Las principales **caracteristicas** de MySQL son:

- Permite acceder a su código fuente.

- Es capaz de funcionar en diferentes plataformas (Windows, Linux u OS X) por lo que no estás atado a una plataforma.

- Puede trabajar con más de un procesador simultáneamente.

- Exisen multitud de herramientas compatibles.

- Existen infinidad de lenguajes de programación capaces de acceder a MySQL.

- Es muy facil encontrar solución a tus problemas ya que tiene una comunidad muy amplia.

Para **acceder a MySQL** desde la terminal hay que escribir:

mysql -u root -p (al pulsar intro te pedira que introduzcas la clave).

Para **abandonar el monitor** hay que escribir ***quit***.

El comando **show databases** permite visualizar las bases de datos activas.

**Ejemplo**:

mysql>

show databases;

+————+

| Database |

+————+

| mysql |

| test |

+————+

2 rows in set (0.01 sec)

El comando **use** nos permite seleccionar que base de datos vamos a utilizar.

**Ejemplo**: "mysql> use test"

Pare **Crear Bases de Datos** se utiliza el comando create **database nombreElegido.**

**Ejemplo**:

mysql> create database nombreElegido;

Query OK, 1 row affected (0.00 sec)

Despues de crearla como hemos indicado mas arriba para utilizarla abrá que indicarlo con use.

En el caso de que queramos **borrar una posible base de datos creada anteriormente con el mismo nombre** que la que vamos a crear podemos utilizar el comando **drop database if exists nombreElegido;**

**Ejemplo:**

mysql>drop database if exists nombreElegido;

Query OK, 0 rows affected (0.00 sec)

Con el comando **show create database nombreElegido;** se pueden ver la configuracion de la base de datos creada.

Los tipos de tablas más utilizados son : InnoDB, MyISAM, MEMORY y MERGE.

Para **indicar que tipo de tabla** queremos crear se pondria de la siguiente forma:

USE nombreElegido;

DROP TABLE IF EXISTS usuario;

CREATE TABLE usuario

(

idusuario INT NOT NULL,

edad INT NOT NULL DEFAULT 0

)

**ENGINE = InnoDB;**

Si no se indica de que tipo se quiere **por defecto será InnoDB.**

Para crear tablas dentro de la base de datos debes de haber echo un "use" hacia la base de datos que se quiere cerar las tablas.

Para **crear las tablas** se utiliza el comando ***Create table nombreElegido*;** , este comando requiere el nombre de la tabla y una lista de definiciones de campos separados por coma.

**Ejemplo**:

mysql> CREATE TABLE escritor (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(200) NOT NULL,

apellido VARCHAR(150),

edad int NOT NULL,

description TEXT);

Query OK, 0 rows affected (0.37 sec)

Las definiciones de la tabla es todo lo que va entre parentesís.

Para **visualizar la estructura de una tabla** se utiliza el comando ***Describe nombreElegido;***

Los tipos de datos para **texto** son: CHAR, VARCHAR, BINARY (equivale al tipo CHAR), VARBINARY (equivale al tipo varchar), BLOB, TEXT (este a diferencia de BLOB no distinge entre mayusculas y minúnsculas) y ENUM.

Tipos de **datos Numéricos:**

Enteros: TINYINT (rango -128 a 127 con signo, o de 0 a 255 sin signo), SMALLINT (rango -32768 a 32767, si no incluye signo 65535), MEDIUMINT (rango -8388608 a 8388607, sin signo 16777215), INT (rango -2147483648 a 2147483647, sin signo 4294967295) y BIGINT ( rango -9223372036854775808 hasta 9223372036854775807)

Para indicar que queremos **almacenar enteros sin signo** le podemos asignar **UNSIGNED**.

Ejemplo:

create table prueba (

id\_prueba int UNSIGNED

);

Para almacenar **numeros con parte decimal** se utiliza **FLOAT** o **DOUBLE.**

Ejemplo:

create table prueba (

id\_prueba int UNSIGNED

cantidad float (6,2) "con 6,2 se esta indicando que se quiere de **6 digitos y 2 decimales**"

);

**Para insertar FECHAS y HORAS** tenemos varios comandos:

DATE: AAAA-MM-DD (año, mes y día), entre 1000-01-01 y 9999-12-31.

TIME: hh:mm:ss

DATETIME: AAAA-MM-DD hh:mm:ss

YEAR: indicado con 2 o 4 digitos.

Para **BORRAR una tabla** se utiliza el comando **DROP TABLE nombreTabla;**

**Modificaciones de la tabla.**

Agregar un nuevo campo**: ALTER TABLE nombreTabla ADD COLUMN nombreColumna varchar(255)**-> esto seria los parámetros de configuración.

Eliminar columna de una tabla: **ALTER TABLE nombreTabla DROP COLUMN nombreColumna;**

Para insertar un nuevo campo en una posicion de la tabla se utiliza FIRST,AFTER y sin nada lo pondría al final de la tabla como predeterminado.

Ejemplo:

**ALTER TABLE** prueba **add column** nombre VARCHAR(255) **AFTER** id\_prueba;

Esto me insertaria la nueva columna detras de id\_prueba.

Para **cambiar el nombre de una columna** se utiliza **ALTER TABLE nombreTabla CHANGE COLUMN nombreAcambiar nombreNuevo + atributos;**

Con este ejemplo también se podría **cambiar los atributos** simplemente añadiendole los atributos que se quiera al final.

Para modificar los atributos o tipos de dato tambien se podria hacer con MODIFY COLUMN

Ejemplo: **ALTER** **TABLE** nombreTabla **MODIFY** **COLUMN** name varchar(59);

Añadir **FOREIGN** **KEY** a un campo de una tabla.

Se utiliza **ALTER** **TABLE** nombreTabla **ADD** **FOREIGN** **KEY** (nombre del campo que lleva la clave) **REFERENCES** nombreTabla(nombrecampoDeReferencia);

**Comando INSERT INTO.**

Este comando se utiliza para insertar valores en una tabla su sintaxis es la siguiente:

INSERT INTO nombreTabla (columna1,columna2,columna3,...) values (valorcolumna1,valorcolumna2,valorcolumna3,..)

Se podria añadir varios valores siguiento esa sintaxis añadiendo una coma al final e añadiendo entre parentesis como las primeras insercciones.

Ejemplo:

**INSERT** **INTO** nombreTabla (columna1,columna2,columna3,...) **values** (valorcolumna1,valorcolumna2,valorcolumna3,..), (valorcolumna1, valorcolumna2, valorcolumna3,…)

Si no se especifican los parentesis de despues de nombreTabla por defecto cogería el orden en el que se hubiesen insertado.

**Actualización** de la tabla con el comando **UPDATE**.

La sintaxis básica de este comando es **UPDATE nombreTabla SET columna1 = valor1, columna2 = valor2 , …**

Lo que hace el comando update lo que hace es actualizar los campos con de la tabla con nuevos valores.

La cláusula **SET** indica los campos a modificar y los valores compatibles.

También se puede asignar una expresión a un campo:

**UPDATE** t2 **SET** total = cuantía \* precio;

Si en la expresión se utiliza el mismo campo que se quiere asignar se coge el valor que tiene al principio.

Esta sentencia se puede restringir con el comando WHERE.

Ejemplo:

**UPDATE nombreTabla SET columna1 = valor1, columna2 = valor2,… WHERE condiciónLogica.**

Con esto solo se modificarían las filas que cumplan la condiciónLogica.

También existe la posibilidad de limitar el alcance de la actualización a un numero de registros.

Ejemplo:

**UPDATE nombreTabla SET columna1 = valor1, columna2 = valor2, … WHERE condiciónLogica LIMIT numero;**

Para **borrar registros** de una tabla se utiliza el comando **DELETE**. Su sintaxis es la siguiente:

**DELETE** **FROM** nombreTabla **WHERE** condiciónLogica;

También se podría limitar como hemos visto anteriormente con LIMIT.

**Consulta de datos.**

El comando **SELECT … FROM** tabla se indica después del **SELECT** la lista de columnas que queremos seleccionar separadas por comas y detrás del **FROM** se pondrían las tablas a las que se le hace referencia también separadas por comas. Si después del **SELECT** se pone un asterisco \* se está indicando que se cogen todos los campos de las tablas nombradas en el **FROM**.

En esta sentencia también se puede utilizar el comando LIMIT para limitar a cierta cantidad de filas.

Ejemplo: **SELECT** \* **FROM** prueba **LIMIT** 10,5 (esto mostraría 5 filas después del décimo registro).

Con el comando ORDER BY se puede ordenar los datos resultantes de la consulta por el campo se que quiera y también se puede indicar si se quiere de modo descendente con “desc” o ascendente con el comando “asc”.

Ejemplo: SELECT \* FROM prueba WHERE condiciónLogica ORDER BY nombre desc LIMIT 10;

Con los operadoras **IS NULL** e **IS NOT NULL** se verifica si la expresión determinada es nula o no.

Existe también un operador para comprobar si una expresión está comprendida en un determinado rango de valores. Este operador es el siguiente:

<expresión>**BETWEEN** mínimo **AND** máximo .

<expresión>**NOT** **BETWEEN** mínimo **AND** máximo .

Operadores lógicos:

Operador Y = AND o &&.

Operador O = OR o ||

Operador de negación = NOT o ¡.

Operadores para cadenas:

El operador **LIKE** se usa para hacer referencias entre cadenas y patrones, devuelve 1 si la cadena se ajusta al patrón y 0 en caso contrario. Su sintaxis es:

<expresión> **LIKE** <patrón>[ESCAPE ‘carácter\_escape’]

También se puede usar **NOT** **LIKE** que realiza la inversa de **LIKE**.

**Funciones matemáticas:**

ABS Devuelve el valor absoluto

CEILING y

CEIL Redondeo hacia arriba

DEGREES Conversión de grados a radianes

EXP Cálculo de potencias de e

FLOOR Redondeo hacia abajo

MOD o % Resto de una división entera

PI Valor del número π

POW o POWER Valor de potencias

RADIANS Conversión de radianes a grados

RAND Valores aleatorios

ROUND Cálculo de redondeos

SIGN Devuelve el signo

SQRT Cálculo de la raíz cuadrada

TRUNCATE Elimina decimales

La cláusula **GROUP BY** se usa para agrupar filas que tienen los mismos valores.

Ejemplo: **SELECT** \* **FROM** nombreTabla **GROUP** **BY** nombreColumna.

También permite agrupar por mas de una columna separando con una coma el nobreColumna.

Se pueden restringir consultas usando HAVING. Sirve para las ocasiones en las que queremos restringir nuestros resultados con un determinado criterio.

Ejemplo: **SELECT \* FROM películas GROUP BY id\_categoria,anio\_lanzamiento HAVING id\_categoria = 8;**

**Subconsultas**.

Una subconsulta es una instrucción SELECT anidada dentro de otra instrucción HAVING o WHERE. Su estructura es la siguiente:

SELECT listaColumnas FROM listaTablas WHERE <expresión><condición>(SELECT nombreColumna FROM listaTablas WHERE condición).

Esto permite realizar la unión de dos o más tablas.

Ejemplo **test de comparación**: SELECT ci,nombre,apellido FROM empleado WHERE dno = (SELECT dnumero FROM departamento WHERE dnombre = “Investigación);

Ejemplo **test de conjunto**: SELECT ci, nombre, apellido FROM empleado WHERE ci IN (SELECT eci FROM carga\_f WHERE YEAR(fecha\_naciemiento) BETWEEN ‘1980’ AND ‘1990’);

También se puede utilizar su negación que seria NOT IN.

Ejemplo **test de existencia**: SELECT nombre,apellido,dno FROM empleado WHERE EXISTS (SELECT \* FROM departamento WHERE dnumero = “4” AND dno = dnumero);

También se puede utilizar su negación que seria NOT EXISTS

Ejemplo **test cuantificados**:

**ANY**: SELECT dnombre FROM departamento WHERE dnumero = ANY(SELECT dno FROM empleado WHERE salario = “2500”);

Tambíen se puede realizar con SOME.

**ALL**: SELECT e.nombre, e.apelldido FROM empleado e WHERE e.salario > ALL (SELECT s.salario FROM empleado s, empleado e WHERE e.superci = s.ci AND e.superci is not null);

**Uso de JOIN para consultas.**

Tipos:

INNER JOIN o JOIN – Este tipo de unión te ayuda a combinar varias tablas, y te devuelve únicamente los datos que estén disponibles en todas las tablas a la vez.

Ejemplo: select f.dep\_nom, f.relacion from empleado e inner join carga\_f f on e.ci = f.eci where nombre = "Elena" and apellido = "Tapia";

LEFT JOIN – Devuelve todos los resultados que coincidan en la primera tabla, con los datos que tenga de la segunda.

Ejemplo: SELECT usuarios.username, juegos.juegoname FROM usuarios LEFT JOIN juegousuario ON usuarios.ID = juegousuario.ID\_usuario LEFT JOIN juegos ON juegousuario.ID\_juego = juegos.ID

RIGHT JOIN- es como la anterior pero con la diferencia de que devuelve todos los datos de la tabla con la que sse relaciona la anterior.

Ejemplo: SELECT usuarios.username, juegos.juegoname FROM usuarios RIGHT JOIN juegousuario ON usuarios.ID = juegousuario.ID\_usuario RIGHT JOIN juegos ON juegousuario.ID\_juego = juegos.ID

OUTER JOIN o FULL OUTER JOIN - recupera todos los datos que haya en ambas tablas, tanto los que tienen contenido en ambos extremos, como los que no. Para poder hacer este tipo de consultas tendrás que echar mano de UNION, que sirve precisamente para combinar los resultados de varios SELECT en una sola consulta.