanuméricos:

| | |
|--|---|
| <i>character</i> o <i>char(n)</i> | Permite guardar una cadena de caracteres de longitud fija <i>n</i> . |
| <i>character varying</i> o <i>varchar(n)</i> | Permite guardar una cadena de caracteres de longitud variable, aunque el usuario con el valor <i>n</i> indicará su tamaño máximo. |

► Otros tipos de datos:

| | |
|--|---|
| <i>boolean</i> o <i>bit</i> | Permite guardar valores ciertos (true) o falsos (false). |
| <i>binary</i> , <i>varbinary</i> , <i>blob</i> , <i>text</i> , <i>enum</i> y <i>set</i> | Para campos más grandes. Pueden servir para guardar imágenes u objetos multimedia, entre otros. |

► [MySQL :: MySQL 8.0 Reference Manual :: 13 Data Types](#)

Lenguaje SQL

7

► DDL – Data Definition Language:

- CREATE
- ALTER
- DROP
- RENAME
- TRUNCATE
- COMMENT
- SHOW
- DESCRIBE

DDL - CREATE

8

- ▶ **CREATE** . Instrucción para la creación de objetos: base de datos, tablas, vistas, usuarios, procedimientos y funciones, triggers, índices, etc.

- DATABASE

```
CREATE DATABASE bdcine;
```

- Para entrar en una bbdd en concreto : **USE** nombre_BBDD;

- TABLE

```
CREATE TABLE `cine` (  
  `idCine` int(11) NOT NULL PRIMARY KEY,  
  `nombreCine` varchar(44)  
  `poblacionCine` varchar(33)  
)
```


DDL - CREATE

9

- TABLE

Clave Primaria compuesta por dos columnas:

```
CREATE TABLE `sala` (  
  `idCine` int(11) NOT NULL,  
  `idSala` int(11) NOT NULL,  
  `butacasSala` int(11) NOT NULL,  
  PRIMARY KEY (idCine, idSala)  
) ;
```

```
CREATE TABLE alumno (  
  id_alumno smallint(6) auto_increment not null,  
  nombre VARCHAR(20),  
  apellido1 VARCHAR(50),  
  apellido2 VARCHAR(50),  
  nom_completo VARCHAR(255) AS (CONCAT(apellido1, ' ', apellido2, ' ', nombre)),  
  edad smallint(3),  
) ;
```

► CREATE (continuación)

- INDEX

```
create index idActorPK
on actor (
    idActor
) with primary;
```

```
create unique index AlumnoNSS -- valor único
on alumno (
    NSS
) with disallow null; -- no permite valores nulos
```

► **CREATE** (continuación)

- FOREIGN KEY. Al crear la tabla (o modificarla), se puede añadir la restricción de clave foránea:

```
CONSTRAINT claveForanea FOREIGN KEY  
REFERENCES tablaRelacionada (campoFK);
```

- ▶ **ALTER TABLE:** modificaciones en la tabla

- ▶ Sintaxis y opciones:

```
ALTER TABLE nombre_tabla
  ADD COLUMN nombre_campo tipo [tamaño | NOT NULL | CONSTRAINT tipo_constraint]
  ADD CONSTRAINT restriccion (campos)
  ALTER COLUMN nombre_campo tipo [tamaño | NOT NULL | CONSTRAINT tipo_constraint]
  DROP COLUMN nombre_campo
  DROP CONSTRAINT restriccion
  DROP FOREIGN KEY fk_claveforanea;
```

- ▶ Ejemplos:

```
ALTER TABLE ACTOR ADD COLUMN (NOMBREACTOR VARCHAR(30));
ALTER TABLE ALUMNO ALTER COLUMN (APELLIDOS VARCHAR(40) NOT NULL);
ALTER TABLE ALUMNO DROP CONSTRAINT PRIMARY_KEY;
ALTER TABLE ALUMNO ADD CONSTRAINT PRIMARY_KEY (DNI, CURSO);
```

► **ALTER TABLE** (continuación):

- Modificación de una columna. Cuando se necesita realizar un ALTER COLUMN , es posible cambiar el **2º ALTER** por CHANGE o MODIFY
 - **CHANGE**: permite modificar el nombre de la columna, tipo o tamaño.
 - **MODIFY/ALTER** : no permiten modificar el nombre de la columna, solo tipo o tamaño

► **ALTER TABLE:** Añadir clave foránea

```
ALTER TABLE sala
ADD CONSTRAINT fkcine
FOREIGN KEY (idCine)
REFERENCES cine (idCine)
ON UPDATE CASCADE;
```

```
-- con esta opcion se asigna nombre
-- automáticamente a la fk
ALTER TABLE sala
ADD FOREIGN KEY (idCine) -- idCine de sala
REFERENCES cine (idCine); -- idCine, pk de cine
```

► Cambiar campo para que sea **autonumérico**:

```
ALTER TABLE `cliente`
MODIFY `idCliente` int(11) NOT NULL
AUTO_INCREMENT, AUTO_INCREMENT=8; -- inicia en 8
```

*Un autonúmero no es necesario
incluirlo en la sentencia insert*

DDL – DROP/RENAME

15

- ▶ **DROP** : Elimina el objeto de la base de datos

- TABLE / INDEX

```
DROP TABLE ALUMNO;  
DROP INDEX NOM_INDICE ON NOMBRETABLA;  
-- EQUIVALENTE A LA ANTERIOR :  
DROP INDEX NOMBRETABLA.NOM_INDICE;
```

- DATABASE

```
DROP DATABASE nombre_bbdd;
```

- ▶ **RENAME** : Renombra tablas e índices

- TABLE / INDEX

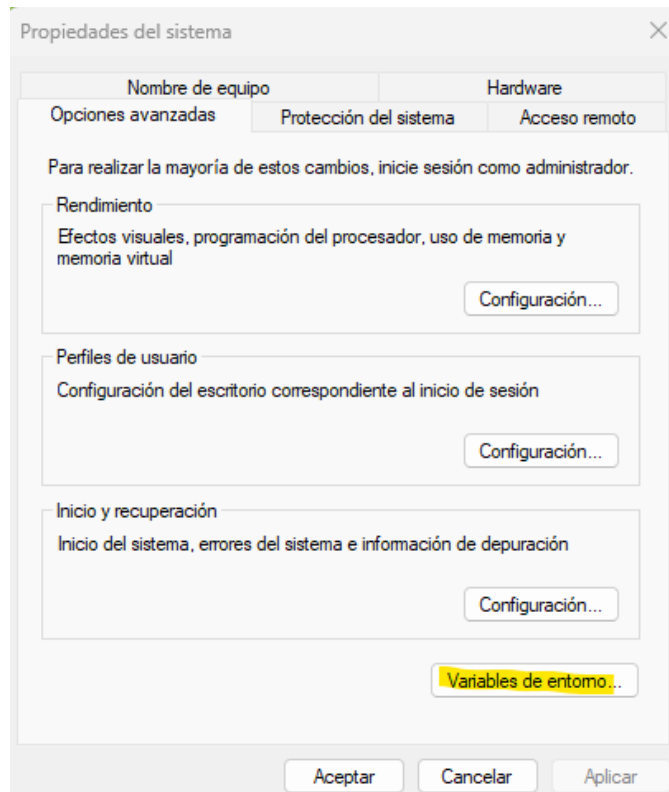
```
RENAME TABLE NOMBRE_ACTUAL TO NOMBRE_NUEVO;  
RENAME INDEX NOMBRE_ACTUAL TO NOMBRE_NUEVO;
```

- ▶ **TRUNCATE** : Se aplica a las tablas realizando un vaciado de todos los datos.

```
TRUNCATE TABLE SALA;
```

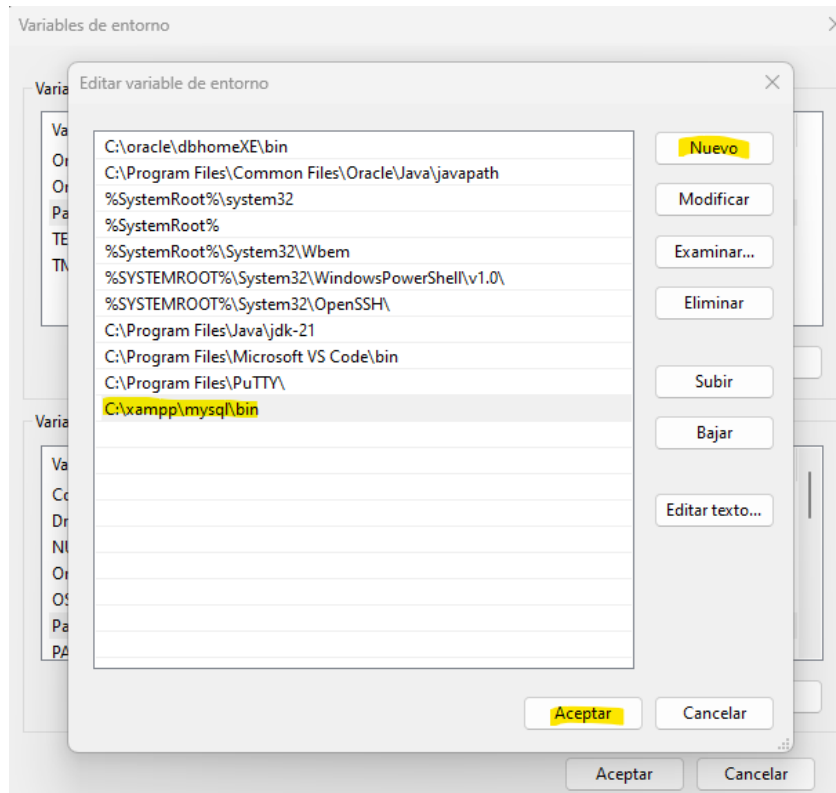
- ▶ **DESCRIBE** : Muestra la estructura de una tabla.
- ▶ **SHOW**: Los comandos show muestran información sobre los diferentes objetos de la base de datos:
 - ▶ show tables
 - ▶ show create table nombre_tabla
 - ▶ etc

- Añadir **mysql** al **PATH** del sistema para poder ejecutarlo desde cualquier directorio



1. Buscar “Editar las variables de entorno del sistema”
2. Click en Variables de Entorno
3. Variables del Sistema. Seleccionar Path y Editar.

- Añadir mysql al PATH del sistema:



1. Buscar "Editar las variables de entorno del sistema"
2. Click en Variables de Entorno
3. Variables del Sistema. Seleccionar Path y Editar.
4. Añadir el directorio en el que se encuentra instalado mysql (C:\xampp\mysql\bin)

MYSQL (source y scripts.sql)

19

- ▶ Ejecutar scripts con extensión .sql
 - ▶ Para la carga de datos en las tablas, en las que interesa garantizar que se utilice el juego de caracteres adecuado:

```
C:\Users\CFGs>mysql --default-character_set=utf8mb4 -u usuario nombrebd < LoadTable.sql
```

- ▶ También se pueden ejecutar scripts .sql desde mysql, con el comando '**source**'

```
MariaDB [cine]> source script.sql
```

** Hay que tener en cuenta la ubicación del archivo .sql, si no se encuentra en el directorio actual, se tiene que indicar la ruta absoluta del archivo al ejecutar el comando.*