SMImport suite

The SMImport suite for Delphi/C++Builder allows to import the data into linked recordset (any TDataset descenders) from different sources:

1. MS Excel spreadsheets/workbooks (including MS Office12 files)
2. Text file (fixed length)
3. CSV (comma delimited)
4. HTML file
5. XML file
6. Lotus 1-2-3 spreadsheet
7. QuattroPro file
8. DBase/FoxPro table
9. MS Access database
10. MS Word document
11. Paradox table
12. any ADO connection
13. Advantage table
14. any BDE source
15. DBISAM table
16. Clarion table
17. from MS Windows clipboard
18. from Windows Adress Book
19 VCalendar/ICalendar
20. another dataset.

Also SMImport suite contains the compound visual dialog for easier defining all settings for import process. Using this visual wizard you may provide the fast and powerful process of data loading where end-user may setup/reuse any import parameters.

Visit the web site at http://www.scalabium.com to see if updated information is available.

Visual Expression Builder

This is a visual dialog where you may build the expression in easy way. Just open the any group of commands and select the item from context list:

Comparision/Logical operators

Valid relational operators are:

= equal
<> not equal
> greater than, 7>3 will return true
>= greater than or equal
< less than, 1<4 will return true
<= less than or equal

AND (A and B) is True only if both A and B are True
NOT not True will return the False
not False will return the True
OR (A or B) is True only if either A is True or B is True

Math/Arithmetic operators

The basic mathematical operators that you will use very often:

11+3 adds 3 to 11 and returns 14
17-5 subtracts 5 from 17 and returns 12
9*5 multiplies 9 by 5 and returns 45
12/3 divides 12 by 3 and returns 4
2^3 raises 2 to 3 and returns 8

Constants

Constants can be: integer, floating point, string and date.

An Integer must begin with ['0'..'9']. For example, 16782

The floating point like 32.81e-4 or 12.8
Floating point values may not contain thousand seperators and are internally converted to double precision floating point values.

The string must begin and end with quote ("). For example, "test string"

The date format is the same as your computers locale settings.

String functions

The next text/string functions supported:

Length returns the number of characters in a string or elements in an array.
function Length(S: **string**): **Integer**;
For example, Length('one')=3

Lower converts an ASCII string to lowercase.
function LowerCase(S: **string**): **string**;
For example, Lower("aNy TExt")="any text"

Upper returns a copy of a string in uppercase.
function UpperCase(S: **string**): **string**;
For example, Lower("aNy TExt")="ANY TEXT"

PadL returns a string, padded with spaces or characters to a specified length on the left side. If last character is not specified, space character used
function PadL(S: **string;** Count: **Integer;** CharToAdd: **Char**): **string**;
For example, PadL("mystr", 8, "1")="111mystr"

PadR returns a string, padded with spaces or characters to a specified length on the right side. If last character is not specified, space character used
function PadR(S: **string;** Count: **Integer;** CharToAdd: **Char**): **string**;
For example, PadR("mystr", 8, "1")="mystr111"

Pos returns the index value of the first character in a specified substring that occurs in a given string.
function Pos(Substr: **string**; S: **string**): **Integer**;
For example, Pos("tex", "any text")=5

ReplaceStr returns a string with occurrences of one substring replaced by another substring
function ReplaceStr(S: **string;** oldS: **string;** newS: **string;** IgnoreCase: **Boolean;** ReplaceAll: **Boolean**): **string**;
For example, ReplaceStr("my1str1", "1", "2", True, True)="my2str2"

SubString returns a substring of a string.
function SubString(S: **string**; Index, Count: **Integer**): **string**;
For example, SubString("any text", 1, 3)="any"

Trim function trims leading and trailing spaces and control characters from a string.
function Trim(S: **string**): **string**;

TrimLeft trims leading spaces and control characters from a string.
function TrimLeft(S: **string**): **string**;

TrimRight trims trailing spaces and control characters from a string.
function TrimRight(S: **string**): **string**;

Math functions

The parser supports the next fundtions:

Abs returns an absolute value.
function Abs(X: **Number**): **Number**;
For example, abs(-2) returns 2

Frac returns the fractional part of a real number.
function Frac(X: **Number**): **Number**;
For example, Frac(123.456) = 0.456

Int returns the integer part of a real number.
function Int(X: **Number**): **Number**;
For example, Int(123.456)=123.0

Random generates random numbers within a specified range.
function Random [(Range: **Integer**)];
If Range is not specified, the result is a real-type random number within the range 0 <= X < 1.

Round function rounds a real-type value to an integer-type value.
function Round(X: **Number**): **Integer**;
Returns the value of X rounded to the nearest whole number.

SQR returns the square of a number.
function Sqr(X: **Number**): **Number**;

SQRT returns the square root of X.
function Sqrt(X: **Number**): **Number**;

Exp returns the value of e raised to the power of X, where e is the base of the natural logarithms.
function Exp(X: **Number**): **Number**;

Ln returns the natural log of a real expression (Ln(e) = 1)
function Ln(X: **Number**): **Number**;

Log returns the log of a real expression.
function Log(X: **Number**): **Number**;
Note that Log(X)=Ln(X)/Ln(10)

Power is a same that ^-operator.
function Power(X, Y: **Number**): **Number**;
For example, 2^3 raises 2 to 3 and returns 8

Div returns the value of x/y rounded in the direction of zero to the nearest integer.
function Div(X, Y: **Number**): **Integer**;

Mod returns the remainder obtained by dividing its operands.
function Mod(X, Y: **Number**): **Integer**;
In other words, x mod y = x - (x div y) * y.

Percent returns the calculated percent from value (45%).
function Percent(X, Y: **Number**): **Integer**;
For example, Percent(130, 200) returns 65

Note that the runtime error occurs when y is zero in an expression of the form x/y, x div y, or x mod y.

Date/Time functions

The following date/time functions are supported:

GetDate returns the current date and time.
function GetDate: **TDateTime**;

Year returns the year from date value.
function Year(D: **TDateTime**): **Integer**;

Month returns the month from date value.
function Month(D: **TDateTime**): **Integer**;

Day returns the day from date value.
function Day(D: **TDateTime**): **Integer**;

Hour returns the hours from time value.
function Hour(D: **TDateTime**): **Integer**;

Minute returns the minutes from time value.
function Minute(D: **TDateTime**): **Integer**;

Second returns the seconds from time value.
function Second(D: **TDateTime**): **Integer**;

DatePart returns the date part (without time) from date-time value.
function DatePart(D: **TDateTime**): **TDateTime**;

TimePart returns the time part (without date) from date-time value
function TimePart(D: **TDateTime**): **TDateTime**;

DateTimeFromString convert the string to date-time value by specified format.
function DateTimeFromString(S, Format: **string**; DateSeparator, TimeSeparator: **Char**): **TDateTime**;
For example, DateTimeFromString('12-Jan-1999 03:04:05', 'dd-mmm-yyyy hh:nn:ss', '-', ':')

Aggregate functions

AVG returns the average value of the elements in an array.
function AVG(X, Y [, ...]: **Number**): **Number**;
For example, AVG(2, 4, 9)=5

SUM returns the sum of the elements in an array.
function SUM(X, Y [, ...]: **Number**): **Number**;
For example, SUM(2, 4, 9)=15

COUNT returns the number of parameters in an array.
function COUNT(X, Y [, ...]: **Number**): **Number**;
For example, COUNT(2, 4, 9)=3

MAX returns the greater of numeric values.
function MAX(X, Y [, ...]: **Number**): **Number**;
For example, MAX(2, 4, 9)=9

MIN returns the lesser of numeric values.
function MIN(X, Y [, ...]: **Number**): **Number**;
For example, MIN(2, 4, 9)=2

Conversion

Parser has several build-in conversion routines:

CHR returns the character by code.
function Chr(X: **Integer**): **Char**;
For example, CHR(65)='A'

ORD returns the code of character.
function Ord(X: **Char**): **Integer**;
For example, ORD('A')=65

IntToStr converts an integer to a string.
function IntToStr(Value: **Integer**): **string**;

StrToInt converts a string representing an integer (decimal or hex notation) to a number.
function StrToInt(Value: **string**): **Integer**;

FloatToStr converts a floating point value to a string.
function FloatToStr(Value: **Number**): **string**;

StrToFloat converts given string to a floating-point value.
function StrToFloat(S: **string**): **Number**;

DateToStr converts a variable of type TDateTime to a formatted string
function DateToStr(Date: **TDateTime**): **string**;

StrToDate converts a string to a date format.
function StrToDate(S: **string**): **TDateTime**;

Program flow functions

Every script language needs some sort of conditional braching.

IIF returns one of two values depending on the value of a logical expression.
function IIF(Condition: Boolean; X, Y: **Variant**): **Variant**;
For example, IIF(10>4, "one", "two") will return the "one"

IsNULL returns the True if parameter is not defined or empty
function IsNull(X): **Boolean**;

Data sources

You may select the field name from list for every available data source. There are listed both target and source datasets/queries/files...

Note that field name contains the space character or is a system function, you must quote such field name. For example, "Customer ID" or "MONTH"

Data source functions

Parser support a few functions for data processing:

Substitute allows to change the parsed value from list.
function Substitute(CodeValue, CodeList, ValueList: **string**): **string**;
For example, to post the 1 instead Male and 2 instead Female:
SEX=SUBSTITUTE('Male', 'Male;Female', '1;2')

Lookup allows to locate the parsed value in another dataset and post in target dataset the value of another field value if required record found.
function Lookup(DatasetName, CodeFieldName, CodeValue, NameFieldName: **string**): **string**;

For example:
STATE=LOOKUP(qryStates, 'CODE', 'IL', 'NAME') {will return the "Illinois"}