SQLSCALAR - *Calling SQL Functions from RPG*

Using an SQL Scalar Function can be a lot easier than the RPG alternative:-

* Use the UPPER function instead of defining two translation strings and using %XLATE
* Use DAYNAME function instead of calculating the day of the week and using an array to provide the day name.
* Using the REGEXP\_ functions as opposed to calling the regular expression APIs
* Using the ENCRYPT/DECRYPT functions instead of calling APIs
* Using the SOUNDEX function instead of - well, there is no alternative.

The SQLSCALAR service program contains subprocedure wrappers for the SQL scalar functions ABS, ACOS, ANTILOG, ASCII, ASIN, ATAN, ATAN2, ATANH, BIT\_LENGTH, CEIL, CHR, COALESCE, COS, COSH, COT, DATABASE, DAYNAME, DAYOFWEEK, DAYOFWEEK\_ISO, DAYOFYEAR, DECRYPT\_CHAR, DEGREES, DIFFERENCE, ENCRYPT\_AES, ENCRYPT\_RC2, ENCRYPT\_TDES, EXP, FLOOR, GETHINT, HEX, IFNULL, JULIAN\_DAY, LAND, LAST\_DAY, LCASE, LEFT, LN, LNOT, LOG10, LOR, LOWER, LPAD, MAX, MIDNIGHT\_SECONDS, MIN, MONTHNAME, NEXT\_DAY, PI, QUARTER, RADIANS, RAND, REGEXP\_COUNT, REGEXP\_INSTR, REGEXP\_REPLACE, REGEXP\_SUBSTR, REPLACE, RIGHT, ROUND\_TIMESTAMP, RPAD, SIN, SINH, SOUNDEX, TAN, TANH, TRUNC\_TIMESTAMP, TRUNCATE, UCASE, UPPER, VALUE, WEEK, WEEK\_ISO and XOR. Subprocedures are not provided for SQL scalar functions that have a direct RPG equivalent (e.g. SUBSTR) or that only make sense when used will multiple rows (e.g. the XML functions).

Installation

The SQLSCALAR library was created and save on a system at V7R1.

The first step is to install the SQLSCALAR library.

* Create a save file on the system
* FTP the downloaded SQLSCALAR.SAVF file to the save file (make sure you use binary mode)
* Restore the library SQLSCALAR from the save file

Using SQL Scalar Functions in RPG

Using an SQL scalar function in embedded SQL is very straight forward. There are two ways it can be done. Using VALUES INTO

exec SQL

values upper(:textOut) into :textOut;

or using SET

exec SQL

set :textOut = upper(:textOut);

My personal preference is to use VALUES INTO simply because SET can be used in other ways e.g. using SET OPTIONS to set the SQL run time environment.

Wrapping an Scalar Function in a Subprocedure

Like any BIF in RPG, our wrapper subprocedures will need to return a single. This is a wrapper subprocedure for the UPPER SQL Scalar Function:-

dcl-Proc rSQL\_upper export;

dcl-Pi \*n varchar(32000);

textIn varchar(32000) const;

end-Pi;

dcl-S textOut varchar(32000);

textOut = %trimR(textIn);

exec SQL

values upper(:textOut) into :textOut;

setCodes();

return %trimR(textOut);

end-Proc;

The subprocedure accepts a VARCHAR filed of up to 32000 characters in length and returns a VARCHAR field of up to 32000 characters in length. The use of the CONST keyword on the input parameter means that a character expression or field may also be passed as a parameter (as opposed to only VARCHAR).

Using the Subprocedures

To make use of the subprocedures, you need to include the source member SQLSCALAR/SQLSCALAR,PSQLSCALE1 in your program.

You will also need to specify the SQLSCALAR binding directory (in the control-spec or on the BNDDIR parameter) when you are creating your program. Alternatively, just add an entry for the SQLSCALAR service program to a binding directory you are already using.

PSQLSCALE1 contains the prototypes for the wrapper subprocedures. The subprocedures are names rSQL\_ followed by the name of the SQL Scalar Function.

The Test Program

You can call the program TESTSCALE to see how a scalar function works. The program will prompt for the name of a function (just the SQL name - no rSQL\_ prefix) and will then show an example of the input parameters and returned value for the subprocedure call. It will also show the values of SQLCODE and SQLSTATE - in case there was a problem!

This would be the result of calling TESTSCALE and specifying UPPER as the function:-

DSPLY Select Function:

upper

DSPLY Parameter is this is lower

DSPLY Result is THIS IS LOWER

DSPLY SQL Code is 0

DSPLY SQL State is 00000

To see an example of calling one of the subprocedures - just browse the source of TESTSCALE for the name of the Scalar Function. This is the code used to call the UPPER function:-

elseIf (function = 'UPPER');

dsply ('Parameter is ' + %trim(lowercase));

showIt = ('Result is ' + %trim(rSQL\_upper(lowercase)));

dsply showIt;

Parameters

RPG is much more stringent about the definition of variables than SQL. In all cases, I had to chose an arbitrary maximum length for VARCHAR variables (which varies, depending on the function, but usually between 500 and 32000) and the length and decimal precision for numbers (usually 30:16). It should not be too much of a concern when these definitions relate to input parameters: since all input parameters are defined with the CONST keyword, you can provide any character or numeric variable you wish. You need only be concerned if the definition is not large enough for the required parameter or returned value.

Some of the procedures will have optional parameters e.g. the rSQL\_coalesce() subprocedure can accept two to ten parameters.

For functions that deal specifically with null values (e.g. rSQL\_coalesce()), a empty value is considered null.

Check for Errors

Two extra subprocedures (rSQL\_SQLCode() and rSQL\_SQLState()) can be called, after a call to one of the wrapper subprocedures, and will return the value of SQLCODE or SQLSTATE that was set by the underlying VALUES INTO statement.

Extra Subprocedures

There are a few other extra subprocedures to be aware of, as well as rSQL\_SQLCode() and rSQL\_SQLState().

* There are two versions of the LAST\_DAY function. rSQL\_last\_Day() will return a date and rSQL\_last\_Day\_Stamp() will return a timestamp.
* The subprocedure rSQL\_set\_Encryption\_Password() can be used to set a default password (and optional hint), as opposed to passing an optional password (and optional hint) on calls to rSQL\_encrypt\_AES(), rSQL\_encrypt\_RC2(), rSQL\_encrypt\_TDES() and rSQL\_decrypt\_CHAR().
* The subprocedure rSQL\_full\_Text\_Date() will return a full text description of a date - as follows:-

DSPLY Select Function:

full\_text\_date

DSPLY Parameter is 2016-09-07

DSPLY = Wednesday, September 7, 2016

DSPLY SQL Code is 0

DSPLY SQL State is 00000

Roll Your Own

I hope this service program gives you a starting point to building your own SQL Scalar Function utilities.