CHAPTER 8

SQL Functions

About this chapter

This chapter describes built-in functions in SQL. Functions are used to return information from the database. They are allowed anywhere an expression is allowed.

NULL value
Unless otherwise stated, any function that receives the NULL value as a parameter returns a NULL value.

Contents

Aggregate functions

Aggregate functions summarize data over a group of rows from the database. The groups are formed using the GROUP BY clause of the SELECT statement. Aggregate functions are allowed only in the select list and in the HAVING and ORDER BY clauses of a SELECT statement.

List of functions

The following aggregate functions are available:

- "AVG function " on page 282
- "COUNT function " on page 289
- "LIST function" on page 308
- "MAX function" on page 311
- "MIN function" on page 311
- "SUM function " on page 331

Date and time functions

Date and time functions perform operations on date and time data types or return date or time information.

List of functions

The following date and time functions are available:

- "DATE function" on page 290
- "DATEFORMAT function" on page 292
- "DATENAME function" on page 293
- "DATETIME function" on page 294
- "DAY function" on page 294
- "DAYNAME function" on page 295
- "DAYS function" on page 295
- "DOW function " on page 298
- "HOUR function " on page 303
- "HOURS function " on page 303
- "MINUTE function" on page 312
- "MINUTES function" on page 312
- "MONTH function" on page 313
- "MONTHNAME function" on page 314
- "MONTHS function" on page 314
- "NOW function" on page 316
- "QUARTER function" on page 321
- "SECOND" on page 325
- "SECONDS function" on page 325
- "TODAY function" on page 332
- "WEEKS function" on page 336
- "YEARS function" on page 337
- "YMD function" on page 337

Date parts

Many of the date functions use dates built from **date parts**. The following table displays allowed values of date-parts.

Date Part	Abbreviation	Values
Year	yyyy	1753 – 9999
Quarter	qq	1 - 4
Month	mm	1 - 12
Week	wk	1 - 54
Day	dd	1 - 31
Dayofyear	dy	1 - 366
Weekday	dw	1 - 7 (Sun.-Sat.)
Hour	hh	0 - 23
Minute	mi	0 - 59
Second	ss	0 - 59
Millisecond	ms	0 – 999
Calyearofweek	cyw	Integer. The year in which the week begins. The week containing the first few days of the year can be part of the last week of the previous year, depending on the weekday on which the year started. Years starting on Monday through Thursday have no days that are part of the previous year, but years starting on Friday through Sunday start their first week on the first Monday of the year.
Calweekofyear	cw	An integer from 1 to 54 representing the week number within the year that contains the specified date.
Caldayofweek	cdw	The day number within the week (Sunday = 1, Saturday = 7)

Numeric functions

Numeric functions perform mathematical operations on numerical data types or return numeric information.

List of functions

The following numeric functions are available:

- "ABS function" on page 279
- "ACOS function" on page 279
- "ASIN function" on page 280
- "ATAN function" on page 281
- "ATAN2 function" on page 281
- "CEILING function" on page 284
- "COS function" on page 289
- "COT function" on page 289
- "DEGREES function" on page 298
- "EXP function" on page 300
- "FLOOR function" on page 301
- "LOG function" on page 309
- "LOG10 function" on page 309
- "MOD function" on page 313
- "PI function" on page 318
- "POWER function" on page 319
- "RADIANS function" on page 321
- "RAND function" on page 322
- "REMAINDER function" on page 322
- "ROUND function" on page 324
- "SIGN function" on page 326
- "SIN function" on page 327
- "TAN function" on page 332
- ""TRUNCATE" function" on page 334
- "TRUNCNUM function" on page 334

String functions

String functions perform conversion, extraction or manipulation operations on strings, or return information about strings.

When working in a multi-byte character set, check carefully whether the function being used returns information concerning characters or bytes.

List of functions

The following string functions are available:

- "ASCII function " on page 280
- "BYTE_LENGTH function" on page 282
- "BYTE_SUBSTR function" on page 283
- "CHAR function" on page 285
- "CHARINDEX function" on page 285
- "CHAR_LENGTH function" on page 286
- "DIFFERENCE function" on page 298
- "LCASE function" on page 306
- "LEFT function" on page 307
- "LENGTH function" on page 307
- "LOCATE function" on page 308
- "PATINDEX function" on page 317
- "REPEAT function" on page 322
- "REPLICATE function" on page 323
- "RIGHT function" on page 323
- "RTRIM function" on page 324
- "SIMILAR function" on page 326
- "SOUNDEX function" on page 327
- "STR function" on page 329
- "STRING function" on page 329
- "STUFF function" on page 330
- "SUBSTR function" on page 330
- "TRIM function" on page 333
- "UCASE function" on page 335

- "UPPER function" on page 335

Text and image functions

Text and image functions operate on text and image data types. Adaptive Server Adaptive Server Anywhere supports only the **textptr** text and image function.

Compatibility

- The Adaptive Server Enterprise **textvalid** function is not currently supported by Adaptive Server Anywhere.

List of functions

The following text and image function is available:

- "TEXTPTR function" on page 332.

Data type conversion functions

These functions convert arguments from one data type to another.

Compatibility

- The Adaptive Server Anywhere **cast** function is not currently supported by Adaptive Server Enterprise.

List of functions

The following data type conversion functions are available:

- "CAST function" on page 284
- "CONVERT function" on page 287
- "HEXTOINT function" on page 302
- "INTTOHEX function" on page 305

System functions

System functions return system information.

List of functions

The following system functions are available:

- "CONNECTION_PROPERTY function " on page 286
- "DATALENGTH function " on page 296
- "DB_ID function" on page 296
- "DB_NAME function " on page 297
- "DB_PROPERTY function" on page 297
- "NEXT_CONNECTION function" on page 315
- "NEXT_DATABASE function" on page 315
- "PROPERTY function" on page 320
- "PROPERTY_DESCRIPTION function " on page 319
- "PROPERTY_NAME function" on page 320
- "PROPERTY_NUMBER function" on page 320

Miscellaneous functions

Miscellaneous functions perform operations on arithmetic, string or date/time expressions, including the return values of other functions.

List of functions

The following miscellaneous functions are available:

- "ARGN function" on page 279
- "COALESCE function" on page 286
- "ESTIMATE function " on page 299
- "ESTIMATE_SOURCE function" on page 299
- "EXPERIENCE_ESTIMATE function " on page 300
- "IFNULL function " on page 304
- "INDEX_ESTIMATE function" on page 304
- "ISNULL function" on page 306
- "NULLIF function" on page 316
- "NUMBER function" on page 317
- "PLAN function" on page 318
- "TRACEBACK function" on page 333

Java and SQL user-defined functions

There are two mechanisms for creating user-defined functions in Adaptive Server Anywhere. You can use the SQL language to write the function, or you can use Java.

User-defined functions in SQL

You can implement your own functions in SQL using the "CREATE FUNCTION statement" on page 397. The RETURN statement inside the CREATE FUNCTION statement determines the data type of the function.

Once a SQL user-defined function is created, it can be used anywhere a built-in function of the same data type is used.

For more information on creating SQL functions, see "Using Procedures, Triggers, and Batches" on page 221 of the book *Adaptive Server Anywhere User's Guide*.

User-defined functions in Java

Although SQL functions are useful, Java classes provide a more powerful and flexible way of implementing user-defined functions, with the additional advantage that they can be moved from the database server to a client application if desired.

Any **class method** of an installed Java class can be used as a user-defined function anywhere a built-in function of the same data type is used.

Instance methods are tied to particular instances of a class, and so have different behavior to the standard idea of user-defined functions.

For more information on creating Java classes, and on class methods, see "A Java seminar" on page 439 of the book *Adaptive Server Anywhere User's Guide*.

Alphabetic list of functions

This section describes each function individually.

ABS function [Numeric]

Function Returns the absolute value of the numeric expression.

Syntax **ABS** (*numeric-expression*)

Parameters

numeric expression The numeric-expression passed into the function.

Examples The statement

```
SELECT ABS( -66 )
```

returns the value `1.23945`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise.

ACOS function [Numeric]

Function Returns the arc-cosine of a numeric expression in radians.

Syntax **ACOS** (*numeric-expression*)

Parameters

numeric expression The numeric-expression passed into the function that represents the cosine of the angle.

Examples The statement

```
SELECT ACOS( 0.52 )
```

returns the value 1.023945.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise.

See also "ASIN function" on page 280

ARGN function [Miscellaneous]

Function Returns a selected argument from a list of arguments.

Syntax **ARGN** (*integer-expression*, *expression* [, ...])

Parameters

integer expression The integer expression used to determine the position within the list of expressions.

expression An expression of any data type passed into the function. All supplied expressions must be of the same data type.

Examples

The statement

```
SELECT ARGN( 6, 1,2,3,4,5,6 )
```

returns the value 6.

Usage

Using the value of the integer-expression as n, returns the n'th argument (starting at 1) from the remaining list of arguments. While the expressions can be of any data type, they must all be of the same data type. The integer expression must be from one to the number of expressions in the list or NULL is returned. Multiple expressions are separated by a comma.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise.

ASCII function [String]

Function

Returns the integer ASCII value of the first byte in the string-expression, or 0 for the empty string.

Syntax

ASCII (*string-expression*)

Parameters

string-expression The string passed into the function. If the string consists of the characters A-Z, the string must appear in quotes.

Examples

The statement

```
SELECT ASCII( 'Z' )
```

returns the value `90`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise.

ASIN function [Numeric]

Function

Returns the arc-sine of the numeric-expression in radians.

Syntax

ASIN (*numeric-expression*)

Parameters

numeric-expression The numeric-expression passed into the function that represents the sine of the angle.

Examples

The statement

```
SELECT ASIN( 0.52 )
```

returns the value `.546850`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise.

See also

"ACOS function" on page 279

ATAN function [Numeric]

Function

Returns the arc-tangent of the numeric-expression in radians.

Syntax

ATAN (*numeric-expression*)

Parameters

numeric-expression The expression passed into the function that represents the tangent of the angle.

Examples

The statement

```
SELECT ATAN( 0.52 )
```

returns the value `.479519`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise.

See also

"ATAN2 function" on page 281

ATAN2 function [Numeric]

Function

Returns the arc-tangent of the ratio of two numeric expressions (numeric-expression1/numeric-expression2) in radians.

Syntax

ATAN2 (*numeric-expression1*, *numeric-expression2*)

Parameters

numeric-expression1 The tangent of the angle, expressed as a column name, variable, or constant of type float, real, double precision, or any data type that can be implicitly converted to one of these types.

numeric-expression2 The second numeric expression that divides the first numeric expression to obtain the arc tangent.

Examples

The statement

```
SELECT ATAN2( 0.52, 060 )
```

returns the value 0.008666.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise.

See also

"ATAN function" on page 281

AVG function [Aggregate]

Function

Computes the average of a numeric-expression or of a set unique values for each group of rows.

Syntax

AVG (*numeric-expression* | **DISTINCT** *column-name*)

Parameters

numeric-expression The numeric expression passed into the function.

DISTINCT column-name Computes the average of the unique values in *column-name*. This is of limited usefulness, but is included for completeness.

Examples

The statement

```
SELECT AVG( salary ) FROM employee
```

returns the value `49988.6`.

Usage

This average does not include rows where the *numeric expression* is the NULL value. Returns the NULL value for a group containing no rows.

Standards and compatibility

- **SQL/92** SQL/92 compatible.
- **Sybase** Compatible with Adaptive Server Enterprise.

See also

"SUM function " on page 331
"COUNT function " on page 289

BYTE_LENGTH function [String]

Function

Returns the number of bytes in the string.

Syntax

BYTE_LENGTH (*string-expression*)

Parameters

string-expression A string passed into the function.

Examples

The statement

```
SELECT BYTE-LENGTH( 'Test Message' )
```

returns the value `12`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise.

See also "LENGTH function" on page 307

BYTE_SUBSTR function [String]

Function Returns the substring of the string-expression starting at the given start position (origin 1), in bytes.

Syntax **BYTE_SUBSTR** (*string-expression*, *integer-expression*
[, *integer-expression*])

Parameters

string- expression The string expression passed into the function.

integer-expression The integer expression passed into the function that specifies the start of the substring. A positive integer starts from the beginning of the string whereas a negative integer specifies a substring starting from the end of the string.

integer-expression The optional *integer expression* specifes the length of the substring. A positive length specifies the number of positions that will appear to the right of the starting point. A negative length specifes the number of positions that will appear to the left of the starting point.

Examples The statement

```
SELECT BYTE_SUBSTR( 'Test Message',-1,-3 )
```

returns the value '`sage`'.

Usage If the *length* is specified, the substring is restricted to that number of bytes. Both *start* and *length* can be negative. A negative starting position specifies a number of bytes from the end of the string instead of the beginning. A positive *length* specifies that the substring ends *length* bytes to the right of the starting position, while a negative *length* specifies that the substring ends *length* bytes to the left of the starting position. Using appropriate combinations of negative and positive numbers, you can get a substring from either the beginning or end of the string.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Not supported in Adaptive Server Enterprise.

CAST function [Data type conversion]

Function Returns the value of an expression converted to the supplied data type.

Syntax **CAST** (*expression* **AS** *data type*)

Parameters

expression The expression passed into the function.

data type The data type that is returned by the function.

Examples For example, the following function ensures a string is used as a date:

```
CAST( '1992-10-31' AS DATE )
```

The database server assigns the length for the following CAST

```
CAST( 1 + 2 AS CHAR )
```

You can use the CAST function to shorten strings:

```
CAST( Surname AS CHAR(10) )
```

Usage If the length is omitted for character string types, the database server chooses an appropriate length. If neither precision nor scale is specified for a DECIMAL conversion, appropriate values are selected.

Standards and compatibility

- ♦ **SQL/92** This function is SQL/92 compatible.
- ♦ **Sybase** Not supported in Adaptive Server Enterprise.

See also "CONVERT function" on page 287

CEILING function [Numeric]

Function Returns the ceiling (smallest integer not less than) of the numeric-expression.

Syntax **CEILING** (*numeric-expression*)

Parameters

numeric expression The integer expression passed into the function used to determine the ceiling.

Examples The statement

```
SELECT CEILING( 59.84567 )
```

returns the value `60`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise.

See also "FLOOR function" on page 301

CHAR function [String]

Function Returns the character with the ASCII value of the integer-expression.

Syntax **CHAR** (*integer-expression*)

Parameters

integer expression The integer expression used to determine the related ASCII character.

Examples The statement

```
SELECT CHAR( 89 )
```

returns the value `Y`.

Usage The character returned corresponds to the supplied numeric expression in the current database character set, according to a binary sort order.

If you are using multi-byte character sets, CHAR may not return a valid character. CHAR returns NULL for integer expressions with values greater than 255.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise.

CHARINDEX function [String]

Function Returns the position of string-expression2 in string-expression1.

Syntax **CHARINDEX** (*string-expression1, string-expression2*)

Parameters

string expression1 The string you are searching for within *string expression2.*

string expression2 The name of the string that contains the search string *string expression1.*

Examples The statement

```
SELECT emp_lname, emp_fname
FROM employee
WHERE CHARINDEX('K', emp_lname ) = 1
```

returns the values:

emp_lname	Emp_fname
Klobucher	James
Kuo	Felicia
Kelly	Moira

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise.

CHAR_LENGTH function [String]

Function Returns the number of characters in the string expression. The returned value includes any trailing whitespace characters. For a NULL string, CHAR_LENGTH returns NULL.

Syntax **CHAR_LENGTH** (*string-expression*)

Parameters **string-expression** The string expression passed into the function. The number of characters in this string will be the value returned from the function.

Examples The statement

```
SELECT CHAR_LENGTH( 'Chemical' )
```

returns the value `8`.

Standards and compatibility

- **SQL/92** This function is SQL/92 compatible.
- **Sybase** Compatible with Adaptive Server Enterprise.

COALESCE function [Miscellaneous]

Function Returns the value of the first expression from the list that is not NULL.

Syntax **COALESCE** (*expression*, *expression* [, ...])

Parameters **numeric-expression** Any expression passed into the function.

Examples The statement

```
SELECT COALESCE( NULL, 34, 13, 0 )
```

returns the value `34`.

Standards and compatibility

- **SQL/92** SQL/92.
- **Sybase** Not supported in Adaptive Server Enterprise.

CONNECTION_PROPERTY function [System]

Function Returns the value of the given property as a string.

Syntax

CONNECTION_PROPERTY ({ *integer-expression* | *string-expression* } ... [, *integer-expression*])

Parameters

integer expression The connection property id of the current connection.

string-expression The connection property name of the current connection. Either the property id or the property name must be specified.

integer-expression The connection id of the current database connection. The current connection is used if this argument is omitted.

Examples

The statement

```
SELECT connection_property( asademo )
```

returns the value `6`, representing the database connection property number for the database **asademo**.

Usage

The current connection is used if the second argument is omitted.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise.

See also

"Connection properties" on page 868

CONVERT function [Data type conversion]

Function

Returns the expression converted to data type.

Syntax

CONVERT (*data type*, *expression* [, *format-style*])

Parameters

data type The data type that the expression will be converted to.

expression The expression passed into the function that is to be converted.

format-style For converting strings to date or time data types and vice versa, the *format-style* is a style code number that describes the date format string to be used. The values of the *format-style* argument have the following meanings:

(yy)	(yyyy)	Output
-	0 or 100	mon dd yyyy hh:miAM (or PM)
1	101	mm/dd/yy[yy]
2	102	[yy]yy.mm.dd
3	103	dd/mm/yy[yy]

(yy)	(yyyy)	Output
4	104	dd.mm.yy[yy]
5	105	dd-mm-yy[yy]
6	106	dd mon yy[yy]
7	107	mon dd, yy[yy]
8	108	hh:mm:ss
-	9 or 109	mmm dd yyyy hh:mi:ss:mmmAM (or PM)
10	110	mm-dd-yy[yy]
11	111	[yy]yy/mm/dd
12	112	[yy]yymmdd

If no *format-style* argument is provided, Style Code 0 is used.

Examples

The following statements illustrate the use of format styles:

```
SELECT CONVERT( CHAR( 20 ), order_date, 104 )
FROM sales_order
```

order_date
16.03.1993
20.03.1993
23.03.1993
25.03.1993

```
SELECT CONVERT( CHAR( 20 ), order_date, 7 )
FROM sales_order
```

order_date
mar 16, 93
mar 20, 93
mar 23, 93
mar 25, 93

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise.

See also

"CAST function" on page 284

COS function [Numeric]

Function

Returns the cosine of the numeric-expression, expressed in radians.

Syntax

COS (*numeric-expression*)

Parameters

numeric expression The numeric expression passed into the function that specifies an angle.

Examples

The statement

```
SELECT COS( 0.52 )
```

returns the value `.86781`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise.

See also

"SIN function" on page 327
"COT function" on page 289
"TAN function" on page 332

COT function [Numeric]

Function

Returns the cotangent of the numeric-expression, expressed in radians.

Syntax

COT (*numeric-expression*)

Parameters

integer expression The angle passed into the function.

Examples

The statement

```
SELECT COT( 0.52 )
```

returns the value `1.74653`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise.

See also

"COS function" on page 289
"TAN function" on page 332
"SIN function" on page 327

COUNT function [Aggregate]

Function

Counts the number of rows in a group depending on the specified parameters.

Syntax **COUNT** (* | *expression* | **DISTINCT** *column-name*)

Parameters

***** Returns the number of rows in each group.

expression Returns the number of rows in each group where the *expression* is not the null value.

DISTINCT column-name Returns the number of different values in the column with name *column-name*. Rows where the value is the NULL value are not included in the count.

Examples The statement

```
SELECT Count( 80 ) from employee
```

returns the value `75`.

Standards and compatibility

- **SQL/92** SQL/92 compatible.
- **Sybase** Compatible with Adaptive Server Enterprise.

See also "AVG function " on page 282
"SUM function " on page 331

DATALENGTH function [System]

Function Returns the length of the expression in bytes.

Syntax **DATALENGTH** (*expression*)

Parameters

expression The expression is usually a column name. If the expression is a string constant, it must be enclosed in quotes.

Examples The statement

```
SELECT MAX( DATALENGTH( company_name ) )
FROM customer
```

returns the value `27`, the longest string in the `company_name` column.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise.

DATE function [Date and time]

Function Converts the expression into a date, and removes any hours, minutes or seconds. Conversion errors may be reported.

Syntax **DATE** (*expression*)

Parameters

expression The *expression* passed into the function to be converted to date format.

Examples

The statement

```
SELECT DATE( '1989-01-02:21:20:53' )
```

returns the value `1989-01-01` as a date.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise.

DATEADD function [Date and time-TSQL]

Function

Returns the date produced by adding the specified number of the specified date parts to the date.

Syntax

YMD (*integer-expression*, *integer-expression*, *integer-expression*)

Parameters

date-part The date-part that is passed into the function.

For a complete listing of allowed date-parts, see "Date parts" on page 270.

numeric-expression The number of *date-parts* to be added to the date.

date-expression The date passed into the function to be modified.

Examples

The statement

```
SELECT dateadd( month, 102, '1987/05/02' )
```

returns the value: `1995-11-02 00:00:00.000`.

Usage

The numeric_expression can be any numeric type; the value is truncated to an integer.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise.

DATEDIFF function [Date and time-TSQL]

Function

Returns date2 - date1, measured in the specified date part.

Syntax

DATEDIFF (*date-part*, *date-expression1*, *date-expression2*)

Parameters

date-part Specifies the date-part that is passed into the function.

For a complete listing of allowed date-parts, see "Date parts" on page 270.

date-expression1 The first *date-expression* argument passed into the function. This value is subtracted from *date-expression2* to return the number of *date-parts* between the two arguments.

date-expression2 The second *date-expression* argument passed into the function. *Date-expression1* is subtracted from this value to return the number of *date-parts* between the two arguments.

Examples

The statement

```
SELECT datediff( month, '1987/05/02', '1995/11/15' )
```

returns the value 102 to signify that there are 102 months between `1987/05/02` and `1995/11/15`.

Standards and compatibility

- ♦ **SQL/92** Transact-SQL extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise.

DATEFORMAT function [Date and time]

Function Returns a string representing the date-expression in the format specified by the string-expression.

Syntax **DATEFORMAT** (*datetime-expression*, *string-expression*)

Parameters

date-expression The *date-expression* passed into the function that specifies the date to be converted.

string-expression The *string-expression* passed into the function that specifies the format of the converted date.

For information on date format decriptions, see "DATE_FORMAT option" on page 150.

Examples

The statement

```
DATEFORMAT( '1989-01-01', 'Mmm Dd, yyyy' )
```

returns the value `Jan 1, 1989`.

Usage Any allowable date format can be used for the string-expression.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise.

> **Year 2000 compliance**
> It is possible in using the DATEFORMAT function to produce a string with the year value represented by only two digits. This can cause problems with year 2000 compliance even though no error has occurred.
>
> For more information on year 2000 compliance, please see "Date to string conversions" on page 262.

See also "DATE_FORMAT option" on page 150

DATENAME function [Date and time-TSQL]

Function Returns the name of the specified part (such as the month "June") of a DATETIME value, as a character string.

Syntax **DATENAME** (*date-part*, *date-expression*)

Parameters

date-part The date-part that is passed into the function.

For a complete listing of allowed date-parts, see "Date parts" on page 270.

date-expression The date passed into the function. This *date-expression* must contain the *date-part* field.

Examples The statement

```
SELECT datename( month , '1987/05/02' )
```

returns the value `May`.

Usage If the result is numeric, such as `23` for the day, it is still returned as a character string.

Standards and compatibility

- **SQL/92** Transact-SQL extension.
- **Sybase** Compatible with Adaptive Server Enterprise.

DATEPART function [Date and time-TSQL]

Function Returns an integer value for the specified part of a DATETIME value.

Syntax **DATEPART** (*date-part*, *date-expression*)

Parameters

date-part The date-part that is passed into the function.

For a complete listing of allowed date-parts, see "Date parts" on page 270.

date-expression The date passed into the function. The date must contain the *date-part* field.

Examples

The statement

```
SELECT datepart( month , '1987/05/02' )
```

returns the value `5`.

Standards and compatibility

- **SQL/92** Transact-SQL extension.
- **Sybase** Compatible with Adaptive Server Enterprise.

DATETIME function [Date and time]

Function

Converts the expression into a timestamp. Conversion errors may be reported.

Syntax

DATETIME (*expression*)

Parameters

expression The *expression* passed into the function that is returned as a timestamp.

Examples

The statement

```
SELECT DATETIME( '1998-09-09 12:12:12.000' )
```

returns the value `1998-09-09 12:12:12.000`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise.

DAY function [Date and time]

Function

Returns a number from 1 to 31 corresponding to the day of the given date.

Syntax

DAY (*date-expression*)

Parameters

date-expression The *date-expression* passed into the function that contains the day.

Examples

The statement

```
SELECT DAY( '1998-09-12' )
```

returns the value `12`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

DAYNAME function [Date and time]

Function

Returns the name of the day from the supplied date expression.

Syntax

DAYNAME(*date-expression*)

Parameters

date-expression The *date-expression* passed into the function that contains the day.

Examples

The statement

```
SELECT DAYNAME ( '1987/05/02' )
```

returns the value `Saturday`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

DAYS function [Date and time]

Function

Returns the number of days since the specified date and time, between two specified times or adds the specified integer-expression days to a time.

Syntax

DAYS (*datetime-expression* | *datetime-expression, datetime-expression* | *datetime-expression, integer-expression*)

Parameters

datetime-expression The *datetime-expression* passed into the function that specifies the date and time.

integer-expression The number of days to be added to the *datetime-expression*. If the *integer-expression* is negative, the appropriate number of days are subtracted from the date/time.

Using this syntax, the *datetime-expression* must be passed into the function as a proper date. For more information about this, see the "CAST function" on page 284

Examples

The statement

```
SELECT DAYS( '1998-07-13 06:07:12' )
```

returns the value 729889, while the statement:

```
SELECT DAYS( '1998-07-13 06:07:12', '1997-07-12
10:07:12' )
```

returns the value `-366` to signify the difference between the two dates.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

DATALENGTH function [System]

Function Returns the length of the expression in bytes.

Syntax **DATALENGTH** (*expression*)

Parameters

expression The expression is usually a column name. If the expression is a string constant, it must be enclosed in quotes.

Examples The statement

```
SELECT MAX( DATALENGTH( company_name ) )
FROM customer
```

returns the value `27`, the longest string in the `company_name` column.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

DB_ID function [System]

Function Returns the database ID number.

Syntax **DB_ID** ([*string-expression*])

Parameters

numeric expression The supplied *database-name* must be a string expression; if it is a constant expression, it must be enclosed in quotes. If no *database_name* is supplied, the ID number of the current database is returned.

Examples The statement

```
SELECT DB_ID( 'asademo' )
```

returns the value 0, as the database ID, while the statement:

```
SELECT DB_ID()
```

Also returns the value `0`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

DB_NAME function [System]

Function Returns the name of the current database.

Syntax **DB_NAME** ([*integer-expression*])

Parameters **database-id** The ID of the database passed into the function. The *database_id* must be a numeric expression.

Examples The statement

```
SELECT DB_NAME( 0 )
```

returns the database name `'asademo'`.

Usage If no database ID is supplied, the name of the current database is returned.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

DB_PROPERTY function [System]

Function Returns the value of the given property as a string.

Syntax **DB_PROPERTY** ({ *integer-expression* | *string-expression* }
... [,{ *integer-expression* | *string-expression* }])

Parameters **property-id** An *integer-expression* that specifies the property-id passed into the function.

property-name A *string-expression* specifying the database property name.

database-id An *integer-expression* specifying the database-id.

database-name A *string-expression* specifying the name of the database.

Examples The statement

```
SELECT DB_PROPERTY( 'PAGESIZE' )
```

returns the value `1.23945`.

Usage The current database is used if the second argument is omitted.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

DEGREES function [Numeric]

Function Converts a numeric-expression, from radians to degrees.

Syntax **DEGREES** (*numeric-expression*)

Parameters

numeric-expression The angle in radians passed into the function.

Examples The statement

```
SELECT DEGREES( 0.52 )
```

returns the value `29.793805`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

DIFFERENCE function [String]

Function Returns the difference in the soundex values between the two string expressions.

Syntax **DIFFERENCE** (*string-expression*, *string-expression*)

Parameters

string-expression1 The *string-expression* passed into the function that specifies the first soundex argument.

string-expression2 The *string-expression* passed into the function that specifies the second soundex argument.

Examples The statement

```
SELECT DIFFERENCE( 'test', 'chest' )
```

returns the value `3`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

DOW function [Date and time]

Function Returns a number from 1 to 7 representing the day of the week of the given date, with Sunday=1, Monday=2, and so on.

Syntax **DOW** (*date-expression*)

Parameters

date-expression The *date-expression* passed into the function.

Examples

The statement

```
SELECT DOW( '1998-07-09' )
```

returns the value `5`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

ESTIMATE function [Miscellaneous]

Function Provides column estimates based on specified parameters.

Syntax **ESTIMATE** (*column-name* [, *number* [, *relation-string*]])

Parameters

column-name The name of the column that is used in the estimate.

number If *number* is specified, the function returns as a REAL the percentage estimate that the query optimizer uses.

relation-string The *relation-string* must be a comparison operator enclosed in single quotes; the default is '='.

Examples

The statement

```
SELECT DISTINCT ESTIMATE( emp_id, 200, '>' )
FROM employee
```

returns the value `81.304607`.

Standards and compatibility

- **SQL/92**
- **Sybase** Not supported in Adaptive Server Enterprise.

ESTIMATE_SOURCE function [Miscellaneous]

Function Provides column estimates based on specified parameters.

Syntax **ESTIMATE_SOURCE** (*column-name* [, *number* [, *relation-string*]]

Parameters

column-name The name of the column that is used in the estimate.

number If *number* is specified, the function returns as a REAL the percentage estimate that the query optimizer uses.

relation-string The *relation-string* must be a comparison operator enclosed in single quotes; the default is '='.

Examples

The statement

```
SELECT ABS( -66 )
```

returns the value `1.23945`.

Standards and compatibility

- **SQL/92** Transact-SQL extension.
- **Sybase** Compatible with Adaptive Server Enterprise

EXP function [Numeric]

Function

Returns the exponential function, e to the exponent power of numeric-expression.

Syntax

EXP (*numeric-expression*)

Parameters

numeric-expression The argument passed into the function that specifies the exponent power.

Examples

The statement

```
SELECT EXP( 15 )
```

returns the value `7.38905`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

EXPERIENCE_ESTIMATE function [Miscellaneous]

Function

This function is the same as the ESTIMATE function, except that it always looks in the frequency table.

Syntax

EXPERIENCE_ESTIMATE (*column-name* [, *number* [, *relation-string*]])

Parameters

column-name The name of the column that is used in the estimate.

number If *number* is specified, the function returns as a REAL the percentage estimate that the query optimizer uses.

relation-string The *relation-string* must be a comparison operator enclosed in single quotes; the default is '='.

Examples

The statement

```
SELECT DISTINCT EXPERIENCE_ESTIMATE( emp_id, 200, '>' )
FROM employee
```

returns the value `NULL`.

Standards and compatibility

- **SQL/92**
- **Sybase** Not supported in Adaptive Server Enterprise.

FLOOR function [Numeric]

Function Returns the floor of (largest integer not greater than) the specified numeric-expression.

Syntax **FLOOR** (*numeric-expression*)

Parameters **numeric- expression** The argument, usually a float, passed into the function.

Examples

Value	FLOOR (Value)
123	123
123.45	123
-123.45	-124

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

See also "CEILING function" on page 284

GETDATE function [Date and time-TSQL]

Function Returns the current date and time.

Syntax **GETDATE** ()

Examples The statement

```
SELECT getdate( )
```

returns the system date and time.

Standards and compatibility

- **SQL/92** Transact-SQL extension.
- **Sybase** Compatible with Adaptive Server Enterprise

Usage The following table displays allowed values of *date-part*.

Date Part	Abbreviation	Values
Year	yyyy	1753 – 9999
Quarter	qq	1 - 4
Month	mm	1 - 12
Week	wk	1 - 54
Day	dd	1 - 31
Dayofyear	dy	1 - 366
Weekday	dw	1 - 7 (Sun.-Sat.)
Hour	hh	0 - 23
Minute	mi	0 - 59
Second	ss	0 - 59
Millisecond	ms	0 – 999
Calyearofweek	cyw	Integer. The year in which the week begins. The week containing the first few days of the year can be part of the last week of the previous year, depending on the weekday on which the year started. Years starting on Monday through Thursday have no days that are part of the previous year, but years starting on Friday through Sunday start their first week on the first Monday of the year.
Calweekofyear	cw	An integer from 1 to 54 representing the week number within the year that contains the specified date.
Caldayofweek	cdw	The day number within the week (Sunday = 1, Saturday = 7)

HEXTOINT function [Data type conversion]

Function Returns the decimal integer equivalent of a hexadecimal string.

Syntax **HEXTOINT** (*hexadecimal-string*)

Parameters

hexadecimal-string The *hexadecimal-string* passed into the function to be converted to an integer.

Examples The statement

```
SELECT HEXTOINT ( '1A4' )
```

returns the value `420`.

Standards and compatibility

- **SQL/92** Transact-SQL extension.
- **Sybase** Compatible with Adaptive Server Enterprise

See also "INTTOHEX function" on page 305

HOUR function [Date and time]

Function Returns a number from 0 to 23 corresponding to the hour component of the given datetime expression.

Syntax **HOUR** (*datetime-expression*)

Parameters

date-expression The *date-expression* passed into the function.

Examples The statement

```
SELECT HOUR( '1998-07-09 21:12:13' )
```

returns the value `21`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

HOURS function [Date and time]

Function Returns the number of hours since the specified date and time, between two specified times or adds the specified integer-expression amount of hours to a time.

Syntax1 **HOURS** (*datetime-expression* | *datetime-expression, datetime-expression* | *datetime-expression, integer-expression*)

Parameters

datetime-expression The *datetime-expression* passed into the function that specifies the date and time.

integer-expression The number of hours to be added to the *datetime-expression*. If the *integer-expression* is negative, the appropriate number of hours are subtracted from the date/time.

Using this syntax, the *datetime-expression* must be passed into the function as a proper date. For more information about this, see the "CAST function" on page 284

Examples The statement

```
SELECT HOURS( '1998-07-13 06:07:12' )
```

returns the value 17517342. While the statement:

```
SELECT HOURS( '1998-07-13 06:07:12', '10:07:12' )
```

returns the value 4 to signify the difference between the two dates.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

IFNULL function [Miscellaneous]

Function If the first expression is the NULL value, then the second expression is returned. Otherwise, the value of the third expression is returned if it was specified. If there was no third expression and the first expression is not NULL, the NULL value is returned.

Syntax **IFNULL** (*expression1*, *expression2* [, *expression3*])

Parameters

expression The numeric expression passed into the function.

Examples The statement

```
SELECT IFNULL( NULL, -66 )
```

returns the value `-66`, while the statement:

```
SELECT IFNULL( -66, NULL )
```

returns the NULL value.

Standards and compatibility

- **SQL/92** Transact-SQL extension.
- **Sybase** Compatible with Adaptive Server Enterprise

INDEX_ESTIMATE function [Miscellaneous]

Function This function is the same as the ESTIMATE function, except that it always looks only in an index.

Syntax **INDEX_ESTIMATE**(*column-name*, *number* [, *relation-string*])

Parameters

column-name The name of the column that is used in the estimate.

number If *number* is specified, the function returns as a REAL the percentage estimate that the query optimizer uses.

relation-string The *relation-string* must be a comparison operator enclosed in single quotes; the default is '='.

Examples

The statement

```
SELECT DISTINCT ESTIMATE( emp_id, 200, '>' )
FROM employee
```

returns the value `81.304607`.

Standards and compatibility

- **SQL/92**
- **Sybase** Not supported in Adaptive Server Enterprise.

INSERTSTR function [String]

Function

Inserts string-expression2 in string-expression1 at the specified character position the specified integer-expression.

Syntax

INSERTSTR (*numeric-expression*, *string-expression1*, *string-expression2*)

Parameters

integer expression The position that *string*-expression1 will be inserted into *string-expression2.*

string-expression1 The *string*-expression passed into the function that is to contain *string-expression2.*

string-expression2 The *string*-expression passed into the function that will be inserted into *string-expression1.*

Examples

The statement

```
SELECT INSERTSTR( 1, 'office ', 'back' )
```

returns the value `'backoffice'`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Not supported in Adaptive Server Enterprise.

INTTOHEX function [Data type conversion]

Function

Returns a string containing the hexadecimal equivalent of a decimal integer.

Syntax

INTTOHEX (*integer-expression*)

Parameters

integer expression The *integer expression* passed into the function that is to be converted to hexadecimal.

Examples

The statement

```
SELECT INTTOHEX( 156 )
```

returns the value `9c`.

Standards and compatibility

- **SQL/92** Transact-SQL extension.
- **Sybase** Compatible with Adaptive Server Enterprise

See also

"HEXTOINT function" on page 302

ISNULL function [Miscellaneous]

Function

Returns the first non-null character in the parameter list.

Syntax

ISNULL (*expression*, *expression* [, ...])

Parameters

expression The numeric expression passed into the function. Atleast two expressions must be passed into the function.

Examples

The statement

```
SELECT ISNULL( NULL ,-66, 55, 45, NULL, 16 )
```

returns the value `-66`.

Standards and compatibility

- **SQL/92** Transact-SQL extension.
- **Sybase** Compatible with Adaptive Server Enterprise

LCASE function [String]

Function

Converts all characters in the string-expression to lower case.

Syntax

LCASE (*string-expression*)

Parameters

string-expression The uppercase *string-expression* passed into the function which will be converted to lower case.

Examples

The statement

```
SELECT LCASE( 'LOWER CasE' )
```

returns the value `'lower case'`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** LCASE is not supported in Adaptive Server Enterprise; you can use LOWER to get the same functionality.

See also

"LOWER function" on page 310
"UCASE function" on page 335

"UPPER function" on page 335

LEFT function [String]

Function Returns the leftmost number of characters of specifed by the integer-expression of the string-expression.

Syntax **LEFT** (*string-expression*, *numeric-expression*)

Parameters

string-expression The string expression passed into the function.

integer expression The integer expression passed into the function that specifies the number of characters to return.

Examples The statement

```
SELECT LEFT( 'chocolate', 5 )
```

returns the value `'choco'`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

See also "RIGHT function" on page 323

LENGTH function[String]

Function Returns the number of characters in the specified string.

Syntax **LENGTH** (*string-expression*)

Parameters

string-expression The string-expression, usually of unknown length, passed into the function.

Examples The statement

```
SELECT LENGTH( 'chocolate' )
```

returns the value `9`.

Usage If string-expression is of binary data type, the LENGTH function behaves as BYTE_LENGTH.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Not supported in Adaptive Server Enterprise.

See also "BYTE_LENGTH function" on page 282

LIST function [Aggregate]

Function

The string-expression parameter returns a string containing a comma-separated list composed of all the values for the string-expression in each group of rows. The DISTINCT column-name parameter returns a string containing a comma-separated list composed of all the different values for string-expression in each group of rows.

Syntax

LIST (*string-expression* | *DISTINCT column-name*)

Parameters

string-expression The *string-expression* argument that you are querying.

DISTINCT column-name The name of the column that you are using in the query.

Examples

The statement

```
SELECT LIST( street ) FROM employee
WHERE emp_fname = 'Paul'
```

returns the value `'112 Endicott Street'`.

Usage

The Rows where string-expression is the NULL value are not added to the list. Returns the NULL value for a group containing no rows.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Not supported in Adaptive Server Enterprise.

LOCATE function [String]

Function

Returns the character offset (base 1) into the string specified by string-expression1 of the first occurrence of the string specified by string-expression2.

Syntax

LOCATE (*string-expression*, *string-expression* [, *numeric-expression*])

Parameters

string-expression1 The *string-expression* passed into the function. This string can be any length.

string-expression2 The *string-expression* passed into function that specifies the search string. This string is limited in length to 255 bytes or it will return the NULL value.

integer-expression The optional *integer-expression* that specifies the number of positions into the string to begin the search.

Examples

The statement

```
SELECT LOCATE( 'office party this week - rsvp as soon as
possible', 'party', 2 )
```

returns the value `8`.

Usage

If *integer-expression* is specified, the search starts at that offset into the string. LOCATE is not supported in Adaptive Server Enterprise.

The first string can be a long string (longer than 255 bytes), but the second is limited to 255 bytes. If a long string is given as the second argument, the function returns a NULL value. If the string is not found, 0 is returned. Searching for a zero-length string will return 1. If any of the arguments are NULL, the result is NULL.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

LOG function [Numeric]

Function Returns the logarithm (base) of the numeric-expression.

Syntax **LOG** (*numeric-expression*)

Parameters

integer expression The argument passed into the function.

Examples

The statement

```
SELECT LOG( 50 )
```

returns the value `3.912023`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

See also "LOG10 function" on page 309

LOG10 function [Numeric]

Function Returns the logarithm (base 10) of the numeric-expression.

Syntax **LOG10** (*numeric-expression*)

Parameters

integer expression The argument passed into the function.

Examples

The statement

```
SELECT LOG10( 50 )
```

returns the value `1.698970`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

See also "LOG function" on page 309

LOWER function [String]

Function Same as LCASE.

Syntax **LOWER** (*string-expression*)

Parameters

string-expression The uppercase *string-expression* passed into the function which will be converted to lower case.

Examples The statement

```
SELECT LOWER( 'LOWER CasE' )
```

returns the value `'lower case'`.

Standards and compatibility

- **SQL/92** This function is SQL/92 compatible.
- **Sybase** Compatible with Adaptive Server Enterprise.

See also "LCASE function" on page 306
"UCASE function" on page 335
"UPPER function" on page 335

LTRIM function [String]

Function Returns the specified string-expression with leading blanks removed.

Syntax **LTRIM** (*string-expression*)

Parameters

string-expression The *string-expression* with leading blanks.

Examples The statement

```
SELECT LTRIM( '     Test Message' )
```

returns the value `'test message'` with all leading blanks removed..

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

See also "RTRIM function" on page 324

MAX function [Aggregate]

Function Returns the maximum *expression* value found in each group of rows.

Syntax **MAX** (*expression* | *DISTINCT column name*)

Parameters

expression The expression argument passed into the function..

DISTINCT column-name Returns the same as MAX(*expression*), and is included for completeness.

Examples The statement

```
SELECT MAX( salary )from employee
```

returns the value `138948`, representing the maximum salary in the employee table.

Usage Rows where expression is the NULL value are ignored. Returns the NULL value for a group containing no rows.

Standards and compatibility

- **SQL/92** SQL/92 compatible.
- **Sybase** Compatible with Adaptive Server Enterprise

See also "MIN function" on page 311

MIN function [Aggregate]

Function Returns the minimum expression value found in each group of rows.

Syntax **MIN** (*expression* | *DISTINCT column name*)

Parameters

expression The expression argument passed into the function..

DISTINCT column-name Returns the same as MIN(*expression*), and is included for completeness.

Examples The statement

```
SELECT MIN( salary ) from employee
```

returns the value `24903`, representing the minimum salary in the employee table.

Usage Rows where expression is the NULL value are ignored. Returns the NULL value for a group containing no rows.

Standards and compatibility

- **SQL/92** SQL/92 compatible.
- **Sybase** Compatible with Adaptive Server Enterprise

See also "MAX function" on page 311

MINUTE function [Date and time]

Function

Returns a number from 0 to 59 corresponding to the minute component of the given date/time.

Syntax

MINUTE (*datetime-expression*)

Parameters

datetime-expression The *datetime-expression* passed into the function that contains the minutes field.

Examples

The statement

```
SELECT MINUTE( '1998-07-13 12:22:34 )
```

returns the value `22`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

MINUTES function [Date and time]

Function

Returns the number of minutes since the specified date and time, returns the number of minutes between two specified dates, and adds or subtracts *integer-expression* minutes to a specified date.

Syntax

MINUTES (*datetime-expression* | *datetime-expression, datetime-expression* | *datetime-expression, integer-expression*)

Parameters

datetime-expression The date passed into the function.

integer-expression The number of minutes to be added to the *datetime-expression*. If the *integer-expression* is negative, the appropriate number of minutes are subtracted from the date/time.

Using this syntax, the *datetime-expression* must be passed into the function as a proper date. For more information about this, see the "CAST function" on page 284

Examples

The statement

```
SELECT MINUTES( '1998-07-13 06:07:12' )
```

returns the value `1051040527`. While the statement:

```
SELECT MINUTES( '1998-07-13 06:07:12', '10:07:12' )
```

returns the value `240` to signify the difference between the two dates.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

MOD function [Numeric]

Function Returns the remainder when the dividend is divided by the divisor.

Syntax **MOD** (*dividend*, *divisor*)

Parameters

dividend The dividend, the first expression, of the equation.

divisor The divisor, the second expression, of the equation.

Examples The statement

```
SELECT MOD( 5,3 )
```

returns the value 2.

Usage Division involving a negative dividend will give a negative or zero result. The sign of the divisor has no effect.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Not supported in Adaptive Server Enterprise. The % operator is used as a modulo operator in Adaptive Server Enterprise.

MONTH function [Date and time]

Function Returns a number from 1 to 12 corresponding to the month of the given date.

Syntax **MONTH** (*date-expression*)

Parameters

date-expression The *date-expression*, that contains a month value, passed into the function.

Examples The statement

```
SELECT MONTH( '1998-07-13' )
```

returns the value `7`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

MONTHNAME function [Date and time]

Function Returns the name of the month from the supplied date expression.

Syntax **MONTHNAME** (*date-expression*)

Parameters

date-expression The *date-expression*, that contains a month value, passed into the function.

Examples The statement

```
SELECT MONTHNAME( '1998-09-05' )
```

returns the value `'September'`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

MONTHS function [Date and time]

Function Returns the number of months since the specified date, the number of months between two specified dates or adds a specified number of months to the specified date.

Syntax **MONTHS** (*datetime-expression* **|** *datetime-expression, datetime-expression* **|** *datetime-expression, integer-expression*)

Parameters

date-time-expression The date, that contains a month field, passed into the function.

integer-expression The number of months to add to the specified date. If the new date is past the end of the month (such as MONTHS('1992-01-31', 1) the result is set to the last day of the month. If the *integer-expression* is negative, the appropriate number of months are subtracted from the date. Hours, minutes, and seconds are ignored.

Using this syntax, the *datetime-expression* must be passed into the function as a proper date. For more information about this, see the "CAST function" on page 284

Examples The statement

```
SELECT MONTHS( '1998-07-13 06:07:12' )
```

returns the value `23982`. While the statement:

```
SELECT MONTHS( '1998-07-13 06:07:12', '1978-07-13
10:07:12' )
```

returns the value `-240` to signify the difference between the two dates.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

NEXT_CONNECTION function [System]

Function Returns the next connection number, or the first connection if the parameter is NULL.

Syntax **NEXT_CONNECTION** ({ **NULL** | *string-expression* })

Parameters

string-expression A *string-expression* passed into the function that specifies the database connection id.

Examples The statement

```
SELECT next_connection( NULL )
```

returns the value `569851433`, the first connection value. The statement

```
SELECT next_connection( 569851433 )
```

returns the value `1661140050`.

Standards and compatibility

- **SQL/92** Transact-SQL extension.
- **Sybase** Compatible with Adaptive Server Enterprise

NEXT_DATABASE function [System]

Function Returns the next database number, or the first connection if the parameter is NULL.

Syntax **NEXT_DATABASE** ({ **NULL** | *string-expression* })

Parameters

database-id A *string-expression* that specifies the id number of the database.

Examples The statement

```
SELECT next_database( NULL )
```

returns the value 0, the first database value. The statement

```
SELECT next_connection( 0 )
```

returns the value `569851433,` as the next database value.

Standards and compatibility

- **SQL/92** Transact-SQL extension.
- **Sybase** Compatible with Adaptive Server Enterprise

NOW function [Date and time]

Function Returns the current date and time. This is the historical syntax for CURRENT TIMESTAMP.

Syntax **NOW** (*)

Parameters

***** The asterisk.

Examples The statement

```
SELECT NOW(*)
```

returns the current date and time..

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

NULLIF function [Miscellaneous]

Function To provide an abbreviated CASE statement by comparing expressions.

Syntax **NULLIF** (*expression1*, *expression2*)

Examples

- The statement

  ```
  SELECT NULLIF( 'a', 'b' )
  ```

  returns 'a'.
- The statement

  ```
  SELECT NULLIF( 'a', 'a' )
  ```

  returns NULL.

Usage NULLIF compares the values of the two expressions. If the first expression equals the second expression, NULLIF returns NULL. If the first expression does not equal the second expression, NULLIF returns the first expression. The NULLIF function provides a short way to write some CASE expressions.

Standards and compatibility

- **SQL/92** Transact-SQL extension.
- **Sybase** Supported by Adaptive Server Enterprise.

NUMBER function [Miscellaneous]

Function

Generates numbers starting at 1 for each successive row in the results of the query.

Syntax

NUMBER (*)

Examples

The statement

```
Select NUMBER( * ) FROM department where dept_id > 5
```

Returns a numbered list.

Usage

Although the NUMBER(*) function is useful for generating primary keys when using the INSERT from SELECT statement (see "INSERT statement" on page 498), the AUTOINCREMENT column is a preferred mechanism for generating sequential primary keys. For information on the AUTOINCREMENT default, see "CREATE TABLE statement" on page 415.

You should not use the NUMBER(*) function anywhere but in a select-list. If you do use NUMBER(*) rather than the preferred AUTOINCREMENT, you should check your results carefully, as the behavior is not reliable in several circumstances. For example, including the function in a WHERE clause or a HAVING clause produces unpredictable results, and you should not include NUMBER(*) in a UNION operation.

In Embedded SQL, care should be exercised when using a cursor that references a query containing a NUMBER(*) function. In particular, this function returns negative numbers when a database cursor is positioned using relative to the end of the cursor (an absolute position with a negative offset).

Standards and compatibility

- **SQL/92** Transact-SQL extension.
- **Sybase** Compatible with Adaptive Server Enterprise

PATINDEX function [String]

Function

Returns an integer representing the starting position in characters of the first occurrence of the pattern in the specified string expression, or a zero if the specified pattern is not found.

Syntax

PATINDEX ('*%pattern%*', *string_expression*)

Parameters

%pattern% The pattern passed into the function. If the leading percent wild card is omitted, PATINDEX returns one (1) if pattern occurs at the beginning of the string, and zero if not. If *pattern* starts with a percent wild card, the two leading percent wild cards are treated as one.

string-expression The *string-expression* passed into the function. This string can be any length.

Examples

- The statement...

  ```
  SELECT PATINDEX( '%hoco%', 'chocolate' )
  ```

  returns the value `2`.

- The statement ...

  ```
  select patindex ('%4_5_', '0a1A 2a3A 4a5A')
  ```

  ... returns the value 11.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

PI function [Numeric]

Function Returns the numeric value PI.

Syntax **PI** (*)

Parameters

***** The asterisk.

Examples The statement

```
SELECT PI( * )
```

returns the value `3.141592653`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** The PI() function is supported in Adaptive Server Enterprise, but PI(*) is not.

PLAN function [Miscellaneous]

Function Returns the optimization strategy of the SELECT statement string-expression, as a string.

Syntax **PLAN** (*string-expression*)

Parameters

string-expression The *string-expression* passed into the function. The *string-expression* must be a valid ISQL statement.

Examples The statement

```
SELECT PLAN( 'select * from department where dept_id >
100' )
```

returns information about the query. This is useful if you are querying on a system with limited resources.

Standards and compatibility

- ♦ **SQL/92** Transact-SQL extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

POWER function [Numeric]

Function Raises numeric-expression1 to the power numeric-expression2.

Syntax **POWER** (*numeric-expression1*, *numeric-expression2*)

Parameters

numeric-expression1 The expression passed into the function the specifies the base.

numeric-expression2 The expression passed into the function the specifies the exponent.

Examples The statement

```
SELECT Power( 2, 6 )
```

returns the value `64`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

PROPERTY_DESCRIPTION function [System]

Function Returns a description of the property with the supplied property-name or property-number.

Syntax **PROPERTY_DESCRIPTION** ({ *integer-expression* | *string-expression* })

Parameters

integer-expression The property-number of the database property.

string-expression The name of the database property.

Examples The statement

```
SELECT PROPERTY_DESCRIPTION( 126 )
```

returns the description: database page size.

Usage While each property does have a number as well as a name, the number is subject to change between releases, and should not be used as a reliable identifier for a given property.

Standards and compatibility

- **SQL/92** Entry level function.
- **Sybase** Compatible with Adaptive Server Enterprise

PROPERTY function [System]

Function Returns the value of the specified property as a string.

Syntax **PROPERTY** ({ *integer-expression* | *string-expression* })

Parameters

numeric-expression The database property passed into the function.

Examples The statement

```
SELECT PROPERTY( 1024 )
```

returns the value `1.23945`.

Standards and compatibility

- **SQL/92** Entry level function.
- **Sybase** Compatible with Adaptive Server Enterprise

PROPERTY_NAME function [System]

Function Returns the name of the property with the supplied property-number.

Syntax **PROPERTY_NAME** (*integer-expression*)

Parameters

integer-expression The property-number of the database property.

Examples The statement

```
SELECT PROPERTY_NAME( 126 )
```

returns the property name `'PageSize'`.

Standards and compatibility

- **SQL/92** Entry level function.
- **Sybase** Compatible with Adaptive Server Enterprise

PROPERTY_NUMBER function [System]

Function Returns the number of the property with the supplied property-name.

Syntax **PROPERTY_NUMBER** (*string-expression*)

Parameters

property-name A *string-expression* that specifies the database property function.

Examples

The statement

```
SELECT PROPERTY_NUMBER( 'PAGESIZE' )
```

returns the value `126`.

Standards and compatibility

- **SQL/92** Entry level function.
- **Sybase** Compatible with Adaptive Server Enterprise

QUARTER function [Date and time]

Function

Returns the quarter from the supplied date expression.

Syntax

QUARTER(*date-expression*)

Parameters

date- expression The *date- expression* passed into the function.

Examples

The statement

```
SELECT QUARTER ( '1987/05/02' )
```

returns the value `2`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

RADIANS function [Numeric]

Function

Converts a numeric-expression, from degrees to radians.

Syntax

RADIANS (*numeric-expression*)

Parameters

numeric-expression The number of degrees. This angle is converted to radians.

Examples

The statement

```
SELECT RADIANS( 0.52 )
```

Returns the value `.0090757`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

RAND function [Numeric]

Function Returns a random number in the interval 0 to 1, with integer-expression as an optional seed.

Syntax **RAND** ([*integer-expression*])

Parameters **integer expression** The optional seed used to create a random number. This argument allows you to create a random number based on the argument itself.

Examples The statement

```
SELECT RAND( 4 )
```

returns the value `.0554504`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

REMAINDER function [Numeric]

Function Same as the MOD function.

Syntax **REMAINDER** (*numeric-expression*, *numeric-expression*)

Parameters **dividend** The dividend, the first expression, of the equation.

divisor The divisor, the second expression, of the equation.

Examples The statement

```
SELECT REMAINDER( 5,3 )
```

returns the value `2`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Not supported in Adaptive Server Enterprise. The % (modulo) operator and the division operator can be used to produce a remainder.

REPEAT function [String]

Function Returns a string composed of integer-expression instances of string-expression, concatenated together.

Syntax **REPEAT** (*string-expression*, *numeric-expression*)

Parameters **string-expression** The *string-expression* to be repeated.

integer-expression The number of times the *string-expression* will be repeated.

Examples The statement

```
SELECT REPEAT( 'repeat', 3 )
```

returns the value `'repeatrepeatrepeat'`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** REPEAT is not supported in Adaptive Server Enterprise, but REPLICATE provides the same capabilities.

See also "REPLICATE function" on page 323

REPLICATE function [String]

Function Same as the REPEAT function.

Syntax **REPLICATE** (*string-expression*, *numeric-expression*)

Parameters

string-expression The *string-expression* to be repeated.

integer-expression The number of times the *string-expression* will be repeated.

Examples The statement

```
SELECT REPLICATE( 'repeat', 3 )
```

returns the value `'repeatrepeatrepeat'`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

See also "REPEAT function" on page 322

RIGHT function [String]

Function Returns the rightmost characters of the string-expression.

Syntax **RIGHT** (*string-expression*, *numeric-expression*)

Parameters

string-expression The string expression passed into the function.

integer expression The integer expression passed into the function that specifies the number of characters to return.

Examples The statement

```
SELECT RIGHT( 'chocolate', 5 )
```

returns the value `'olate'`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

See also "LEFT function" on page 307

ROUND function [Numeric]

Function Rounds the *numeric-expression* to the specified integer-expression amount of places after the decimal point.

Syntax **ROUND** (*numeric-expression*, *integer-expression*)

Parameters

numeric-expression The number, passed into the function, to be rounded..

integer-expression A positive integer specifies the number of significant digits to the right of the decimal point to round to. A negative expression specifies the number of significant digits to the left of the decimal point.

Examples The statement

```
SELECT ROUND( 123.234, 1 )
```

returns the value `123.200`.

Usage A positive integer determines the number of significant digits to the right of the decimal point; a negative integer, the number of significant digits to the left of the decimal point.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

RTRIM function [String]

Function Returns the string-expression with trailing blanks removed.

Syntax **RTRIM** (*string-expression*)

Parameters

string-expression The string-expression passed into the function with leading blanks.

Examples The statement

```
SELECT RTRIM( 'Test Message     ' )
```

returns the output `'test message'` with all trailing blanks removed.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

See also "LTRIM function" on page 310

SECOND function [Date and time]

Function Returns a number from 0 to 59 corresponding to the second component of the given date.

Syntax **SECOND** (*datetime-expression*)

Parameters

datetime-expression The *datetime-expression,* that contains the second field, that is passed into the function.

Examples The statement

```
SELECT SECOND( '1998-07-13:21:21:25' )
```

returns the value `21`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

SECONDS function [Date and time]

Function Return the number of seconds since the specified date, the number of seconds between two specified dates or adds a specified amount of seconds to the specified date.

Syntax **SECONDS** (*datetime-expression* | *datetime-expression, datetime-expression* | *datetime-expression, integer-expression*)

Parameters

date-time-expression The date, that contains a seconds field, passed into the function.

integer-expression The number of seconds to add to the specified date. If the *integer-expression* is negative, the appropriate number of seconds are subtracted from the date/time.

Using this syntax, the *datetime-expression* must be passed into the function as a proper date. For more information about this, see the "CAST function" on page 284

Examples The statement

```
SELECT SECONDS( '1998-07-13 06:07:12' )
```

returns the value `63062431632`. While the statement:

```
SELECT SECONDS( '1998-07-13 06:07:12', '1998-07-12
10:07:12' )
```

returns the value `-72000` to signify the number of seconds between the two dates.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

SIGN function [Numeric]

Function Returns the sign of the specified numeric-expression.

Syntax **SIGN** (*numeric-expression*)

Parameters

numeric-expression The signed or unsigned expression passed into the function. The return value 1 signifies a positive sign while the return value –1 signifies a negative value.

Examples The statement

```
SELECT SIGN( -550 )
```

returns the value:`-1`

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

SIMILAR function [String]

Function Returns an integer between 0 and 100 representing the similarity between the two strings. The result can be interpreted as the percentage of characters matched between the two strings (100 percent match if the two strings are identical).

Syntax **SIMILAR** (*string-expression*, *string-expression*)

Parameters

string-expression1 The first *string-expression* passed into the function that specifies the first comparative search argument.

string-expression2 The second *string-expression* passed into the function that specifies the second comparative search argument. The first argument is compared to this argument and a relative similarity percentage value is derived and returned.

Examples The statement

```
SELECT SIMILAR( 'toast', 'coast' )
```

returns the value `75`. This signifies that the two values are % 75 similar.

Usage This function can be very useful for correcting a list of names (such as customers). Some customers may have been added to the list more than once with slightly different names. Join the table to itself and produce a report of all similarities greater than 90 percent but less than 100 percent.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Not supported by Adaptive Server Enterprise.

SIN function [Numeric]

Function Returns the sine of the numeric-expression, expressed in radians.

Syntax **SIN** (*numeric-expression*)

Parameters **integer expression** The integer expression passed into the function that specifies the angle.

Examples The statement

```
SELECT SIN( 0.52 )
```

returns the value `.496880`.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

See also "COS function" on page 289
"COT function" on page 289
"TAN function" on page 332

SOUNDEX function [String]

Function Returns a number representing the sound of the string-expression. Although it is not perfect, soundex will normally return the same number for words that sound similar and that start with the same letter.

Syntax **SOUNDEX** (*string-expression*)

Parameters **string-expression** The *string-expression* passed into the function.

Examples The statement

```
soundex( 'Smith' ) = soundex( 'Smythe' )
```

The soundex function value for a string is based on the first letter and the next three consonants other than H, Y, and W. Doubled letters are counted as one letter. For example,

```
soundex( 'apples' )
```

is based on the letters A, P, L and S. Multi-byte characters are ignored by the SOUNDEX function.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

SPACE function [Miscellaneous]

Function Returns the amount of spaces specified by the integer-expression.

Syntax **SPACE** (*integer-expression*)

Parameters

integer expression The number of spaces to be returned by the function.

Examples The statement

```
SELECT SPACE( 10 )
```

returns the 10 spaces..

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

SQRT function [Miscellaneous]

Function Returns the square root of the specified numeric-expression.

Syntax **SQRT** (*numeric-expression*)

Parameters

integer expression The numeric expression passed into the function that specifies the operand of the square root operation.

Examples The statement

```
SELECT SQRT( 9 )
```

returns the value `3`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

STR function [String]

Function

Returns the string equivalent of the specified number.

Syntax

STR (*numeric_expression* [, *length* [, *decimal*]])

Parameters

numeric-expression Any approximate numeric (float, real, or double precision) expression.

length Sets the number of characters to be returned (including the decimal point, all digits to the right and left of the decimal point, and blanks). The default is 10.

decimal Sets the number of decimal digits to be returned. The default is 0.

Examples

For example, the following statement returns the result 1235:

```
SELECT ( 1234.56 )
```

The following statement returns the result 1234.6

```
SELECT STR( 1234.56, 10, 1 )
```

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

The *length* value must be at least enough to hold the required numbers. The following statement returns an invalid result:

```
SELECT STR( 1234.56, 3, 1 )
```

STRING function [String]

Function

Concatenates the strings into one large string. NULL values are treated as empty strings ('').

Syntax

STRING (*string-expression* [, ...])

Parameters

string-expression The *string-expression* passed into the function. If only one argument is supplied, the *string-expression* is converted into a single expression. If more than one argument is supplied, they are concatenated into a single string.

Examples

The statement

```
SELECT STRING(
'test','message','for','string','function' )
```

returns the value `'testmessageforstringfunction'`.

Usage

Any numeric or date parameters are automatically converted to strings before concatenation. The STRING function can also be used to convert any single expression to a string by supplying that expression as the only parameter.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

STUFF function [String]

Function

Returns the string formed by deleting a specified number of characters from one string and replacing them with another string.

Syntax

STUFF (*string-expression1*, *start*, *length*, *string-expression2*)

Parameters

string-expression1 The *string-expression* that is modified by the STUFF function.

start The character position at which to begin deleting characters.

length The number of characters to delete.

string-expression2 The *string-expression* that is inserted into *string-expression1*. To delete a portion of a string using STUFF, use a replacement string of NULL.

Examples

The statement

```
SELECT STUFF( 'chocolate cake', 11, 4, 'pie' )
```

returns the value `'chocolate pie'`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

SUBSTR function [String]

Function

Returns the substring of the string-expression starting at the specified start position (origin 1).

Syntax

SUBSTR (*string-expression*, *integer-expression* [, *integer-expression*])

Parameters

string-expression The *string-expression* passed into the function.

start The *integer-expression* passed into the function that specifies the start position of the substring to return. A negative starting position specifies a number of characters from the end of the string instead of the beginning.

length The *integer-expression* passed into the function that specifies the length of the substring to return. A positive *length* specifies that the substring ends *length* characters to the right of the starting position, while a negative *length* specifies that the substring ends *length* characters to the left of the starting position.

Examples

The result of the statement:

```
select substr( 'back yard',1,4 )
```

is 'back'. While the same statement with negative arguments,

Usage

If the length is specified, the substring is restricted to that length. Both start and length can be negative.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

```
SELECT substr( 'back yard',-1, -4 )
```

produces 'yard'. Using appropriate combinations of negative and positive numbers, you can easily get a substring from either the beginning or end of the string. If *string-expression* is of binary data type, the SUBSTR function behaves as BYTE_SUBSTR.

SUM function [Aggregate]

Function

Returns the total of the specified expression for each group of rows.

Syntax

SUM (*expression* | **DISTINCT** *column-name*)

Parameters

expression The argument passed into the function.

DISTINCT column-name This is of limited usefulness, but is included for completeness.

Examples

The statement

```
SELECT SUM( salary )
FROM Employee
```

returns the value `3749146`.

Usage

Rows where the specified expression is the NULL value are not included. Returns NULL for a group containing no rows.

Standards and compatibility

- **SQL/92** SQL/92 compatible.
- **Sybase** Compatible with Adaptive Server Enterprise

See also

"COUNT function " on page 289
"AVG function " on page 282

TAN function [Numeric]

Function Returns the tangent of the numeric-expression, expressed in radians.

Syntax **TAN** (*numeric-expression*)

Parameters **integer expression** The *integer expression* passed into the function that specifies the angle in radians.

Examples The statement

```
SELECT TAN( 0.52 )
```

returns the value `.572561`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

TEXTPTR function [String]

Function Returns the 16-byte binary pointer to the first page of the specified text column.

Syntax **textptr** (*column-name*)

Parameters **column-name** The name of a text column.

Example

- Use the **textptr** function to locate the text column, copy, associated with au_id 486-29-1786 in the author's blurbs table.

  The text pointer is put into a local variable @val and supplied as a parameter to the readtext command, which returns 5 bytes, starting at the second byte (offset of 1).

```
DECLARE @val VARBINARY(16)
SELECT @val = TEXTPTR(copy)
FROM blurbs
WHERE au_id = "486-29-1786"
READTEXT blurbs.copy @val 1 5
```

TODAY function [Date and time]

Function Returns today's date. This is the historical syntax for CURRENT DATE.

Syntax **TODAY** (*)

Parameters * The integer expression used to determine the position within the list of expressions.

Examples The statement

```
SELECT TODAY( * )
```

returns the current day according to the system clock.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

TRACEBACK function [Miscellaneous]

Function Returns a string containing a traceback of the procedures and triggers that were executing when the most recent exception (error) occurred.

Syntax **TRACEBACK** (*)

Examples To use the traceback function, enter the following after an error occurs while executing a procedure:

```
SELECT TRACEBACK ( * )
```

Usage This is useful for debugging procedures and triggers

Standards and compatibility

- ♦ **SQL/92** Transact-SQL extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise

TRIM function [String]

Function Returns the string-expression with both leading and trailing blanks removed.

Syntax **TRIM** (*string-expression*)

Parameters

string-expression The *string-expression* passed into the function with leading and/or trailing blanks.

Examples The statement

```
SELECT TRIM( '   chocolate   ' )
```

returns the value `'chocolate'` with no leading or trailing blanks.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Compatible with Adaptive Server Enterprise.

See also "LTRIM function" on page 310
"RTRIM function" on page 324

"TRUNCATE" function [Numeric]

Function Truncates the numeric-expression at the integer-expression amount of places after the decimal point. TRUNCNUM is the same function, and is the preferred function name.

Syntax "**TRUNCATE**" (*numeric-expression*, *integer-expression*)

Parameters

numeric-expression The numeric expression passed into the function which is truncated.

integer-expression The expression passed into the function that specifies the point at which the *numeric-expression* is truncated.

Examples The statement

```
SELECT "TRUNCATE"( 655, -2 )
```

returns the value 600, while the statement:

```
SELECT "TRUNCATE"( 655.345, 2 )
```

returns the value `655.340`.

Usage This function is the same as TRUNCNUM. Using TRUNCNUM is recommended as there are then no problems with keyword conflicts.

A positive integer determines the number of significant digits to the right of the decimal point; a negative integer, the number of significant digits to the left of the decimal point.

The double quotes are required because of a keyword conflict with the TRUNCATE TABLE statement. You can only use TRUNCATE if the quoted_identifier option is set to OFF.

The value of the integer-expression is relative to the decimal point. Negative numbers are to the left of the decimal point.

Standards and compatibility

- ♦ **SQL/92** Vendor extension.
- ♦ **Sybase** Not supported in Adaptive Server Enterprise.

TRUNCNUM function [Numeric]

Function Truncates the numeric-expression at the integer-expression amount of places after the decimal point.

Syntax **TRUNCNUM** (*numeric-expression*, *integer-expression*)

Parameters

numeric-expression The numeric expression passed into the function which is truncated.

integer-expression The expression passed into the function that specifies the point at which the *numeric-expression* is truncated.

Examples The statement

```
SELECT TRUNCNUM( 655, -2 )
```

returns the value 600, while the statement:

```
SELECT TRUNCNUM( 655.345, 2 )
```

returns the value `655.340`.

Usage This function is the same as TRUNCATE, but TRUNCNUM has no problems with keyword conflicts.

A positive integer determines the number of significant digits to the right of the decimal point; a negative integer, the number of significant digits to the left of the decimal point.

The value of the integer-expression is relative to the decimal point. Negative numbers are to the left of the decimal point.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Not supported in Adaptive Server Enterprise.

UCASE function [String]

Function Converts all characters in the string-expression to uppercase.

Syntax **UCASE** (*string-expression*)

Parameters

string-expression The *string-expression* passed into the function that is converted to an upper case string.

Examples The statement

```
SELECT UCASE( 'ChocoLate' )
```

returns the value `'CHOCOLATE'`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** UCASE is not supported by Adaptive Server Enterprise, but UPPER provides the same feature in a compatible manner.

See also "UPPER function" on page 335

UPPER function [String]

Function Same as UCASE.

Syntax **UPPER** (*string-expression*)

Parameters

string-expression The *string-expression* passed into the function that is converted to an upper case string.

Examples The statement

```
SELECT UPPER( 'ChocoLate' )
```

returns the value `'CHOCOLATE'`.

Standards and compatibility

- **SQL/92** This function is SQL/92 compatible.
- **Sybase** Compatible with Adaptive Server Enterprise.

See also "UCASE function" on page 335
"LCASE function" on page 306
"LOWER function" on page 310

WEEKS function [Date and time]

Function Return the number of weeks since the specified date, the number of weeks between two specified dates or adds the integer-expression amount of weeks to the specified date.

Syntax **WEEKS** (*datetime-expression* | *datetime-expression, datetime-expression* | *datetime-expression, integer-expression*)

Parameters

date-time-expression The date that is passed into the function.

integer-expression The number of weeks to add to the specified date. If the *integer-expression* is negative, the appropriate number of weeks are subtracted from the date/time.

Using this syntax, the *datetime-expression* must be passed into the function as a proper date. For more information about this, see the "CAST function" on page 284

Examples The statement

```
SELECT WEEKS( '1998-07-13 06:07:12' )
```

returns the value 104270. While the statement:

```
SELECT WEEKS( '1998-07-13 07:07:12', '1998-07-26
10:07:26' )
```

returns the value `1` to signify the number of weeks between the two dates.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

YEARS function [Date and time]

Function

Return the number of years since the specified date, the number of years between two specified dates or adds the integer-expression amount of years to the specified date.

Syntax

YEARS (*datetime-expression* | *datetime-expression, datetime-expression* | *datetime-expression, integer-expression*)

Parameters

date-time-expression The date that is passed into the function.

integer-expression The number of years to add to the specified date. If the *integer-expression* is negative, the appropriate number of years are subtracted from the date/time. Hours, minutes, and seconds are ignored.

Using this syntax, the *datetime-expression* must be passed into the function as a proper date. For more information about this, see the "CAST function" on page 284

Examples

The statement

```
SELECT YEARS( '1998-07-13 06:07:12' )
```

returns the value 1998. While the statement:

```
SELECT YEARS( birthdate, CURRENT DATE )
```

can be used to determine a person's age.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise

YMD function [Date and time]

Function

Returns a date value corresponding to the given year, month, and day of the month.

Syntax

YMD (*integer-expression*, *integer-expression*, *integer-expression*)

Parameters

integer expression The year number.

integer expression The number of the month. If the month is outside the range 1-12, the year is adjusted accordingly.

integer expression Specifies the day number. The day is allowed to be any integer, the date is adjusted accordingly.

Examples

The statement

```
SELECT YMD( 1998, 06, 12 )
```

returns the value 1998-06-12. However, if the values are outside there normal range, the date will adjust accordingly. For example:

```
YMD( 1992, 15, 1 )
```

returns the value '`Mar 1, 1993'`.

Standards and compatibility

- **SQL/92** Vendor extension.
- **Sybase** Compatible with Adaptive Server Enterprise