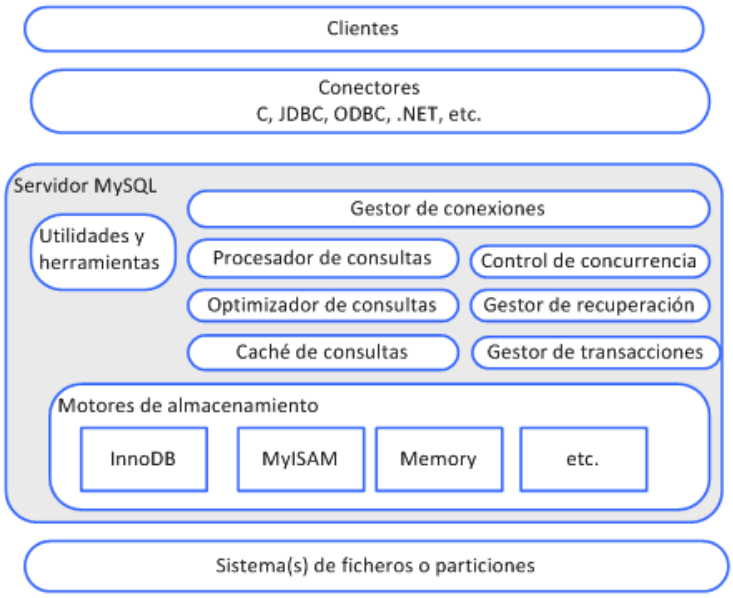
CONOCIMIENTO PREVIOS PARA EL ACCESO A LOS DATOS DE UNA BASE DE DATOS DESDE PHP

1. MySQL.

MySQL es un SGBD relacional, fácil de administrar y utilizar**, basado en el modelo cliente/servidor** y por ello es preciso disponer de:

- Clientes, a través de los cuales se accede al servidor MySQL. Estos pueden ser:
  - Modo texto: es decir, se accede al servidor desde la consola o línea de comandos.(ej: mysql)
  - Modo gráfico:
    - Workbench
    - PhpMyAdmin
    - MySQLCC
    - Otros (https://byspel.com/5-excelentes-clientes-para-bases-datos-mysql/)
- Servidor, programa que proporciona las bases de datos que se encuentren almacenadas en memoria o disco duro.



componentes de MySQL

Un Gestor de Bases de Datos sirve de intermediario entre los datos y los programas utilizados para su manipulación.

Las funciones esenciales de un GBD son:

- **Proporcionar un diccionario de datos o catálogo** accesible por el usuario que contiene la descripción de los datos de la base de datos (metadatos).
- **Proporcionar un lenguaje de descripción o definición (DDL).** Sirve para describir los objetos del sistema de información, las relaciones entre ellos y además las restricciones de integridad que haya. Se utiliza en el momento del diseño lógico de la BD y en las sucesivas modificaciones de su estructura.
- **Proporcionar un lenguaje de manipulación de datos (DML).** Se utiliza para insertar, modificar, suprimir y consultar información de la BD. Además, en muy habitual que proporcione instrucciones condicionales, iterativas y de asignación como un lenguaje de programación convencional.
- **Proporcionar un lenguaje de control de datos (DCL)**. Con él se dispone de herramientas para:
  - Crear usuario y otorgar privilegios
  - Obtener copias de seguridad
  - Reconstruir una BD si está dañada
  - Monitorizar la BD, etc.

Crear objetos de la BD	Sentencia CREATE	**Sentencias de definición de datos**
Eliminar objetos de la BD	Sentencia DROP	

Modificar objetos de la BD	Sentencia ALTER	**(DDL)**
Consultar la BD	Sentencia SELECT	**Sentencias de manipulación de datos (DML)**
Añadir tuplas a las tablas de la BD	Sentencia INSERT	
Eliminar tuplas de las tablas de la BD	Sentencia DELETE	
Modificar tuplas de las tablas de la BD	Sentencia UPDATE	
Crear privilegios de acceso a los datos	Sentencia GRANT	**Sentencias de control de datos (DCL)**
Quitar privilegios de acceso a los datos	Sentencia REVOKE	

- **Mantener la Integridad.** El diseño de una BD obliga a la implantación de una serie de restricciones que aseguren la consistencia de los datos y es el Gestor el encargado de que dichas restricciones se respeten.
- **Mantener la seguridad de acceso.** Es por ello que también es el encargado de que se respeten las restricciones impuestas a los usuarios con respecto al acceso a los datos.
- **Seguridad y recuperación de datos.** Debe permitir la recuperación de los datos al punto previo a un fallo en la BD.
- **Acceso concurrente.** Teniendo en cuenta que varios usuarios pueden acceder a la vez a un mismo dato, debe encargarse de que los datos obtenidos por todos ellos sean siempre consistentes.
- **Interacción con el gestor de archivos.** La estructura utilizada para almacenar los archivos varía en función del sistema operativo de una máquina. El GBD al actuar de intermediario entre los datos a nivel de almacenamiento físico y las aplicaciones diseñadas para su manipulación, debe transformar las órdenes propias de la manipulación de datos en un lenguaje que comprenda el sistema operativo.

2. LENGUAJE DE DEFINICIÓN DE DATOS (DDL): CREATE, DROP y ALTER.

2.1. CREATE.

Antes de crear las tablas de una BD, debemos crear la propia BD.

Para ello se utiliza también la sentencia CREATE, teniendo en cuenta el siguiente formato:

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_specification] ...

create_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
```

Por ejemplo, CREATE DATABASE IF NOT EXISTS Ejemplo

Creada la BD, se crearán las tablas a partir del diseño Relacional obtenido, teniendo en cuenta **las restricciones (CONSTRAINTS)** derivadas de él, para el almacenamiento correcto de la información y su integridad. Las restricciones pueden ser definidas a nivel de columna o campo o a nivel de tabla.

Para crear una tabla utilizaremos el siguiente formato de la sentencia CREATE:

```
CREATE TABLE [IF NOT EXISTS] NombreTabla
        (
        NombreColumna TipoDato [Restricciones a nivel de columna]
        [, NombreColumna TipoDato [Restricciones a nivel de columna]…]
        [Restricciones a nivel de tabla [, Restricciones a nivel de tabla]…]
        ) [ENGINE =tipo_motor (ej:InnoBD)];
```

Las restricciones frecuentes son:

- **PRIMARY KEY**, para indicar qué campo o columna es la clave primaria. En caso de estar formada por varios campos o columnas debe también ser definida a nivel de tabla ya que sólo se puede especificar una restricción PRIMARY KEY por tabla.
- **NULL |NOT NULL,** para indicar la posibilidad o no de que un campo o columna tome valor NULO.
- **DEFAULT valor,** de tal forma que, en ausencia de un valor, un campo o columna tome el valor por defecto especificado en esta restricción.
- **UNIQUE,** permitiendo valores únicos para un campo o columna sin ser éste clave primaria.
- **CKECK (expresión),** pudiendo comprobar el valor que toma un campo o columna.
- **REFERENCES NombreTablaReferenciada [(NombreColumnaReferenciada)],** para indicar que una columna o campo es clave foránea o ajena. Si los nombres de las columnas coinciden puede omitirse el nombre de la columna referenciada en esta restricción.

  Recordemos que son las claves foráneas las que permiten mantener la integridad referencial en una BD, de tal manera que un sistema relacional no permitiría borrar (**DELETE**) o modificar (**UPDATE**) una fila de una tabla referenciada, lo cual es una restricción inherente (opción por defecto) a este tipo de sistemas cuando existen tuplas en otra tabla donde un atributo sea clave foránea y se refiera a la anterior. Sin embargo, existen las siguientes opciones al definir una clave foránea, pudiéndose especificar varias en la restricción:

  - **CASCADE**, el borrado o modificación de una fila de la tabla referenciada lleva consigo el borrado o modificación en cascada de las filas de la tabla que contiene la clave ajena.
  - **SET NULL**, el borrado o modificación de una fila de la tabla referenciada pone a NULL los valores de la clave ajena.
  - **SET DEFAULT**, las operaciones anteriores conllevan que la clave ajena tome el valor especificado por defecto.
- **AUTO_INCREMENT**, con él se genera un valor numérico único para cada fila. No es una restricción propiamente dicha, sino más bien una función que se añade al tipo de dato INT con el fin de generar automáticamente el valor de un campo o columna, siendo éste el que tendrá atribuida la función de clave primaria de una tabla.

```
CREATE TABLE certificado (
       denominacion VARCHAR(30) CHECK (denominacion='TITULO %'),
       dni CHAR(10),
       anno emision DATE NOT NULL,
       CONSTRAINT pk certificado PRIMARY KEY (denominacion,dni),
       CONSTRAINT fk demandante certificado FOREIGN KEY (dni)
       REFERENCES demandante (dni) ON DELETE CASCADE
       );
```

```
CREATE TABLE demandante (
       dni CHAR(10) PRIMARY KEY,
       apellidos VARCHAR (15) NOT NULL,
       nombre VARCHAR(15) NOT NULL
       );
```

```
CREATE TABLE prestamos (
       num prestamo INT(2) PRIMARY KEY,
       socio no INT(4),
       CONSTRAINT FK_SOCIO_PRESTAMOS FOREIGN KEY (socio_no)
       REFERENCES socios (socio_no) ON UPDATE SET NULL
       );
```

```
CREATE TABLE socio (
       socio no INT(4) PRIMARY KEY,
       apellidos VARCHAR (15) NOT NULL,
       nombre VARCHAR(15) NOT NULL,
       dirección VARCHAR(45)UNIQUE
       );
```

Ejemplos creación de Tablas

2.2. ALTER.

Con la sentencia ALTER, podremos modificar diferentes elementos. Así, podremos modificar las características generales de la BD con el siguiente formato:

```
    ALTER {DATABASE | SCHEMA} [db_name]
      alter_specification …
   alter_specification:
      [DEFAULT] CHARACTER SET [=] charset_name | [DEFAULT] COLLATE [=]
collation_name
```

Sin embargo, lo que más nos interesa ahora es modificar la estructura de una tabla ya creada. Para ello utilizaremos el siguiente formato genérico que iremos detallando:

```
    ALTER TABLE NombreTabla
      EspecificaciónModificación [, EspecificaciónModificación…]
```

Donde EspecificaciónModificación, hace referencia a las modificaciones más habituales en una tabla:

- Añadir una columna nueva

  `ADD [COLUMN] NombreColumna TipoDato [ResticciónColumna]`

- Borrar una columna

  `DROP [COLUMN] NombreColumna`

- Modificar la definición de una columna manteniendo su nombre

  `MODIFY [COLUMN] NombreColumna TipoDatoNuevo [RestricciónNueva]`

- Modificar la definición de una columna y también su nombre

  `CHANGE [COLUMN] NombreColumnaAntiguo NombreColumnaNuevo TipoDatosNuevo [RestricciónNueva]`

- Añadir una restricción nueva

  `ADD {RestricciónColumna |RestricciónTabla}`

  - `RestricciónColumna: PRIMARY KEY, FOREIGN KEY, UNIQUE ó CHECK`
  - `RestricciónTabla: PRIMARY KEY, FOREIGN KEY, UNIQUE`

- Borrar una restricción

  `DROP {PRIMARY KEY | FOREIGN KEY NombreConstraint}`

- Renombrar la tabla

  `RENAME [TO] NombreTablaNueva`

  Nota: existe otra opción que es utilizando la sentencia `RENAME TABLE NombreAntiguo TO NombreNuevo.`

2.3. DROP.

Como en los apartados anteriores, es posible **borrar la BD**:

```
      DROP {DATABASE | SCHEMA} [IF EXISTS] db_name;
```

Pero lo que más nos interesa en **borrar una tabla**. Para ello utilizaremos el siguiente formato:

```
      DROP TABLE [IF EXISTS] NombreTabla;
```

3. LENGUAJE DE MANIPULACIÓN DE DATOS (DML).

Hasta el apartado anterior básicamente hemos creado la estructura de una base de datos. La actualización de la BBDD se trata en profundidad en el tema 8; sin embargo es necesario adelantar el formato simple de las sentencias que nos permiten INSERTAR, BORRAR y MODIFICAR la información de una tabla.

3.1. INSERT.

Con la sentencia INSERT almacenaremos datos en las tablas creadas.

Formato básico:

```
INSERT INTO NombreTabla [(NombreColumna [,NombreColumna…])]
       {VALUES | VALUE} (lista_valores) [, (lista_valores)]…;
```

Es posible dar valor a todos los campos o sólo a alguno de ellos. Si se da valor a todos los campos, el número de valores de lista debe coincidir con el número de campos o columnas de la tabla.

Ejemplos:

```
# especificando algunos campos de la tabla; el campo Codigo, al ser PRIMARY KEY y AUTO_INCREMENT, toma
# valores correlativos desde el 1

#ejemplo 1:

INSERT INTO Opositor(NombreCompleto, TitulacionAc, Idioma)
       VALUES ("José Javier García","Técnico Superior","español");


#ejemplo 2:

INSERT INTO Opositor(NombreCompleto, TitulacionAc)
       VALUES ("Miguel Román","Arquitecto"),
              ("Ana Iglesias","Técnico Superior Laboratorio"),
              ("Pedro Alonso","Técnico Superior Administración de Sistemas"),
              (""Olga Martínez","Técnico Superior Administración de Sistemas");
```

```
# sin especificar el nombre de los campos, ya que se da valor a todos. Interesante el valor null que se
# utiliza en este caso como comodín para el campo Codigo.


INSERT INTO Opositor
       VALUES (null,"Rosa Gómez","Ingeniera","inglés");
```

3.2. UPDATE.

Son múltiples las ocasiones en las cuales es necesario modificar la información de las tablas. Para poder hacerlo SQL cuenta con la sentencia UPDATE.

```
       UPDATE NombreTabla
              SET NombreColumna=valor,[ NombreColumna=valor]…
              [WHERE condición]
              [ORDER BY…]
              [LIMIT…]
```

3.3. DELETE.

Para el borrado de tuplas utilizaremos el siguiente formato simplificado y sobre una tabla. Si no se especifica la cláusula WHERE se borrará toda la información de la tabla pero no la tabla en sí.

```
DELETE NombreTabla
       [WHERE condición]
```

3.4. SELECT (CONSULTAS SENCILLAS)

El formato que vamos a utilizar para este tipo de consultas será el siguiente:

```
SELECT {*|[ALL/DISTINCT] ExpresiónColumna [[AS] AlíasColumna]
      [,ExpresiónColumna [[AS] AlíasColumna]….] }
FROM NombreTabla [AlíasTabla]
[WHERE CondiciónSelección]
[ORDER BY {ExpresiónColumna | Posición} [ASC|DESC]
      [,{ExpresiónColumna | Posición} [ASC|DESC]…]]
[LIMIT [m, ] n];
```

Explicaciones básicas:

- *, permite recuperar todos los atributos de la tabla
- [ALL/DISTINCT] `ExpresiónColumna`, permite recuperar todos los posibles valores de la `ExpresiónColumna` (opción por defecto) o sólo los distintos. `ExpresiónColumna` puede ser el nombre de un atributo, una expresión formada por nombres de atributos, literales, operadores y funciones.
- AlíasColumna, para dar un nuevo nombre a la columna y hacerlo más legible o para renombrar la columna y utilizar ese nombre más adelante. Si sólo está formado por una palabra, no es preciso el uso de comillas simples o dobles en su definición.
- AlíasTabla, para dar un nuevo nombre a la tabla lo cual será útil más adelante.
- ORDER BY, permite ordenar los datos consultados, de forma ascendente (por defecto) o descendente. Se puede especificar varios sistemas de ordenación, siendo siempre el primer criterio el que primero se haya especificado. Si se utiliza la opción `posición` se hará referencia al número de orden que ocupa una `ExpresiónColumna` en el `SELECT`.
- LIMIT, permite indicar el número de tuplas a recuperar; m, indica el número de la tupla que marca el inicio de la recuperación, siendo por defecto su valor 0 y se corresponde con la primera tupla o fila; n, sería el número de tuplas o filas a recuperar.

ACTIVIDAD 1:

Te resultará útil utilizar un editor de texto sencillo, tipo notepad++, para escribir las sentencias SQL. Después copiar la sentencia y pegarla en la línea de comandos. Además podrás almacenarlas y reutilizarlas.

El punto de partida es el siguiente esquema relacional (CICLOS):

CURSO (**id_curso**, deno);

ALUMNO (**id_alumno**, nombre, edad, id_curso);

Pasos:

Crear la **BD denominada ciclos**.

```
CREATE DATABASE CICLOS;
```

Crear la tabla curso:

```
CREATE TABLE curso (id_curso INT, deno VARCHAR(15), CONSTRAINT pk_curso PRIMARY KEY (id_curso))
ENGINE=”InnoDB”;
```

Crea la tabla alumno con los mismos atributos pero con las restricciones a nivel de tabla y motor InnoDB.

```
CREATE TABLE alumno (id_al INT, nombre VARCHAR(15), edad INT, id_curso INT,
CONSTRAINT pk_alumno PRIMARY KEY (id_al),
CONSTRAINT fk_alumno FOREIGN  KEY  (id_curso) REFERENCES curso (id_curso)) ENGINE=”InnoDB”;
```

Insertar información.

En la tabla curso:

- Ejecutar:
  - INSERT INTO curso (id_curso, deno) VALUES (1, “DAW1”);
  - INSERT INTO curso (id_curso, deno) VALUES (2, “DAW2”);

En la tabla alumno:

- Ejecutar:
  - INSERT INTO alumno (id_al, nombre, edad, id_curso) VALUES (1, “Ana”, 18, 1);
  - INSERT INTO alumno (id_al, nombre, edad, id_curso) VALUES (2, “Sergio”, 18, 1);
  - INSERT INTO alumno (id_al, nombre, edad, id_curso) VALUES (3, “Jorge”, 19, 1);
  - INSERT INTO alumno (id_al, nombre, edad, id_curso) VALUES (4, “Maria”, 21, 2);
  - INSERT INTO alumno (id_al, nombre, edad, id_curso) VALUES (5, “Pedro”, 23, 3);
  - INSERT INTO alumno (id_al, nombre, edad, id_curso) VALUES (6, “Juan”, 21);

Consultar la base de datos:

```
SELECT * FROM alumno WHERE edad=18;
SELECT nombre, edad FROM alumno WHERE edad=18;
SELECT nombre FROM alumno WHERE edad !=18;
SELECT nombre FROM alumno WHERE edad=18 OR edad=21;
SELECT nombre FROM alumno WHERE edad>=18 AND edad <=21;
```

ACTIVIDAD 2: (UT2)

1. Construye una **BD denominada cinema** que recoja información sobre los actores y las películas en las que intervienen (obtener la información de la siguiente Base de Datos *Los_Oscar_2020*) y resuelve las siguientes consultas:

Información sobre los ACTORES:

id_actor	nacionalidad	nombre_apellidos	sexo
00001	francés	Jacqueline Bisset	f
00002	española	Antonio Banderas	m
00003	española	Belén Rueda	f
00004	estadounidense	Brad Pitt	m
00005	estadounidense	Laura Dern	f

Información sobre las PELÍCULAS:

titulo	genero	anno_prod
Dolor y gloria	drama	2019
Erase una vez...Hollywood	comedia	2019
Historia de un matrimonio	drama	2019
La piel que habito	thriller	2011

Información sobre la intervención de los ACTORES en las PELÍCULAS:

id_actor	titulo
00002	Dolor y gloria
00004	Erase una vez...Hollywood
00005	Historia de un matrimonio
00002	La piel que habito

A continuación:

a) Mostrar las películas cuyo género sea drama.

b) Mostrar el nombre de las actrices.

c) Mostrar cuántos actores hay de una determinada nacionalidad (se podrá elegir desde un formulario la nacionalidad disponible)

d) ¿En cuántas películas intervino un determinado actor? (se podrá teclear el nombre del actor)

ACTIVIDAD 3: Incorpora a la **BD cinema**, dos nuevas tablas con la siguiente información:

La tabla **cine**:

id_cine	nombre_c	ciudad_c
CINE1	Odeón	Burgos
CINE2	Van Golem	Burgos
CINE3	CineSur	Sevilla
CINE4	Capitol	Madrid
CINE5	Conde Duque	Madrid

La tabla **proyectar:**

titulo	id_cine	fecha_proy
Dolor y gloria	CINE1	2019-11-15
Dolor y gloria	CINE2	2019-11-15
Dolor y gloria	CINE3	2019-11-15
Dolor y gloria	CINE4	2019-11-15
Dolor y gloria	CINE5	2019-11-15
Erase una vez...Hollywood	CINE1	2019-11-01
Erase una vez...Hollywood	CINE3	2019-11-01
Erase una vez...Hollywood	CINE4	2019-11-01
Historia de un matrimonio	CINE3	2019-10-15
Historia de un matrimonio	CINE4	2019-10-15
La piel que habito	CINE1	2010-10-01
La piel que habito	CINE2	2010-10-01
La piel que habito	CINE3	2010-10-01
La piel que habito	CINE4	2010-10-01
La piel que habito	CINE5	2010-10-01

A continuación, resuelve las siguientes consultas:

a) Muestra, de forma ordenada, por cada cine, las películas proyectadas de la forma más legible que sepas.
b) Muestra la dirección del cine en el cual se proyectó la película “Historia de un matrimonio”.
c) ¿Qué películas se proyectaron en 2019?
d) Y…¿en noviembre de 2019?
e) Añade el atributo recaudación a la tabla proyectar y dale valor

f) ¿Cuál fue la película más taquillera?

ACTIVIDAD 4: Dado el esquema relacional asociado a una Librería y su posterior **diseño de BD** (la profesora proporcionará el **archivo librería.sql**)

LIBRO(**ISBN**, título, género, año_publicación, precio, Autor)

ESCRITOR(**Autor**, f_nacimiento, lugar_nacimiento)

Realizar las siguientes consultas:

a. Mostrar todos los datos de los libros no escritos por Arturo Pérez Reverte y posteriores al año 2010.
b. ¿Cuántos libros hay de cada género?
c. ¿Cuál es el precio medio de un libro?.
d. ¿Cuántos escritores tienes más de 60 años?
e. Título de uno de los libros más barato.

Nota: Explicadas las librerías disponibles en PHP para establecer una conexión con una BD (mysqli dual y PDO) , diseña los scripts necesarios para la realización de estas actividades. Se anexa un resumen de las funciones predeterminadas de MySQL que pueden ser de utilidad para realizar algunas de las consultas (Anexo I)

Anexo I

1. FUNCIONES PREDEFINIDAS.

Las funciones pueden ser predefinidas en SQL o programadas por el usuario experto; en cualquier caso, son programas que incrementan la funcionalidad del Gestor de la BD.

Básicamente, una función puede o no recibir una serie de parámetros o argumentos, realizar algún tipo de tratamiento con ellos y devolver un resultado.

Son muchas las funciones predefinidas que existen: numéricas, de caracteres, de fechas, de comparación,… (*https://dev.mysql.com/doc/refman/5.7/en/func-op-summary-ref.html*)

El material que se proporciona es un resumen de las funciones más utilizadas.

1.1. Funciones numéricas.

Estas funciones **devuelven un valor numérico**. Se podría hacer una clasificación:

- Funciones numéricas individuales, reciben como parámetro un parámetro individual, generalmente, un atributo, una expresión aritmética,…
- Funciones numéricas de grupo, reciben como parámetro una columna completa de datos.

FUNCIÓN INDIVIDUALES	**RESULTADO**
ABS(n)	El valor absoluto de n
MOD(n1,n2)	El resto de la división entera entre n1 y n2
PI( )	El número pi
RAND( )	Un número aleatorio entre 0 y 1
ROUND(n1, n2)	n1 redondeado a n2 decimales. Si se omite n2, redondea a 0 decimales.
TRUNCATE(n1,n2)	n1 truncado a n2 decimales. Si se omite n2, tunca a 0 decimales.
SIGN(n)	-1, si n<0 0, si n=0 1, si n>0
FUNCIONES DE GRUPO	**RESULTADO**
COUNT ({* | expresión_aritmética})	Con * se cuentas las tuplas o filas y con expresión, las veces que aparece esa expresión en las tuplas seleccionadas.
SUM(expresión_aritmética)	Suma expresión_aritmética
AVG(expresión_aritmética)	Media aritmética de expresión_aritmética; ignora los valores nulos
MAX(expresión_aritmética)	Valor máximo de expresión_aritmética
MIN(expresión_aritmética)	Valor mínimo de expresión_aritmética

1.2. Funciones de caracteres.

Estas funciones devuelven un **string** o cadena de caracteres. Está es una pequeña muestra:

FUNCIÓN	RESULTADO
LENGTH(string)	Longitud del string
LOWER(string)	El string en minúsculas
UPPER(string)	El string en mayúsculas
CONCAT(string1, string2,…)	Concatena los strings
SUBSTR(string,n1,n2)	Una subcadena de string, comenzando en n1, n2 caracteres
RTRIM(string) y LTRIM(string)	Eliminar los posibles caracteres " " de la dcha. o a la izda. de un string
REPLACE(string, substring, substring_nuevo)	Sustituir en un string una subcadena por otra

1.3. Funciones para trabajar con las fechas.

FUNCIÓN	RESULTADO
ADDDATE(fecha, n)	Devuelve fecha + n días
SUBDATE(fecha, n)	Devuelve fecha - n días
DATE_ADD(fecha, INTERVAL, n formato)	Devuelve fecha + n veces formato (DAY, WEEK, MONTH, YEAR, HOUR, MINUTE, SECOND)
DATE_SUB(fecha, INTERVAL, n formato)	Devuelve fecha - n veces formato (DAY, WEEK, MONTH, YEAR, HOUR, MINUTE, SECOND)
DATEDIFF(fecha1,fecha2)	Días de diferencia entre las fechas; fecha1 debe ser mayor que fecha2
DAYNAME(fecha)	Nombre del día de la semana de fecha
MONTHNAME(fecha)	Nombre del mes de la fecha
DAYOFWEEK(fecha)	Número del día de la semana (1,2,3,…hasta 7)
DAYOFMONTH(fecha)	El día de la fecha
DAYOFYEAR(fecha)	El número del día en el año (1..366)
WEEKOFYEAR(fecha)	El número de semana de la fecha (1..52)
MONTH(fecha)	El mes de la fecha (1..12)
YEAR(fecha)	El año
HOUR(time)	El valor de la hora en el dato time
MINUTE(time)	El valor de los minutos en el dato time
SECOND(time)	El valor de los segundos en el dato time
CURDATE()	Fecha del sistema yyyy-mm-dd
CURTIME()	Hora del sistema hh:mm:ss
SYSDATE() o NOW()	Fecha y hora del sistema

En muchas ocasiones será preciso modificar el formato que por defecto tienen los datos tipo DATE() o tipo TIME().

Para ello se utiliza la función **DATE_FORMAT(fecha, formato) y TIME_FORMAT(hora, formato),** donde el formato es una cadena de caracteres que constituye la máscara a utilizar para el formato deseado, en la cual se pueden utilizar las siguientes secuencias especiales de caracteres:

Secuencia especial	Función
%a	Tres letras para el nombre del día
%b	Tres letras para el nombre del mes
%c	Número del mes (1..12)
%m	Número del mes (01..12)
%e	Número del día
%H	Hora en formato 24 horas
%h	Hora en formato 12 horas
%i	Minutos
%s	Segundos
%M	Nombre del mes
%Y	Año con 4 dígitos
%y	Año con 2 dígitos

Por ejemplo:

```
mysql> SELECT DATE_FORMAT(sysdate(),'%e/%m/%y');
+-----------------------------------+
| DATE_FORMAT(sysdate(),'%e/%m/%y') |
+-----------------------------------+
| 29/01/20                          |
+-----------------------------------+
1 row in set (0.00 sec)
```

```
MariaDB [ciclo]> select TIME_FORMAT(sysdate(),"%h:%i:%s");
+---------------------------------+
| TIME_FORMAT(sysdate(),"%h:%i:%s") |
+---------------------------------+
| 12:04:27                        |
+---------------------------------+
1 row in set (0.001 sec)
```

1.4. Funciones de comparación.

FUNCIÓN	RESULTADO
GREATEST(serie de valores)	Mayor valor de una serie de valores; cada valor puede ser una expresión numérica (no pueden tomar valor NULL)
LEAST(serie d valores)	menor valor de una serie de valores; cada valor puede ser una expresión numérica (no pueden tomar valor NULL)
IFNULL(expresión1, expresión2)	Si la expresión1 es NULL, el resultado es la expresión2, sino es la expresión1. Expresión2 debe ser “” si expresión1 es de tipo alfanumérico y 0 si es de tipo numérico.
ISNULL(expresión)	1 si expresión es NULL y 0 si no lo es.
STRCMP(string1,string2)	1 si string1 y string2 son iguales, 0 si son distintas y NULL si alguna de ellas lo es.

1.5. Otras funciones.

Hay muchas otras funciones como DATABASE(), USER(), VERSION(),… que utilizaremos según nos sean necesarias consultando la ayuda en línea de MySQL 5.7.