1. FUNCIONES ESCALARES.

Las funciones escalares son aquellas que operan sobre un único valor y devuelven) a su vez, un único valor. Las dividiremos en tres categorías, dependiendo de los datos sobre los que actúen:

- Funciones de fechas.
- Funciones de cadenas.
- Funciones matemáticas.

Hay que tener en cuenta que estas funciones son las que más cambian entre diferentes gestores de bases de datos. Posiblemente no puedas emplear las funciones que vamos a estudiar en otro gestor que no sea MySQL. Además, veremos sólo las más útiles.

DE FECHAS.

Estas funciones trabajan con fechas. Se pueden dividir en dos grandes grupos, dependiendo de si el valor que devuelven es de tipo fecha, como en las siguientes:

- CURDATE ( )
- CURTIME ( )
- NOW ( )
- DATE_ADD ( )
- DATE_SUB ( )

Si devuelven números obtenidos a partir de un campo de tipo fecha:

- WEEKDAY ( )
- DAYOFMONTH ( )
- MONTH ( )
- YEAR ( )
- HOUR ( )
- MINUTE ( )
- SECOND ( )

También existe una función DATE_FORMAT que devuelve una cadena. Vamos ahora a estudiar la utilidad de cada una de ellas.

CURDATE, CURTIME, NOW.

Estas funciones sirven para obtener la fecha actual. Esta fecha la obtiene del reloj de nuestro ordenador, de modo que debemos recordar tenerlo en día y hora si queremos emplearlas. La diferencia entre ellas está en los datos concretos que devuelven:

- CURDATE ( ) devuelve sólo la fecha.
- CURTIME ( ) devuelve sólo la hora.
- NOW ( ) devuelve tanto la fecha como la hora.

Ejemplo: Seleccionar los datos de los empleados que hayan empezado hoy mismo a trabajar.

```
SELECT *
FROM empleado
WHERE fechaingreso=CURDATE()
```

Ejemplo: Averigua la fecha y la hora actual.

```
SELECT NOW ( );
```

DAYOFMONTH, MONTH , YEAR.

Las funciones DAYOFMONTH, MONTH y YEAR sirven para obtener el día, el mes y el año de una fecha dada como parámetro. WEEKOFDAY, en cambio, devuelve el día de la semana (1=lunes, 2=martes, etc.).

La sintaxis es:

```
DAYOFMONTH (fecha)
WEEKDAY (fecha)
MONTH (fecha)
YEAR (fecha)
```

Ejemplo: *Seleccionar los datos de los empleados que hayan nacido el día 12 de cualquier mes.*

```
SELECT *  FROM empleado
WHERE DAYOFMONTH(fechanac)=12;
```

Ejemplo: *Seleccionar los datos de los empleados que hayan nacido en martes.*

```
SELECT * FROM empleado
WHERE WEEKDAY(fechanac)=3;
```

Ejemplo: *Seleccionar los datos de los empleados que hayan nacido cualquier día del mes de marzo.*

```
SELECT * FROM empleado
WHERE MONTH(fechanac)=3;
```

Ejemplo: *Seleccionar los datos de los empleados que hayan nacido en 1970.*

```
SELECT *  FROM empleado
WHERE YEAR(fechanac)=1970;
```

Ejemplo: *Seleccionar día, mes y año de la fecha actual*

```
SELECT DAYOFMONTH(CURDATE()) as dia, MONTH(CURDATE())
as mes, YEAR(CURDATE()) as año;
```

HOUR, MINUTE Y SECOND.

Las funciones HOUR, MINUTE y SECOND sirven para obtener la hora, los minutos y los segundos de una fecha dada como parámetro.

Sintaxis:

```
HOUR (fecha)
MINUTE (fecha)
SECOND (fecha)
```

Ejemplo: *Seleccionar la hora, los minutos y los segundos de la hora actual.*

```
SELECT HOUR(NOW() ) AS hora, MINUTE(NOW()) AS minuto,
SECOND(NOW()) AS segundos;
```

DATE_ADD Y DATE_SUB.

Las funciones DATE_ADD y DATE_SUB suman o restan un intervalo a una fecha. Se puede utilizar una **sintaxis** alternativa más intuitiva:

```
DATE_ADD (fecha, INTERVAL intervalo)
fecha + INTERVAL intervalo
DATE_SUB (fecha, INTERVAL intervalo)
fecha - INTERVAL intervalo
```

El intervalo tiene el formato numero tipo donde tipo puede ser YEAR, MONTH, DAY, HOUR, MINUTE o SECOND.

Ejemplo: *Seleccionar los datos del empleado apellidado Vilán añadiendo un campo llamado fecha de jubilación, que cumplirá a los 65 años*

```
SELECT *, fechanac + INTERVAL 65 YEAR AS 'Fecha de
jubilación'
FROM empleado WHERE apel1='Vilán';
```

```
SELECT *, DATE_ADD(fechanac , INTERVAL 65 YEAR) AS
'Fecha de jubilación'
FROM empleado WHERE apel1='Vilán';
```

Ejemplo: Mostrar la fecha 47 dias posterior a la actual.

```
SELECT DATE_ADD(CURDATE() , INTERVAL 47 DAY);

SELECT CURDATE() + INTERVAL 47 DAY;
```

DATEDIFF

La función es DATEDIFF devuelve el intervalo entre dos fechas.

```
DATEDIFF (fechaMayor, fechaMenor)
```

Ejemplo: *Mostrar los días transcurridos entre dos fechas proporcionadas como cadenas.*

```
SELECT DATEDIFF('1997-12-31','1997-12-30');
```

Devolverá 1

Ejemplo: *Mostrar cuantos días lleva en su cargo el gerente del departamento 2.*

```
SELECT DATEDIFF(CURDATE(), FECHAINICGERENTE)
FROM DEPARTAMENTO
WHERE NUMDEP =2;
```

DATE_FORMAT.

La función DATE_FORMAT devuelve una cadena con la fecha pasada como parámetro escrita según el formato descrito por su segundo parámetro. Ese formato está formado de caracteres normales y una serie de caracteres especiales o especificadores que tienen un significado especial y cambiarán según la fecha que estemos tratando.

```
DATE_FORMAT (fecha, formato)
```

En la tabla siguiente tenemos algunos de los especificadores más utilizados.

Ejemplo: *Seleccionar la fecha de nacimiento de los empleados en formato día/mes/año, con 4 dígitos en el año*

```
SELECT  DATE_FORMAT(fechanac, '%d/%m/%Y')
AS 'Fecha de nacimiento'  FROM empleado;
```

Ejemplo: *Seleccionar la fecha de nacimiento de los empleados en formato día-mes-año, con 2 dígitos en el año y un dígito en el mes.*

```
SELECT  DATE_FORMAT(fechanac, '%d-%c-%y')
AS 'Fecha de nacimiento'  FROM empleado;
```

Ejemplo: *Seleccionar la fecha actual en formato hora completa, de 24 horas*

```
SELECT  DATE_FORMAT(CURTIME(), '%T') ;
```

Especificador	Significado
%Y	Año, numérico, 4 dígitos
%y	Año, numérico, 2 dígitos
%d	Día del mes, numérico (00..31)
%e	Día del mes, numérico (0..31)
%m	Mes, numérico (00..12)
%c	Mes, numérico (0..12)
%H	Hora (00..23)
%k	Hora (0..23)
%h	Hora (01..12)
%l	Hora (1..12)
%i	Minutos, numérico (00..59)
%r	Hora completa, 12 horas (hh:mm:ss [AP]M)
%T	Hora completa, 24 horas (hh:mm:ss)
%S	Segundos (00..59)
%p	AM o PM
%%	El carácter %

DE CADENAS.

Estas funciones trabajan con cadenas. Normalmente devuelven una cadena tras hacerle diversas transformaciones, como por ejemplo quitarles espacios en blanco:

- **LTRIM** (x) elimina los espacios en blanco a la izquierda del primer carácter que no sea un espacio.
- **RTRIM** (x) elimina los espacios en blanco al final de la cadena.
- **TRIM** (x) elimina los espacios en blanco tan al principio como al final de la cadena.

Ejemplo: *Seleccionar los apellidos de los empleados quitando los espacios de la derecha, y el nombre*

quitandole todos los espacios.

```
SELECT RTRIM(apel1),
 RTRIM(apel2),TRIM( Nome)
FROM empleado;
```

Hay un par de funciones para convertir a mayúsculas o minúsculas:

- **UPPER** (x) (o UCASE) convierte x a mayúsculas.
- **LOWER** (x) (o LOASE) convierte x a mayúsculas.

Ejemplo: Seleccionar los apellidos de los empleados poniéndolos en mayúsculas y en minúsculas.

```
SELECT UCASE(apel1), LCASE(apel2)
FROM empleado;
```

La función **LENGTH** (x) devuelve el número de caracteres de x.

Ejemplo: *Seleccionar el primer apellido y la longitud del mismo.*

```
SELECT apel1, LENGTH(apel1) AS 'Tamaño del apellido'
FROM empleado;
```

CONCATENACIÓN.

Las funciones CONCAT y CONCAT_WS devuelven una cadena que resulta de la concatenación de todas las que se pasan como parámetro. La diferencia entre ambos es que el primer parámetro pasado a CONCAT_WS se toma como separador de cadenas, insertándose entre ellas.

Sintaxis:

```
CONCAT (cadenal, cadena2,...)
CONCAT_WS (separador, cadenal, cadena2,...)
```

Ejemplo: *Averiguar los códigos de la tarjeta de los empleados. Estos códigos se obtienen a partir del NSS del empleado seguido de su apellido en mayúsculas.*

```
SELECT CONCAT(nss, UPPER(apel1) ) AS tarjeta
FROM empleado;
```

Ejemplo: *Seleccionar en un sólo campo los dos apellidos de cada empleado, separados por una coma.*

```
SELECT  CONCAT_WS (', ', apel1, apel2) AS 'Apellidos'
FROM empleado;
```

Ejemplo: *Seleccionar en un sólo campo el nombre completo de cada empleado, separando los apellidos*

del nombre por una coma.

```
SELECT  concat_WS(',  ', CONCAT_WS(' ', apel1, apel2)
, nombre) AS 'nombre completo'
FROM empleado;
```

Ejemplo: *Mostrar la fecha ordenada como año, mes, dia, con guiones de separación. (lo mejor sería usar el formato de fecha adecuado, pero también podríamos hacer lo siguiente :)*

```
SELECT concat_ws('-',DAYOFMONTH(CURDATE()) ,
MONTH(CURDATE()) , YEAR(CURDATE()) ) ;
```

SUBCADENAS.

Hay tres funciones para extraer un fragmento de una cadena:

- **LEFT**(cadena, x) devuelve los x primeros caracteres de la cadena.
- **RIGHT**(cadena, x) devuelve los x últimos caracteres de la cadena.
- **SUBSTRING**(cadena, pos[, x]) devuelve x caracteres de la cadena situados a partir del carácter número pos. En MySQL el primer carácter tiene el número 1, y no 0, como en muchos lenguajes de programación. Si no se especifica x coge todos los caracteres hasta el final de la cadena. También se puede emplear MID como sinónimo, aunque en ese caso deberemos indicar x obligatoriamente.

Ejemplo: *Extrae cuatro caracteres de cada apellido de los empleados, por la derecha y por la izquierda, y por el medio a partir del 2º carácter, inclusive*

```
SELECT LEFT(apel1,4), RIGHT(apel2,4), MID( apel1,2,4)
FROM empleado;
```

MATEMÁTICAS.

Estas funciones trabajan con datos numéricos. No obstante, las funciones numéricas más utilizadas no son escalares sino las agregadas que veremos en el próximo capítulo.

MySQL dispone de un par de funciones relacionadas con el signo de un número:

- **ABS( )** devuelve el valor absoluto de un número, es decir, que si el número que recibe como parámetro es positivo lo deja tal cual, pero si es negativo, le cambia el signo.
- **SIGN( )**devuelve el signo sin valor. Es decir, devuelve 1 si el número es positivo, -1 si es negativo y O si es cero.

Podemos concluir que ABS(x)*SIGN(x)=x, si nos gustan las matemáticas. También tenemos a nuestra disposición algunas funciones de exponenciación.

- **POW**(x,y) devuelve x elevado a y
- **EXP**(x) devuelve e elevado a x
- **LN**(x) devuelve el logaritmo natural (es decir, en base e) de x.
- **LOG**(b,x) devuelve el logaritmo en base b de x.

El número e, recordémoslo, es aproximadamente 2'71.

Por último veremos otras tres funciones más:

- PI( ) devuelve el valor del número PI. Muestra por defecto 6 decimales, pero se guarda internamente como un número real en doble precisión,
- RAND([x]) devuelve un número aleatorio situado entre 0 y 1. Si se facilita un número como parámetro, se emplea éste como semilla.
- SQRT(x) devuelve la raíz cuadrada de x

REDONDEO.

MySQL dispone de tres funciones que realizan la labor de redondeo de un número real. Son éstas:

- **ROUND**(x[,d]): Redondea al entero más cercano. d indica el número de decimales que deseamos mantener, por defecto 0.
- **CEIL**(x): Devuelve el entero más pequeño cuyo valor es mayor que X.
- **FLOOR**(x): Devuelve el entero más grande inferior o igual a X

Ejemplo: *Este ejemplo nos permite observar el distinto comportamiento de cada función de redondeo ante distintos datos, positivos y negativos y cercanos bien al entero más pequeño o al más grande.*

```
SELECT
ROuND(3.468,2),  Floor(3.4), Ceil(3.4);
select
ROuND(-3.5), FLOOR(-3.5), CEILING(-3.5);
select
ROuND(3.5), FLOOR(3.5), CEILING(3.5);
select
ROuND(-3.6), FLOOR(-3.6), CEILING(-3.6);
```

Tambien dispone de funciones trigonométricas