Chapter 2.

Core & Script Writing

Desktop, Web, Mobile

“Trace Functions & Methods”	Working with the Trace Window.	34
“Variable Functions & Methods”	Working with Variables.	35
“Work Queue”	Save work for use at a later time.	36

Core Functions

CORE FUNCTIONS FOR WRITING EXPRESSIONS

The following list of functions is recommended as the starting point for the programmer who is writing Xbasic expressions.

Function	Description
Character Functions (Core)	
ALLTEXT()	Returns at character string of all of the field values in the current record.
ALLTRIM()	Removes leading and trailing blanks from a *Input_String*.
ASC()	Returns the integer *ASCII_Value* of the first letter in the specified *Character_String*.
AT()	Returns a position integer indicating where the Nth occurrence of the *Find_String* begins in the *Search_In_String*. Case insensitive.
ATC()	Returns a position integer indicating where the Nth occurrence of the *Find_String* begins in the *Search_In_String*. Case sensitive.
CHR()	Returns the *Single_Character* corresponding to the integer *ASCII_Value*.
CONTAINS()	Returns .T. (TRUE) if all words in the *Find_Words* string appear at least once in the *Search_String*.
LEFT()	Returns a character string containing the left-most *Number_Of_Characters* of the *Input_String*.
LEN()	Returns an integer equal to the number of characters in the *Input_String*.
LOWER()	Converts the specified character string to lowercase.
LTRIM()	Removes any specified leading characters in the *Input_String*.
OCCURS()	Returns the number of times the *Find_String* is found in the *Search_In_String*.
QUOTE()	Places quotes around *Input_String* and returns the result as a string.
RTRIM()	Removes any specified trailing characters in the Input_String.
STRITRAN()	Replaces every occurrence of *Find_String* in *Input_String* with a *Replace_String*. Case insensitive.
STRTRAN()	Replaces every occurrence of *Find_String* in *Input_String* with a Replace_String. Case sensitive.
STR()	Converts a *Number* to a right-justified character string.

SUBSTR()	Returns a *Substring* of a *Input_String*. The *Substring* begins at the specified *Starting_Position* and continues for the specified *Number_Of_Characters*.
UPPER()	Converts all the text in a *Input_String* to uppercase.
VAL()	Converts a *Character_String* to a numeric value.
WORD()	Returns one or more words, specified by the *Word_Number* and *Word_Count* and delimited by *Word_Delimiter*, from the supplied *Input_String*.
WORDAT()	Returns the word number of *Word* in *Input_String*. Case insensitive.
WORDATC()	Returns the word number of *Word* in *Input_String*. Case sensitive.
Character Functions	See also “Character Types” on page 39.
Date and Time Functions (Core)	
ADD_BUS_DAYS()	Adds *Add_Days* number of business days to a *Starting_Date* and returns an *Ending_Date*.
ADDMONTHS()	Adds the specified *Number_Of_Months* to the *Starting_Date.*
ADDYEARS()	Adds the specified *Number_Of_Year* s to a *Date* and returns an Ending_Date.
AGE()	Returns the integer number of years between *Starting_Date* and *Ending_Date* (default now).
CDOW()	Returns a character string containing the name of the day of the week for the specified *Date*.
CMONTH()	Returns the *Month_Name* in the specified *Date*
CTOD()	Converts a character string containing date information and returns a Date value.
CYEAR()	Returns a character string containing the year specified in *Date_Value* in the form "YYYY".
DATE()	Returns todays date.
DATE_VALUE()	Returns a Date value for the specified year, month and day.
DAY()	Returns an integer corresponding to the *Date's* day of the month.
MONTH()	Returns an integer equal to the month number (1-12) of the specified *Date_Value*.
NOW()	Returns the current date/time, correct to the millisecond, in the Time data type.
TIME()	Returns a *Formatted_String* containing the system time.

TOTIME()	Returns a formatted character string containing a time based on the specified *Number_Of_Seconds* past midnight.
YEAR()	Returns a four digit integer equal to the year portion of the specified *Date_Value*.
Date & Time Functions	See also “Date & Time Types” on page 65.
Expression Evaluation Functions and Methods (Core)	
BETWEEN()	Returns .T. (TRUE) if *Expression* is greater than or equal to *Lower_Value* and less than or equal to *Higher_Value*.
CASE()	Returns the value (*Value* _1 ... *Value* _16) specified by the first expression (*Expression* _ *1* ... *Expression* _ *16*) that evaluates to .T. (TRUE).
CCVALID()	Returns .T. (TRUE) if a field contains a valid *Credit_Card_Number*.
EVAL()	Evaluates *Expression* and returns the result of that expression. •See also “Reports” on page 98 > *CURRENT_EXPRESSION().
IIF()	Evaluates the *Logical_Expression*. If the expression evaluates to TRUE, the *True_Value* is returned, otherwise the *False_Value* is returned.
Expression Functions	See also “Expression Evaluation Functions” on page 93.
File Functions and Methods (Core)	
EOF()	Returns .T. (TRUE) if the record pointer is at the end of the file for the table.
Database Functions	See also
Directory Functions	See also
Filename Functions	See also
List Processing Functions (Core)	
LINE_COUNT()	Returns the number of lines in a CR-LF delimited character string list.
List Processing Functions	See also “List Processing Functions” on page 57.
Mathematical Functions (Core)	
CEILING()	Returns the smallest integer that is greater than or equal to the specified *Number*.
FLOOR()	Returns the largest integer that is either less than or equal to the specified *Number*.
INT()	Truncates the decimal portion of the specified Number and returns the remaining integer.

Mathematical Functions	See also “Numeric & Logical Types” on page 79.
Query Functions and Methods (Core)	
CURRENT_FILTER_EXPN()	Returns the filter expression of the primary table in a session.
CURRENT_ORDER_EXPN()	Returns the order expression of the primary table in a session.
Query Functions and Methods	See also “Queries” on page 90.
Record Functions (Core)	
AVERAGE()	Returns the average value of the *Expression* evaluated for a group of records.
COUNT()	A summary function that returns the number of records in a group of records.
MAXIMUM()	Returns the maximum value of the *Expression* evaluated for a group of records.
MINIMUM()	Returns the minimum value of the *Expression* evaluated for a group of records.
MRECNO()	Returns the record number of the current composite record in a set.
RECCOUNT()	Returns the number of records in the specified table.
RECNO()	Returns the record number of the current record in the specified table.
RUN_COUNT()	Returns the running count of records of the *Expression* evaluated from the first record to the current record.
RUN_TOTAL()	Returns the total value of the *Expression* evaluated from the first record to the current record.
SCANNING()	Returns .T. (TRUE) if the current composite record buffer includes data from *Table_Name* ; otherwise, it returns .F. (FALSE).
TOTAL()	Returns the total value of the *Expression* evaluated for a group of records.
Record Functions	See also
Report Functions and Methods (Core)	
PAGECOUNT()	Returns the number of pages in a report.
PAGENO()	Returns the current page number of a printed layout.
Report Functions and Methods	See also “Reporting” on page 97..

Type Checking Functions (Core)	
ISALPHA()	Returns . T. (TRUE) if the first character of the specified *Character_String* is a letter; otherwise, returns .F. (FALSE).
ISDATE()	Returns .T. (TRUE) if *Date_String* is a valid date; otherwise it returns .F. (FALSE).
ISNUMBER()	Returns .T. (TRUE) if a character string contains only numbers; otherwise returns .F. (FALSE).
Type Checking Functions	See also “Type Checking Functions” on page 95.y

Array Functions & Methods

ARRAY FUNCTIONS & METHODS

The following functions and methods are available for manipulating Arrays. Prefix the following methods with the <ARRAY> pointer or the name of the array variable.

Method		Description
.APPEND_ARRAY()	V6	Appends *Array2* to *Array1* to produce *New_Array*.
.APPEND_ARRAYS_FILTERED()	V6	Appends selected elements from *Array2* to *Array1* to produce *New_Array*.
.CLEAR()	V5	Clears the contents of the array.
.COPY_FROM()	V6	Copies elements from a source array into a target array, overwriting existing elements and resizing the target array as necessary.
.DELETE()	V5	Deletes *Count* entries in an array starting at *Position*.
.DUMP_PROPERTIES()	V5	Creates a CR-LF delimited string of a property array's contents.
.DUMP()	V5	Creates a CR-LF delimited string of the array's contents.
.FIND()	V5	Finds an element in an array and returns its *Index* number.
.FIRST_EMPTY()	V5	Returns the index of the first NULL element in an array.
.GetDynamic()	V10	Returns the dynamic sizing state of an array
.INITIALIZE_FROM_TABLE()	V5	Loads field values from a table into a property array.
.INITIALIZE_PROPERTIES()	V5	Loads field values from a data string into a property array.

.INITIALIZE()	V5	Populates an array with values contained in a character *Data_String*.
.INSERT()	V5	Inserts *Count* entries in an array starting at *Position*.
.MATCHING()	V6	Returns a CR-LF delimited list of array elements matching a filter.
.MOVE()	V5	Moves an array element *From* the specified position specified *To* the specified position.
.PROPERTY_ORDER()	V6	Creates a list containing the sequence of reordered array elements.
.REORDER()	V6	Reorders an array based on an order list of the numbers of array elements.
.RESIZE()	V6	Increases the size of an array, without losing any of the existing data.
.SetDynamic()	V10	Controls dynamic sizing of an array
.SIZE()	V5	Returns size of an array.
.SORT()	V5	Sorts an array.
***ARRAY_APPEND()**	V5	Re-dimensions an array and assigns a value to the new array element.
***ARRAY_RANGE_ASSIGN()**	V6	Copies elements from one array to another.
***ARRAY_RANGED REFERNCE_TO_RANGE()**	V6	Takes a list of array variable definitions and returns a new list that can be used to properly DIM them.
<TBL>.POPULATE_GRID()	V6	Populates arrays quickly with data from a table.
A5_ADO_TO_ARRAY()	V6	Retrieves records from an ADO compliant database and populates an array.
ArrayType()	V6	Returns the data type of an array.
ISARRAY()	V6	Indicates whether an object is an array.
SORT_ARRAY()	V5	Sorts the specified character, numeric, or date array.
UI_GET_LIST_ARRAY()	V5	Displays a dialog box with a list box filled with the contents of a character array. Desktop only.

Collection Methods

COLLECTION METHODS

A collection is an Xbasic datatype, similar to an array, but whose elements are referenced with a "key" rather than an index.

- A key is a value (character, numeric or date) that uniquely identifies an element in a collection.

The following methods are available for the Collection Datatype.

- See also “StringDictionary” on page 120.

Function		Description
<COLLECTION>.DELETE()	V5	Deletes the collection element referenced by key.
<COLLECTION>.DUMP()	V5	Returns all of the data (key and data) in the specified collection in a CR-LF delimited string.
<COLLECTION>.ENUM_ALL()	V5	Returns all of the keys in the specified collection in a CR-LF delimited string.
<COLLECTION>.EXIST()	V5	Returns .T. if the specified key exists in the specified collection.
<COLLECTION>.FIRST()	V5	Returns the first key in a collection.
<COLLECTION>.FROM_TABLE()	V5	Populates a collection with data from a table.
<COLLECTION>.GET()	V5	Returns the data associated with a specified key.
<COLLECTION>.INITIALIZE()	V5	Populates a collection from a CR-LF delimited string.
<COLLECTION>.NEXT()	V5	Returns the key following the specified key.
<COLLECTION>.SET()	V5	Sets the data for the specified key in the specified collection.
<COLLECTION>.SIZE()	V5	Returns the number of entries in a collection.

Developer Functions & Methods

DEVELOPER FUNCTIONS

The following functions and methods will be useful to the Alpha Anywhere developer.

Method		Description
A5.GET_EXE_NAME()	V8	Returns the base name of the Alpha Anywhere executable.
A5.GET_MASTER_NAME()	V5	Returns the name of the master database if the workstation is shadowed.
A5.GET_NAME()	V5	Returns the filename of the current database.

A5_ASCII_TABLE()	V5	Displays an ASCII lookup table and returns a value. Desktop only.
A5_AXCONTROL_AVAILABLE()	V7	Determines whether an ActiveX control is installed and registered.
A5_Compile_Scripts()	V5	Displays the **Compile Script Library** dialog box.
A5_CREATE_SHORTCUT()	V5	Displays the **Create Shortcut Genie** dialog box. Desktop only.
A5_GetAppVersionNumber()	V6	Returns the version of the application.
A5_GET_EXPRESSION()	V7	Displays the **Expression Builder** and returns an expression.
A5_InstallAmyuni()	V6	Installs Alpha Anywhere printer drivers. Desktop only.
A5_IS_DATABASE_OPEN()	V6	Reports whether any database is open.
A5_IS_RUNTIME()	V5	Returns .T. if the Alpha Anywhere Runtime version is running.
A5_IS_RUNTIMEPLUS()	V7	Returns .T. if the Alpha Anywhere Runtime Plus version is running.
A5_IsHomeEdition()	V6	Determines if the current application is the **Alpha Anywhere Home Edition**.
A5_ISMAXIMIZED()	V5	Indicates whether the MDI child window is maximized.
A5_MDAC_VERSION()	V6	Returns the version number of the installed Microsoft Data Access Components.
A5_REGISTRATION_LICENSE_INUSE()	V6	Counts the number of concurrent users in a multi-user application.
A5_SetAppVersionNumber()	V6	Sets the application's version number.
A5_SHOW_VARIABLE()	V5	Displays a dialog box with tabs showing the values of each of the currently defined variables. Desktop only.
A5_SYSTEM_ADDIN_VERSION()	V5	Returns the version number of the installed addin functions.
A5_TempFileListFileName()	V6	Returns the name of the file that contains the temp file list.
COMPILE_LIBRARY()	V6	Compiles .ALB library files into .AEX format.
COMPILE_TEMPLATE()	V6	An alternative way to define and invoke functions.

FUNCTION_DESCRIPTION_GET()	V5	Returns the description of a function.
FUNCTION_LIMITATIONS_GET()	V5	Returns the limitations of the named function.
FUNCTION_PROTOTYPE_GET()	V5	Returns the prototype for the named function.
FUNCTIONS_GET()	V5	Returns a list of functions.
HOURGLASS_CURSOR()	V5	Turns on an hourglass cursor to indicate to the user that a task is being performed. Desktop only.
LOAD_LIBRARY()	V5	Loads the named library (.AEX file).
MOST_RECENTLY_USED_DATA-BASE_ENUM()	V5	Returns a CR-LF delimited list containing the fully qualified filenames of the eight different most recently used databases. Desktop only.
ON_XBASIC_IDLE()	V6	Registers code to run when Alpha Anywhere is idle.
PLAY_SOUND()	V5	Plays a sound file (.WAV extension). Desktop only.
PROFILER_BEGIN()	V5	Begins the profiling of a script.
PROFILER_END()	V5	Ends the profiling of a script.
USER_VARIABLES_ENUM()	V5	Returns a CR-LF delimited list of user variables in the variable frame referenced by *Pointer*.
VARIABLE_CONTEXT_NAME()	V5	Returns the name of the variable frame (if any) of the referenced variable.
VARIABLES_ENUM()	V5	Returns a CR-LF delimited string of variable names in the variable frame referenced by *Pointer*.
VERSION()	V5	Returns the version number and other information about Alpha Anywhere.
XBASIC_ERROR_LOG()	V6	Defines an error log for a thread's code.
XBASIC_TRACE_END()	V5	Terminates writing Xbasic debug information to a file.
XBASIC_TRACE_START()	V5	Starts writing Xbasic debug information to a file.

Chapter 11.

SQL Data Type

Desktop, Web, Mobile

Notes on this chapter

In most cases, SQL is used for web applications, but developers are also using it in Xdialog.

We have included the properties and methods for each of the SQL objects.

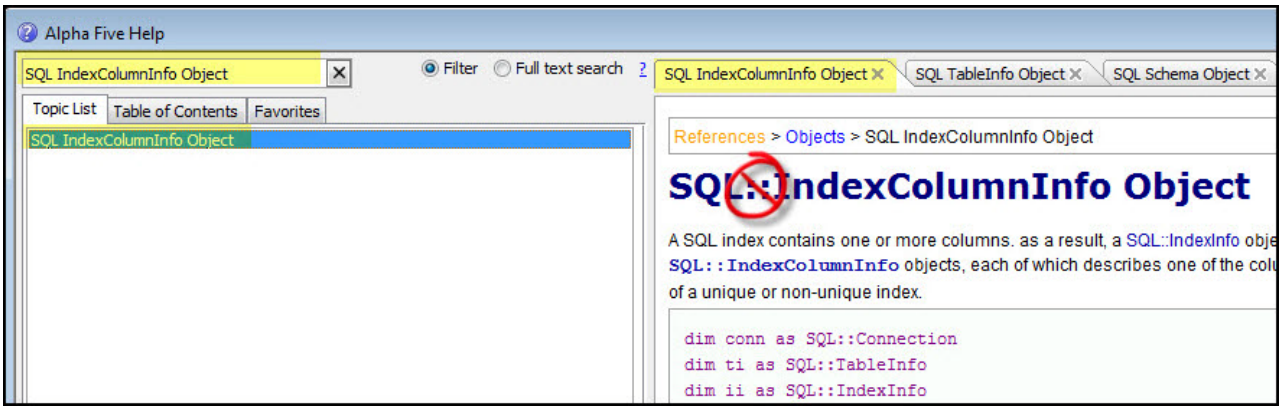
SYNTAX

We refer you to each individual page for a discussion of the syntax and sample scripts.

SEARCHING

As usual, the page title is in the left column. Note that the double colons are not used in Documentation page title.

In other words, if you are looking for the **SQL::IndexColumnInfo** object, drop the double colons and search on **SQL IndexColumnInfo** Object.



Portable SQL

PORTABLE SQL

Contrary to popular belief, SQL is not a standard syntax. Some terms:

- **SQL Syntax**: A set of rules describing how a SQL command is to be formatted.
- **Native SQL:** The SQL syntax specific to a particular vendor (" Oracle " syntax or " MySQL " syntax).
- **Portable SQL:** An Alpha supported SQL syntax intended to be translated into native SQL for execution.

To solve the problem of the lack of SQL standardization, the folks at Alpha Software developed Portable SQL.

- Portable SQL is a database-independent, standardized SQL syntax.

Using Portable SQL, the same syntax will work across all database platforms. Alpha Anywhere invisibly translates Portable SQL Syntax to Native SQL syntax.

GO TO A5 HELP

There are several Documentation pages on Portable SQL. To locate them, execute the following search.

- Documentation > Filter: Enter **portab**. Choose:
  - *Portable SQL*, *Portability Functions* (see below), *Supported Portable SQL Syntax*.

SQL Portability Functions

PORTABILITY FUNCTIONS

The set of functions below are deemed “Portability” functions.

Not all back-end databases support all of these functions. Known incompatibilities are noted. If you use a function that Portable SQL does not recognize, it will pass it through to the back-end database. You will get whatever error the database provides.

NOTE

We have adapted this page to fit our book format. Please see the Documentation page (*Portability Functions*) for more information on syntax.

Function	Description	Not Supported By
Abs()	Returns the absolute value of *Number*.	
Acos()	Returns the ArcCosine of *Number*.	Access, Ingres
AddDaysToDate()	Adds *Number* days to *Date_Expression* and returns a *Date*.	
AddDaysToDateTime()	Adds *Number* days to *Date_Time_Expression* and returns a *Date_Time*.	

AddHoursToDateTime()	Adds *Number* hours to *Date_Time_Expression* and returns a *Date_Time*.	
AddMinutesToDateTime()	Adds *Number* minutes to *Date_Time_Expression* and returns a *Date_Time*.	
AddMonthsToDate()	Adds *Number* months to *Date_Expression* and returns a *Date*.	
AddMonthsToDateTime()	Adds *Number* months to *Date_Time_Expression* and returns a *Date_Time*.	
AddSecondsToDateTime()	Adds *Number* seconds to *Date_Time_Expression* and returns a *Date_Time*.	
AddYearsToDate()	Adds *Number* years to *Date_Expression* and returns a *Date*.	
AddYearsToDateTime()	Adds *Number* years to *Date_Time_Expression* and returns a *Date_Time*.	
AllTrim()	Removes white space characters both before and after the *String*.	Firebird, Ingres
ASCII()	Returns the ASCII value of a *Character*.	Informix, Ingres
Asin()	Returns the ArcSine of *Number* in radians.	Access, Ingres
Atan()	Returns the ArcTangent of *Number* in radians.	Access
Atan2()	Returns the ArcTangent of *Number* in radians. **?????**	Access, Ingres
Avg()	Calculates the average value of all rows or all rows that satisfy *Filter_Expression*.	
Ceiling()	Returns the smallest integer greater than or equal to *Number*.	Informix, Ingres, Access
Character()	Returns character value of a numeric expression.	Informix, Ingres
Concatenate()	Concatenates the character strings and returns the result.	
Cos()	Returns the Cosine of *Number* in radians.	
CoT()	Returns the Cotangent of *Number* in radians.	Oracle, Informix, Excel, Access, Ingres
Count()	Calculates the count of all rows or all rows that satisfy *Filter_Expression*.	

CurrentDate()	Returns the current *Date* according to the SQL database server.	Firebird
CurrentDateTime()	Returns the current *Date_Time* according to the SQL database server.	
CurrentTime()	Returns the current *Time* according to the SQL database server.	Firebird
CurrentUser()	Returns the ID of the current user according to the SQL database server.	
Database()	Returns the current database name.	
Day()	Returns the day of the month.	
DayName()	Returns the day name of the week.	
DayOfWeek()	Returns the day number of the week.	
DayOfYear()	Returns the day number of the year.	
DaysBetweenDates()	Returns the number of days between *Date_Expression1* and *Date_Expression2*.	
DaysBetweenDateTimes()	Returns the number of days between *Date_Time_Expression1* and *Date_Time_Expression2*.	
Degrees()	Returns the degrees equivalent of *Radians*.	Oracle, Informix, Firebird, Access, Ingres
Exp()	Returns the number **e** raised to the power of *Number*.	Firebird
Floor()	Returns the largest integer less than or equal to *Number*.	Ingres, Informix
Hour()	Returns the hour component of *Date_Expression* or *Date_Time_Expression*.	
HoursBetweenDateTimes()	Returns the hours between *Date_Time_Expression1* and *Date_Time_Expression2*.	
If()	Returns *TrueValue* or *FalseValue* based on the evaluation of *Expression*.	
IfNull()	Returns *Replacement_Value* if *Expression* evaluates to NULL.	Firebird
Left()	Returns *Count* characters from the beginning of *String*.	Firebird, Oracle, OracleLite, PostgreSQL, Informix

Log()	Returns the natural logarithm of *Number*.	
Log10()	Returns the base 10 logarithm of *Number*.	Access, Ingres
Lower()	Changes all upper case characters to lower case and returns the result.	
LTrim()	Removes white space characters before the *String*.	Ingres, Excel
Max()	Calculates the maximum value of all rows or all rows that satisfy *Filter_Expression*.	
Min()	Calculates the minimum value of all rows or all rows that satisfy *Filter_Expression*.	
Minute()	Returns the minutes value of a *Date_Expression* or *Date_Time_Expression*.	
MinutesBetweenDateTimes( )	Returns the minutes between *Date_Time_Expression1* and *Date_Time_Expression2*.	
Mod()	Returns the integer remainder of a number divided by another	
Month()	Returns the month value of a *Date_Expression* or *Date_Time_Expression*.	
MonthName()	Returns the month name of a *Date_Expression* or *Date_Time_Expression*.	
MonthsBetweenDates()	Returns the months between *Date_Expression1* and *Date_Expression2*.	
MonthsBetweenDateTimes()	Returns the months between *Date_Time_Expression1* and *Date_Time_Expression2*.	
NextDay()	Returns the day after *Date_Time_Expression* as a *Date_Time_Value*.	
Now()	Returns the current time as a *Date_Time_Value*.	
Pi()	Returns an approximate value of **Pi**.	Oracle, DB2
Position()	Returns the position of *Find_String* within *Search_In_String*.	
Power()	Raises *Number* to the *Exponent* power.	Firebird
Quarter()	Returns the quarter of the year that contains *Date_Time_Expression*.	

Radians()	Converts a number in *Degrees* to radians.	Oracle, Informix, Firebird, Access, Ingres
Rand()	Returns a random value between 0 and 1.	Oracle, Informix
Replace()	Replaces all occurrences of *Search_For_String* with *Replace_String* in *Search_in_String*.	Ingres, firebird
Right()	Returns *Count* characters from the end of *String*.	Firebird, Oracle, OracleLite, PostgreSQL, Informix
Round()	Rounds off a *Number* to *Places* decimal places.	Firebird, Ingres
RTrim()	Removes white space characters after the *String*.	
Second()	Returns the seconds value of a *Date_Expression* or *Date_Time_Expression*.	
SecondsBetweenDateTimes()	Returns the seconds between *Date_Time_Expression1* and *Date_Time_Expression2*.	
SessionUser()	Returns the name of the session user according to the SQL database server.	
Sign()	Returns 1 if *Number* is positive, 0 if zero, and -1 if negative.	Informix, Ingres
Sin()	Returns the Sine of *Number* in radians.	
Square()	Returns the square of *Number*.	
SquareRoot()	Returns the square root of *Number*.	
StdDev()	Calculates the standard deviation of all rows or all rows that satisfy *Filter_Expression*.	
StringLength()	Returns the length of *String*.	
StringPad()	Returns a string right padded with *PadString* to a length of *Length* characters.	
StringPadLeft()	Returns a string left padded with *PadString* to a length of *Length* characters.	
SubString()	Returns the portion of *String* starting at *Position* and extending *Length* characters.	
Sum()	Returns the sum of all rows or all rows that satisfy *Filter_Expression*.	
SystemUser()	Returns the name of the system user according to the SQL database server.	

Tan()	Returns the Tangent of *Number* in radians.	
Truncate()	Returns *Number* truncated to *Number2* decimal places.	
Upper()	Changes all lower case characters to upper case and returns the result.	
Variance()	Returns the variance of all rows or all rows that satisfy *Filter_Expression*.	
Week()	Returns the week number of a *Date_Expression* or *Date_Time_Expression*.	
Year()	Returns the year of a *Date_Expression* or *Date_Time_Expression*.	
YearsBetweenDates()	Returns the years between *Date_Expression1* and *Date_Expression2.*	
YearsBetweenDateTimes()	Returns the years between *Date_Time_Expression1* and *Date_Time_Expression2.*	

Portable SQL Functions for Geographies

PORTABLE SQL FUNCTIONS FOR GEOGRAPHIES V11

These functions are only available from Alpha Anywhere when Portable SQL is used. They are intended to insulate you from the differences between the native SQL geography implementations in the supported databases.

All of the examples assume that you will be substituting the correct Spatial-ReferenceID value for the database currently in use for the argument :SRID.

Function	Description
GeogAsBinary()	Return the object description in the Well Known Binary (WKB) format.
GeogAsText()	Return the object description in the Well Known Text (WKT) format.
GeogCreateFromBinary()	Create a geography object from Well Known Binary (WKB) format.
GeogCreateFromText()	Create a geography object from Well Known Text (WKT) format.
GeogCreateLine()	Constructs a geographic line from two or more longitude/ latitude pairs.
GeogCreateLocation()	Constructs a geographic location from a longitude/latitude pair.

GeogCreatePolygon()	Constructs a geographic polygon from three or more longitude/latitude pairs.
GeogDistanceBetween()	Returns the distance between two objects in the default unit, typically meters.
GeogLatitude()	Returns the latitude value for a geographic location.
GeogLocationIntersectsLine()	Returns true if the location intersects the line or is within the tolerance distance from it.
GeogLocationIsWithinPolygon()	Returns true if the location is contained within the polygon or within the tolerance distance from it.
GeogLocationIsWithinRadius()	Returns true if the location is within the radius defined around the point or within the tolerance defined.
GeogLongitude()	Returns the longitude value for a geographic location.
GeogSRID()	Returns the Spatial Reference Identifier (SRID) assigned to the object.
GeogType()	Returns LOCATION, LINE, or POLYGON if the geography object is one of them.

SQL Basics

Following is a basic list of SQL objects and functions. The full list follows.

SQL Objects

ALPHADAO OBJECTS

AlphaDAO provides the following SQL objects.

- Each object has properties and some also have methods.

Object		Description
SQL::Argument •See “SQL Argument Object” on page 186.	V8	This object defines a single argument to be passed to a SQL function in a **`SQL::Arguments`** collection.
SQL::Arguments •See “SQL Arguments Object” on page 187.	V8	A collection of arguments to be passed to a SQL function.
SQL::CallResult •See “SQL CallResult Object” on page 188.	V8	This object provides a more complete description of a function call on an object in the SQL name space.

SQL::Connection •See "SQL Connection Object Functions" on page 190.	V8	Each instance of **SQL::Connection** represents a potential or active database session. •See "SQL Argument Object" on page 186.
SQL::DataTypeInfo •See "SQL DataTypeInfo Object" on page 196.	V8	A single column in a table or result set. The description includes the name, size, precision, Alpha Anywhere type, the native type, and an intermediate type (see SQL::IntermediateType)* which contains more precise type information. This intermediate type makes it possible to recreate a table using the closest possible matching type in the target table.
SQL::IndexColumnInfo •See "SQL IndexColumnInfo Properties" on page 201.	V8	A description of one of the columns in an index. A SQL::IndexInfo contains one or more columns that make up the primary key, the foreign key, of a unique or non-unique index.
SQL::IndexInfo •See "SQL IndexInfo Object" on page 202.	V8	A description of an index on a SQL table. Each table in a SQL database has one or more relationships, constraints or indexes. The primary key, foreign keys, and unique and non-unique indexes are each described by a SQL::IndexInfo instance. The SQL::TableInfo object contains a list of those indexes.
SQL::ResultSet •See "SQL ResultSet Object" on page 203.	V8	A collection of rows returned from the execution of a query. When a SQL query returns one or more rows of data, you can use the SQL::ResultSet object to get the description of the result set and to navigate within the rows of the results to retrieve the data.
SQL::Row •See "SQL Row Object" on page 205.	V8	The data returned by the SQL::ResultSet.NextRow() method.
SQL::Schema •See "SQL Schema Object" on page 206.	V8	A collection of SQL table definitions. SQL::Schema is a container for one or more SQL::TableInfo objects.
SQL::TableInfo •See "SQL TableInfo Object" on page 208.	V8	A fairly complete description of a SQL table, its columns, and its indexes. Many of the SQL functions either ask you to provide a SQL::TableInfo object, or will create one for you. Using this object, it is possible to retrieve a description of a table from one database and construct a copy of that table in the same or another database.

*. See "SQL Enumerated Types" on page 211.