Funciones para Strings

Fuente: http://www.tutorialesprogramacionya.com

NO debe haber espacios entre un nombre de función y los paréntesis porque MySQL puede confundir una llamada a una función con una referencia a una tabla o campo que tenga el mismo nombre de una función.
MySQL tiene algunas funciones para trabajar con cadenas de caracteres. Estas son algunas:

-ord(caracter): Retorna el código ASCII para el caracter enviado como argumento.
Ejemplo: select ord('A'); retorna 65.

-char(x,..): retorna una cadena con los caracteres en código ASCII de los enteros enviados como argumentos. Ejemplo: select char(65,66,67); retorna "ABC".

-concat(cadena1,cadena2,...): devuelve la cadena resultado de concatenar los argumentos. Ejemplo: select concat('Hola,',' ','como esta?'); retorna "Hola, como esta?".

-concat_ws(separador,cadena1,cadena2,...): "ws" son las iniciales de "with separator". El primer argumento especifica el separador que utiliza para los demás argumentos; el separador se agrega entre las cadenas a concatenar.
Ejemplo: select concat_ws('-','Juan','Pedro','Luis'); retorna "Juan-Pedro-Luis".

-length(cadena): retorna la longitud de la cadena enviada como argumento. Ejemplo: select length('Hola'); devuelve 4.

-locate(subcadena,cadena): retorna la posición de la primera ocurrencia de la subcadena en la cadena enviadas como argumentos. Devuelve "0" si la subcadena no se encuentra en la cadena. Ejemplo: select locate('o','como le va'); retorna 2.

- lpad(cadena,longitud,cadenarelleno): retorna la cadena enviada como primer argumento, rellenada por la izquierda con la cadena enviada como tercer argumento hasta que la cadena retornada tenga la longitud especificada como segundo argumento. Si la cadena es más larga, la corta. Ejemplo: select lpad('hola',10,'0'); retorna "000000hola".

- rpad(cadena,longitud,cadenarelleno): igual que "lpad" excepto que rellena por la derecha.

- left(cadena,longitud): retorna la cantidad (longitud) de caracteres de la cadena comenzando desde la izquierda, primer caracter. Ejemplo:
select left('buenos dias',8); retorna "buenos d".

- right(cadena,longitud): retorna la cantidad (longitud) de caracteres de la cadena comenzando desde la derecha, último caracter. Ejemplo:
select right('buenos dias',8); retorna "nos dias".

-ltrim(cadena): retorna la cadena con los espacios de la izquierda eliminados. Ejemplo:

select ltrim(' Hola '); retorna "Hola "

- rtrim(cadena): retorna la cadena con los espacios de la derecha eliminados. Ejemplo:
select rtrim(' Hola '); retorna " Hola"

-trim([[both|leading|trailing] [subcadena] from] cadena): retorna una cadena igual a la enviada pero eliminando la subcadena prefijo y/o sufijo. Si no se indica ningún especificador (both, leading o trailing) se asume "both" (ambos). Si no se especifica prefijos o sufijos elimina los espacios. Ejemplos:

select trim(' Hola '); retorna 'Hola'.
select trim (leading '0' from '00hola00'); retorna "hola00".
select trim (trailing '0' from '00hola00'); retorna "00hola".
select trim (both '0' from '00hola00'); retorna "hola".
select trim ('0' from '00hola00'); retorna "hola".

-replace(cadena,cadenareemplazo,cadenareemplazar): retorna la cadena con todas las ocurrencias de la subcadena reemplazo por la subcadena a reemplazar.
Ejemplo: select replace('yyy.mysql.com','y','w'); retorna "www.mysql.com'.

-repeat(cadena,cantidad): devuelve una cadena consistente en la cadena repetida la cantidad de veces especificada. Si la "cantidad" es menor o igual a cero, retorna una cadena vacía. Ejemplo: select repeat('hola',3); retorna "holaholahola".

-reverse(cadena): devuelve la cadena invirtiendo el orden de los caracteres. Ejemplo:
select reverse('Hola'); retorna "aloH".

-lcase(cadena) y lower(cadena): retornan la cadena con todos los caracteres en minúsculas. Ejemplos:

select lower('HOLA ESTUDIAnte'); retorna "hola estudiante".
select lcase('HOLA ESTUDIAnte'); retorna "hola estudiante".

-ucase(cadena) y upper(cadena): retornan la cadena con todos los caracteres en mayúsculas. Ejemplos:

select upper('HOLA ESTUDIAnte'); retorna "HOLA ESTUDIANTE".
select ucase('HOLA ESTUDIAnte'); retorna "HOLA ESTUDIANTE".

-strcmp(cadena1,cadena2): retorna 0 si las cadenas son iguales, -1 si la primera es menor que la segunda y 1 si la primera es mayor que la segunda. Ejemplo:
select strcmp('Hola','Chau'); retorna 1.

Ejemplo práctico:

```
create table libros(
  codigo int unsigned auto_increment,
  titulo varchar(40) not null,
  autor varchar(30),
  editorial varchar (20),
  precio decimal(5,2) unsigned,
  primary key(codigo)
 );

insert into libros (titulo,autor,editorial,precio)
  values('El Aleph','Borges','Paidos',33.4);
insert into libros (titulo,autor,editorial,precio)
  values('Alicia en el País de las Maravillas','L. Carroll','Planeta',16);

select concat_ws('-',titulo,autor)
  from libros;

select left(titulo,15)
  from libros;

select lower(titulo), upper(editorial)
 from libros;
```

Los operadores aritméticos son "+","-","*" y "/". Todas las operaciones matemáticas retornan "null" en caso de error. Ejemplo:
select 5/0;

Funciones para valores numéricos

MySQL tiene algunas funciones para trabajar con números. Aquí presentamos algunas. RECUERDE que NO debe haber espacios entre un nombre de función y los paréntesis porque MySQL puede confundir una llamada a una función con una referencia a una tabla o campo que tenga el mismo nombre de una función.

-abs(x): retorna el valor absoluto del argumento "x". Ejemplo: select abs(-20); retorna 20.

-ceiling(x): redondea hacia arriba el argumento "x". Ejemplo: select ceiling(12.34),retorna 13.

-floor(x): redondea hacia abajo el argumento "x". Ejemplo: select floor(12.34); retorna 12.

-greatest(x,y,..): retorna el argumento de máximo valor.

-least(x,y,...): con dos o más argumentos, retorna el argumento más pequeño.

-mod(n,m): significa "módulo aritmético"; retorna el resto de "n" dividido en "m". Ejemplos:
select mod(10,3); retorna 1.
select mod(10,2); retorna 0.

-power(x,y): retorna el valor de "x" elevado a la "y" potencia. Ejemplo: select power(2,3); retorna 8.

-rand(): retorna un valor de coma flotante aleatorio dentro del rango 0 a 1.0.

-round(x): retorna el argumento "x" redondeado al entero más cercano. Ejemplos:
select round(12.34); retorna 12.
select round(12.64); retorna 13.

-sqrt(): devuelve la raiz cuadrada del valor enviado como argumento.

-truncate(x,d): retorna el número "x", truncado a "d" decimales. Si "d" es 0, el resultado no tendrá parte fraccionaria. Ejemplos:
select truncate(123.4567,2); retorna 123.45;
select truncate (123.4567,0); retorna 123.

Todas retornan null en caso de error.

Ejemplo práctico:

```
drop table if exists libros;

create table libros(
  codigo int unsigned auto_increment,
  titulo varchar(40) not null,
  autor varchar(30),
  editorial varchar (20),
  precio decimal(5,2) unsigned,
  primary key(codigo)
 );

insert into libros (titulo,autor,editorial,precio)
  values('El aleph','Borges','Paidos',33.46);
insert into libros (titulo,autor,editorial,precio)
  values('Alicia en el pais de las maravillas','L. Carroll','Planeta',16.31);
insert into libros (titulo,autor,editorial,precio)
  values('Alicia a traves del espejo','L. Carroll','Planeta',18.89);

select titulo, ceiling(precio),floor(precio)
  from libros;

select titulo, round(precio) from libros;
```

```
select titulo,truncate(precio,1)
  from libros;
```

Columnas calculadas

Es posible obtener salidas en las cuales una columna sea el resultado de un cálculo y no un campo de una tabla.
Si queremos ver los títulos, precio y cantidad de cada libro escribimos la siguiente sentencia:

```
select titulo,precio,cantidad
  from libros;
```

Si queremos saber el monto total en dinero de un título podemos multiplicar el precio por la cantidad por cada título, pero también podemos hacer que MySQL realice el cálculo y lo incluya en una columna extra en la salida:

```
 select titulo, precio,cantidad,precio*cantidad
  from libros;
```

Si queremos saber el precio de cada libro con un 10% de descuento podemos incluir en la sentencia los siguientes cálculos:

```
select titulo, precio,precio*0.1,precio-(precio*0.1)
  from libros;
```

Ejemplo práctico:

```
drop table if exists libros;

 create table libros(
  codigo int unsigned auto_increment,
  titulo varchar(40) not null,
  autor varchar(30),
  editorial varchar(15),
  precio decimal(5,2) unsigned,
  cantidad smallint unsigned,
  primary key (codigo)
 );

 insert into libros (titulo,autor,editorial,precio,cantidad)
  values('El Aleph','Borges','Planeta',15,100);
 insert into libros (titulo,autor,editorial,precio,cantidad)
  values('Martin Fierro','Jose Hernandez','Emece',22.20,200);
 insert into libros (titulo,autor,editorial,precio,cantidad)
  values('Antologia Poética','Borges','Planeta',40,150);
 insert into libros (titulo,autor,editorial,precio,cantidad)
  values('Aprenda PHP','Mario Molina','Emece',18.20,200);
 insert into libros (titulo,autor,editorial,precio,cantidad)
  values('Cervantes y el Quijote','Borges','Paidos',36.40,100);
```

```
insert into libros (titulo,autor,editorial,precio,cantidad)
 values('Manual de PHP', 'J.C. Paez', 'Paidos',30.80,100);
insert into libros (titulo,autor,editorial,precio,cantidad)
 values('Harry Potter y la Piedra Filosofal','J.K. Rowling','Paidos',45.00,500);
insert into libros (titulo,autor,editorial,precio,cantidad)
 values('Harry Potter y la Cámara Secreta','J.K. Rowling','Paidos',46.00,300);
insert into libros (titulo,autor,editorial,precio,cantidad)
 values('Alicia en el País de las Maravillas','Lewis Carroll','Paidos',null,50);

select titulo, precio,cantidad,precio*cantidad
 from libros;

select titulo, precio,precio*0.1,precio-(precio*0.1)
 from libros;
```

Group by

Es posible obtener salidas en las cuales una columna sea el resultado de un cálculo y no un campo de una tabla.
Si queremos ver los títulos, precio y cantidad de cada libro escribimos la siguiente sentencia:

select titulo,precio,cantidad from libros;

Si queremos saber el monto total en dinero de un título podemos multiplicar el precio por la cantidad por cada título, pero también podemos hacer que MySQL realice el cálculo y lo incluya en una columna extra en la salida:

select titulo, precio,cantidad,precio*cantidad from libros;

Si queremos saber el precio de cada libro con un 10% de descuento podemos incluir en la sentencia los siguientes cálculos:

select titulo, precio,precio*0.1,precio-(precio*0.1) from libros;

Queremos saber la cantidad de visitantes de cada ciudad, podemos tipear la siguiente sentencia:

select count(*) from visitantes where ciudad='Cordoba';

y repetirla con cada valor de "ciudad":

select count(*) from visitantes where ciudad='Alta Gracia';
select count(*) from visitantes where ciudad='Villa Dolores';

Pero hay otra manera, utilizando la cláusula "group by":

select ciudad, count(*) from visitantes group by ciudad;

Entonces, para saber la cantidad de visitantes que tenemos en cada ciudad utilizamos la función "count()", agregamos "group by" y el campo por el que deseamos que se realice el agrupamiento, también colocamos el nombre del campo a recuperar.
La instrucción anterior solicita que muestre el nombre de la ciudad y cuente la cantidad agrupando los registros por el campo "ciudad". Como resultado aparecen los nombres de las ciudades y la cantidad de registros para cada valor del campo.

Para obtener la cantidad de visitantes con teléfono no nulo, de cada ciudad utilizamos la función "count()" enviándole como argumento el campo "telefono", agregamos "group by" y el campo por el que deseamos que se realice el agrupamiento (ciudad):

select ciudad, count(telefono) from visitantes group by ciudad;

Como resultado aparecen los nombres de las ciudades y la cantidad de registros de cada una, sin contar los que tienen teléfono nulo. Recuerde la diferencia de los valores que retorna la función "count()" cuando enviamos como argumento un asterisco o el nombre de un campo: en el primer caso cuenta todos los registros incluyendo los que tienen valor nulo, en el segundo, los registros en los cuales el campo especificado es no nulo.
Para conocer el total de las compras agrupadas por sexo:

select sexo, sum(montocompra) from visitantes group by sexo;

Para saber el máximo y mínimo valor de compra agrupados por sexo:

select sexo, max(montocompra), min(montocompra) from visitantes group by sexo;

Para calcular el promedio del valor de compra agrupados por ciudad:

select ciudad, avg(montocompra) from visitantes group by ciudad;

Podemos agrupar por más de un campo, por ejemplo, vamos a hacerlo por "ciudad" y "sexo":

select ciudad, sexo, count(*) from visitantes group by ciudad,sexo;

También es posible limitar la consulta con "where".
Vamos a contar y agrupar por ciudad sin tener en cuenta "Cordoba":

select ciudad, count(*) from visitantes where ciudad<>'Cordoba' group by ciudad;

Ejemplo Práctico:

```
create table visitantes(
  nombre varchar(30),
  edad tinyint unsigned,
  sexo char(1),
  domicilio varchar(30),
  ciudad varchar(20),
```

```
  telefono varchar(11),
  montocompra decimal (6,2) unsigned
 );

insert into visitantes (nombre,edad, sexo,domicilio,ciudad,telefono,montocompra)
  values ('Susana Molina', 28,'f','Colon 123','Cordoba',null,45.50);
insert into visitantes (nombre,edad, sexo,domicilio,ciudad,telefono,montocompra)
  values ('Marcela Mercado',36,'f','Avellaneda 345','Cordoba','4545454',0);
insert into visitantes (nombre,edad, sexo,domicilio,ciudad,telefono,montocompra)
  values ('Alberto Garcia',35,'m','Gral. Paz 123','Alta Gracia','03547123456',25);
insert into visitantes (nombre,edad, sexo,domicilio,ciudad,telefono,montocompra)
  values ('Teresa Garcia',33,'f','Gral. Paz 123','Alta Gracia','03547123456',0);
insert into visitantes (nombre,edad, sexo,domicilio,ciudad,telefono,montocompra)
  values ('Roberto Perez',45,'m','Urquiza 335','Cordoba','4123456',33.20);
insert into visitantes (nombre,edad, sexo,domicilio,ciudad,telefono,montocompra)
  values ('Marina Torres',22,'f','Colon 222','Villa Dolores','03544112233',25);
insert into visitantes (nombre,edad, sexo,domicilio,ciudad,telefono,montocompra)
  values ('Julieta Gomez',24,'f','San Martin 333','Alta Gracia','03547121212',53.50);
insert into visitantes (nombre,edad, sexo,domicilio,ciudad,telefono,montocompra)
  values ('Roxana Lopez',20,'f','Triunvirato 345','Alta Gracia',null,0);
insert into visitantes (nombre,edad, sexo,domicilio,ciudad,telefono,montocompra)
  values ('Liliana Garcia',50,'f','Paso 999','Cordoba','4588778',48);
insert into visitantes (nombre,edad, sexo,domicilio,ciudad,telefono,montocompra)
  values ('Juan Torres',43,'m','Sarmiento 876','Cordoba','4988778',15.30);

-- Para saber la cantidad de visitantes que tenemos de cada ciudad tipeamos:
select ciudad, count(*)
  from visitantes
  group by ciudad;

-- Necesitamos conocer la cantidad visitantes con teléfono no nulo, de cada ciudad:
select ciudad, count(telefono)
 from visitantes
  group by ciudad;

-- Queremos conocer el total de las compras agrupadas por sexo:
select sexo, sum(montocompra) from visitantes
  group by sexo;

-- Para obtener el máximo y mínimo valor de compra agrupados por sexo:
select sexo, max(montocompra) from visitantes
  group by sexo;
select sexo, min(montocompra) from visitantes
  group by sexo;

-- Se pueden simplificar las 2 sentencias anteriores en una sola sentencia, ya que usan el
mismo "group by":
select sexo, max(montocompra),
```

```
  min(montocompra)
  from visitantes
  group by sexo;

-- Queremos saber el promedio del valor de compra agrupados por ciudad:
select ciudad, avg(montocompra) from visitantes
  group by ciudad;

-- Contamos los registros y agrupamos por 2 campos, "ciudad" y "sexo":
select ciudad, sexo, count(*) from visitantes
  group by ciudad,sexo;

-- Limitamos la consulta, no incluimos los visitantes de "Cordoba", contamos y agrupar por
ciudad:
select ciudad, count(*) from visitantes
  where ciudad<>'Cordoba'
  group by ciudad;
```

Having

Así como la cláusula "where" permite seleccionar (o rechazar) registros individuales; la cláusula "having" permite seleccionar (o rechazar) un grupo de registros.
Si queremos saber la cantidad de libros agrupados por editorial usamos la siguiente instrucción ya aprendida:

select editorial, count(*) from libros group by editorial;

Si queremos saber la cantidad de libros agrupados por editorial pero considerando sólo algunos grupos, por ejemplo, los que devuelvan un valor mayor a 2, usamos la siguiente instrucción:

select editorial, count(*) from libros group by editorial having count(*)>2;

Se utiliza "having", seguido de la condición de búsqueda, para seleccionar ciertas filas retornadas por la cláusula "group by".
Veamos otros ejemplos. Queremos el promedio de los precios de los libros agrupados por editorial:

select editorial, avg(precio) from libros group by editorial;

Ahora, sólo queremos aquellos cuyo promedio supere los 25 pesos:

select editorial, avg(precio) from libros group by editorial having avg(precio)>25;

En algunos casos es posible confundir las cláusulas "where" y "having". Queremos contar los registros agrupados por editorial sin tener en cuenta a la editorial "Planeta".
Analicemos las siguientes sentencias:

select editorial, count(*) from libros where editorial<>'Planeta' group by editorial;
select editorial, count(*) from libros group by editorial having editorial<>'Planeta';

Ambas devuelven el mismo resultado, pero son diferentes.
La primera, selecciona todos los registros rechazando los de editorial "Planeta" y luego los agrupa para contarlos. La segunda, selecciona todos los registros, los agrupa para contarlos y finalmente rechaza la cuenta correspondiente a la editorial "Planeta".

No debemos confundir la cláusula "where" con la cláusula "having"; la primera establece condiciones para la selección de registros de un "select"; la segunda establece condiciones para la selección de registros de una salida "group by".
Veamos otros ejemplos combinando "where" y "having".
Queremos la cantidad de libros, sin considerar los que tienen precio nulo, agrupados por editorial, sin considerar la editorial "Planeta":

```
select editorial, count(*) from libros
where precio is not null
group by editorial
having editorial<>'Planeta';
```

Aquí, selecciona los registros rechazando los que no cumplan con la condición dada en "where", luego los agrupa por "editorial" y finalmente rechaza los grupos que no cumplan con la condición dada en el "having".
Generalmente se usa la cláusula "having" con funciones de agrupamiento, esto no puede hacerlo la cláusula "where". Por ejemplo queremos el promedio de los precios agrupados por editorial, de aquellas editoriales que tienen más de 2 libros:

```
select editorial, avg(precio) from libros
group by editorial
having count(*) > 2;
```

Podemos encontrar el mayor valor de los libros agrupados por editorial y luego seleccionar las filas que tengan un valor mayor o igual a 30:

```
select editorial, max(precio) from libros
group by editorial
having max(precio)>=30;
```

Esta misma sentencia puede usarse empleando un "alias", para hacer referencia a la columna de la expresión:

```
select editorial, max(precio) as 'mayor' from libros
group by editorial
having mayor>=30;
```

Ejemplo:

```
drop table if exists libros;

create table libros(
  codigo int unsigned auto_increment,
  titulo varchar(60) not null,
  autor varchar(30),
  editorial varchar(15),
  precio decimal(5,2) unsigned,
  primary key (codigo)
 );

insert into libros (titulo,autor,editorial,precio)
  values('El Aleph','Borges','Planeta',15);
insert into libros (titulo,autor,editorial,precio)
  values('Martin Fierro','Jose Hernandez','Emece',22.20);
insert into libros (titulo,autor,editorial,precio)
  values('Antología poética','Borges','Planeta',40);
insert into libros (titulo,autor,editorial,precio)
  values('Aprenda PHP','Mario Molina','Emece',18.20);
insert into libros (titulo,autor,editorial,precio)
  values('Cervantes y el Quijote','Borges','Paidos',36.40);
insert into libros (titulo,autor,editorial,precio)
  values('Manual de PHP', 'J.C. Paez', 'Paidos',30.80);
insert into libros (titulo,autor,editorial,precio)
  values('Harry Potter y la Piedra Filosofal','J.K. Rowling','Paidos',45.00);
insert into libros (titulo,autor,editorial,precio)
  values('Harry Potter y la Cámara Secreta','J.K. Rowling','Paidos',46.00);
insert into libros (titulo,autor,editorial,precio)
  values('Alicia en el País de las Maravillas','Lewis Carroll','Paidos',null);

-- Queremos averiguar la cantidad de libros agrupados por editorial:
select editorial, count(*) from libros
  group by editorial;

-- Queremos conocer la cantidad de libros agrupados por editorial pero considerando
-- sólo los que devuelvan un valor mayor a 2
select editorial, count(*) from libros
  group by editorial
  having count(*)>2;

-- Necesitamos el promedio de los precios de los libros agrupados por editorial:
select editorial, avg(precio)
  from libros
```

```
  group by editorial;

-- sólo queremos aquellos cuyo promedio supere los 25 pesos:
select editorial, avg(precio)
  from libros
  group by editorial
  having avg(precio)>25;

-- Queremos contar los registros agrupados por editorial sin tener en cuenta
-- a la editorial "Planeta"
select editorial, count(*) from libros
  where editorial<>'Planeta'
  group by editorial;
select editorial, count(*) from libros
  group by editorial
  having editorial<>'Planeta';

 -- Queremos la cantidad de libros, sin tener en cuenta los que tienen precio nulo,
 -- agrupados por editorial, rechazando los de editorial "Planeta"
select editorial, count(*) from libros
  where precio is not null
  group by editorial
  having editorial<>'Planeta';

-- promedio de los precios agrupados por editorial, de aquellas editoriales
-- que tienen más de 2 libros
select editorial, avg(precio) from libros
  group by editorial
  having count(*) > 2;

-- mayor valor de los libros agrupados por editorial y luego seleccionar las filas
-- que tengan un valor mayor o igual a 30
select editorial, max(precio)
  from libros
  group by editorial
  having max(precio)>=30;

-- Para esta misma sentencia podemos utilizar un "alias" para hacer referencia a la
-- columna de la expresión
select editorial, max(precio) as 'mayor'
  from libros
  group by editorial
  having mayor>=30;
```

Procedimientos almacenados

Los procedimientos almacenados se crean en la base de datos seleccionada.

En primer lugar se deben tipear y probar las instrucciones que se incluyen en el procedimiento almacenado, luego, si se obtiene el resultado esperado, se crea el procedimiento.
Los procedimientos almacenados pueden hacer referencia a tablas, vistas y otros procedimientos almacenados.
Un procedimiento almacenado pueden incluir cualquier cantidad y tipo de instrucciones.
Para crear un procedimiento almacenado empleamos la instrucción "create procedure".
La sintaxis básica parcial es:

```
create procedure NOMBREPROCEDIMIENTO()
begin
 INSTRUCCIONES;
end
```

Con las siguientes instrucciones creamos un procedimiento almacenado llamado "pa_libros_limite_stock" que retorna todos los libros de los cuales hay menos de 10 disponibles:

```
create procedure pa_libros_limite_stock()
 begin
   select * from libros
   where stock<=10;
 end
```

Para llamar luego al procedimiento almacenado debemos utilizar la cláusula 'call' y seguidamente el nombre del procedimiento almacenado:

```
call pa_libros_limite_stock();
```

```
 create procedure pa_libros_mayor_precio()
 begin
   select * from libros
   where precio>=15;
 end

 call pa_libros_mayor_precio();
```