Preparador Informática

MANUAL 3

BASES DE DATOS

SQL (Básico)

1. Introducción

SQL (Structured Query Language) es el lenguaje fundamental de los SGBD relacionales. Es uno de los lenguajes más utilizados en informática en todos los tiempos. SQL surgió como una evolución de SEQUEL (Structured English Query Language), lenguaje definido por IBM en 1977. ANSI lo definió como estándar unos años más tarde (1986), y fue adoptado también por ISO al año siguiente. Es un lenguaje declarativo y, por tanto, lo más importante es definir qué se desea hacer, y no cómo hacerlo. De esto último ya se encarga el SGBD. Hablamos por tanto de un lenguaje normalizado que nos permite trabajar con cualquier tipo de lenguaje (ASP o PHP) en combinación con cualquier tipo de base de datos (Access, SQL Server, MySQL, Oracle, etc.). El lenguaje SQL puede ser utilizado tanto de forma interactiva (en una consola del SGBD) como inmerso dentro de un lenguaje anfitrión.

SQL posee dos características muy apreciadas, potencia y versatilidad, que contrastan con su facilidad para el aprendizaje, ya que utiliza un lenguaje bastante natural. Es por esto que las instrucciones son muy parecidas a órdenes humanas. Por esta característica se le considera un Lenguaje de Cuarta Generación. Aunque frecuentemente se oye que SQL es un "lenguaje de consulta", ten en cuenta que no es exactamente cierto ya que contiene muchas otras capacidades además de la de consultar la base de datos, como son la definición de la propia estructura de los datos, su manipulación, y la especificación de conexiones seguras.

2. Elementos del lenguaje. Normas de escritura.

El lenguaje SQL está compuesto por comandos, cláusulas, operadores, funciones y literales. Todos estos elementos se combinan en las instrucciones y se utilizan para crear, actualizar y manipular bases de datos.

- **COMANDOS**: Van a ser las instrucciones que se pueden crear en SQL. Se pueden distinguir en tres grupos:
    - **De definición de datos (DDL)**, que permiten crear y definir nuevas bases de datos, tablas, campos, etc.

Comandos DDL. Lenguaje de Definición de Datos.	
Comando:	**Descripción:**
CREATE	Se utiliza para crear nuevas tablas, campos e índices.
DROP	Se utiliza para eliminar tablas e índices.
ALTER	Se utiliza para modificar tablas.

    - **De manipulación de datos (DML)**, que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos, así como inserciones, modificaciones y eliminación de datos.

Comandos DML. Lenguaje de Manipulación de Datos.	
Comando:	**Descripción:**
SELECT	Se utiliza para consultar filas que satisfagan un criterio determinado.
INSERT	Se utiliza para cargar datos en una única operación.
UPDATE	Se utiliza para modificar valores de campos y filas específicos.
DELETE	Se utiliza para eliminar filas de una tabla.

- **De control y seguridad de datos (DCL, Data Control Language)**, que administran los derechos y restricciones de los usuarios.

Comandos DCL. Lenguaje de Control de Datos.	
Comando:	**Descripción:**
GRANT	Permite dar permisos a uno o varios usuarios o roles para realizar tareas determinadas.
REVOKE	Permite eliminar permisos que previamente se han concedido con GRANT.

- **CLÁUSULAS:** Llamadas también condiciones o criterios, son palabras especiales que permiten modificar el funcionamiento de un comando.

Cláusulas	
Cláusulas:	**Descripción:**
FROM	Se utiliza para especificar la tabla de la que se van a seleccionar las filas.
WHERE	Se utiliza para especificar las condiciones que deben reunir las filas que se van a seleccionar.
GROUP BY	Se utiliza para separar las filas seleccionadas en grupos específicos.
HAVING	Se utiliza para expresar la condición que debe satisfacer cada grupo.
ORDER BY	Se utiliza para ordenar las filas seleccionadas de acuerdo a un orden específico.

- **OPERADORES:** Permiten crear expresiones complejas. Pueden ser aritméticos (+, -, *, /, ...) o lógicos (< , >, , < >, And, Or, etc.).

Operadores de comparación.	
Operadores:	**Descripción:**
<	Menor que.
>	Mayor que.
< >	Distinto de.
< =	Menor o igual.
> =	Mayor o igual.
=	Igual.
BETWEEN	Se utiliza para especificar un intervalo de valores.
LIKE	Se utiliza para comparar.
IN	Se utiliza para especificar filas de una base de datos.

Operadores lógicos.	
Operadores:	**Descripción:**
AND	Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Devuelve el valor contrario de la expresión.

- **FUNCIONES:** Para conseguir valores complejos. Existen muchas funciones distintas, por ejemplo:

Funciones de agregado.	
Función:	**Descripción:**
AVG	Calcula el promedio de los valores de un campo determinado.
COUNT	Devuelve el número de filas de la selección.
SUM	Devuelve la suma de todos los valores de un campo determinado.
MAX	Devuelve el valor más alto de un campo determinado.
MIN	Devuelve el valor mínimo de un campo determinado.

- **LITERALES:** Les podemos llamar también constantes y serán valores concretos, como por ejemplo un número, una fecha, un conjunto de caracteres, etc.

Literales	
Literales:	**Descripción:**
23/03/97	Literal fecha.
María	Literal caracteres.
5	Literal número.

Y, además, tendremos que seguir unas normas sencillas pero primordiales:

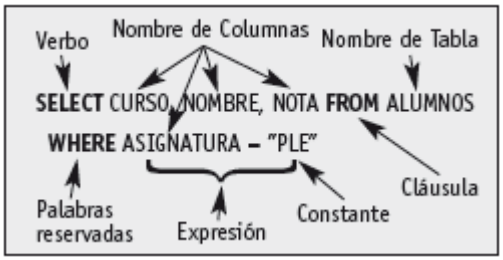
- Todas las instrucciones terminan con un signo de punto y coma.
- No se distingue entre mayúsculas y minúsculas.
- Cualquier comando puede ser partido con saltos de línea o espacios para facilitar su lectura y comprensión.
- Los comentarios comienzan por /* y terminan con */ (excepto en algunos SGBD).

3. Tipos de sentencias SQL

Resumiendo, los tipos de sentencias de las que dispone SQL exponiendo las sentencias más importantes y las usadas con mayor frecuencia y organizadas dependiendo de las tareas son:

	SENTENCIA	DESCRIPCIÓN
DML	*Manipulación de datos*	
	SELECT	Recupera datos de la base de datos.
	INSERT	Añade nuevas filas de datos a la base de datos.
	DELETE	Suprime filas de datos de la base de datos.
	UPDATE	Modifica datos existentes en la base de datos.
DDL	*Definición de datos*	
	CREATE TABLE	Añade una nueva tabla a la base de datos.
	DROP TABLE*	Suprime una tabla de la base de datos.
	ALTER TABLE*	Modifica la estructura de una tabla existente.
	CREATE VIEW*	Añade una nueva vista a la base de datos.
	DROP VIEW*	Suprime una vista de la base de datos.
	CREATE INDEX*	Construye un índice para una columna.
	DROP INDEX*	Suprime el índice para una columna.
	CREATE SYNONYM*	Define un alias para un nombre de tabla.
	DROP SYNONYM*	Suprime un alias para un nombre de tabla.
DCL	*Control de acceso*	
	GRANT	Concede privilegios de acceso a usuarios.
	REVOKE	Suprime privilegios de acceso a usuarios.
	Control de transacciones	
	COMMIT	Finaliza la transacción actual.
	ROLLBACK	Aborta la transacción actual.

Casi todas las sentencias SQL tienen una forma básica. Todas comienzan con un verbo, que es una palabra clave que describe lo que hace la sentencia (por ejemplo, SELECT, INSERT, UPDATE, CREATE). A continuación, le siguen una o más cláusulas que especifican los datos con los que opera la sentencia. Estas también comienzan con una palabra clave como WHERE o FROM. Algunas son opcionales y otras obligatorias. Muchas contienen nombres de tablas o de columnas, palabras reservadas, constantes o expresiones adicionales.



4. Tipos de datos

Cuando se crea una tabla instrucción CREATE TABLE se debe especificar el tipo de dato para cada una de sus columnas, estos tipos de datos definen el dominio de valores que cada columna puede contener. Los tipos de datos SQL se clasifican en 13 tipos de datos primarios y de varios sinónimos válidos reconocidos por dichos tipos de datos. Los tipos de datos primarios son:

Tipo de Datos	Longitud	Descripción
BINARY	1 byte	Para consultas sobre tabla adjunta de productos de bases de datos que definen un tipo de datos Binario.
BIT	1 byte	Valores Si/No ó True/False
BYTE	1 byte	Un valor entero entre 0 y 255.
COUNTER	4 bytes	Un número incrementado automáticamente (de tipo Long)
CURRENCY	8 bytes	Un entero escalable entre 922.337.203.685.477,5808 y 922.337.203.685.477,5807.
DATETIME	8 bytes	Un valor de fecha u hora entre los años 100 y 9999.
SINGLE	4 bytes	Un valor en punto flotante de precisión simple con un rango de - 3.402823*1038 a -1.401298*10-45 para valores negativos, 1.401298*10- 45 a 3.402823*1038 para valores positivos, y 0.
DOUBLE	8 bytes	Un valor en punto flotante de doble precisión con un rango de - 1.79769313486232*10308 a -4.94065645841247*10-324 para valores negativos, 4.94065645841247*10-324 a 1.79769313486232*10308 para valores positivos, y 0.
SHORT	2 bytes	Un entero corto entre -32,768 y 32,767.
LONG	4 bytes	Un entero largo entre -2,147,483,648 y 2,147,483,647.
LONGTEXT	1 byte por carácter	De cero a un máximo de 1.2 gigabytes.
LONGBINARY	Según se necesite	De cero 1 gigabyte. Utilizado para objetos OLE.
TEXT	1 byte por carácter	De cero a 255 caracteres.

La siguiente tabla recoge los sinónimos de los tipos de datos definidos dependiendo del SGBD:

Tipo de Dato	Sinónimos
BINARY	VARBINARY
BIT	BOOLEAN LOGICAL LOGICAL1 YESNO
BYTE	INTEGER1
COUNTER	AUTOINCREMENT

CURRENCY	MONEY
DATETIME	DATE TIME TIMESTAMP
SINGLE	FLOAT4 IEEESINGLE REAL
DOUBLE	FLOAT FLOAT8 IEEEDOUBLE NUMBER NUMERIC
SHORT	INTEGER2 SMALLINT
LONG	INT INTEGER INTEGER4
LONGBINARY	GENERAL OLEOBJECT
LONGTEXT	LONGCHAR MEMO NOTE
TEXT	ALPHANUMERIC CHAR - CHARACTER STRING - VARCHAR

De todo lo anterior los más habituales y utilizados son los siguientes, (aunque se han mostrado los sinónimos que tienen cada uno de ellos y en los ejercicios se podrán utilizar indistintamente):

Tipo	Bytes	Descripción
INT o INTEGER	4	Números enteros. Existen otros tipos de mayor o menor longitud específicos de cada base de datos.
DOUBLE o REAL	8	Números reales (grandes y con decimales). Permiten almacenar todo tipo de número no entero.
CHAR	1/caracter	Alfanuméricos de longitud fija predefinida
VARCHAR	1/caracter+1	Alfanuméricos de longitud variable
DATE	3	Fechas, existen multiples formatos específicos de cada base de datos
BOOLEAN	1	Almacenan un bit de información (verdadero o falso)

En este manual y en las prácticas utilizaré "**MySQL**" para ejecutar las sentencias de SQL, por lo que veamos cuales son los tipos de datos que utiliza y los rangos de los mismos. Repasando la documentación oficial podríamos dividir en 3 grandes grupos estos datos:

- Numéricos
- Fecha
- String

Tipos de dato numéricos

Listado de cada uno de los **tipos de dato numéricos en MySQL**, su ocupación en disco y valores.

- **INT (INTEGER)**: Ocupación de 4 bytes con valores entre -2147483648 y 2147483647 o entre 0 y 4294967295.
- **SMALLINT**: Ocupación de 2 bytes con valores entre -32768 y 32767 o entre 0 y 65535.
- **TINYINT**: Ocupación de 1 bytes con valores entre -128 y 127 o entre 0 y 255.
- **MEDIUMINT**: Ocupación de 3 bytes con valores entre -8388608 y 8388607 o entre 0 y 16777215.
- **BIGINT**: Ocupación de 8 bytes con valores entre -8388608 y 8388607 o entre 0 y 16777215.
- **DECIMAL (NUMERIC)**: Almacena los números de coma flotante como cadenas o string.
- **FLOAT (m,d)**: Almacena números de **coma flotante**, donde 'm' es el número de dígitos de la parte entera y 'd' el número de decimales.
- **DOUBLE (REAL)**: Almacena número de coma flotante con precisión doble. Igual que FLOAT, la diferencia es el rango de valores posibles.
- **BIT (BOOL, BOOLEAN)**: Número entero con valor 0 o 1.

Tipos de dato con formato fecha

Listado de cada uno de los **tipos de dato con formato fecha en MySQL**, su ocupación en disco y valores.

- **DATE**: Válido para almacenar una fecha con año, mes y día, su rango oscila entre '1000-01-01' y '9999-12-31'.
- **DATETIME**: Almacena una fecha (año-mes-día) y una hora (horas-minutos-segundos), su rango oscila entre '1000-01-01 00:00:00' y '9999-12-31 23:59:59'.
- **TIME**: Válido para almacenar una hora (horas-minutos-segundos). Su rango de horas oscila entre -838-59-59 y 838-59-59. El formato almacenado es 'HH:MM:SS'.
- **TIMESTAMP**: Almacena una fecha y hora UTC. El rango de valores oscila entre '1970-01-01 00:00:01' y '2038-01-19 03:14:07'.
- **YEAR**: Almacena un año dado con 2 o 4 dígitos de longitud, por defecto son 4. El rango de valores oscila entre 1901 y 2155 con 4 dígitos. Mientras que con 2 dígitos el rango es desde 1970 a 2069 (70-69).

Tipos de dato con formato string

Listado de cada uno de los **tipos de dato con formato string en MySQL**, su ocupación en disco y valores.

- **CHAR**: Ocupación fija cuya longitud comprende de 1 a 255 caracteres.
- **VARCHAR**: Ocupación variable cuya longitud comprende de 1 a 255 caracteres.

- **TINYBLOB**: Una longitud máxima de 255 caracteres. Válido para objetos binarios como son un fichero de texto, imágenes, ficheros de audio o vídeo. No distingue entre minúsculas y mayúsculas.
- **BLOB**: Una longitud máxima de 65.535 caracteres. Válido para objetos binarios como son un fichero de texto, imágenes, ficheros de audio o vídeo. No distingue entre minúsculas y mayúsculas.
- **MEDIUMBLOB**: Una longitud máxima de 16.777.215 caracteres. Válido para objetos binarios como son un fichero de texto, imágenes, ficheros de audio o vídeo. No distingue entre minúsculas y mayúsculas.
- **LONGBLOB**: Una longitud máxima de 4.294.967.298 caracteres. Válido para objetos binarios como son un fichero de texto, imágenes, ficheros de audio o vídeo. No distingue entre minúsculas y mayúsculas.
- **SET**: Almacena 0, uno o varios valores una lista con un máximo de 64 posibles valores.
- **ENUM**: Igual que **SET** pero solo puede almacenar un valor.
- **TINYTEXT**: Una longitud máxima de 255 caracteres. Sirve para almacenar texto plano sin formato. Distingue entre minúsculas y mayúsculas.
- **TEXT**: Una longitud máxima de 65.535 caracteres. Sirve para almacenar texto plano sin formato. Distingue entre minúsculas y mayúsculas.
- **MEDIUMTEXT**: Una longitud máxima de 16.777.215 caracteres. Sirve para almacenar texto plano sin formato. Distingue entre minúsculas y mayúsculas.
- **LONGTEXT**: Una longitud máxima de 4.294.967.298 caracteres. Sirve para almacenar texto plano sin formato. Distingue entre minúsculas y mayúsculas.

5. Pasos para implementar una base de datos

PASO	DESCRIPCIÓN	SQL
1	Crear la base de datos	CREATE DATABASE…
2	Crear las diferentes tablas	CREATE TABLE…
3	Insertar las filas de las diferentes tablas	INSERT INTO…
4	Actualizar los datos que cambien con el tiempo en las tablas si es necesario	UPDATE…
5	Eliminar las filas que ya no se requieran en las diferentes tablas	DELETE…
6	Realizar las consultas deseadas	SELECT…
7	Crear vistas si es necesario Crear índices si es necesario	CREATE VIEW… CREATE INDEX…

6. Creación de la base de datos

Básicamente, la creación de la base de datos consiste en crear las tablas que la componen. Con el estándar de SQL la instrucción a usar sería Create Database, pero cada SGBD tiene un procedimiento para crear las bases de datos. Crearíamos una base de datos con el nombre que se indique a continuación:

```
CREATE DATABASE NombredemiBasedeDatos;
```

En caso de que quisiéramos eliminar una base de datos usamos:

```
DROP DATABASE NombredemiBasedeDatos;
```

Si queremos ver las bases de datos que tenemos creadas usamos:

```
show databases;
```

Cuando queremos empezar a utilizar una base de datos ya creada utilizamos:

```
USE NombredemiBasedeDatos;
```

7. Creación de tablas.

Los objetos básicos con los que trabaja SQL son las tablas, que como ya sabemos es un conjunto de filas y columnas cuya intersección se llama celda. Es ahí donde se almacenarán los elementos de información, los datos que queremos recoger. Antes de crear la tabla es conveniente planificar algunos detalles:

- Qué nombre le vamos a dar a la tabla.
- Qué nombre le vamos a dar a cada una de las columnas.
- Qué tipo y tamaño de datos vamos a almacenar en cada columna.
- Qué restricciones tenemos sobre los datos.
- Alguna otra información adicional que necesitemos.

La sintaxis básica del comando que permite crear una tabla es la siguiente:

```
CREATE TABLE [esquema.] NombredeTabla (
columna1 Tipo_Dato [ModificadordeTipo],
columna2 Tipo_Dato [ModificadordeTipo],
…
columnaN Tipo_Dato [ModificadordeTipo]  ) ;
```

donde:

columna1, columna2, ..., columnaN son los nombres de las columnas que contendrá la tabla.

Tipo_Dato indica el tipo de dato de cada columna. Podremos usar alguno de los vistos en apartados anteriores.

ModificadordeTipo/Restricciones: Una restricción es una condición que una o varias columnas deben cumplir obligatoriamente. Entre otras podemos destacar NULL (Admite

valores nulos), NOT NULL (no permite valores nulos) y UNIQUE (no admite valores repetidos). Al definir la tabla podemos indicar cuál de las columnas constituye la clave principal (PRIMARY KEY) que es el cual nos permite distinguir entre varios registros de datos y cuáles son las claves secundarias, indicando el nombre del atributo y la tabla con el que se relaciona (REFERENCES FOREIGN KEY). Las restricciones CKECK exigen la integridad del dominio mediante la limitación de los valores que puede aceptar una columna.

Hemos de hacer algunas observaciones:

- Las definiciones individuales de columnas se separan mediante comas.
- No se pone coma después de la última definición de columna.
- Las mayúsculas y minúsculas son indiferentes a la hora de crear una tabla.

Ejemplo 1:

```
CREATE TABLE ALUMNOS
(
      NUMERO_MATRICULA INT NOT NULL PRIMARY KEY,
      NOMBRE VARCHAR(15) NOT NULL,
      FECHA_NACIMIENTO DATE,
      DIRECCION VARCHAR(30),
      LOCALIDAD VARCHAR(15)
);
```


NOTA: Si al crear la tabla tenemos errores de sintaxis se nos avisa y si se ha creado con éxito nos dará también un mensaje sobre ello. Para comprobar las tablas que hay creadas podemos usar el comando:

```
SHOW TABLES;
```

Y si queremos ver las características de una tabla en concreto podemos usar el comando:

```
DESCRIBE NombredemiTabla;
```

Ejemplo 2:

```
CREATE TABLE EMPLEADO
(
      NOMBRE VARCHAR(25) PRIMARY KEY,
      EDAD TINYINT CHECK (EDAD BETWEEN 18 AND 35),
      COD_PROVINCIA TINYINT
);
```

Ejemplo 3:

```
CREATE TABLE PROVIN
(
      CODIGO TINYINT PRIMARY KEY,
      NOMBRE VARCHAR(25)
);

CREATE TABLE EMPLEADO
(
      NOMBRE VARCHAR(25),
      EDAD TINYINT,
      COD_PROVINCIA TINYINT,
      CONSTRAINT PK_EMPLEADO PRIMARY KEY (NOMBRE),
      CONSTRAINT CK_EDAD CHECK(EDAD BETWEEN 18 AND 35),
      CONSTRAINT FK_EMPLEADO FOREIGN KEY (COD_PROVINCIA)
      REFERENCES PROVIN (CODIGO) ON DELETE CASCADE
);
```

El nombre de las restricciones es opcional. También es válida esta sentencia CREATE TABLE:

```
CREATE TABLE EMPLEADO
(
      NOMBRE VARCHAR(25),
      EDAD TINYINT,
      COD_PROVINCIA TINYINT,
      PRIMARY KEY (NOMBRE),
      CHECK (EDAD BETWEEN 18 AND 35),
      FOREIGN KEY (COD_PROVINCIA) REFERENCES PROVIN (CODIGO)
      ON DELETE CASCADE
);
```

Ejemplo 4:

```
CREATE TABLE PROVINCIAS
(
      CODPROVINCIA TINYINT PRIMARY KEY,
      NOM_PROVINCIA VARCHAR(15)
);
```

```
CREATE TABLE PERSONAS
(
      DNI INT PRIMARY KEY,
      NOMBRE VARCHAR(15),
      DIRECCION VARCHAR(25),
      POBLACION VARCHAR(20),
      CODPROVIN TINYINT NOT NULL,
      FOREIGN KEY (CODPROVIN) REFERENCES PROVINCIAS
      (CODPROVINCIA) ON DELETE CASCADE
);
```

Ejemplo 5:

```
CREATE TABLE EJEMPLO5
(
      NIF VARCHAR(9) PRIMARY KEY,
      NOMBRE VARCHAR(30),
      FECHA DATE
);
```

Ejemplo 6:

```
CREATE TABLE EJEMPLO6
(
      NIF VARCHAR(9) NOT NULL PRIMARY KEY,
      NOMBRE VARCHAR(30) NOT NULL,
      EDAD TINYINT CHECK (EDAD BETWEEN 5 AND 20),
      CURSO TINYINT CHECK (CURSO IN(1, 2, 3))
);
```

Ejemplo 7:

```
CREATE TABLE EJEMPLO7
(
      NIF VARCHAR(9) PRIMARY KEY,
      NOM VARCHAR(30) UNIQUE,
      EDAD TINYINT
);
```

8. Modificación de tablas.

a) Si queremos cambiar el nombre de una tabla:

```
RENAME TABLE NombreViejo TO NombreNuevo;
```

b) Si queremos añadir columnas a una tabla: las columnas se añadirán al final de la tabla.

```
ALTER TABLE NombreTabla ADD
( ColumnaNueva1 Tipo_Dato [ModificadordeTipo],
ColumnaNueva2 Tipo_Dato [ModificadordeTipo],
…
ColumnaNuevaN Tipo_Dato [ModificadordeTipo]  );
```

c) Si queremos eliminar columnas de una tabla: se eliminará la columna indicada sin poder deshacer esta acción. Además, se eliminan todos los datos que contenga la columna.

```
ALTER TABLE NombreTabla DROP COLUMN (Columna1 [, Columna2, …] );
```

d) Si queremos modificar columnas de una tabla: podemos modificar el tipo de dato y las propiedades de una columna. Todos los cambios son posibles si la tabla no contiene datos.

```
ALTER TABLE NombreTabla MODIFY (Columna1 Tipo_Dato
[ModificadordeTipo] [, columna2 Tipo_Dato [ModificadordeTipo]
...] );
```


e) Si queremos renombrar columnas de una tabla:

```
ALTER TABLE NombreTabla RENAME COLUMN NombreAntiguo TO
NombreNuevo;
```

f) Podemos añadir y eliminar las siguientes restricciones de una tabla: CHECK, PRIMARY KEY, NOT NULL, FOREIGN KEY y UNIQUE. Para añadir restricciones usamos la orden:

```
ALTER TABLE nombretabla ADD CONSTRAINT nombrerestricción ...
```

Para eliminar restricciones usamos la orden:

```
ALTER TABLE nombretabla DROP CONSTRAINT nombrerestriccion ...
```

OJO: En la mayoría de los SGBD se usa DROP CONSTRAINT, en MySQL hay que especificar el tipo de restricción. Por ejemplo:

- UNIQUE:

  ```
  ALTER TABLE nombretabla DROP INDEX  nombrerestriccion;
  ```

- PRIMARY KEY:

  ```
  ALTER TABLE nombretabla DROP PRIMARY KEY;
  ```

- FOREIGN KEY:

  ```
  ALTER TABLE nombretabla DROP FOREIGN KEY nombrerestriccion;
  ```

Ejemplo 1:

```
ALTER TABLE EJEMPLO7 ADD (SEXO CHAR(1));
```

Ejemplo 2:

```
ALTER TABLE EMPLEADO ADD (SALARIO DOUBLE);
ALTER TABLE EMPLEADO ADD CONSTRAINT SALMAYOR CHECK (SALARIO > 0);
```

Ejemplo 3:

```
ALTER TABLE EMP ADD CONSTRAINT APELLIDO_UQ UNIQUE(APELLIDO);
```

Ejemplo 4:

```
ALTER TABLE EMP DROP CONSTRAINT APELLIDO_UQ,
(MySQL)ALTER TABLE EMP DROP INDEX APELLIDO_UQ,
```


9. Eliminación de tablas.

Podemos eliminar una tabla siempre y cuando contemos con privilegios para ello. En tal caso debemos escribir:

```
DROP TABLE NombreTabla [CASCADE];
```

CASCADE elimina las restricciones de integridad referencial que remitan a la clave primaria de la tabla borrada.

10. Inserción de datos

Empezamos la manipulación de datos de una tabla con la orden INSERT. Con ella se añaden filas de datos en una tabla. El formato de esta orden es el siguiente:

```
INSERT INTO NombreTabla [(columna [, columna] ...)] VALUES (valor
[, valor] ...);
```

Donde:

NombreTabla es la tabla en la que se van a insertar las filas.

[(columna [, columna] ...)] representa la columna o columnas donde se van a introducir valores. Si las columnas no se especifican en la cláusula INSERT, se consideran, por defecto, todas las columnas de la tabla.

(valor [, valor] ...) representa los valores que se van a dar a las columnas. Éstos se deben corresponder con cada una de las columnas que aparecen; además, deben coincidir con el tipo de dato definido para cada columna. Cualquier columna que no se encuentre en la lista de columnas recibirá el valor NULL, siempre y cuando no esté definida como NOT NULL, en cuyo caso INSERT fallará. Si no se da la lista de columnas, se han de introducir valores en todas las columnas.

Ejemplo: Tenemos la tabla PROFESORES con los siguientes atributos:

```
COD_CENTRO TINYINT NOT NULL
DNI INT PRIMARY KEY
APELLIDOS VARCHAR(30)
ESPECIALIDAD VARCHAR(15)
```


Damos de alta a una profesora con estos apellidos y nombre: 'Pérez Fernández, Ana', de la especialidad 'INFORMÁTICA' y con el código de centro 78. Las columnas a las que damos valores son: APELLIDOS, ESPECIALIDAD y COD_CENTRO:

```
INSERT INTO PROFESORES (APELLIDOS, ESPECIALIDAD, COD_CENTRO,
DNI)
VALUES ('Pérez Fernández, Ana', 'INFORMÁTICA', 78, 14151617);
```

Insertamos ahora a un profesor de apellidos y nombre 'Sánchez Márquez, Mario' en el código de centro 34, con DNI 14141414 y de la especialidad de 'SAI':

```
INSERT INTO PROFESORES VALUES (34, 14141414, 'Sánchez Márquez,
Mario', 'SAI');
```

No es necesario especificar el nombre de las columnas ya que las hemos dado un valor en la fila que insertamos para todas las columnas. Este valor ha de ir en el mismo orden en que las columnas estén definidas en la tabla.

11. Modificación de datos

Para actualizar los valores de las columnas de una o varias filas de una tabla utilizamos la orden UPDATE, cuyo formato es el siguiente:

```
UPDATE NombreTabla SET columna1=valor1, ..., columnan=valorn
WHERE condición;
```

Donde:

- *NombreTabla es la tabla cuyas columnas se van a actualizar.*
- *SET indica las columnas que se van a actualizar y sus valores.*
- *WHERE selecciona las filas que se van a actualizar. Si se omite, la actualización afectará a todas las filas de la tabla.*

Ejemplo:

```
UPDATE CENTROS SET DIRECCION = 'C/Pilón 13', NUM_PLAZAS = 295
WHERE COD_CENTRO = 22;
```

12. Borrado de datos

Para eliminar una fila o varias filas de una tabla se usa la orden DELETE. La cláusula WHERE es esencial para eliminar sólo aquellas filas deseadas. Sin la cláusula WHERE, DELETE borrará todas las filas de la tabla. Éste es su formato:

```
DELETE FROM NombreTabla WHERE Condición;
```

Ejemplo:

```
DELETE FROM CENTROS WHERE COD_CENTRO=50;
```

13. Rollback, commit y autocommit

La mayoría de los SGBD incluyen las funciones rollback, commit y autocommit. Supongamos que queremos borrar una fila de una tabla, pero, al teclear la orden SQL, se nos olvida la cláusula WHERE y borramos todas las filas de la tabla. Esto no es problema pues podemos dar marcha atrás a un trabajo realizado mediante la orden ROLLBACK, siempre y cuando no hayamos validado los cambios en la base de datos mediante la orden COMMIT.

Cuando hacemos transacciones sobre la base de datos, es decir, cuando insertamos, actualizamos y eliminamos datos en las tablas, los cambios no se aplicarán a la base de datos hasta que no hagamos un COMMIT. Esto significa que, si durante el tiempo que hemos estado realizando transacciones, no

hemos hecho ningún COMMIT y de pronto se va la luz, todo el trabajo se habrá perdido, y nuestras tablas estarán en la situación de partida.

Para validar los cambios que se hagan en la base de datos tenemos que ejecutar la orden COMMIT:

```
COMMIT;
```

En algunos sistemas se nos permite validar automáticamente las transacciones sin tener que indicarlo de forma explícita. Para eso sirve el parámetro AUTOCOMMIT. El valor de este parámetro se puede mostrar con la orden SHOW, de la siguiente manera:

```
SHOW AUTOCOMMIT;
```

OFF es el valor por omisión, de manera que las transacciones (INSERT, UPDATE y DELETE) no son definitivas hasta que no hagamos COMMIT. Si queremos que INSERT, UPDATE Y DELETE tengan un carácter definitivo sin necesidad de realizar la validación COMMIT, hemos de activar el parámetro AUTOCOMMIT con la orden SET:

```
SET AUTOCOMMIT ON;
```

Ahora, cualquier INSERT, UPDATE y DELETE se validará automáticamente.

La orden ROLLBACK aborta la transacción volviendo a la situación de las tablas de la base de datos desde el último COMMIT siempre y cuando no esté activado el AUTOCOMMIT:

```
ROLLBACK;
```

OJO: En MySQL el autocommit viene activado por defecto (AUTOCOMMIT=1) y en el caso de que no lo queramos tener activo debemos utilizar:

```
SET AUTOCOMMIT=0
```