2.4.6 Funciones incorporadas

En los programas almacenados pueden seguirse utilizando la mayoría de las funciones incluidas en MySQL y que se utilizan para formar las sentencias SQL, excepto las que trabajan con grupos de datos (cláusula GROUP BY) puesto que las variables en los programas almacenados son escalares y almacenan un solo valor. Por eso funciones como SUM, COUNT, MIN, MAX y AVG pueden emplearse en programas almacenados siempre y cuando devuelvan una fila y no varias (como consecuencia p.ej. en este último caso de utilizar la cláusula GROUP BY).

A continuación se detallarán las más importantes. Para más información consultar el manual.

FUNCIONES MATEMÁTICAS

FUNCIÓN	DEVUELVE	EJEMPLO Y RESULTADO
ABS(num)	Valor absoluto de num	SELECT ABS(-3) → 3
SIGN(num)	-1, 0 o 1 en función del valor de num	SELECT SIGN(2), SIGN(-2), SIGN(0) → 1, -1, 0
MOD(num1, num2)	Resto de la división de num1 por num2	SELECT MOD (5,2) → 1
FLOOR(num)	Mayor valor entero inferior a num	SELECT FLOOR(23.9) → 23
CEILING(num)	Menor valor entero superior a num	SELECT CEILING(23.9) → 24
ROUND(num)	Redondeo entero más próximo	SELECT ROUND(23.5), ROUND(23.4); → 24 23
ROUND(num,d)	Redondeo a d decimales más próximo	SELECT ROUND(23.545,2), ROUND(23.44,1) → 23,55 23,4
TRUNCATE (num, d)	Num truncado a d decimales	SELECT TRUNCATE (22.89, 1), TRUNCATE (15326,-3) → 22,8 5000
POW(num1, num2)	Num1 elevado a la num2 potencia	SELECT POW(2,5) → 32
SQRT (num)	Raíz cuadrada de num	SELECT SQRT(36) → 6

FUNCIONES DE CADENA

FUNCIÓN	DEVUELVE	EJEMPLO Y RESULTADO
LIKE(plantilla)	Resultado de comparar una cadena con una plantilla	SELECT 'ALBERTO' LIKE 'ALBER%' → 1 (cierto)
NOT LIKE (plantilla)	Lo contrario a la fila anterior	SELECT 'ALBERTO' NOT LIKE 'ABIERTO' → 1
_ (subrayado)	Se trata de un comodín que reemplaza un carácter en una cadena	SELECT 'ALBERTO' LIKE 'ALBERT_' → 1
%	Como el caso anterior pero para uno o más caracteres	SELECT 'ALBERTO' LIKE 'ALBER%' → 1 (cierto)
\	Como en otros lenguajes se trata del carácter de escape, si precede al comodín elimina su función y lo trata como un carácter más	SELECT '30%' LIKE '30\%' → 1

FUNCIONES DE CADENA (*Continuación)*		
BINARY	Por defecto en las comparaciones entre cadenas no se distingue mayúsculas de minúsculas salvo que se indique esta opción	SELECT 'ALBERTO' LIKE BINARY 'Alberto' → 0 (falso)
STRCMP(cad1, cad2)	-1 si cad1 < cad2, 0 si cad1=cad2 o 1 si cad1 > cad2	SELECT STRCMP('ALBERTO', 'ABIERTO') → 1
UPPER(cad)	La cadena cad en mayúsculas	SELECT UPPER('Alberto') → ´ALBERTO’
LOWER(cad)	La cadena cad en minúsculas	SELECT LOWER('Alberto') → ‘alberto’

FUNCIONES DE FECHA		
FUNCIÓN	**DEVUELVE**	**EJEMPLO Y RESULTADO**
NOW()	Fecha y hora según el formato ‘aaaa-mm-dd hh:mm:ss’	SELECT NOW() → 2006-08-01 00:40:25
DAYOFWEEK(fecha)	Cifra que representa el día de la semana (1 – domingo, 2 –lunes…)	SELECT DAYOFWEEK('1966-11-03') → 5 |
WEEKDAY(fecha)	Ídem de DAYOFWEEK pero con otros valores: 0 – lunes, 1 – martes…	SELECT WEEKDAY('1966-11-03') → 3
DAYOFMONTH(fecha)	Día del mes (entre 1 y 31)	SELECT DAYOFMONTH('1966-11-03') → 3
DAYOFYEAR(fecha)	Día del año (entre 1 y 366)	SELECT DAYOFYEAR('1966-11-03') → 307
MONTH(fecha);	Mes del año (entre 1 y 12)	SELECT MONTH('1966-11-03') → 11
DAYNAME(fecha)	Nombre del día de la fecha	SELECT DAYNAME('1966-11-03') → ‘Thursday’
MONTHNAME(fecha)	Nombre del mes	SELECT MONTHNAME('1966-11-03') → ‘November’
QUARTER(fecha)	Trimestre del año (entre 1 y 4)	SELECT QUARTER('1966-11-03') → 4
WEEK(fecha [,inicio])	Semana del año (entre 1 y 52). Inicio especifica el comienzo de la semana. Si no se especifica vale 0 (domingo). Para empezar el lunes utilizar el 1	SELECT WEEK('2006-12-20',1) → 51
YEAR(fecha)	Año (entre 1000 y 9999)	SELECT YEAR('2006-12-20') → 2006
HOUR(fecha)	La hora	SELECT HOUR(NOW()) →1
MINUTE(fecha)	Los minutos	SELECT MINUTE(NOW()) → 5
SECOND(fecha)	Los segundos	SELECT SECOND(NOW()) → 58

FUNCIONES DE FECHA *(Continuación)*		
TO_DAYS(fecha)	Número de días transcurridos desde el año 0 hasta la fecha	SELECT TO_DAYS('2006-08-01') → 732889
DATE_ADD(fecha, INTERVAL valor tipo de intervalo	La fecha sumado el intervalo especificado	SELECT DATE_ADD('2006-08-01', INTERVAL 1 MONTH) → '2006-09-01'
DATEDIFF(fecha1, fecha2)	El número de días transcurridos entre fecha1 y fecha2	SELECT DATEDIFF('2006-08-01', '2006-07-26') → 6
CURDATE() CURRENT_DATE()	Fecha actual según el formato 'aaaa-mm-dd'	SELECT CURRENT_DATE() → '2006-08-01'
CURTIME() CURRENT_TIME()	Fecha actual según el formato 'hh:mm:ss'	SELECT CURRENT_TIME() → '01:12:43'
DATE_FORMAT (fecha, formato)	Devuelve la fecha en el formato especificado. Consultar el manual para las posibilidades de la opción formato.	SELECT DATE_FORMAT(NOW(), 'Hoy es %d de %M de %Y') → 'Hoy es 01 de August de 2006'

FUNCIONES DE CONTROL		
FUNCIÓN	**DESCRIPCIÓN**	**EJEMPLO Y RESULTADO**
IF(expr1, expr2, expr3)	Si la expresión expr1 es cierta, devuelve expr2, sino expr3	SET @A=20; SET @B=15; SELECT IF(@A<@B, @A+@B, @A - @B); → 5
IFNULL(expr1, expr2)	Si la expresión expr1 es NULL devuelve expr2, sino expr1	SET @A=20; SELECT IFNULL(@A, 0); → 20
NULLIF(expr1, expr2)	Si la expresión expr1 es igual a expr2, devuelve NULL sino expr1	SET @A=20; SET @B=15; SELECT NULLIF(@B, @A); → 15
CASE valor WHEN comp1 THEN res1 [WHEN comp2 THEN res2] [ELSE reselse] END	Compara el valor con cada una de las expresiones comp. Si se verifica la igualdad entonces devuelve el valor res asociado, en cualquier otro caso devuelve reselse	SELECT CASE WEEKDAY(NOW()) WHEN 5 THEN 'Fin de semana' WHEN 6 THEN 'Fin de semana' ELSE 'No es fin de semana' END;

FUNCIONES DE AGREGACIÓN		
FUNCIÓN	**DEVUELVE**	**EJEMPLO Y RESULTADO**
AVG(columna)	Media de los valores de la columna especificada	SELECT AVG(salario) FROM empleados → 302.9412
COUNT (columna | *)	Número de valores no nulos de la columna (si esta se especifica como argumento). Utilizando el carácter * devuelve el número total de valores incluyendo los nulos	SELECT COUNT(comision) FROM empleados → 14 SELECT COUNT(*) FROM empleados → 34 (luego 20 trabajadores no tienen comisión)
MIN(columna)	Valor mínimo de la columna	SELECT MIN(salario) FROM empleados → 100
MAX(columna)	Valor máximo de la columna	SELECT MAX(salario) FROM empleados → 720
SUM(columna)	Suma de valores contenidos en la columna	SELECT SUM(salario) FROM empleados → 10300

OTRAS FUNCIONES		
FUNCIÓN	**DESCRIPCIÓN**	**EJEMPLO Y RESULTADO**
CAST (expresión AS tipo) CONVERT (expresión, tipo)	Convierte la expresión al tipo indicado	SELECT CONVERT(20060802, DATE) → '2006-08-02'
LAST_INSERT_ID()	Devuelve el valor creado por una columna de tipo AUTO_INCREMENT en la última inserción	SELECT LAST_INSERT_ID() → 0
VERSION()	Devuelve la versión del servidor MySQL	SELECT VERSION() → '5.0.20a-nt'
CONNECTION_ID()	Devuelve el identificador de conexión	SELECT CONNECTION_ID() → 4
DATABASE()	Devuelve la base de datos actual	SELECT DATABASE() → 'test'
USER()	Devuelve el usuario actual	SELECT USER() → root@localhost

2.4.7 Bloques de instrucciones

Hasta ahora hemos estado trabajando con procedimientos con un solo bloque de instrucciones, comenzando con la sentencia BEGIN y terminando con la sentencia END: