***FUNCIONES***

1. **Funciones matemáticas**
2. **Funciones de cadena de caracteres**
3. **Funciones de fecha y hora**
4. **Funciones de control de flujo**
5. **Funciones de conversión**
6. **Funciones de búsqueda sobre índices FULLTEXT**
7. **Funciones de agregado**
8. **Otras funciones**

***1.- Funciones matemáticas***

Todas las funciones matemáticas devuelven NULL en caso de error:

**ABS(*X*)**

Devuelve el valor absoluto de *X*.

*mysql> SELECT ABS(2);*

*-> 2*

*mysql> SELECT ABS(-32);*

*-> 32*

**ACOS(*X*)**

Devuelve los radianes correspondientes el arco coseno de X. Si X no está comprendido entre –1 y +1 devuelve NULL.

*mysql> SELECT ACOS(1);*

*-> 0.000000*

*mysql> SELECT ACOS(1.0001);*

*-> NULL*

*mysql> SELECT ACOS(0);*

*-> 1.570796*

**ASIN(*X*)**

Devuelve los radianes correspondientes el arco seno de X. Si X no está comprendido entre –1 y +1 devuelve NULL.

*mysql> SELECT ASIN(0.2);*

*-> 0.201358*

*mysql> SELECT ASIN('Madrid');*

*-> 0.000000*

**ATAN(*X*)**

Devuelve los radianes correspondientes el arco tangente de X.

mysql> SELECT ATAN(2);

-> 1.107149

mysql> SELECT ATAN(-2);

-> -1.107149

**CEILING(*X*)**

**CEIL(*X*)**

Devuelve el entero resultado de redondear hacia arriba un número con decimales.

*mysql> SELECT CEILING(1.23);*

*-> 2*

*mysql> SELECT CEIL(-1.23);*

*-> -1*

**COS(*X*)**

Devuelve el coseno de X, donde X se da en radianes.

*mysql> SELECT COS(PI());*

*-> -1.000000*

**COT(*X*)**

Devuelve la cotangente de X, donde X está dado en radianes.

*mysql> SELECT COT(12);*

*-> -1.57267341*

*mysql> SELECT COT(0);*

*-> NULL*

**CRC32(*expr*)**

Obtiene el código de redundancia cíclica correspondiente a una determinada entrada de tipo cadena. Devuelve un entero.

*mysql> SELECT CRC32('MySQL');*

*-> 3259397556*

**DEGREES(*X*)**

Convierte X, de radianes a grados.

*mysql> SELECT DEGREES(PI());*

*-> 180.000000*

**EXP(*X*)**

Devuelve el resultado de elevar el número **e** al valor indicado en X.

*mysql> SELECT EXP(2);*

*-> 7.389056*

*mysql> SELECT EXP(-2);*

*-> 0.135335*

**FLOOR(*X*)**

Obtiene el entero correspondiente al redondeo por abajo de un número con decimales.

*mysql> SELECT FLOOR(1.23);*

*-> 1*

*mysql> SELECT FLOOR(-1.23);*

*-> -2*

**LN(*X*)**

Obtiene el logarirmo neperiano de X.

*mysql> SELECT LN(2);*

*-> 0.693147*

*mysql> SELECT LN(-2);*

*-> NULL*

**LOG(*X*)**

**LOG(*B*,*X*)**

Con un parámetro devuelve el logaritmo neperiano de X.

*mysql> SELECT LOG(2);*

*-> 0.693147*

*mysql> SELECT LOG(-2);*

*-> NULL*

Con dos parámetros devuelve el logaritmo de X en la base B.

*mysql> SELECT LOG(2,65536);*

*-> 16.000000*

*mysql> SELECT LOG(1,100);*

*-> NULL*

**LOG2(*X*)**

Devuelve el logaritmo en base 2 de X.

mysql> SELECT LOG2(65536);

-> 16.000000

LOG2() se suele usar para obtener el número de bits necesarios para representar N valores distintos.

**LOG10(*X*)**

Devuelve el logaritmo en base 10 de X.

mysql> SELECT LOG10(2);

-> 0.301030

mysql> SELECT LOG10(100);

-> 2.000000

**MOD(*N*,*M*)**

***N* % *M***

***N* MOD *M***

Devuelve el módulo o resto de dividir N entre M

*mysql> SELECT MOD(234, 10);*

*-> 4*

*mysql> SELECT MOD(29,9);*

*-> 2*

*mysql> SELECT MOD(34.5,3);*

*-> 1.5*

**PI()**

Devuelve el valor de PI. Por defecto el número de decimales que devuelve es cinco. Pero se puede usar un truco para que devuelva más.

*mysql> SELECT PI();*

*-> 3.141593*

*mysql> SELECT PI()+0.000000000000000000;*

*-> 3.141592653589793116*

**POW(*X*,*Y*)**

**POWER(*X*,*Y*)**

Devuelve el resultado de elevar X a Y.

*mysql> SELECT POW(2,2);*

*-> 4.000000*

*mysql> SELECT POW(2,-2);*

*-> 0.250000*

**RADIANS(*X*)**

Convierte X de grados a radianes.

*mysql> SELECT RADIANS(90);*

*-> 1.570796*

**RAND()**

**RAND(*N*)**

Devuelve un valor float aleatorio en el rango de 0 a 1.0. Si se usa el parámetro entero N, ese valor se usa para que el valor se genere tomando como base para el algoritmo de generación, el número N.

*mysql> SELECT RAND();*

*-> 0.9233482386203*

*mysql> SELECT RAND(20);*

*-> 0.15888261251047*

*mysql> SELECT RAND(20);*

*-> 0.15888261251047*

*mysql> SELECT RAND();*

*-> 0.63553050033332*

mysql> SELECT RAND();

-> 0.70100469486881

Se puede usar random para seleccionar de forma aleatoria varias filas de una tabla, por ejemplo:

*mysql> SELECT \* FROM* tbl\_name *ORDER BY RAND() limit 4;*

**ROUND(*X*)**

**ROUND(*X*,*D*)**

Con un sólo parámetro obtiene X redondeado al entero más próximo. Con dos argumentos, redondea X a un decimal que contiene en la parte decimal el número de cifras indicado en D.

*mysql> SELECT ROUND(-1.23);*

*-> -1*

*mysql> SELECT ROUND(-1.58);*

*-> -2*

*mysql> SELECT ROUND(1.58);*

*-> 2*

*mysql> SELECT ROUND(1.298, 1);*

*-> 1.3*

*mysql> SELECT ROUND(23.298, -1);*

*-> 20*

**SIGN(*X*)**

Devuelve el signo de X como -1, 0, o 1, dependiendo de que X sea negativo, cero, o positivo.

**SIN(*X*)**

Devuelve el seno de X, donde X está dado en radianes.

*mysql> SELECT SIN(PI());*

*-> 0.000000*

**SQRT(*X*)**

Devuelve la raíz cuadrada de X.

*mysql> SELECT SQRT(4);*

*-> 2.000000*

**TAN(*X*)**

Devuelve la tangente de X, donde X está dado en radianes.

*mysql> SELECT TAN(PI()+1);*

*-> 1.557408*

**TRUNCATE(*X*,*D*)**

Devuelve el número X truncado a D decimales.

*mysql> SELECT TRUNCATE(1.223,1);*

*-> 1.2*

*mysql> SELECT TRUNCATE(1.999,1);*

*-> 1.9*

*mysql> SELECT TRUNCATE(1.999,0);*

*-> 1*

*mysql> SELECT TRUNCATE(-1.999,1);*

*-> -1.9*

*mysql> SELECT TRUNCATE(122,-2);*

*-> 100*

***2.- Funciones de cadena de caracteres***

**Para todas las funciones de cadena que operen con posiciones en la cadena, la primera posición se numera con 1.**

**ASCII(*str*)**

Devuelve el ASCII del primer carácter de la cadena. Devuelve 0 si cadena vacía y NULL si la cadena es NULL.

*mysql> SELECT ASCII('2');*

*-> 50*

*mysql> SELECT ASCII('dx');*

*-> 100*

**BIN(*N*)**

Devuelve una cadena que representa un número BIGINT en binario.

*mysql> SELECT BIN(12);*

*-> '1100'*

**BIT\_LENGTH(*str*)**

Devuelve el tamaño de la cadena **str** en bits.Returns the length of the string *str* in bits.

*mysql> SELECT BIT\_LENGTH('text');*

*-> 32*

**CHAR(*N*,...)**

CHAR() interpreta los parámetros como enteros que representan códigos ASCII y devuelve la cadena equivalente. Los valores NULL no los tiene en cuenta.

*mysql> SELECT CHAR(77,121,83,81,'76');*

*-> 'MySQL'*

*mysql> SELECT CHAR(77,77.3,'77.3');*

*-> 'MMM'*

**CHAR\_LENGTH(*str*)**

Devuelve el número de caracteres de la cadena str.

*mysql> SELECT CHAR\_LENGTH('text');*

*-> 4*

**CONCAT(cad1*1*,cad2,...)**

Devuelve la cadena resultado de concatenar todas las cadenas pasadas como parámetros. Devuelve NULL si alguna de las cadenas es NULL. Si hay algún parámetro numérico, se realiza su conversión a cadena.

*mysql> SELECT CONCAT('My', 'S', 'QL');*

*-> 'MySQL'*

*mysql> SELECT CONCAT('My', NULL, 'QL');*

*-> NULL*

*mysql> SELECT CONCAT(14.3);*

*-> '14.3'*

**CONCAT\_WS(*separador*,cad1,*cad2*,...)**

Es igual que CONCAT aunque al concatenar separa las cadenas con el separador que se indique.

mysql> SELECT CONCAT\_WS('-','Herrero','de Cos','Luis');

-> 'Herrero-de Cos-Luis'

mysql> SELECT CONCAT\_WS(',','First name',NULL,'Last Name');

-> 'First name,Last Name'

**CONV(*N*, base1, *base2*)**

Convierte un número expresado en base1 a número expresado en base2. La base máxima es 36.

*mysql> SELECT CONV('a',16,2);*

*-> '1010'*

*mysql> SELECT CONV('6E',18,8);*

*-> '172'*

*mysql> SELECT CONV(-17,10,-18);*

*-> '-H'*

**ELT(*N*, cad1, cad2, cad3,...)**

Devuelve cad1 si *N* = 1, cad2 if *N* = 2, y así sucesivamente. Esta función es la complementaria a FIELD.

*mysql> SELECT ELT(4, 'ej', 'Heja', 'hej', 'foo');*

*-> 'foo'*

**FIELD(cadena,cad1*1*, *cad2*, *cad3*,...)**

Devuelve la posición donde se encuentra cadena dentro de la lista de cadenas (cad1, cad2, ...) que se indican a continuación.

*mysql> SELECT FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo');*

*-> 2*

*mysql> SELECT FIELD('fo', 'Hej', 'ej', 'Heja', 'hej', 'foo');*

*-> 0*

**FIND\_IN\_SET(*cad*, *cadlist*)**

Devuelve un valor de 1 a N si la cadena cad está dentro de las subcadenas de cadlist. Una lista de cadenas no es mas que una cadena compuesta de varias subcadenas separadas con comas. Devuelve 0 si la cadena no está en la lista. En la lista se puede especificar una variable de tipo SET.

mysql> SELECT FIND\_IN\_SET('b','a,b,c,d');

-> 2

**HEX(*Numero o cadena*)**

Con un número devuelve la representación hexadecimal del número. Si se pasa una cadena, devuelve la representación ASCII en hexadecimal de la cadena.

*mysql> SELECT HEX(255);*

*-> 'FF'*

*mysql> SELECT HEX('abc');*

*-> 616263*

**INSERT(*cadena, posición, longitud, nueva\_cad*)**

Devuelve el resultado de insertar cadena en nueva\_cad desde la posición indicada en nueva\_cad y con el número de caracteres indicados en longitud, reemplazando a los correspondientes caracteres de nueva\_cad.

mysql> SELECT INSERT('Quadratic', 3, 4, 'What');

-> 'QuWhattic'

mysql> SELECT INSERT('Quadratic', -1, 4, 'What');

-> 'Quadratic'

mysql> SELECT INSERT('Quadratic', 3, 100, 'What');

-> 'QuWhat'

**INSTR(*cadena*, *subcadena*)**

Devuelve la posición a partir de la cual se encuentra la primera aparición de subcadena en cadena. Si no se encuentra la subcadena en cadena, devuelve cero.

*mysql> SELECT INSTR('foobarbar', 'bar');*

*-> 4*

*mysql> SELECT INSTR('xbar', 'foobar');*

*-> 0*

**LEFT(cadena, N)**

Devuelve los N primeros caracteres por la izquierda de cadena.

*mysql> SELECT LEFT('foobarbar', 5);*

*-> 'fooba'*

**LENGTH(*cadena*)**

Devuelve la longitud de cadena medida en bytes. Si se está usando UNICODE, una cadena de N letras tendría el doble de bytes.

*mysql> SELECT LENGTH('text');*

*-> 4*

**LOAD\_FILE(*fichero*)**

Devuelve el contenido del fichero como una cadena.

mysql> UPDATE *tbl\_name*  SET *blob\_column*=LOAD\_FILE('\\tmp\\picture') WHERE id=1;

**LOCATE(*subcad*, *cad*)**

**LOCATE(*subcad*, *cad*, *pos*)**

La primera sintaxis devuelve la posición de la primera aparición de subcadena en cadena. La segunda devuelve la posición de la primera aparición de subcadena en cadena desde la posición indicada en pos (busca desde pos pero devuelve la posición real en la cadena). Devuelve 0 si la subcadena no se encuentra en cadena.

*mysql> SELECT LOCATE('bar', 'foobarbar');*

*-> 4*

*mysql> SELECT LOCATE('xbar', 'foobar');*

*-> 0*

*mysql> SELECT LOCATE('bar', 'foobarbar',5);*

*-> 7*

**LCASE(cadena)**

**LOWER(cadena)**

Devuelve la cadena con todas las letras en minúscula.

*mysql> SELECT LOWER('QUADRATICALLY');*

*-> 'quadratically'*

**LPAD(*cad*, *longitud*, *cad2*)**

Devuelve cad añadiéndole por la izquierda la cadena cad2 repetida tantas veces como sea necesario para que el conjunto de cad y los caracteres añadidos lleguen a tener tantos caracteres como los indicados en longitud.

*mysql> SELECT LPAD('hi',13,'abc');*

*abcabcabcabhi|*

*mysql> SELECT LPAD('hi',1,'??');*

*-> 'h'*

**LTRIM(*cad*)**

Devuelve la cadena cad tras eliminarle los espacios que tuviera por la izquierda.

*mysql> SELECT LTRIM(' barbar');*

*-> 'barbar'*

**REPEAT(*cad, N*)**

Devuelve la cadena cad repetida N veces.

*mysql> SELECT REPEAT('MySQL', 3);*

*-> 'MySQLMySQLMySQL'*

**REPLACE(*cad*, *cad\_origen, cad\_destino*)**

Devuelve la cadena cad sustituyendo las ocurrencias de cad\_origen en cad por ocurrencias de cad\_destino.

*mysql> SELECT REPLACE('www.mysql.com', 'w', 'Ww');*

*-> 'WwWwWw.mysql.com'*

**REVERSE(*str*)**

Devuelve la cadena str con los caracteres en orden inverso.

*mysql> SELECT REVERSE('abc');*

*-> 'cba'*

**RIGHT(*cad , N*)**

Devuleve los N caracteres de la derecha de la cadena cad.

*mysql> SELECT RIGHT('foobarbar', 4);*

*-> 'rbar'*

**RPAD(*cad*, *longitud*, *cad2*)**

Devuelve cad añadiéndole por la derecha la cadena cad2 repetida tantas veces como sea necesario para que el conjunto de cad y los caracteres añadidos lleguen a tener tantos caracteres como los indicados en longitud.

*mysql> SELECT RPAD('hi',5,'?0');*

*-> 'hi?0?'*

*mysql> SELECT RPAD('hi',1,'?');*

*-> 'h'*

**RTRIM(*cad*)**

Devuelve la cadena cad eliminando los espacios que tenga por la derecha.

*mysql> SELECT RTRIM('barbar ');*

*-> 'barbar'*

**SPACE(*N*)**

Devuelve una cadena formada por N espacios.

*mysql> SELECT SPACE(6);*

*-> ' '*

**SUBSTRING(*cad*, *pos*)**

**SUBSTRING(cad FROM *pos*)**

**SUBSTRING(cad, *pos*, *len*)**

**MID(cad, pos, len)**

**SUBSTRING(*cad*  FROM *pos* FOR *len*)**

Las sintaxis que no usan el parámetro **len** devuelven una subcadena formada por los caracteres de la cadena **cad** a partir de la posición **pos** de ésta. Las sintaxis con parámetro **len** realizan lo mismo que la anterior pero devolviendo únicamente len caracteres.

*mysql> SELECT SUBSTRING('Quadratically',5);*

*-> 'ratically'*

*mysql> SELECT SUBSTRING('foobarbar' FROM 4);*

*-> 'barbar'*

*mysql> SELECT SUBSTRING('Quadratically',5,6);*

*-> 'ratica'*

**SUBSTRING\_INDEX(*cad*, *delimitador*, *N*)**

Si N es positivo, devuelve una subcadena formada por los caracteres que hay desde el inicio de cad hasta la aparición N del carácter delimitador. Si N es negativo, devuelve una subcadena formada por los caracteres que hay desde la aparición N del delimitador en cad hasta el final de ésta.

mysql> SELECT SUBSTRING\_INDEX('www.mysql.com', '.', 2);

-> 'www.mysql'

mysql> SELECT SUBSTRING\_INDEX('www.mysql.com', '.', -2);

-> 'mysql.com'

**TRIM([{BOTH | LEADING | TRAILING} [*cadborr*] FROM] *str*)**

**TRIM([*cadborr* FROM] *str*)**

Devuelve la cadena **cad** eliminando las apariciones de **cadborr** a la izquierda y a la derecha de la cadena. Por defecto se borra por los dos lados, si se unas LEADING se borra sólo por la izquierda y si se usa TRAILING se borra por la derecha. BOTH borra a ambos lados. Si no se especifica **cadborr**, se eliminan espacios.

*mysql> SELECT TRIM(' bar ');*

*-> 'bar'*

*mysql> SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx');*

*-> 'barxxx'*

*mysql> SELECT TRIM(BOTH 'x' FROM 'xxxbarxxx');*

*-> 'bar'*

*mysql> SELECT TRIM(TRAILING 'xyz' FROM 'barxxyz');*

*-> 'barx'*

**UNHEX(*cad*)**

Es la función opuesta a HEX(cad). Devuelve la cadena correspondiente resultado de representar en ASCII cada pareja de dígitos hexadecimales pasados como parámetros.

mysql> SELECT UNHEX('4D7953514C');

-> 'MySQL'

mysql> SELECT 0x4D7953514C;

-> 'MySQL'

**UPPER(*cad*)**

**UCASE(*cad)***

Devuelve la cadena cad con todos sus caracteres alfabéticos cambiados a mayúsculas.

*mysql> SELECT UPPER('Hej');*

*-> 'HEJ'*

**STRCMP(*cad1*, *cad2*)**

Devuelve 0 si las cadenas cad1 y cad2 son iguales, -1 si la primera es menor alfabéticamente y +1 si es mayor. No diferencia entre mayúsculas y minúscula.

*mysql> SELECT STRCMP('text10', 'text2');*

*-> -1*

*mysql> SELECT STRCMP('text2', 'TEXT2');*

*-> 0*

*mysql> SELECT STRCMP('text', 'text');*

*-> 0*

*mysql> SELECT STRCMP('text101', 'text10');*

*-> 1*

***3.- Funciones de Fecha y Hora***

Las funciones que operan con fechas normalmente pueden trabajar con datos que contienen también la hora, en cuyo caso la ignoran. De igual forma, las funciones que operan con horas pueden trabajar con datos que contienen fechas y horas ignorando la parte correspondiente a la fecha.

Si una fecha usada por una función contiene un 0 en día o en mes, o la fecha está incompleta, la función devolverá 0. Si una fecha o una hora usadas por una función son incorrectas, entonces la función devuelve NULL.

**ADDDATE(*fecha*, INTERVAL *expr*  *tipo*)**

**DATE\_ADD(*fecha*, INTERVAL *expr*  *tipo*)**

**ADDDATE(*fecha*, *días*)**

Cuando se usan las sintaxis con INTERVAL obtiene la fecha resultado de sumar al parámetro fecha el intervalo de días, meses, años o semanas (DAY, MONTH, YEAR) indicados en **expr** que podrá ser negativo restando a la fecha el intervalo**.** También se pueden usar otros tipos de intervalos que se representan en una tabla relativa a los parámetros de la función DATE\_SUB y se pueden expresar fechas-horas**.** Si no se usa INTERVAL, se obtiene la fecha correspondiente de sumar al parámetro fecha los días indicados.

*mysql> SELECT DATE\_ADD('2004-01-02', INTERVAL 31 DAY);*

*-> '2004-02-02'*

*mysql> SELECT ADDDATE('2004-12-02', INTERVAL 30 DAY);*

*-> '2005-01-01'*

*mysql> SELECT ADDDATE('2004-12-02', 100);*

*-> '2005-03-12'*

*mysql> SELECT ADDDATE('2004-12-02', INTERVAL 6 MONTH);*

*-> '2005-06-02'*

*mysql> SELECT ADDDATE('2004-12-30', INTERVAL 4 WEEK);*

*-> '2005-06-02'*

*mysql> SELECT DATE\_ADD('2004-01-02 23:50:50', INTERVAL 12 hour);*

*-> ' 2004-01-03 11:50:50'*

**ADDTIME(*expr*,*expr2*)**

Añade ***expr2*** a ***expr*** y devuelve el resultador. Las dos expresiones pueden representar horas o fechas-horas.

*mysql> SELECT ADDTIME('2004-12-31 23:59:59.999999', '1 0:0:1.000002');*

*-> '2005-01-02 00:00:01.000001'*

*mysql> SELECT ADDTIME('01:00:59', '02:04:00');*

*-> ' 03:04:59'*

**CONVERT\_TZ(*dt*, *zona\_or*, *zona\_dest*)**

Convierte un valor fecha-hora **dt** de una zona horaria zona\_or a la fecha y hora de otra zona horaria **zona\_dest.** Para zonas horarias se pueden usar valores como **'Europe/Helsinki', 'GMT', 'US/Eastern', 'MET', 'SYSTEM', etc**. También se pueden usar dos horas diferenciadas. ***Para que se pueda trabajar con la función tiene que existir una tabla de sistema con las zonas horarias.***

*mysql> SELECT CONVERT\_TZ('2004-01-01 12:00:00','GMT','MET');*

*-> '2004-01-01 13:00:00'*

*mysql> SELECT CONVERT\_TZ('2004-01-01 12:00:00','+00:00','-07:00');*

*-> '2004-01-01 05:00:00'*

**CURDATE()**

**CURRENT\_DATE**

**CURRENT\_DATE()**

Devuelven la fecha actual en formato 'AAAA-MM-DD' o AAAAMMDD, dependiendo de si la función se usa o no en un contexto numérico.

*mysql> SELECT CURDATE();*

*-> '2004-11-04'*

*mysql> SELECT CURDATE() + 0;*

*-> 20041104*

**CURTIME()**

**CURRENT\_TIME**

**CURRENT\_TIME()**

Devuelven la hora actual en formato 'HH:MM:SS' o HHMMSS, dependiendo de si se usa o no en contexto numérico.

*mysql> SELECT CURTIME();*

*-> '17:50:26'*

*mysql> SELECT CURTIME() + 0;*

*-> 175026*

**DATE(*expr*)**

Obtiene la parte correspondiente a la fecha del contenido de **expr.**

*mysql> SELECT DATE('2003-12-31 01:02:03');*

*-> '2003-12-31'*

**DATEDIFF(*expr*,*expr2*)**

Devuelve el número de días entre comprendidos entre **expr2** y **expr.** Para el cálculo sólo se usa la parte correspondiente a la fecha de las expresiones y no se tiene en cuenta la hora.

*mysql> SELECT DATEDIFF('2004-12-31 23:59:59','2004-12-30 00:00:01');*

*-> 1*

*mysql> SELECT DATEDIFF('2004-11-30 23:59:59','2004-12-31');*

*-> -31*

**DATE\_SUB(*fecha*, INTERVAL *expr* *tipo*)**

**Fecha** es de tipo DATE o DATETIME y especifica una fecha desde la que se restan el número de unidades de tiempo especificadas en **expr.** El parámetro **tipo** expresa el tipo de unidad de tiempo. Si en expr pasamos un valor negativo se suman las unidades de tiempo a fecha.

|  |  |
| --- | --- |
| ***tipo*** | **Valor indicado en expr** |
| MICROSECOND | Microsegundos |
| SECOND | Segundos |
| MINUTE | Minutos |
| HOUR | Horas |
| DAY | Días |
| WEEK | Semanas |
| MONTH | Meses |
| QUARTER | Trimestres |
| YEAR | Años |
| SECOND\_MICROSECOND | Segundos.microsegundos |
| MINUTE\_MICROSECOND | Minutos.microsegundos |
| MINUTE\_SECOND | Minutos:segundos |
| HOUR\_MICROSECOND | Horas.microsegundos |
| HOUR\_SECOND | Horas:minutos:segundos |
| HOUR\_MINUTE | Horas:minutos |
| DAY\_MICROSECOND | 'dias.microsegundos' |
| DAY\_SECOND | 'dias horas:minutos:segundos' |
| DAY\_MINUTE | 'días horas:minutos' |
| DAY\_HOUR | 'días horas' |
| YEAR\_MONTH | 'años-meses' |

'

*mysql> SELECT DATE\_ADD('1997-12-31 23:59:59', INTERVAL '1:1' MINUTE\_SECOND);*

*-> '1998-01-01 00:01:00'*

*mysql> SELECT DATE\_SUB('1998-01-01 00:00:00', INTERVAL '1 1:1:1' DAY\_SECOND);*

*-> '1997-12-30 22:58:59'*

*mysql> SELECT DATE\_SUB('1998-01-01 00:00:00', INTERVAL '-1 10' DAY\_HOUR);*

*-> ' 1998-01-02 10:00:00'*

*mysql> SELECT DATE\_SUB('1998-01-02', INTERVAL 31 DAY);*

*-> '1997-12-02'*

*mysql> SELECT DATE\_SUB('2002-01-01 00:00:01.000002' ,INTERVAL '1.999999' SECOND\_MICROSECOND);*

*-> ' 2001-12-31 23:59:59.000003'*

**DATE\_FORMAT(*fecha*, *formato*)**

Devuelve la fecha expresado con el formato que se indique. Para el formato se pueden usar los siguientes especificadores de formato. El formato debe estar encerrado entre comillas.

|  |  |
| --- | --- |
| **Especificador** | **Descripción** |
| %a | Día de la semana abreviado (Sun..Sat) |
| %b | Nombre del mes abreviado (Jan..Dec) |
| %c | Mes numérico (1..12) |
| %D | Día del mes con sufijo inglés (1st, 2nd, 3rd, ...) |
| %d | Día del mes numérico con dos cifras (01..31) |
| %e | Día del mes numérico con 1 o 2 cifras (1..31) |
| %f | Microsegundos (000000..999999) |
| %H | Hora (00..23) |
| %h | Hora pm (01..12) |
| %I | Hora pm (01..12) |
| %i | Minutos (00..59) |
| %j | Día del año (001..366) |
| %k | Hora con 1 o 2 cifras (0..23) |
| %l | Hora PM con 1 o 2 cifras (1..12) |
| %M | Nombre completo del mes (January..December) |
| %m | Mes numérico (01..12) |
| %p | AM o PM |
| %r | Hora de 12 horas (**hh:mm:ss** seguido de **AM** o **PM**) |
| %S | Segundos (00..59) |
| %s | Segundos (00..59) |
| %T | Hora de 24 horas (hh:mm:ss) |
| %U | Semana (00..53), siendo el domingo el primer día de la semana. |
| %u | Semana (00..53), siendo el lunes el primer día de la semana |
| %V | Semana (01..53), siendo el domingo el primer día de la semana. |
| %v | Semana (01..53), siendo el lunes el primer día de la semana |
| %W | Nombre completo del día de la semana (Sunday..Saturday) |
| %w | Día de la semana numérico (0=Sunday..6=Saturday) |
| %X | Año con cuatro dígitos para usarse con %V |
| %x | Año con cuatro dígitos para usarse con %v |
| %Y | Año con cuatro dígitos |
| %y | Año con dos dígitos |
| %% | El carácter '%'. |

*mysql> SELECT DATE\_FORMAT('1997-10-04 22:23:00', '%W %M %Y');*

*-> 'Saturday October 1997'*

*mysql> SELECT DATE\_FORMAT('1997-10-04 22:23:00', 'Son las %H:%i horas');*

*-> 'Son las 22:23 horas'*

*mysql> SELECT DATE\_FORMAT('1997-10-04 22:23:00', '%D %y %a %d %m %b %j');*

*-> '4th 97 Sat 04 10 Oct 277'*

*mysql> SELECT DATE\_FORMAT('1997-10-04 22:23:00', '%H %k %I %r %T %S %w');*

*-> '22 22 10 10:23:00 PM 22:23:00 00 6'*

*mysql> SELECT DATE\_FORMAT('1999-01-01', '%X %V');*

*-> '1998 52'*

**DAY(*date*)**

**DAYOFMONTH(*fecha*)**

Devuelve el día del mes correspondiente a fecha.

*mysql> SELECT DAYOFMONTH('2004-02-03');*

*-> 3*

**DAYNAME(*fecha*)**

Devuelve el nombre, en inglés, del día de la semana correspondiente a fecha.

mysql> SELECT DAYNAME('2004-11-05');

-> 'Friday'

**DAYOFWEEK(*date*)**

Devuelve el día de la semana numérico correspondiente a fecha *date* (1 = Sunday, 2 = Monday, ..., 7 = Saturday). Estos valores corresponden al estándar ODBC.

mysql> SELECT DAYOFWEEK('2004-11-05');

-> 6

**DAYOFYEAR(*fecha*)**

Devuelve el día del año correspondiente a fecha.

mysql> SELECT DAYOFYEAR('2004-11-05');

-> 310

**EXTRACT(*tipo* FROM *fecha-hora*)**

Usa los mismos tipos que DATE\_ADD() or DATE\_SUB(), y extrae la parte indicada en tipo del parámetro fecha-hora.

mysql> SELECT EXTRACT(YEAR FROM '2004-07-02');

-> 2004

mysql> SELECT EXTRACT(YEAR\_MONTH FROM '2004-07-02 01:02:03');

-> 200407

mysql> SELECT EXTRACT(DAY\_MINUTE FROM '2004-07-02 01:02:03');

-> 20102

mysql> SELECT EXTRACT(MICROSECOND FROM '2003-01-02 10:30:00.00123');

-> 123

**FROM\_DAYS(*N*)**

Dado un número de días N, devuelve la fecha equivalente a esos días.

*mysql> SELECT FROM\_DAYS(732251);*

*-> ' 2004-11-05'*

**GET\_FORMAT(DATE|TIME|DATETIME, 'EUR'|'USA'|'JIS'|'ISO'|'INTERNAL')**

Devuelve una cadena de formato. Esta función suele usarse en combinación con DATE\_FORMAT() y con STR\_TO\_DATE(). Como puede verse hay tres posibles valores para el primer parámetro y cinco posibles valores para el segundo. Suele usarse como parámetro para funciones que usan formatos para trabajar. Las posibles combinaciones de valores para los parámetros de GET\_FORMAT y los formatos equivalentes se presentan a continuación:

|  |  |
| --- | --- |
| **Llamada a la función** | **Resultado** |
| GET\_FORMAT(DATE,'USA') | '%m.%d.%Y' |
| GET\_FORMAT(DATE,'JIS') | '%Y-%m-%d' |
| GET\_FORMAT(DATE,'ISO') | '%Y-%m-%d' |
| GET\_FORMAT(DATE,'EUR') | '%d.%m.%Y' |
| GET\_FORMAT(DATE,'INTERNAL') | '%Y%m%d' |
| GET\_FORMAT(DATETIME,'USA') | '%Y-%m-%d-%H.%i.%s' |
| GET\_FORMAT(DATETIME,'JIS') | '%Y-%m-%d %H:%i:%s' |
| GET\_FORMAT(DATETIME,'ISO') | '%Y-%m-%d %H:%i:%s' |
| GET\_FORMAT(DATETIME,'EUR') | '%Y-%m-%d-%H.%i.%s' |
| GET\_FORMAT(DATETIME,'INTERNAL') | '%Y%m%d%H%i%s' |
| GET\_FORMAT(TIME,'USA') | '%h:%i:%s %p' |
| GET\_FORMAT(TIME,'JIS') | '%H:%i:%s' |
| GET\_FORMAT(TIME,'ISO') | '%H:%i:%s' |
| GET\_FORMAT(TIME,'EUR') | '%H.%i.%S' |
| GET\_FORMAT(TIME,'INTERNAL') | '%H%i%s' |

Ejemplos de uso en combinación con otras funciones:

mysql> SELECT DATE\_FORMAT('2004-10-03',GET\_FORMAT(DATE,'EUR'));

-> '03.10.2004'

mysql> SELECT STR\_TO\_DATE('10.31.2004',GET\_FORMAT(DATE,'USA'));

-> 2004-10-31

**HOUR(*time*)**

Devuelve la hora almacenada en **time.** El valor devuelto normalmente estará comprendido entre 0 y 23.

*mysql> SELECT HOUR('10:05:03');*

*-> 10*

Sin embargo, se admite usar para horas valores mayores por lo cual la función devuelve esos valores.

*mysql> SELECT HOUR('272:59:59');*

*-> 272*

**LAST\_DAY(*Fecha*)**

Devuelve la fecha del último día del mes correspondiente a la fecha dada como parámetro.

*mysql> SELECT LAST\_DAY('2003-02-05');*

*-> '2003-02-28'*

*mysql> SELECT LAST\_DAY('2004-02-05');*

*-> '2004-02-29'*

*mysql> SELECT LAST\_DAY('2004-01-01 01:01:01');*

*-> '2004-01-31'*

*mysql> SELECT LAST\_DAY('2003-03-32');*

*-> NULL*

**MAKEDATE(*año*, *DiaDe Año*)**

Devuelve una fecha a partir del año y día del año (entre 1 y 366) dados como parámetros.

*mysql> SELECT MAKEDATE(2001,31), MAKEDATE(2001,32);*

*-> '2001-01-31', '2001-02-01'*

*mysql> SELECT MAKEDATE(2001,365), MAKEDATE(2004,365);*

*-> '2001-12-31', '2004-12-30'*

*mysql> SELECT MAKEDATE(2001,0);*

*-> NULL*

**MAKETIME(*Hora*, *minutos*, *segundos*)**

Devuelve un dato de tipo TIME a partir de la hora, los minutos y segundos dados como parámetros.

*mysql> SELECT MAKETIME(12,15,30);*

*-> '12:15:30'*

**MICROSECOND(*expr*)**

Devuelve la parte correspondiente a los microsegundos a partir de la expresión **expr** dada en forma de hora o de fecha-hora. El valor devuelto estará comprendido entre 0 y 999999.

*mysql> SELECT MICROSECOND('12:00:00.123456');*

*-> 123456*

*mysql> SELECT MICROSECOND('1997-12-31 23:59:59.000010');*

*-> 10*

**MINUTE(expr)**

Devuelve la parte correspondiente a los minutos a partir de la expresión **expr** dada en forma de hora o de fecha-hora. El valor devuelto estará comprendido entre 0 y 59.

mysql> SELECT MINUTE('00-02-03 10:05:03');

-> 5

**MONTH(*expr*)**

Devuelve la parte correspondiente al mes (numérico) de la expresión **expr** dada en forma de fecha o de fecha-hora. El valor devuelto estará comprendido entre 1 y 12.

*mysql> SELECT MONTH('2004-12-03');*

*-> 12*

**MONTHNAME(*date*)**

Devuelve la parte correspondiente al mes (nombre completo en inglés) de la expresión **expr** dada en forma de fecha o de fecha-hora.

*mysql> SELECT MONTHNAME('1998-02-05');*

*-> 'February'*

**NOW()**

**CURRENT\_TIMESTAMP**

**CURRENT\_TIMESTAMP()**

**LOCALTIME**

**LOCALTIME()**

**LOCALTIMESTAMP**

**LOCALTIMESTAMP()**

**SYSDATE()**

Devuelven la fecha y hora actuales en forma 'AAAA-MM-DD HH:MM:SS' o en forma AAAAMMDDHHMMSS, dependiendo que la función se use en contexto de cadenas o en contexto numérico (resultado opera con algún número).

*mysql> SELECT NOW();*

*-> '2004-11-05 18:50:26'*

*mysql> SELECT NOW() + 0;*

*-> 20041105185026*

**PERIOD\_ADD(*P*,*N*)**

Devuelve en forma AAAAMM el resultado de añadir N meses al periodo P dado en forma AAMM o AAAAMM.

*mysql> SELECT PERIOD\_ADD(0401,2);*

*-> 200403*

**PERIOD\_DIFF(*P1*,*P2*)**

Devuelve los meses comprendidos entre los periodos P1 y P2. Ambos periodos deben estar en los formatos AAMM o AAAAMM.

*mysql> SELECT PERIOD\_DIFF(0402,200303);*

*-> 11*

*mysql> SELECT PERIOD\_DIFF(200303,0402);*

*-> - 11*

**QUARTER(expr)**

Devuelve el trimestre correspondiente a la expresión **expr** dada en forma de fecha o de fecha-hora.

*mysql> SELECT QUARTER('04-04-01');*

*-> 2*

**SECOND(*expr*)**

Devuelve la parte correspondiente a los segundos de la expresión **expr** dada en forma de hora o de fecha-hora. Devuelve valores comprendidos entre 0 y 59.

*mysql> SELECT SECOND('10:05:03'); -> 3*

**SEC\_TO\_TIME(*segundos*)**

Devuelve el resultado de convertir **segundos** a formato 'HH:MM:SS' o formato HHMMSS dependiendo del contexto de utilización de la función.

*mysql> SELECT SEC\_TO\_TIME(2378);*

*-> '00:39:38'*

*mysql> SELECT SEC\_TO\_TIME(2378) + 0;*

*-> 3938*

**STR\_TO\_DATE(*cadena*, *formato*)**

Es la función inversa de la función DATE\_FORMAT(). Devuelve el resultado de convertir una **cadena** representada en el **formato** indicado a un valor de tipo DATETIME, DATE o TIME. Si la cadena no está en el formato indicado, la función devuelve NULL.

*mysql> SELECT STR\_TO\_DATE('03.10.2003 09.20', '%d.%m.%Y %H.%i');*

*-> '2003-10-03 09:20:00'*

*mysql> SELECT STR\_TO\_DATE('10arp', '%carp');*

*-> '0000-10-00 00:00:00'*

*mysql> SELECT STR\_TO\_DATE('2003-15-10 00:00:00', '%Y-%m-%d %H:%i:%s');*

*-> NULL*

**SUBDATE(*fecha*, INTERVAL *expr* *tipo*)**

**SUBDATE(*expr*, *dias*)**

Cuando usa INTERVAL, es equivalente a DATE\_SUB. La segunda sintaxis devuelve la fecha o fecha-hora resultado restar los días indicados a la fecha o fecha-hora indicadas en **expr.**

*mysql> SELECT SUBDATE('2005-01-02 12:00:00', 31);*

*-> '2004-12-02 12:00:00'*

**SUBTIME(*expr*,*expr2*)**

SUBTIME resta *expr2* de *expr* y devuelve el resultado. Las expresiones **expr** y **expr2** pueden representar una hora o una fecha-hora.

mysql> SELECT SUBTIME('2004-12-31 23:59:59.999999', '1 1:1:1.000002');

-> '2004-12-30 22:58:58.999997'

mysql> SELECT SUBTIME('01:00:00.999999', '02:00:00.999998');

-> '-00:59:59.999999'

**TIME(*expr*)**

Devuelve el valor TIME almacenado en **expr.**

*mysql> SELECT TIME('2003-12-31 01:02:03');*

*-> '01:02:03'*

*mysql> SELECT TIME('2003-12-31 01:02:03.000123');*

*-> '01:02:03.000123'*

**TIMEDIFF(*expr*,*expr2*)**

Devuelve las horas:minutos:segundos de diferencia entre los valores TIME o DATETIME de expr y expr2, debiendo ser ambas del mismo tipo.

mysql> SELECT TIMEDIFF('2000:01:01 00:00:00', '2000:01:01 00:00:00.000001');

-> '-00:00:00.000001'

mysql> SELECT TIMEDIFF('1997-12-31 23:59:59.000001', '1997-12-30 01:01:01.000002');

-> '46:58:57.999999'

**TIMESTAMP(*expr*)**

**TIMESTAMP(*expr*,*expr2*)**

Con un parámetro, devuelve la expresión **expr** dada como tipo DATE o DATETIME convertida a tipo DATETIME. Con dos parámetros devuelve el resultado de añadir el valor TIME de **expr2** al valor DATE o DATETIME de **expr.**

*mysql> SELECT TIMESTAMP('2003-12-31');*

*-> '2003-12-31 00:00:00'*

*mysql> SELECT TIMESTAMP('2003-12-31 12:00:00','12:00:00');*

*-> '2004-01-01 00:00:00'*

**TIMESTAMPADD(*intervalo*, *int\_expr*, *datetime\_expr*)**

Devuelve el resultado de añadir el entero *int\_expr* al valor DATE o DATETIME de *datetime\_expr*. El entero expresa las unidades del intervalo que se especifique en el primer parámetro. El intervalo puede tener los valores: FRAC\_SECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, o YEAR.

*mysql> SELECT TIMESTAMPADD(MINUTE,1,'2003-01-02');*

*-> '2003-01-02 00:01:00'*

*mysql> SELECT TIMESTAMPADD(WEEK,1,'2003-01-02');*

*-> '2003-01-09'*

**TIMESTAMPDIFF(*interval*, *datetime\_expr1*, *datetime\_expr2*)**

Devuelve la diferencia en forma de un número de unidades del intervalo indicado entre las expresiones DATE o DATETIME *datetime\_expr1* y *datetime\_expr2*. Las unidades para intervalo son las mismas que las de la función TIMESTAMPADD().

*mysql> SELECT TIMESTAMPDIFF(MONTH,'2003-02-01','2003-05-01');*

*-> 3*

*mysql> SELECT TIMESTAMPDIFF(YEAR,'2002-05-01','2001-01-01');*

*-> -1*

**TIME\_FORMAT(*time*, *formato*)**

Devuelve un TIME correspondiente a la conversión del valor **time** (que debe ser de tipo TIME) representado en el **formato** indicado.

*mysql> SELECT TIME\_FORMAT('100:00:00', '%H %k %h %I %l');*

*-> '100 100 04 04 4'*

**TIME\_TO\_SEC(*time*)**

Devuelve el parámetro *time* convertido a segundos totales desde las 00:00:00.

mysql> SELECT TIME\_TO\_SEC('22:23:00');

-> 80580

mysql> SELECT TIME\_TO\_SEC('00:39:38');

-> 2378

**TO\_DAYS(fecha)**

Devuelve el número de días transcurridos desde el comienzo del año 0 hasta la fecha indicada.

*mysql> SELECT TO\_DAYS(950501);*

*-> 728779*

*mysql> SELECT TO\_DAYS('1997-10-07');*

*-> 729669*

*mysql> SELECT TO\_DAYS('2004-03-07')- TO\_DAYS('04-02-07');*

*-> 29*

**UTC\_DATE()**

Devuelve la fecha UTC actual en formato 'AAAA-MM-DD' o AAAAMMDD, dependiendo del contexto de cadena de caracteres o numérico.

*mysql> SELECT UTC\_DATE(), UTC\_DATE() + 0;*

*-> '2004-11-05', 20041105*

**UTC\_TIME()**

Devuelve la hora UTC actual en formato 'HH:MM:SS' o HHMMSS, dependiendo del contexto de cadena de caracteres o numérico.

*mysql> SELECT UTC\_TIME(), UTC\_TIME() + 0;*

*-> '18:07:53', 180753*

**UTC\_TIMESTAMP()**

Devuelve la fecha-hora UTC actual en formato 'AAAA-MM-DD HH:MM:SS' o AAAAMMDDHHMMSS, dependiendo del contexto de cadena de caracteres o numérico.

*mysql> SELECT UTC\_TIMESTAMP(), UTC\_TIMESTAMP() + 0;*

*-> '2004-11-05 18:08:04', 20041105180804*

**WEEK(*fecha* [,*modo*])**

Devuelve la semana correspondiente a fecha. El parámetro modo permite especificar en que día de la semana comienza la semana y a partir de que semana se empiezan a numerar. Si no se especifica modo se toma como modo por defecto el de valor 0.

|  |  |
| --- | --- |
| **Valor** | **Significado** |
| 0 | Comienza en Domingo. Se numeran de 0 a 53. La primera que comienza en domingo es la 1. |
| 1 | Comienza en Lunes. Se numeran de 0 a 53. La primera que tenga más de 3 días es la 1. |
| 2 | Comienza en Domingo. Se numeran de 1 a 53. La primera que comienza en domingo es la 1. |
| 3 | Comienza en Lunes. Se numeran de 1 a 53. La primera que tenga más de 3 días es la 1. |
| 4 | Comienza en Domingo. Se numeran de 0 a 53. La primera que tenga más de 3 días es la 1. |
| 5 | Comienza en Lunes. Se numeran de 0 a 53. La primera que comienza en lunes es la 1. |
| 6 | Comienza en Domingo. Se numeran de 1 a 53. La primera que tenga más de 3 días es la 1. |
| 7 | Comienza en Lunes. Se numeran de 1 a 53. La primera que comienza en lunes es la 1. |

*mysql> SELECT WEEK('2005-01-01');*

*->0*

*mysql> SELECT WEEK('2005-01-02',0);*

*-> 1*

*mysql> SELECT WEEK('2005-01-04',1);*

*-> 1*

*mysql> SELECT WEEK('2005-01-31',1);*

*-> 5*

**WEEKDAY(*fecha*)**

Devuelve el día numérico de la semana correspondiente a fecha (0 = Lunes, 1 = Martes, ... 6 = Domingo).

*mysql> SELECT WEEKDAY('2005-01-02 22:23:00');*

*-> 6*

*mysql> SELECT WEEKDAY('2005-01-01');*

*-> 5*

**WEEKOFYEAR(*date*)**

Devuelve el número de semana entre 1 y 53. Las semanas comienzan en Domingo. La primera semana que se cuenta es la primera del año que comienza en Domingo. Los días de comienzo del año anteriores al primer Domingo se considerarán que forman parte de la semana 53. Así, dado que el 1 de Enero de 2005 es Sábado:

*mysql> SELECT WEEKOFYEAR('2005-01-01');*

*-> 53*

*mysql> SELECT WEEKOFYEAR('2005-01-02');*

*-> 1*

**YEAR(*fecha*)**

Devuelve el año correspondiente a la fecha dada con cuatro cifras.

*mysql> SELECT YEAR('98-02-03');*

*-> 1998*

**YEARWEEK(*fecha*)**

**YEARWEEK(*fecha*,*start*)**

Devuelven el año y la semana correspondientes a una fecha. El parámetro *start* es el mismo de la función WEEK() para indicar en que día se inician las semanas y como se numeran éstas.

mysql> SELECT YEARWEEK('2005-01-01');

-> 200452

***4.- Funciones de control de flujo***

**CASE *valor* WHEN [*valor1*] THEN *resultado1* [WHEN [*valor2*] THEN *resultado2* ...] [ELSE *resultado*] END**

**CASE WHEN [*condicion1*] THEN *resultado1* [WHEN [*condicion2*] THEN *resultado2* ...] [ELSE *resultado*] END**

La primera sintaxis devuelve el resultado correspondiente al primer valorN que coincida con *valor*. Si ningún valorN coincide con *valor* se devuelve el resultado que hay tras la cláusula ELSE, y si no tuviera esta cláusula se devuelve NULL. En la segunda sintaxis se devuelve el resultado correspondiente a la primera condición que se cumpla.

*mysql> SELECT CASE 1 WHEN 1 THEN 'one' WHEN 2 THEN 'two' ELSE 'more' END;*

*-> 'one'*

*mysql> SELECT CASE WHEN 1>0 THEN 'true' ELSE 'false' END;*

*-> 'true'*

*mysql> SELECT CASE BINARY 'B' WHEN 'a' THEN 1 WHEN 'b' THEN 2 END;*

*-> NULL*

*mysql>SELECT CASE weekday('2005-1-1') WHEN 1 THEN martes WHEN 2 THEN 'miercoles' WHEN 3 THEN 'jueves' WHEN 4 THEN 'viernes' WHEN 5 THEN 'sabado' when 6 then 'domingo' ELSE 'lunes' END;*

*-> 'sabado'*

**IF(*expr1*,*expr2*,*expr3*)**

Si *expr1* es verdadera (*expr1* <> 0 and *expr1* <> NULL), devuelve *expr2*, si no devuelve *expr3*. IF puede devolver expresiones tanto numéricas como alfanuméricas según se empleen en *expr2* y *expr3.* Si expr1 es numérica, es evaluada como entera por lo que una expresión que sea numérica con decimales se convierte a entera para ser analizada.

*mysql> SELECT IF(1>2,2,3);*

*-> 3*

*mysql> SELECT IF(1<2,'yes','no');*

*-> 'yes'*

*mysql> SELECT IF(STRCMP('test','test1'),'no','yes');*

*-> 'no'*

*mysql> SELECT IF(0.1,1,0);*

*-> 0*

*mysql> SELECT IF(0.1<>0,1,0);*

*-> 1*

**IFNULL(*expr1*,*expr2*)**

Si *expr1* no es NULL, devuelve *expr1*, si no devuelve *expr2*.

mysql> SELECT IFNULL(1,0);

-> 1

mysql> SELECT IFNULL(NULL,10);

-> 10

mysql> SELECT IFNULL(1/0,10);

-> 10

mysql> SELECT IFNULL(1/0,'yes');

-> 'yes'

**NULLIF(*expr1*,*expr2*)**

Devuelve NULL si se cumple *expr1* = *expr2*, si no devuelve *expr1*.

mysql> SELECT NULLIF(1,1);

-> NULL

mysql> SELECT NULLIF(1,2);

-> 1

***5.- Funciones de conversión***

**BINARY**

Realmente no es una función sino un operador pero se hace aconsejable mencionarlo aquí por el comportamiento de las funciones que comparan cadenas. El operador BINARY opera sobre cadena de caracteres que se indique a continuación para convertirla a cadena binaria o de bytes. La comparación entre cadenas de binarias diferencia mayúsculas y minúsculas, mientras que no las diferencia una comparación entre cadenas de caracteres no. De esta forma, la principal utilidad de este operador es permitir que se puedan comparar cadenas de caracteres diferenciando mayúsculas y minúsculas. Para ello, basta con convertir una de las cadenas de caracteres a cadena binaria.

*mysql> SELECT 'a' = 'A';*

*-> 1*

*mysql> SELECT BINARY 'a' = 'A';*

*-> 0*

*mysql> SELECT 'a' = 'a ';*

*-> 1*

*mysql> SELECT BINARY 'a' = 'a ';*

*-> 0*

Si queremos hacer lo contrario, es decir, que al comparar cadenas binarias (BLOB) no se diferencie entre mayúsculas y minúsculas, podríamos usar la técnica de convertir la cadena binaria a cadena de caracteres que usa el mismo código que las cadenas de caracteres definidas en MySQL:

SELECT 'A' LIKE CONVERT(*columna\_blob*  USING latin1) FROM *nombre\_tabla;*

**CAST(*valor* AS *tipo*)**

**CONVERT(*valor*, *tipo*)**

**CONVERT(*valor*  USING *transcoding\_name*)**

Estas funciones toman un valor de un tipo y lo devuelven convertido a otro tipo. Los tipos que pueden ser usados en el parámetro *tipo* son:

* BINARY
* CHAR
* DATE
* DATETIME
* SIGNED [INTEGER]
* TIME
* UNSIGNED [INTEGER]

La sintaxis que usa USING sirve para convertir datos entre diferentes conjuntos de caracteres (tablas de codificación).

*SELECT CONVERT('abc' USING utf8);*

Los conjuntos de caracteres que se pueden usar, es posible obtenerlos con el comando SHOW CHARACTER SET. Algunos de esos conjuntos son *latin1, latin2, ASCII, utf8.*

Un ejemplo de uso de estas funciones es para ordenar una consulat sobre una columna ENUM de forma alfabética (por defecto la ordena numérica, es decir, por la posición que ocupa cada dato de la enumeración.

*SELECT enum\_col FROM tabla ORDER BY CAST(enum\_col AS CHAR);*

Para convertir una cadena de caracteres en valor numérico, generalmente no es necesario hecer una conversión con CASTo CONVERT ya que la puede realizar automáticamente MySQL si el contexto lo inplica.

*mysql> SELECT 1+'1';*

*-> 2*

De igual forma se relaizan algunas conversiones de forma automática de numérico a cadena de caracteres, como se muestra en el siguiente ejemplo.

*mysql> SELECT CONCAT("hello you ",2);*

*-> "hello you 2"*

Si se hace una operación entre dos valores sin signo (UNSIGNED), el resultado es sin signo. Para forzar que el resultado con signo, habría que convertir uno de los operandos a numérico con signo (SIGNED). En los siguientes ejemplos se muestra un ejemplo de resta entre dos números sin signo que da error y como se soluciona ese error.

*mysql> SELECT 1- (2 AS UNSIGNED);*

*-> 18446744073709551615*

*mysql> SELECT CAST(1-CAST(2 AS UNSIGNED)) AS SIGNED);*

*-> -1*

***6.- Funciones de búsqueda sobre índices FULLTEXT***

Los índices **FULLTEST** pueden definirse en tablas MyISAM y sólo para columnas de tipo CHAR, VARCHAR o TEXT. En los ejemplos propuestos se usa la tabla ARTICLES n la que se define un índice FULLTEST.

**CREATE TABLE articles (**

**id INT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,**

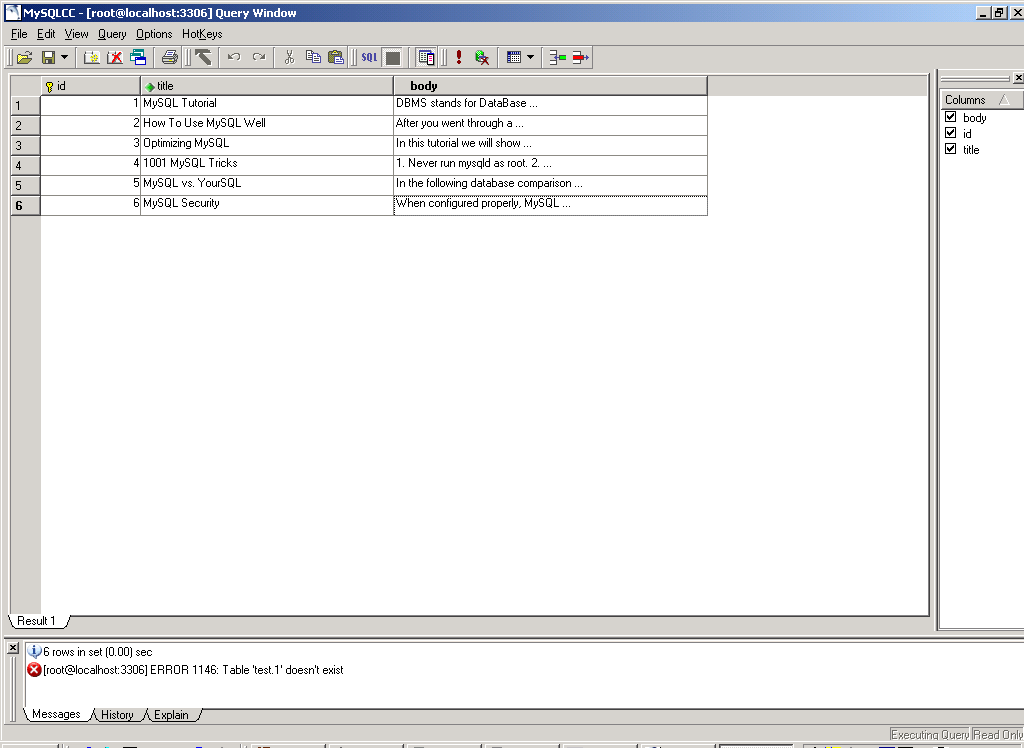
**title VARCHAR(200),**

**body TEXT,**

**FULLTEXT (title,body)**

**);**

Se supone que la tabla contiene los datos:



**MATCH (*col1*,*col2*,...) AGAINST (*expr* [IN BOOLEAN MODE ])**

La función permite buscar la palabras de *expr* dentro de las columnas de tipo índice FULLTEXT que se indiquen. Las palabras de *expr* deben estar separadas por espacios y nunca pueden ser variables o referirse a nombres de columnas. Por cada fila devuelta al aplicar la función MATCH, se devuelve la relevancia de las palabras buscadas en las columnas donde se busca. La relevancia es 0 cuando ninguna de las palabras se encuentran en las columnas en que se buscan. Si se encuentra la palabra, la relevancia se obtiene en función del número de palabras encontradas en cada fila, el número de palabras de cada fila no repetidas, el número de filas que contienen las palabras buscadas, el número total de palabras encontradas. No se pueden buscar palabras de 3 o menos letras. Por defecto no se diferencian mayúsculas y minúsculas.

El siguiente ejemplo, busca la palabra *Tutorial* dentro de las columnas índice FULLTEST de todas las filas de la tabla ARTICLES. En el resultado se puede observar que dicha palabra es relevante (aprece) en las filas de id 1 y 3.

*mysql> SELECT id, MATCH (title,body) AGAINST ('Tutorial') FROM articles;*

*+----+-----------------------------------------+*

*| id | MATCH (title,body) AGAINST ('Tutorial') |*

*+----+-----------------------------------------+*

*| 1 | 0.65545833110809 |*

*| 2 | 0 |*

*| 3 | 0.66266459226608 |*

*| 4 | 0 |*

*| 5 | 0 |*

*| 6 | 0 |*

*+----+-----------------------------------------+*

En lugar de una sola palabra, en expr se puede indicar un conjunto de palabras como se va a realizar en el siguiente ejemplo. En este ejemplo se muestran sólo las filas con relevancia. En la fila 4 aparecen dos palabras por lo que tiene mayor relevancia.

*mysql> SELECT id, body, MATCH (title,body) AGAINST ('Security implications of running MySQL as root') AS score FROM articles WHERE MATCH (title,body) AGAINST ('Security implications of running MySQL as root');*

*+----+--------------------------------------------+----------------- +*

*| id | body | score |*

*+----+--------------------------------------------+---------------- -+*

*| 4 | 1. Never run mysqld as root. 2. ... | 1.5219271183014 |*

*| 6 | When configured properly, MySQL ... | 1.3114095926285 |*

*+----+--------------------------------------------+----------------- +*

Hay palabras llamadas "palabras ruido" y que no se buscan. Se trata de palabras muy comunes en inglés y suelen ser artículos, preposiciones como, por ejemplo, some, never, after.

Si se hace una búsqueda de una palabra que aparece en todas las filas, se indica que tienen relevancia 0 todas las filas.

*mysql> SELECT \* FROM articles WHERE MATCH (title,body) AGAINST ('MySQL');*

*Empty set (0.00 sec)*

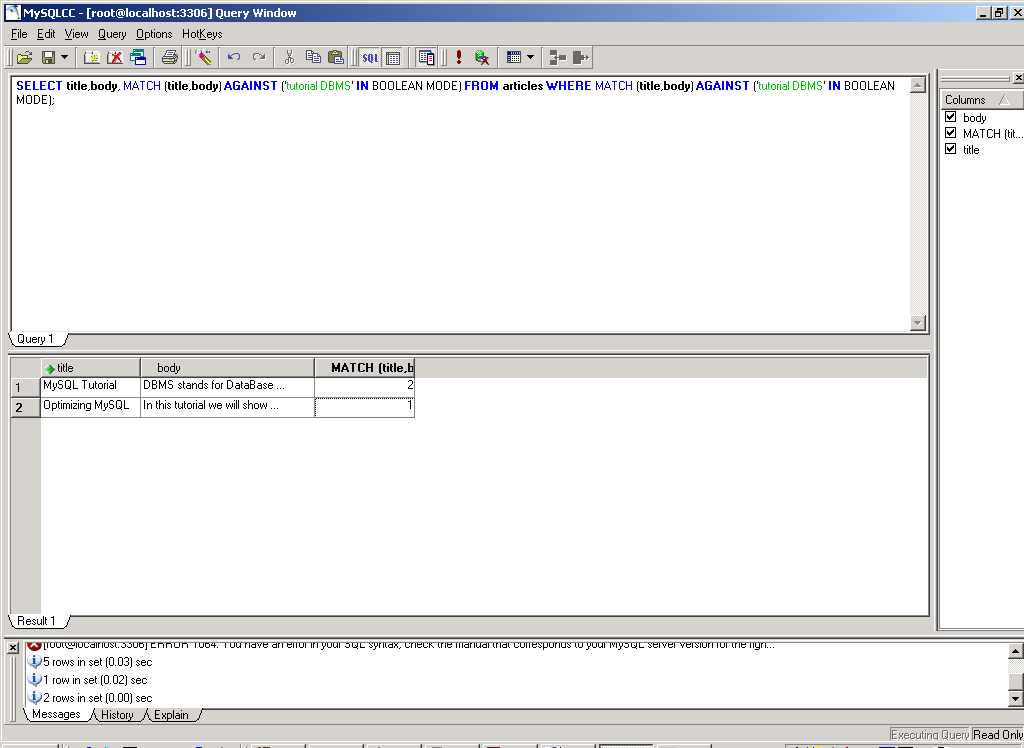
Las palabras que en inglés son poco comunes reciben una mayor ponderación o valoración que otras más comunes que se tienen guardadas en un almacén de palabras en el servidor.

**Búsquedas booleanas de texto completo:**

Con la función MATCH también se pueden buscar combinaciones de palabras, porciones de palabras y otras variantes a través de búsquedas booleanas añadiendo la cláusula IN BOOLEAN MODE y usando una serie de operadores que operan con las palabras a buscar. :

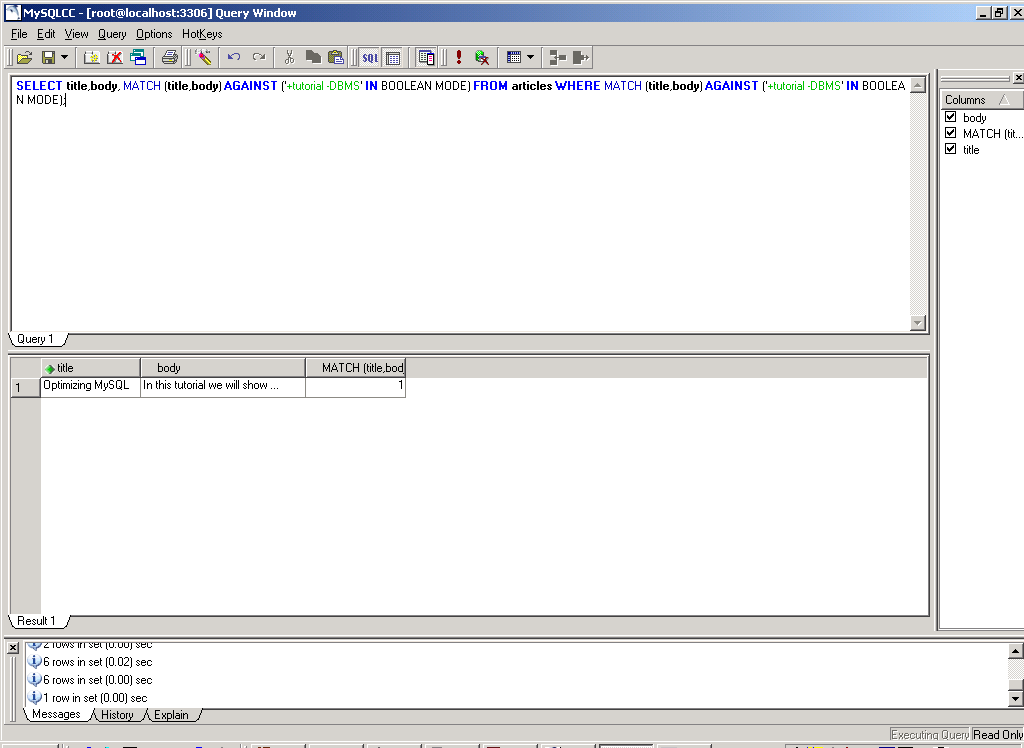
|  |  |
| --- | --- |
| **Operador** | **Descripción** |
| + | La palabra que siga al operador es obligatorio que aparezca en todas las filas |
| - | La palabra no debe aparecer en ninguna de las filas devueltas |
| < | La palabra siguiente se considerará menos relevante que las otras |
| > | La palabra siguiente se considerará más relevante que las otras |
| ( ) | Para agrupar palabras en subexpresiones |
| ~ | La palabra siguiente da contribución negativa a la relevancia |
| \* | Comodín para cualquier cadena |
| " | Entre comillas dobles se representa un conjunto |

*SELECT title,body, MATCH (title,body) AGAINST ('tutorial DBMS' IN BOOLEAN MODE) FROM articles WHERE MATCH (title,body) AGAINST ('tutorial DBMS' IN BOOLEAN MODE);*

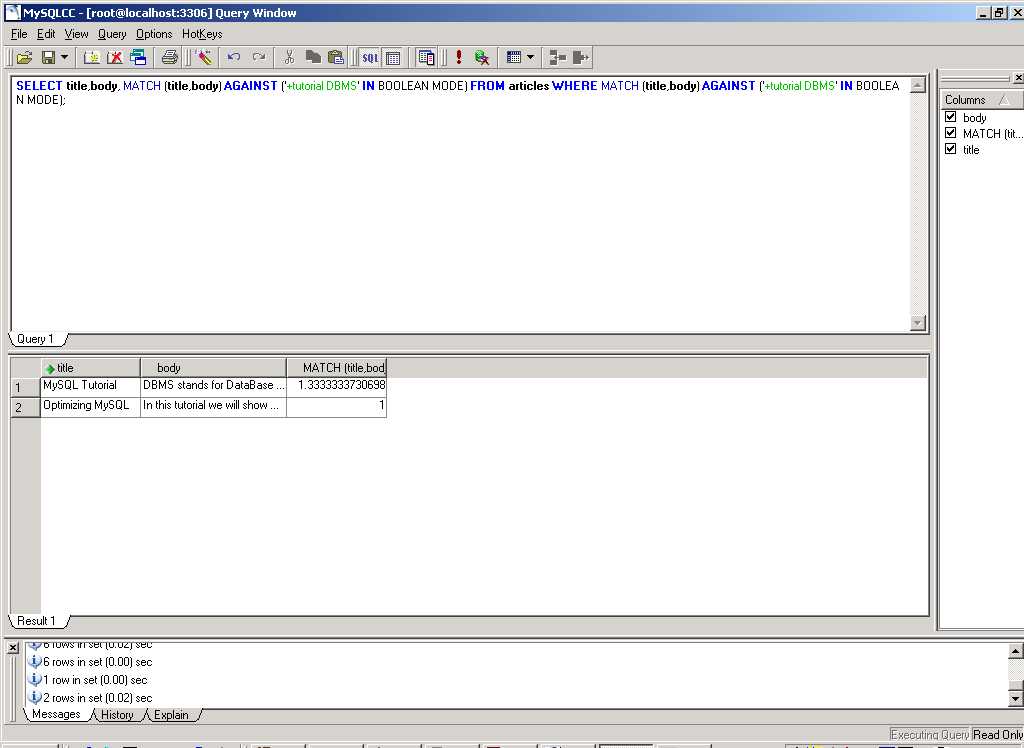


.

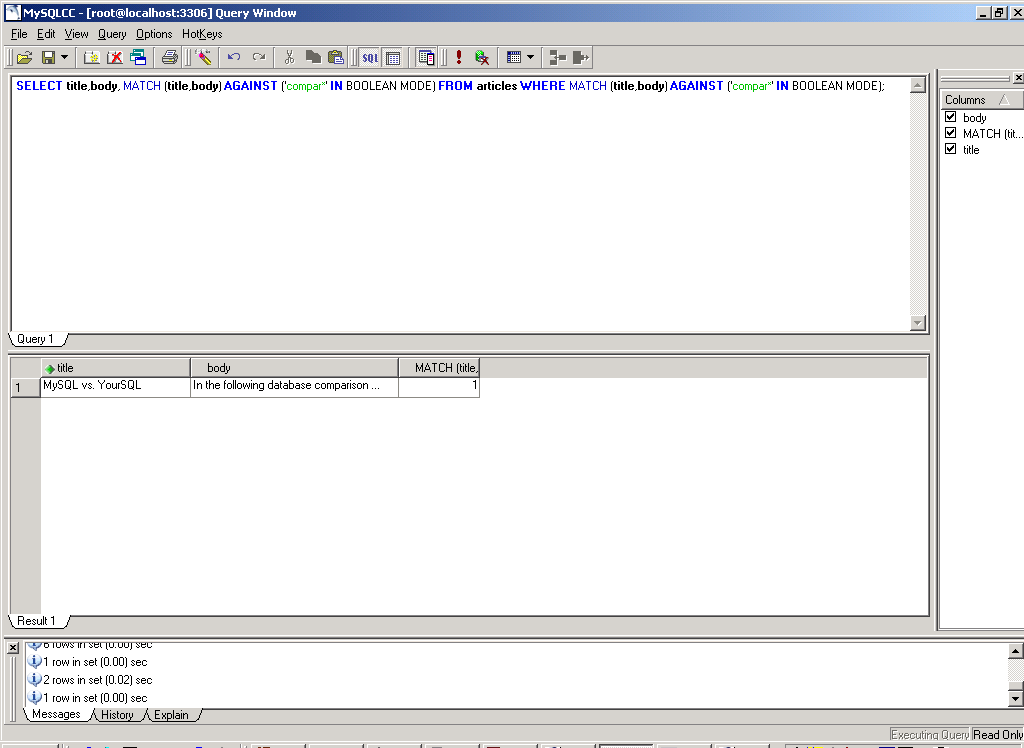
*SELECT title,body, MATCH (title,body) AGAINST ('+tutorial -DBMS' IN BOOLEAN MODE) FROM articles WHERE MATCH (title,body) AGAINST ('+tutorial -DBMS' IN BOOLEAN MODE);*



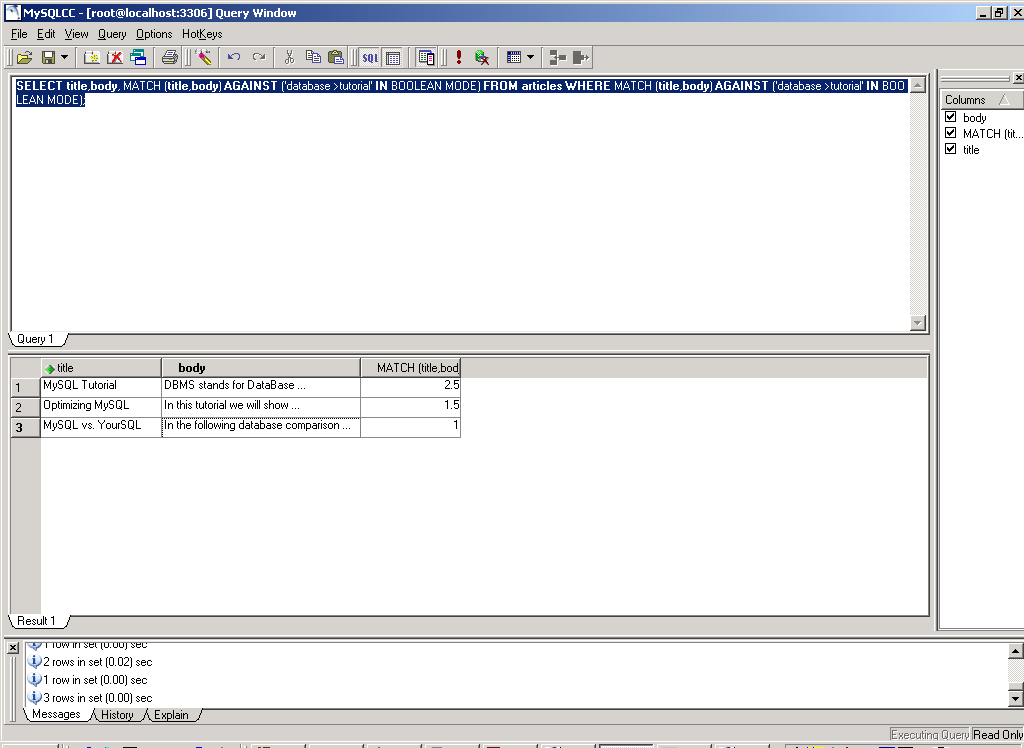
*SELECT title,body, MATCH (title,body) AGAINST ('+tutorial DBMS' IN BOOLEAN MODE) FROM articles WHERE MATCH (title,body) AGAINST ('+tutorial DBMS' IN BOOLEAN MODE);*



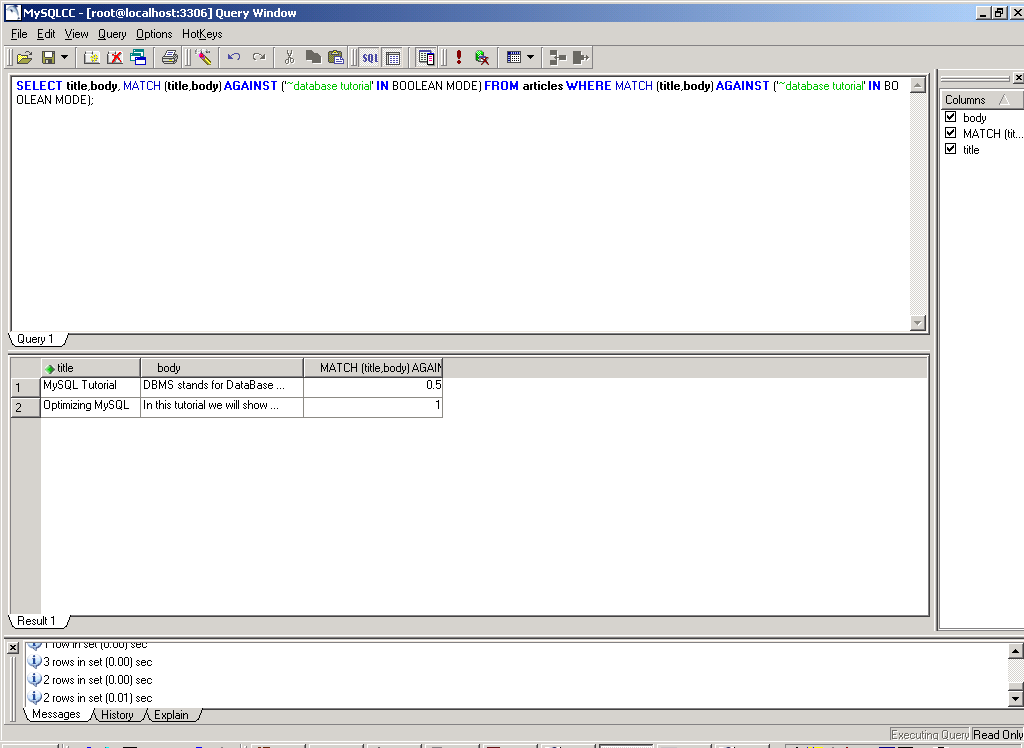
*SELECT title,body, MATCH (title,body) AGAINST ('compar\*' IN BOOLEAN MODE) FROM articles WHERE MATCH (title,body) AGAINST ('compar\*' IN BOOLEAN MODE);*



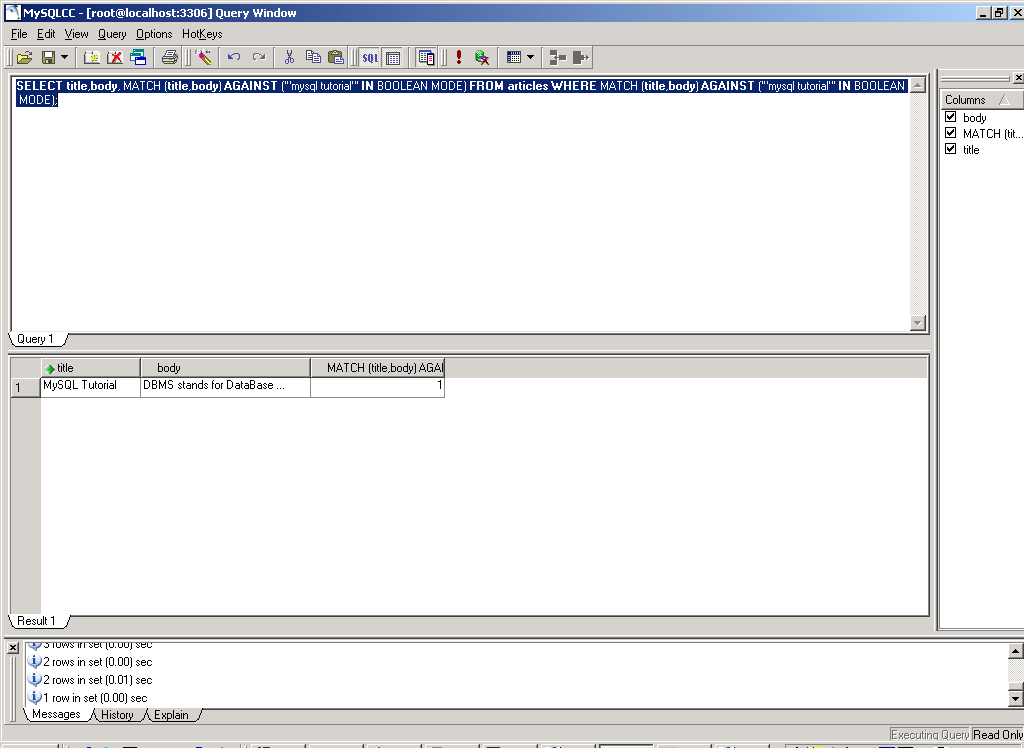
*SELECT title,body, MATCH (title,body) AGAINST ('database >tutorial' IN BOOLEAN MODE) FROM articles WHERE MATCH (title,body) AGAINST ('database >tutorial' IN BOOLEAN MODE);*



*SELECT title,body, MATCH (title,body) AGAINST ('~database tutorial' IN BOOLEAN MODE) FROM articles WHERE MATCH (title,body) AGAINST ('~database tutorial' IN BOOLEAN MODE);*



*SELECT title,body, MATCH (title,body) AGAINST ('"mysql tutorial"' IN BOOLEAN MODE) FROM articles WHERE MATCH (title,body) AGAINST ('"mysql tutorial"' IN BOOLEAN MODE);*



***7.- Funciones de agregado***

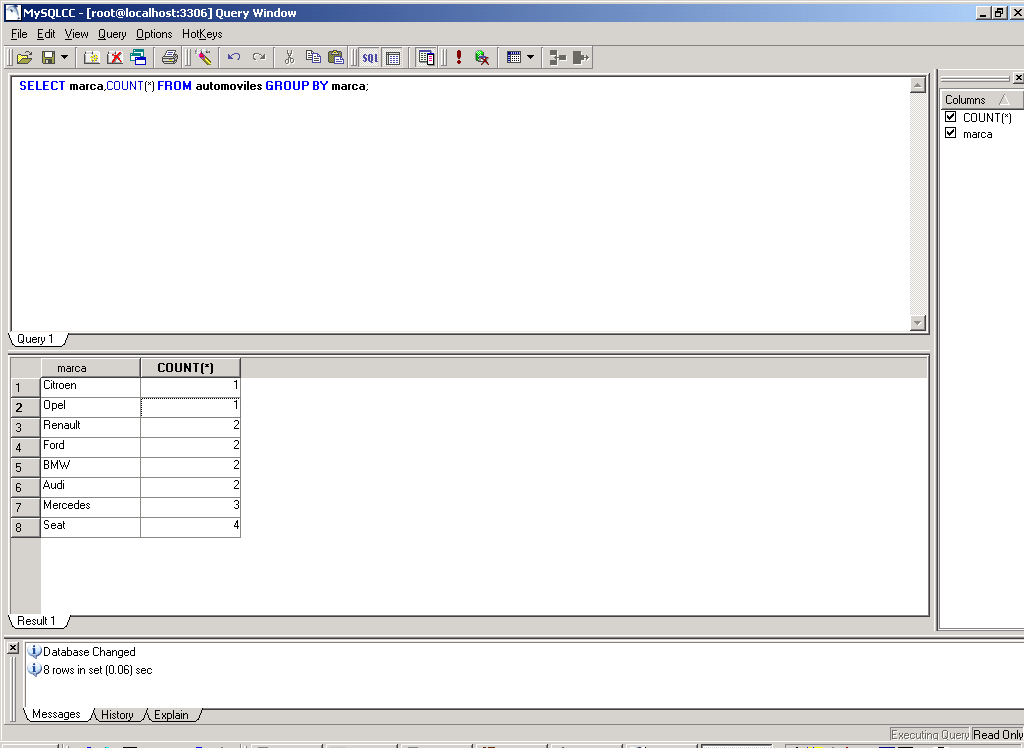
Las funciones de agregado o de grupo sirven para obtener cálculos sobre todas las filas de una tabla o sobre cada conjunto de filas que tiene valores iguales en una o más columnas (cláusula GROUP BY de una SELECT). Algunas de estas funciones, las principales, ya las hemos visto al estudiar las consultas de grupo o de resumen. Todas las funciones de agregado disponibles son:

**COUNT(expr)**

Devuelve el número de filas no nulas devueltas en una consulta. Generalmente se usa para contar todas las filas de una tabla o de cada grupo y para ello, se usa \* en la expresión ya que no puede haber ninguna fila que tenga todos sus valores nulos. Ejemplos:

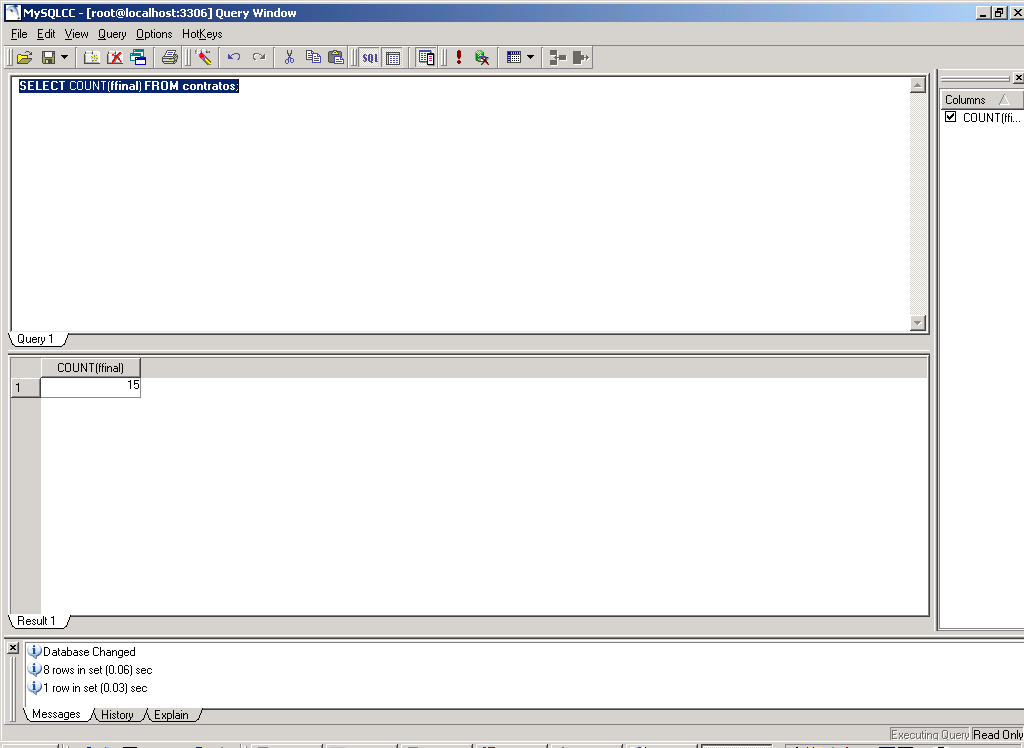
Obtener el número de coches de cada marca de la tabla automoviles.

*SELECT marca,COUNT(\*) FROM automoviles GROUP BY marca;*



Obtener el número de contratos finalizados (columna ffinal no es NULL).

*SELECT COUNT(ffinal) FROM contratos;*



**COUNT(DISTINCT expr,[expr...])**

Devuelve la acntidad de filas con valores no nulos diferentes. Por ejemplo, para saber cuantas marcas distintas de automoviles hay:

*mysql> SELECT COUNT(DISTINCT marcas) FROM automoviles;*

**AVG(expr)**

Devuelve el valor medio de la expresión indicada para todas las filas devueltas o para los grupos definidos con GROUP BY. Los valores NULL no los usa para el cálculo de la media.

Por ejemplo, para obtener el precio medio de los coches de cada marca:

mysql> SELECT marca, AVGprecio) FROM automoviles GROUP BY marca;

**MIN(expr)**

**MAX(expr)**

Devuelven el valor mínimo o máximo de *expr* de todas las filas devueltas en la consulta o de las filas correspondientes a cada grupo definido con GROUP BY.

**SUM(expr)**

Devuelve la suma de expr.

**GROUP\_CONCAT(expr)**

La sintaxis completa es:

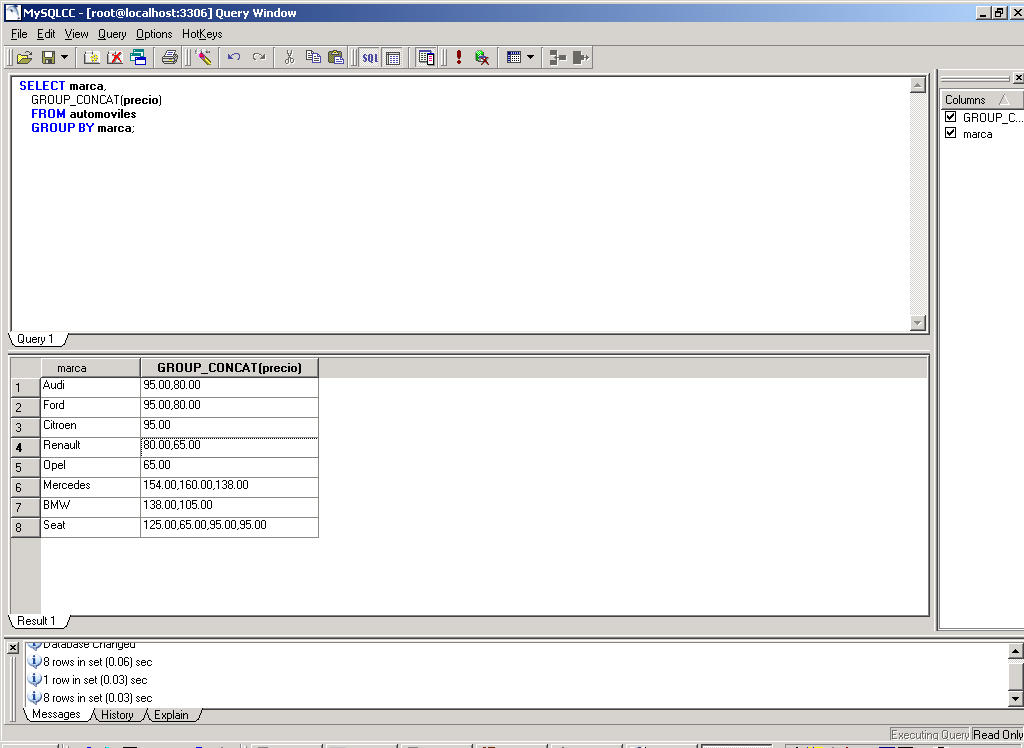
GROUP\_CONCAT([DISTINCT] expr [,expr ...]

[ORDER BY col1 [ASC | DESC] [,col2 ...]]

[SEPARATOR valor\_str])

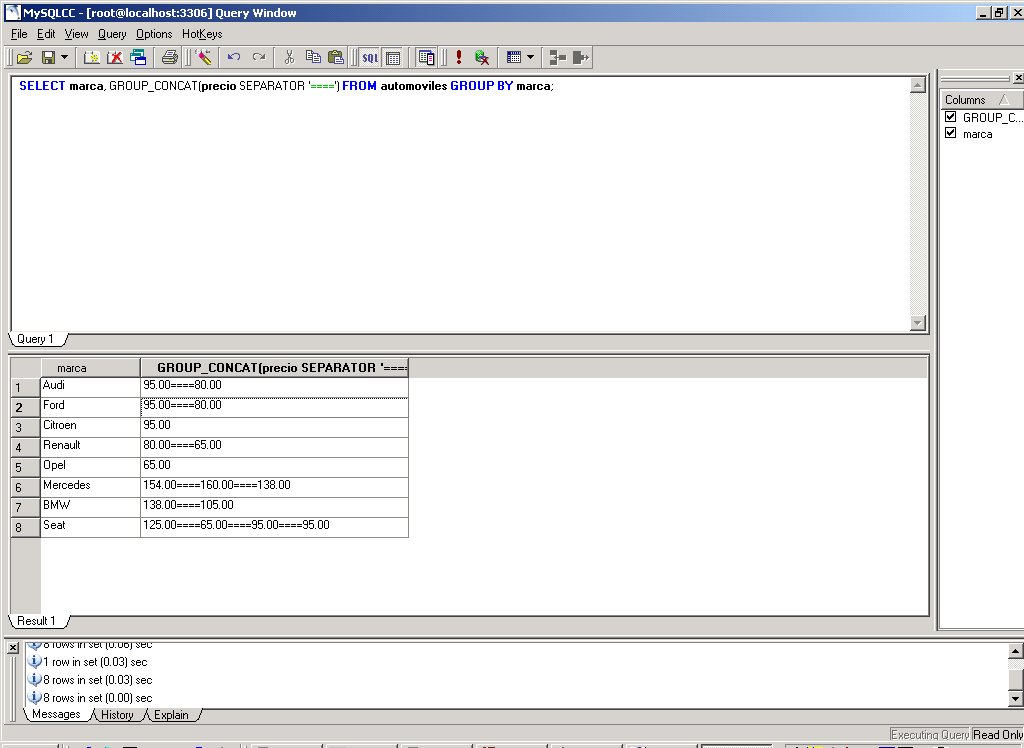
Permite obtener una lista con todos los valores de las expresiones *expr* para todas las filas devueltas o para cada grupo definido con GROUP BY. Para separar los valores se usa el carácter coma o los carácteres que se definan tras SEPARATOR. Por ejemplo, para obtener una lista con todos los precios de alquiler de los coches de cada marca.

SELECT marca, GROUP\_CONCAT(precio) FROM automoviles GROUP BY marca;



Y usando como separador "=====":

SELECT marca, GROUP\_CONCAT(precio SEPARATOR '====') FROM automoviles GROUP BY marca;



**VARIANCE(expr)**

Devuelve la varianza de expr.

**STD(expr)**

**STDDEV(expr)**

Devuelven la desviación estándar de expr.

**BIT\_OR(expr)**

Devuelve el resultado de realizar la operación binaria OR con todas las filas de cada grupo sobre los valores de expr.

**BIT\_XOR(expr)**

Devuelve el resultado de realizar la operación binaria XOR con todas las filas de cada grupo sobre los valores de expr.

**BIT\_AND(expr)**

Devuelve el resultado de realizar la operación binaria AND con todas las filas de cada grupo sobre los valores de expr.

***8.- Otras funciones***

**BIT\_COUNT(*N*)**

Devuelve el número de bits de valor 1 que tiene la representación binaria de N.

mysql> SELECT BIT\_COUNT(29);

-> 4

**AES\_ENCRYPT(*str*,*key\_str*)**

**AES\_DECRYPT(*crypt\_str*,*key\_str*)**

Permiten encriptar y desencriptar información usando una clave de encriptación. Utilizan la técnica AES.

Ejemplo:

INSERT INTO t VALUES (1,AES\_ENCRYPT('text','password'));

El ejemplo anterior ha encriptado la cadena 'text' usando la clave 'password' y ha insertado el valor encriptado en una fila de la tabla t. Para que en esa fila haya la cadena 'text' habrá que desencriptar al consultar la fila usando la misma clave. Un buen método para hacer encriptados y desencriptados es usar la clave que nos permite acceder al cliente MySQL. Para ello podemos crear una variable que contiene esa clave y, después, usar esa variable como clave en todos los encriptados. De esta forma evitamos errores provocados por insertar una fila con una clave en la que nos hemos equivocado en algún carácter, lo que haría imposible recuperar de forma correcta esa información.

SELECT @p:='mi contraseña';

INSERT INTO t VALUES (1,AES\_ENCRYPT('text',@p));

**DECODE(*crypt\_str*,*pass\_str*)**

**ENCODE(*str*,*pass\_str*)**

Desencriptan y encriptan una cadena con un método muy básico. El resultado es una cadena binaria siempre.

.

**DES\_ENCRYPT(*str*[,(*key\_num*|*key\_str*)])**

**DES\_DECRYPT(*crypt\_str*[,*key\_str*])**

Encriptan y desencriptan cadenas usando el método Triple-DES.

**BENCHMARK(*N*, *expr*)**

Ejecuta la expresión *expr* tantas veces como indique N. Devuelve siempre 0 y sirve para poder comprobar cuanto se tarda en ejecutar una función Dado que ejecutando una vez cada función, el tiempo obtenido será cero, podremos forzar a que la sentencia se ejecute muchas veces. En el resultado de la consulta se indicará el tiempo que ha tardado.

mysql> SELECT BENCHMARK(1000000,ENCODE('hello','goodbye'));

+----------------------------------------------+

| BENCHMARK(1000000,ENCODE('hello','goodbye')) |

+----------------------------------------------+

| 0 |

+----------------------------------------------+

1 row in set (4.74 sec)

**CHARSET(*str*)**

Devuelve el conjunto de caracteres o alfabeto al que pertenece *expr.*

*mysql> SELECT CHARSET(nombre) FROM clientes;*

*+-----------------+*

*| CHARSET(nombre) |*

*+-----------------+*

*| latin1 |*

*| latin1 |*

*| latin1 |*

*| latin1 |*

*| latin1 |*

**COLLATION(*str*)**

Devuleve la variante del conjunto de caracteres utilizado.

*mysql> SELECT COLLATION('abc');*

*-> 'latin1\_swedish\_ci'*

**CONNECTION\_ID()**

Devuelve el número de identificador de la conexión cliente MySQL al servidor MySQL. Cada conexión tiene un número único contado desde uno y desde que se inicio el servidor (normalmente al arrancar el ordenador).

.

*mysql> SELECT CONNECTION\_ID();*

* + *23*

**CURRENT\_USER()**

Devuelve el nombre del usuario y del equipo donde éste ha sido autenticado.

*mysql> SELECT CURRENT\_USER();*

*-> 'luis@localhost'*

**DATABASE()**

Devuelve el nombre de la base de datos abierta actualmente.

mysql> SELECT DATABASE();

-> 'alquileres'

**FOUND\_ROWS()**

Permite conocer cuantas filas se podrían haber devuelto en una SELECT ejecutada anteriormente con LIMIT para limitar el número de filas. En esa sentencia anterior es necesario especificar una opción para que después se pueda usar la función FOUND\_ROWS, esa opción es SQL\_CALC\_FOUND\_ROWS.

*mysql> SELECT SQL\_CALC\_FOUND\_ROWS \* FROM*  clientes

*WHERE localidad='Madrid' LIMIT 5;*

*mysql> SELECT FOUND\_ROWS();*

*+--------------+*

*| FOUND\_ROWS() |*

*+--------------+*

*| 12 |*

*+--------------+*

**LAST\_INSERT\_ID()**

Devuelven el último valor insertado en una columna AUTO\_INCREMENT.

mysql> SELECT LAST\_INSERT\_ID();

-> 195

Devuelve el valor que afecta a la última tabla en que se ha insertado un valor AUTO\_INCREMENT. Si en una misma INSERT se insertan varias filas y se ejecuta esta función, devolverá el valor correspondiente a la primera de las filas insertadas en grupo con INSERT (este es un comportamiento defectuoso).

**ROW\_COUNT()**

Devuelve el número de filas que se vieron afectadas por la operación precedente de borrado, inserción o modificación. Si la operación precedente no fue ninguna de las anteriores, la función devuelve –1.

*mysql> INSERT INTO t VALUES(1),(2),(3);*

*Query OK, 3 rows affected (0.00 sec)*

*Records: 3 Duplicates: 0 Warnings: 0*

*mysql> SELECT ROW\_COUNT();*

*+--------------------+*

*| ROW\_COUNT() |*

*+--------------------+*

*| 3 |*

*+--------------------+*

*1 row in set (0.00 sec)*

**USER()**

**SESSION\_USER()**

**SYSTEM\_USER()**

Devuelve el nombre del usuario de la sesión cliente y el nombre del equipo en que se está ejecutando dicha sesión cliente.

*mysql> SELECT USER();*

*-> 'luis@localhost'*

El carácter @ separa el nombre del usuario del nombre del equipo. El nombre del equipo es localhost que quiere decir, lógicamente, que el cliente se está ejecutando en el equipo en el que estamos haciendo esta consulta. El nombre del equipo no tiene por que ser el mismo que el devuelto por la función CURRENT\_USER() ya que ésta indica el nombre del equipo donde se ejecuta el servidor. Se puede extraer el nombre del usuario de lo devuelto por USER() utilizando la función SUBSTRING ya vista anteriormente.

*mysql> SELECT SUBSTRING\_INDEX(USER(),'@',1);*

*-> 'luis'*

**VERSION()**

Devuelve la versión del servidor MySQL..

*mysql> SELECT VERSION();*

*-> ' 5.0.1-alpha-nt'*

**DEFAULT(*columna*)**

Devuelve el valor por defecto de una columna que tiene la restricción DEFAULT.

*Mysql> SELECT DISTINCT DEFAULT(nota1) FROM cursadas;*

*-> 5*

*mysql> UPDATE cursadas SET nota1 = DEFAULT(nota1)+1 WHERE numal < 10;*

**FORMAT(*X*,*D*)**

Da un formato de visualización a un número X en la forma '#,###,###.##', redondeado a D decimales. El resultado devuelto es una cadena de caracteres.

*mysql> SELECT FORMAT(12332.123456, 4);*

*-> '12,332.1235'*

*mysql> SELECT FORMAT(12332.1,4);*

*-> '12,332.1000'*

*mysql> SELECT FORMAT(12332.2,0);*

*-> '12,332'*

.

**UUID()**

Genera aleatoriamente un Identificador Universal Único (UUID) de acuerdo al protocolo DCE 1.1. Un identificador UUID es un número de 128 bits representado con dígitos hexadecimales (32 dígitos) y en forma de cinco campos separados por el carácter menos.

* Los tres primeros campos se generan a partir de la fecha y hora actual.
* El cuarto campo se genera a partir de un contador que previene que pudiera producirse justo en la misma fecha-hora la generación de dos UUID simultáneos para el mismo equipo.
* El quinto campo es un número de nodo IEEE 802 obtenido de la tarjeta de red.

*mysql> SELECT UUID();*

*-> '6ccd780c-baba-1026-9564-0040f4311e29'*