Unidad 4

# El lenguaje SQL. DDL

## Gestión de bases de datos

# Índice

4.1. SQL. Tipos de lenguajes

4.2. Tipos de datos en SQL

4.3. Instalación de MySQL en Debian

4.4. Lenguaje de definición de datos (DDL)

4.4.1. Definición de bases de datos
4.4.2. Definición de tablas
4.4.3. Definición de índices
4.4.4. Definición de tipos de datos

# Introducción

Tiempo después de que E.F Codd creara el modelo relacional, dos trabajadores de IBM Research CeNter definieron un cierto lenguaje de programación que nos permitiera interactuar con los datos almacenados en una base de datos. Como el lenguaje era muy similar a la lengua hablada y cotidiana, en inglés, el primer nombre que recibió este lenguaje fue SEQUEL (structured English query language), que significa lenguaje de consulta estructurado en inglés. Este nombre tenía ciertos choques con otro en lo referente a Copyright y decidieron llamarlo por su nombre actual, SQL.

SQL es un lenguaje de alto nivel que coge sus bases de la teoría del álgebra y cálculo relacional y tiene la intención de comunicarse con un sistema gestor de bases de datos.

Este lenguaje nos ayudará a todo tipo de interacción con la base de datos, desde realizar consultas hasta definir la estructura de la base de datos.

Por último, hay que tener en cuenta que el lenguaje SQL es un lenguaje de programación no procedimental que solo sirve para ejecutar acciones en bases de datos y carece de elementos de otros lenguajes de programación.

# Al finalizar esta unidad

- Conoceremos los condicionantes en que surgió el lenguaje SQL,
- Habremos comprendido el ámbito y estructura del lenguaje SQL.
- Estaremos familiarizados con los tipos de datos del SQL estándar.
- Seremos capaces de crear, modificar y eliminar objetos de la base de datos mediante el DDL.

# 4.1. SQL. Tipos de lenguajes

Las distintas funciones que se pueden realizar con SQL como lenguaje se pueden dividir en tres sublenguajes:

- **DDL (data definition language).** Lenguaje de definición de datos, nos facilita la creación de la estructura de la base de datos, tablas y la propia base de datos.
- **DML (data manipulation language).** Lenguaje de manipulación de datos, es el lenguaje que nos permite trabajar sobre los datos que contiene la base de datos como la modificación o consulta de los datos.
- **DCL (data control language).** Lenguaje de control de datos, es el lenguaje que usaremos para administrar la seguridad de los datos mediante permisos y usuarios.

Como en el lenguaje SQL no hay un a universalidad debido a que se ha ido implementando de distintos modos a lo largo de los años, nosotros vamos a seguir la sintaxis SQL estándar, que es la recogida en la versión SQL:1999.

# 4.2. Tipos de datos en SQL

El estándar de SQL nos define los siguientes tipos de datos, que agruparemos por familias:

### Cadenas de caracteres

- **CHAR (CHARACTER):** se trata de una cadena de caracteres que tienen una longitud fija y que habrá que especificar. Si no se especifica nada, será de un solo carácter.

  Si tenemos que la longitud del dato que almacenamos es menor de la definida, esto se rellenará con espacio. Por lo que podemos decir que realmente el número es un máximo de caracteres a escribir por el usuario. Un ejemplo sería almacenar un DNI en un campo 'cDNI' que se definiría como CHAR(9).

- **VARCHAR (CHARACTER VARYING):** es una cadena de caracteres en la que su longitud puede variar, pero habrá que definir una longitud máxima.

  La diferencia con la anterior es que, aunque se almacene un dato de una longitud menor a la definida, solo se almacena lo que introducimos y no se rellena de ningún modo. Un ejemplo sería el campo 'cNombre' de tipo VARCHAR(60), donde Juan sería un nombre admitido sin problema.

- **CLOB (CHARACCTER LARGE OBJECT):** se trata de una cadena de caracteres que tienen un gran tamaño y por eso se almacenará en un archivo a parte al de la tabla. Algunas veces hay ciertos SGBD que limitan el trabajo con este tipo de datos.

## Números exactos

Su uso debe de ser para almacenar código o valores numéricos necesarios para realizar operaciones. El ejemplo más común es el de un número de teléfono, que, aunque sea exacto, debe de almacenarse como una cadena de caracteres porque no se va a operar con él y no sirve de identificador. Los tipos son los siguientes:

- **INT (INTEGER):** definimos en este tipo los números enteros que se pueden almacenar en 4 bytes. El rango comprendido va desde el – 2 147 483 648 al 2 147 483 647.
- **SMALLINT:** comprende los números enteros almacenados en 2 bytes, su rango ahora va desde el – 32 768 al 32 767.
- **NUMERIC o DEC (DECIMAL):** es un número que tienen una parte decimal de tamaño fijo y para eso deberemos de especificar el número de dígitos totales y el de dígitos decimales que se separan por una coma. Un ejemplo sería el campo 'nNotaSelectividad' que se podría definir como NUMERIC(4,2), dos dígitos enteros y dos decimales.

## Números aproximados

Nos dan la opción de que el rango donde movernos en valores decimales sea mayor a los datos NUMERIC, pero también menos exactos. Sus tipos son:

- **FLOAT o REAL:** definimos aquí un número real en coma flotante.
- **DOUBLE:** se define un número real en coma flotante, pero con doble precisión.

## Fechas, horas e intervalos

- **DATE:** almacenamos la fecha en formato año-mes-día.
- **TIME:** almacenamos la hora en formato hora-minuto-segundo.
- **TIMESTAMP:** se almacena la fecha y la hora en la que se modifique el registro afectado y su formato es año-mes-día-hora-minuto-segundo.
- **INTERVAL:** representa un intervalo a lo largo del tiempo y tenemos dos formatos, años-meses o días-horas-minutos-segundos.

Los SGBD almacenan normalmente los datos de una fecha en concreto como todos los segundos que han transcurrido desde el día 1 de enero de 1970.

Este formato se llama Unix time y es común para otros tipos de software.

## Valores lógicos

- **BOOLEAN:** se incluyen aquí os valores "true" y "false" para dictaminar si algo es verdadero o falso y dependiendo de la implementación también se podría almacenar el valor NULL (nulo).

**Objetos binarios**

> **BLOB (BINARY LARGE OBJECT):** almacenas imágenes, documentos, u otros objetos de carácter binario que normalmente se almacenan en un archivo fuera de la tabla en la que estén referenciados.

**Tipos definidos por el usuario. Los veremos más adelante con detalle.**

# 4.3. Instalación de MySQL en Debian

Para poder trabajar con el lenguaje de bases de datos SQL debemos de tener un Sistema Gestor de Bases de datos instalado.

Existen numerosos sistemas gestores de bases de datos, pero nos vamos a centrar en uno, MySQL Server que se trata de un software libre muy usado en el día a día.

A continuación, vamos a ver el proceso de instalación de este SGBD en un sistema operativo Debian:

1. Lo primero que debemos de hacer es abrir el terminal del sistema.
2. Ahora, ejecutamos el comando su root para loguearnos como superusuario.
3. Hay que añadir el repositorio donde se encuentra el programa en cuestión, ya que no viene por defecto con Debian. Para realizar esto lanzamos el siguiente comando:

```
echo "deb http://ftp.debian.org/debian unstable
main contrib" > /etc/apt/sources.list.d/debian.list
```

```
root@debian:/home/miguel# echo "deb http://ftp.debian.org/debian unstable main contrib"
 > /etc/apt/sources.list.d/debian.list
root@debian:/home/miguel#
```

Imagen 1. Añadir el repositorio para instalar MySQL.

4. Una vez que se ha añadido el repositorio, actualizamos las listas de paquetes con los comandos apt update y apt upgrade.
5. Si ya se han actualizado los paquetes, es el momento de instalar nuestro sistema gestor de bases de datos, con el comando:

```
apt install mysql-server
```

```
root@debian:/home/miguel# apt install mysql-server
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no
son necesarios.
  cpp-10 gir1.2-handy-0.0 libamtk-5-0 libamtk-5-common libcbor0
  libcmis-0.5-5v5 libdpkg-perl libextutils-pkgconfig-perl
  libfile-fcntllock-perl libfluidsynth2 libfwupdplugin1 libidn11 libigdgmm11
  libjuh-java libjurt-java liblibreoffice-java libllvm11 liblua5.2-0 libmms0
  libmusicbrainz5-2 libmusicbrainz5cc2v5 libneon27-gnutls libntfs-3g883
  libofa0 liborcus-0.16-0 liborcus-parser-0.16-0 libperl5.32 libqrcodegencpp1
  libquvi-0.9-0.9.3 libquvi-scripts-0.9 libridl-java libtepl-5-0
  libunoloader-java libusb-0.1-4 libxmlb1 linux-image-5.10.0-9-amd64 lua-bitop
  lua-expat lua-json lua-lpeg lua-socket perl-modules-5.32 pkg-config
Utilice «apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  libaio1 libcgi-fast-perl libcgi-pm-perl libevent-core-2.1-7
  libevent-pthreads-2.1-7 libfcgi-bin libfcgi-perl libfcgi0ldbl
  libhtml-template-perl libmecab2 libprotobuf-lite23 mecab-ipadic
  mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0
  mysql-common mysql-server-8.0 mysql-server-core-8.0
Paquetes sugeridos:
  libipc-sharedcache-perl mailx tinyca
```

Imagen 2. Instalación de MySQL.

6. Y con esto ya tendríamos instalado nuestro sistema gestor de bases de datos y listo para funcionar.

**RECOMENDADO**

Una vez que se hace una instalación de MySQL, este trae una opción que es recomendable que activemos, que es mysql_secure_installation.

Esta opción nos otorga la posibilidad de darle al SGBD una mayor seguridad con respecto a usuarios y contraseñas.

A continuación, se van a describir los pasos a seguir para realizar esta instalación que no es obligatoria, pero sí recomendada.

1. Lo primero que vamos a hacer es lanzar en nuestro terminal el comando:

   ```
   mysql_secure_installation
   ```

2. Si todo funciona como debe, automáticamente nos saldrá un bloque de opciones que hay que configurar:

   a. Lo primero que se nos pide es elegir la complejidad de la contraseña como vemos en la siguiente imagen, en nuestro caso vamos a elegir una complejidad media.

```
root@debian:/home/miguel# mysql_secure_installation

Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD COMPONENT can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD component?

Press y|Y for Yes, any other key for No: y

There are three levels of password validation policy:

LOW    Length >= 8
MEDIUM Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary
           file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1
```

Imagen 3. mysql_secure_isntallation 1.

b. Ahora ponemos esta contraseña siguiendo las restricciones que antes se nos nombraban.

c. Damos a todo que sí, fijándonos en la última opción de la siguiente imagen. Con esta pedimos que se borren todos los usuarios anónimos con el fin de siempre saber quién se encuentra conectado a nuestra base de datos.

```
Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1
Please set the password for root here.

New password:

Re-enter new password:

Estimated strength of the password: 100
Do you wish to continue with the password provided?(Press y|Y for Yes, any other
 key for No) : y
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.
```

Imagen 4. mysql_secure_installation 2.

d. Deshabilitamos el inicio remoto desde nuestra base de datos, aunque más adelante podremos volver a activarlo.

e. Se elimina la base de datos de prueba.

```
Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.


Remove test database and access to it? (Press y|Y for Yes, any other key for No)
 : y
 - Dropping test database...
Success.

 - Removing privileges on test database...
Success.
```

Imagen 5. mysql_secure_installation 3.

f. Volvemos a cargar todos los permisos de las tablas y ya habríamos terminado.

```
Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : n

 ... skipping.
All done!
```

Imagen 6. mysql_secure_installation 4.

Por último, vamos a ver como acceder a la base de datos, ya que tenemos dos medios:

- Si estamos logueados como roota en el terminal, bastará con ejecutar la sentencia mysql y automáticamente entraremos en la consola del SGBD.

```
root@debian:/home/miguel# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.23-3+b1 (Debian)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Imagen 7. MySQL.

- La otra opción que tenemos es la de solicitarle a la base de datos que queremos entrar en la consola como root. Para esto ejecutamos el comando:

  `mysql -u root (-p)`

La opción -p es si queremos que se nos pida una contraseña.

```
root@debian:/home/miguel# mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.23-3+b1 (Debian)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Imagen 8. mysql -u root.

En el ANEXO I referente a esta unidad encontraremos algunos comandos de la consola de MySQL que nos facilitarán el trabajo.

# 4.4. Lenguaje de definición de datos (DDL)

Como se ha comentado anteriormente, la función del lenguaje DDL es la de crear, modificar, y borrar objetos presentes en una base de datos y también la propia base de datos.

## 4.4.1. Definición de bases de datos

### Para crear una base de datos.

```
CREATE DATABASE nombre_bd;
```

En el siguiente ejemplo creamos una base de datos que se va a llamar bduniversae.

```
mysql> create database bduniversae;
Query OK, 1 row affected (0.01 sec)
```

Imagen 9. Ejemplo de creación de base de datos.

Ahora lanzamos el siguiente comando para ver si la base de datos se ha añadido al sistema:

```
show databases;
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| bduniversae        |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.05 sec)
```

Imagen 10. Bases de datos del sistema.

Si nos fijamos, MySQL trae por defecto cuatro bases de datos para la gestión del SGBD, entonces con la que hemos creado serían 5 las bases de datos que tenemos.

### Para borrar una base de datos

```
DROP DATABASE nombre_bd;
```

Siguiendo nuestro ejemplo, borramos la base de datos que hemos creado anteriormente.

```
mysql> drop database bduniversae;
Query OK, 0 rows affected (0.01 sec)
```

Imagen 11. Ejemplo de borrado de una base de datos.

Ahora lanzamos de nuevo el comando para ver las bases de datos que tenemos en el SGBD y comprobamos que ha desaparecido.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)
```

Imagen 12. Bases de datos del sistema.

## 4.4.2. Definición de tablas

### Para crear una tabla

```
CREATE TABLE nombre_tabla (
campo1 tipo[(longitud)] [NOT NULL] [UNIQUE] [PRIMARY
KEY] [CHECK condición] [DEFAULT valor][, [
…
campoN … ]
[,  PRIMARY KEY (campos)]
[, FOREIGN KEY (campos) REFERENCES tabla (campos)
[{ON UPDATE [NO ACTION|SET DEFAULT|SET NULL|CASCADE]
[ON DELETE [NO ACTION|SET DEFAULT|SET NULL|CASCADE]]
}|
{ON DELETE [NO ACTION|SET DEFAULT|SET NULL|CASCADE]
[ON UPDATE [NO ACTION|SET DEFAULT|SET NULL|CASCADE]]
}]]
…
);
```

Para poder crear una tabla, primero tendremos que solicitar usar una base de datos en concreto, con el comando:

```
use nombre_bd;
```

```
mysql> use bdempresa;
Database changed
```

Imagen 13. Usar una base de datos.

Vamos a crear ahora una de las tablas que se crearon en ejemplos de la unidad anterior:

```
mysql> create table TProveedor (
    -> nCodProveedorID NUMERIC(5,0) NOT NULL UNIQUE PRIMARY KEY,
    -> cNombreProveedor VARCHAR(50) NOT NULL,
    -> cWeb VARCHAR(50),
    -> nTelefono NUMERIC(9,0) NOT NULL UNIQUE,
    -> cDireccion VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.07 sec)
```

Imagen 14. Ejemplo de creación de una tabla.

Como se puede ver, los campos 'cNombreProveedor' y ' nTelefono' no pueden estar nulos, y además, el campo para guardar los datos del teléfono también debe ser único, pues dos trabajadores no pueden tener el mismo número de teléfono.

```
mysql> create table TProvision (
    -> nCodProveedorID NUMERIC(5,0) NOT NULL UNIQUE,
    -> nProductoID NUMERIC(7,0) NOT NULL UNIQUE,
    -> CONSTRAINT PK_Provision PRIMARY KEY(nCodProveedorID,nProductoID),
    -> FOREIGN KEY (nCodProveedorID) REFERENCES TProveedor (nCodProveedorID)
    -> ON UPDATE CASCADE
    -> ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.03 sec)
```

Imagen 15. Creación de tabla con clave foránea.

Ahora hemos creado una tabla adicional, la tabla 'TProvision', y como podemos ver, hemos indicado que hay una doble clave primaria y que además uno de los campos que referencia a la clave primaria hace referencia a la clave primaria de 'TProveedor'. Por último, hemos añadido 'ON UPDATE CASCADE ON DELETE CASCADE', que quiere decir que, si se actualiza la clave principal o se elimina un registro, se actualice dicha información en la clave foránea.

## Para modificar una tabla

Para realizar modificaciones en tablas usamos la sentencia **ALTER TABLE** y con esta podemos:

### > Añadir un campo:

```
ALTER TABLE nombre_tabla
ADD campo tipo[(longitud)] [NOT NULL] [UNIQUE]
[PRIMARY KEY]
[CHECK condición] [DEFAULT valor];
```

Vamos a añadir a la tabla 'TProveedor' un campo 'cCorreo' donde escribir la dirección de correo electrónico.

```
mysql> ALTER TABLE TProveedor
    -> ADD cCorreo VARCHAR(256);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Imagen 16. Añadir un campo a una tabla.

### > Modificar un campo:

```
ALTER TABLE nombre_tabla
MODIFY COLUMN campo [tipo(longitud)] | [DROP DEFAULT];
```

**DROP DEFAULT** se usa para borrar el dato que se haya puesto por defecto, si es que lo hay.

Antes habíamos creado el campo 'nTelefono' como numérico, pero realmente si vemos la explicación que dimos anteriormente de los tipos de datos, sería una cadena de caracteres, puesto que no se van a realizar operaciones con los números de teléfono, vamos a cambiarlo.

```
mysql> ALTER TABLE TProveedor
    -> Modify Column nTelefono VARCHAR(9);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Imagen 17. Modificar un campo de una tabla.

> **Borrar un campo:**

```
ALTER TABLE nombre_tabla
DROP campo;
```

Borramos el campo anteriormente creado.

```
mysql> ALTER TABLE TProveedor
    -> DROP cCorreo;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Imagen 18. Imagen 18. Borrar el campo de una tabla.

**Para eliminar una tabla**

```
DROP TABLE nombre_tabla;
```

Vamos a eliminar ahora la última tabla creada, 'TProvision'.

```
mysql> drop table TProvision
    -> ;
Query OK, 0 rows affected (0.02 sec)
```

Imagen 19. Ejemplo de borrado de una tabla.

Para comprobar el estado de una tabla y que campos tiene junto con sus características, se usa el comando:

```
DESC nombre_tabla;
```

Describimos a continuación la tabla que nos queda, que es 'TProveedor'.

```
mysql> desc TProveedor;
+------------------+--------------+------+-----+---------+-------+
| Field            | Type         | Null | Key | Default | Extra |
+------------------+--------------+------+-----+---------+-------+
| nCodProveedorID  | decimal(5,0) | NO   | PRI | NULL    |       |
| cNombreProveedor | varchar(50)  | NO   |     | NULL    |       |
| cWeb             | varchar(50)  | YES  |     | NULL    |       |
| nTelefono        | varchar(9)   | YES  | UNI | NULL    |       |
| cDireccion       | varchar(50)  | YES  |     | NULL    |       |
+------------------+--------------+------+-----+---------+-------+
5 rows in set (0.02 sec)

mysql>
```

Imagen 20. Ejemplo de descripción de una tabla.

## 4.4.3. Definición de índices

Los índices se usan para marcar ciertos campos con la intención de asignarles una importancia o relevancia para luego poder trabajar sobre ellos con mayor facilidad.

A continuación, se muestra una sintaxis algo genérica sobre cómo se trabaja con los índices:

**Para crear un índice**

```
CREATE [UNIQUE] INDEX nombre_indice
ON nombre_tabla
(campo1 [ASC|DESC][, campo2 [ASC|DESC][, …, campoN
[ASC|DESC]]);
```

Si especificamos ASC O DESC en alguno de los campos estamos haciendo referencia a que el orden del índice será ascendente o descendente, por defecto será descendente.

Vamos a crear un índice para agrupar el nombre y el teléfono de un proveedor.

```
mysql> CREATE INDEX iProveedor_Nombre_Telefono
    -> ON TProveedor
    -> (cNombreProveedor, nTelefono);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

Imagen 21. Ejemplo de creación de índice.

### Para borrar un índice

```
DROP INDEX nombre_indice
ON nombre_tabla;
```

Borramos el índice anteriormente creado.

```
mysql> DROP INDEX iProveedor_Nombre_Telefono on TProveedor;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Imagen 22. Ejemplo de borrado de índice.

En MySQL también se puede usar la modificación de tablas para borrar índices como en el ejemplo siguiente:

```
mysql> alter table TProveedor
    -> drop index iProveedor_Nombre_Telefono;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

Imagen 23. Otra forma de borrar índices.

## 4.4.4. Definición de tipos de datos

En algunas bases de datos el usuario puede crear tipos como quiera para reutilizarlo a la hora de crear o modificar campos para las tablas. Estos tipos nuevos realmente son alias de otros tipos.

La sentencia para crear tipos es:

```
CREATE TYPE nombre_tipo AS tipo[(longitud)];
```

En MySQL, por ejemplo, no funciona la creación de tipos.

www.universae.com