

Funciones de Conversión de números

[**BIN\(\)**](#) Conversión a binario

[**CONV\(\)**](#) Convierte números entre distintas bases

[**HEX\(\)**](#) Conversión de números a hexadecimal

[**vOCT\(\)**](#) Convierte un número a octal

[**UNHEX\(\)**](#) Conversión de números de hexadecimal a ASCII

Funciones de control de flujo

[**IF**](#) Elección en función de una expresión booleana

[**IFNULL**](#) Elección en función de si el valor de una expresión es NULL

[**NULLIF**](#) Devuelve NULL en función del valor de una expresión

[**CASE**](#) Devuelve en función de diversas condiciones.

Funciones de casting (conversión de tipos)

[**CAST / CONVERT**](#) Conversión de tipos explícita

Funciones de Conversión de números

BINQ

BIN(N)

Devuelve una cadena que representa el valor binario de N, donde N es un número longlong (BIGINT). Es equivalente a CONV(N,10,2). Devuelve NULL si N es NULL:

```
SELECT BIN(12) ;
```

BIN(12)	BIN(128)
1100	10000000

CONVQ

CONV(N,from_base,to_base)

Convierte números entre distintas bases. Devuelve una cadena que representa el número N, convertido desde la base from_base a la base to_base. Devuelve *NULL* si alguno de los argumentos es *NULL*. El argumento N se interpreta como un entero, pero puede ser especificado como un entero o como una cadena. La base mínima es 2 y la máxima 36. Si to_base es un número negativo, N es tratado como un número con signo. En caso contrario, N se trata como sin signo. **CONV** trabaja con una precisión de 64 bits:

```
SELECT CONV("a",16,2) , CONV("6E",18,8) , CONV("-17",10,-18) ,  
CONV(10+"10"+"10'+0xa",10,10) ;
```

CONV("a",16,2)	CONV("6E",18,8)	CONV("-17",10,-18)	CONV(10+"10"+"10'+0xa",10,10)
1010	172	-H	40

HEXQ

HEX(N_or_S)

Si N_OR_S es un número, devuelve una cadena que representa el valor hexadecimal de N, donde N es un número longlong (BIGINT). Es equivalente a CONV(N,10;16). Si N_OR_S es una cadena, devuelve una cadena hexadecimal de N_OR_S donde cada carácter en N_OR_S se convierte a dos dígitos hexadecimales. Esto es la inversa de las cadenas 0xff.

```
SELECT HEX(255) , HEX("abc") , 0x616263;
```

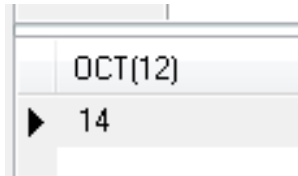
HEX(255)	HEX("abc")	0x6...
FF	616263	abc

oOCT()

OCT(N)

Devuelve una cadena que representa el valor octal de N, donde N es un número BIGINT. Es equivalente a CONV(N,10,8). Devuelve NULL si N es NULL:

```
SELECT OCT(12) ;
```



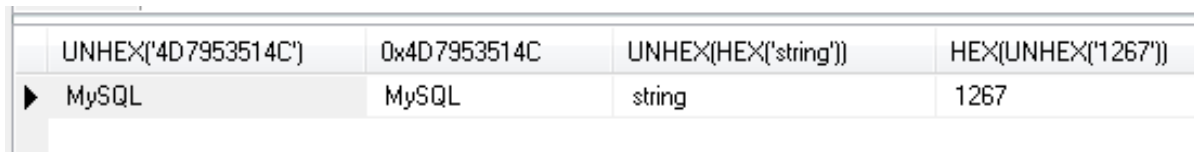
OCT(12)
14

UNHEX

UNHEX(str)

Es la función opuesta a **HEX(str)**. Es decir, interpreta cada par de dígitos hexadecimales del argumento como un número, y lo convierte en el carácter representado por ese número. Los caracteres resultantes se devuelven como una cadena binaria.

```
SELECT UNHEX('4D7953514C'), 0x4D7953514C, UNHEX(HEX('string')),  
HEX(UNHEX('1267')) ;
```



UNHEX('4D7953514C')	0x4D7953514C	UNHEX(HEX('string'))	HEX(UNHEX('1267'))
MySQL	MySQL	string	1267

UNHEX() fue añadido en MySQL 4.1.2.

Funciones de control de flujo

Las funciones de esta categoría son:

IF

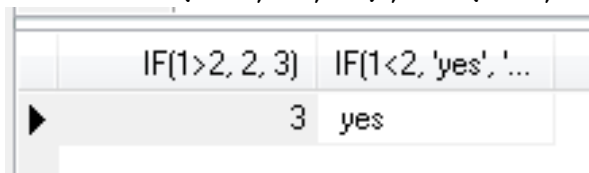
IF(expr1,expr2,expr3)

Expr1 es una condición.

Expr2 es el valor que devuelve la función si la condición es verdadera.

Expr3 es el valor que devuelve la función si la condición es falsa.

```
SELECT IF(1>2, 2, 3), IF(1<2, 'yes', 'no') ;
```



IF(1>2, 2, 3)	IF(1<2, 'yes', 'no')
3	yes

Ejemplo: Hacer una consulta que nos muestre un mensaje "Cuota sin alcanzar" para los empleados que no hayan alcanzado la cuota.

```
SELECT Nombre, Cuota, Ventas, IF(Cuota > Ventas, 'Cuota sin
alcanzar', '')
FROM Empleados
```

Nombre	Cuota	Ventas	IF(Cuota > Ventas, 'Cuota sin alcanzar', '')
Antonio Viguer	3000.00	3100.00	
Alvaro Jaumes	3500.00	3900.00	
Juan Rovira	2850.00	2950.00	
Jose Gozalez	2500.00	2220.00	Cuota sin alcanzar
Vicente Pantalla	3500.00	3600.00	
Luis Antonio	NULL	NULL	
Jorge Gutierrez	3000.00	1500.00	Cuota sin alcanzar
Ana Bustamante	3500.00	3590.00	
Maria Sunta	3000.00	3300.00	
Juan Victor	NULL	NULL	
Luisa Alvarez	3000.00	3100.00	
Jose Luis Lopez	3000.00	3100.00	
Miguel Chinchetru	4000.00	3100.00	Cuota sin alcanzar
Carme Lejardi	3500.00	5900.00	
Maria Arrieta	2850.00	4950.00	
Luisa Terrazo	0.00	0.00	

16 rows fetched in 0,0038s (0,0006s)

IFNULL

IFNULL(expr1,expr2)

Si expr1 no es *NULL*, **IFNULL()** devuelve expr1, en caso contrario, devuelve expr2. **IFNULL()** devuelve un valor numérico o una cadena, dependiendo del contexto en el que se use.

```
SELECT IFNULL(1,0) , IFNULL(NULL,10) , IFNULL(1/0,10) ,
IFNULL(1/0, 'yes') ;
```

	IFNULL(1,0)	IFNULL(NULL,10)	IFNULL(1/0,10)	IFNULL(1/0,'yes')
▶	1	10	10.0000	yes

Ejemplo: Hacer una consulta que para los empleados que no tengan oficina asignada nos muestre el mensaje "Sin Oficina".

```
SELECT Nombre, IFNULL(Oficina, 'Sin Oficina Asignado')
FROM Empleados d;
```

Nombre	IFNULL(Oficina, 'Sin Oficina Asignado')
Antonio Viguer	12
Alvaro Jaumes	21
Juan Rovira	12
Jose Gozalez	12
Vicente Pantalla	13
Luis Antonio	11
Jorge Gutierrez	22
Ana Bustamante	21
Maria Sunta	11
Juan Victor	Sin Oficina Asignado
Luisa Alvarez	12
Jose Luis Lopez	12
Miguel Chinchetru	Sin Oficina Asignado
Carme Lejardi	Sin Oficina Asignado
Maria Arrieta	Sin Oficina Asignado
Luisa Terrazo	12
Maria Sánchez	21
Marta Elorriaga	22

NULLIF

NULLIF(expr1,expr2)

Devuelve *NULL* si expr1 = expr2 es verdadero, si no devuelve expr1. Esto es lo mismo que si se usa la expresión WHEN expr1 = expr2 THEN NULL ELSE expr1 END.

```
SELECT NULLIF(1,1) , NULLIF(1,2) ;
```

NULLIF(1,1)	NULLIF(1,2)
NULL	1

Hay que tener en cuenta que MySQL evalúa expr1 dos veces si los argumentos no son iguales. **NULLIF()** se añadió en MySQL 3.23.15.

CASE

CASE WHEN Condicion1 THEN Accion1

WHEN Condicion2 THEN Accion2

.....

WHEN CondicionN THEN AccionN

ELSE ValorPorDefecto

END

```
SELECT CASE 5 WHEN 1 THEN 'uno'
        WHEN 2 THEN 'dos' ELSE 'otro' END;
```

```
SELECT CASE WHEN 1>10 THEN 'verdadero' ELSE 'falso' END;
```

```
SELECT CASE BINARY 'B'
        WHEN 'a' THEN 1 WHEN 'b' THEN 2 ELSE 'FIN' END;
```

```
SELECT CASE BINARY 'B'
        WHEN 'a' THEN 1 WHEN 'b' THEN 2 END;
```

Ejemplo: Hacer una consulta que para los empleados que sumen menos de 3.000 euros entre el salario y la comisión nos indique "Categoria 1", entre 3.001 y 4.000 "Categoria 2", entre 4.001 y 4.500 "Categoria 3", el resto "Categoria 4".

```
SELECT EmNombre, EmSalario, IFNULL(EmComision, 0), EmSalario +
EmComision AS Sueldo, CASE
WHEN EmSalario + EmComision < 3000 THEN 'Categoria 1'
WHEN EmSalario + EmComision BETWEEN 3001 AND 4000 THEN
'Categoria 2'
WHEN EmSalario + EmComision BETWEEN 4001 AND 4500 THEN
'Categoria 3'
ELSE 'Categoria 4'
END
FROM Empleados;
```

EmNombre	EmSalario	IFNULL(EmCo...	Sueldo	CASE
► PONS, CESAR	3100	0	NULL	Categoria 4
LASA, MARIO	3500	1100	4600	Categoria 4
TEROL, LUCIANO	2900	1100	4000	Categoria 2
PEREZ, JULIO	4400	0	NULL	Categoria 4
AGUIRRE, AURED	3100	1100	4200	Categoria 3
PEREZ, MARCOS	4800	500	5300	Categoria 4
VEIGA, JULIANA	3000	0	NULL	Categoria 4
GALVEZ, PILAR	3800	0	NULL	Categoria 4
SANZ, LAVINIA	2800	1000	3800	Categoria 2
ALBA, ADRIANA	4500	0	NULL	Categoria 4
LOPEZ, ANTONIO	7200	0	NULL	Categoria 4
GARCIA, OCTAVIO	3800	800	4600	Categoria 4
FLOR, DOROTEA	2900	800	3700	Categoria 2
POLO, OTILIA	3800	0	NULL	Categoria 4

34 rows fetched in 0,0058s (0,0008s)

Funciones de casting (conversión de tipos)

CAST / CONVERT

CAST(expression AS type)

CONVERT(expression,type)

CONVERT(expr USING transcoding_name)

Las funciones **CAST()** y **CONVERT()** pueden usarse para tomar un valor de un tipo y obtener uno de otro tipo.

Los valores de 'type' pueden ser uno de los siguientes:

- BINARY
- CHAR
- DATE
- DATETIME
- SIGNED {INTEGER}
- TIME
- UNSIGNED {INTEGER}

CAST() y **CONVERT(... USING ...)** forman parte de la sintaxis de SQL-99. La forma de **CONVERT** sin *USING* pertenece a la sintaxis de **ODBC**.

Las funciones de conversión de tipo son corrientes cuando se quiere crear una columna de un tipo específico en una sentencia **CREATESELECT**:

```
CREATE TABLE Tabla SELECT CAST('2000-01-01' AS DATE);
```

También son útiles para ordenar columnas ENUM por orden alfabético. Normalmente, ordenar columnas ENUM usa los valores del orden numérico interno. Haciendo la conversión a CHAR resulta un orden alfabético:

```
SELECT * FROM departamentos d
ORDER BY CAST(DeTipoDirector AS CHAR);
```

DeCodigo	DeCodigoCentro	De...	DePresupuesto	DeDepartamento	DeNombre	DeDirector
144	NULL	F	4500	121	CONTABILIDAD	NULL
140	NULL	F	3520	121	PRODUCCION	NULL
111	20	F	110000	110	SECTOR INDUSTRIAL	180
120	10	F	30000	100	ORGANIZACION	150
123	NULL	F	100000	121	PERSONAL CONTRATADO	150
130	10	P	20000	100	FINANZAS	310
122	10	P	60000	120	PROCESO DE DATOS	350
121	10	P	20000	120	PERSONAL	150
112	20	P	90000	110	SECTOR SERVICIOS	270
110	20	P	150000	100	DIRECCION COMERCIAL	180
100	10	P	120000	NULL	DIRECCION GENERAL	260

CAST(string AS BINARY) es lo mismo que una cadena *BINARY*. **CAST(expr AS CHAR)** trata la expresión como una cadena con el juego de caracteres por defecto.

NOTA: En MySQL 4.0 la función **CAST()** para *DATE*, *DATETIME* o *TIME* sólo marca la columna para que sea de un tipo específico pero no cambia el valor de la columna.

```
SELECT CAST(NOW() AS DATE);
```

CAST(NOW() AS DATE)
2012-03-24

No se puede usar **CAST()** para extraer datos en diferentes formatos, pero en su lugar se pueden usar funciones de cadena como **LEFT()** o **EXTRACT()**.

Para convertir una cadena a valor numérico, normalmente no hay que hacer nada; sencillamente se usa el valor de la cadena como si se tratase de un número:

```
SELECT 1 + '1';
```

1 + '1'
2

Si se usa un número en un contexto de cadena, el número se convertirá automáticamente a cadena BINARY:

```
SELECT CONCAT("hello you ",2);
```

CONCAT("hello you ",2)
hello you 2

MySQL soporta aritmética con valores de 64 bits, tanto con o sin signo. Si se usan operaciones numéricas (como +) y uno de los operandos es un entero sin signo, el resultado será sin signo. Se puede evitar esto usando una conversión de tipo SIGNED y UNSIGNED para los operadores para convertir el resultado a un entero de 64 bits con o sin signo, respectivamente.

```
SELECT CAST(1-2 AS UNSIGNED), CAST(CAST(1-2 AS UNSIGNED) AS SIGNED);
```

CAST(1-2 AS UNSIGNED)	CAST(CAST(1-2 AS UNSIGNED) AS SIGNED)
18446744073709551615	-1

Por otra parte, si cualquiera de los operandos en un valor en punto flotante, el resultado será un valor en punto flotante y no resulta afectado por la regla anterior. (En ese contexto, los valores DECIMAL serán promocionados a valores en punto flotante.)

```
SELECT CAST(1 AS UNSIGNED) - 2.0;
```

CAST(1 AS UNSIGNED) - 2.0
-1.0

Si se usa una cadena en una operación aritmética, será convertida a un número en punto flotante.

CONVERT() con *USING* se usa para convertir datos entre diferentes juegos de caracteres. En MySQL, los nombres de traducción son los mismos que los nombres de los juegos de caracteres correspondientes. Por ejemplo, esta sentencia convierte la cadena 'abc' en el juego de caracteres por defecto del servidor a la cadena correspondiente en el juego de caracteres utf8:

```
SELECT CONVERT('abcñ' USING utf8);
```

