Paper 50

SQL Query Window Classes: How To Incorporate the SQL Column Expression Builder Class In Custom SAS/AF® Applications

Christopher A. Roper, Qualex Consulting Services, Inc. Fairfax, Va.

Abstract

SQL (Structured Query Language) is the standard language used for querying all major databases, including SAP/R3, SYBASE, ORACLE, RED BRICK, DB2, and others. The SAS Query window creates SQL expressions using SAS/AF Classes that are available to the SAS/AF developer. One of these class entries is the Column Expression Builder Class, QWCLEXPR.CLASS in the SASHELP.SQL catalog. This is the class used by the Query window to build columns in SQL. This class may be called by any SAS/AF entry to present the same interface the SAS Query window uses as a column expression builder. This paper will cover what the Column Expression Builder Class is, how to incorporate it into a SAS/AF application, and the nuances of using this class effectively.

Introduction

The SQL Column Expression Builder class is essentially a frame called by the "Build A Column" pushbutton in the lower center portion of the SAS Query Columns window. Many users of the SAS System have used this feature of the Query window, and are very comfortable with it. It has a clear, simple interface that is easy to use and well thought out. Little is to be gained by taking the time to develop a different SQL column expression builder for a custom AF application. This makes it an obvious choice for the AF developer to include in a custom application. For the remainder of this paper, this class will be referenced by the name QWCLEXPR.CLASS.

Default Settings Used For Examples

The examples shown in this paper will use the following defaults. The data set used is SASHELP.RETAIL, available to anyone licensing Base SAS. All other settings and options will be left at the default settings. An example column expression will be built using the name "RET100", the label "Retail Sales/100", and will equal the SALES column from the data set SASHELP.RETAIL divided by 100. Note that the SCL list id values are set upon execution of the methods and code, and may differ from those shown in the following examples and figures in this paper.

Methods of the SQL Column Expression Builder Class (QWCLEXPR.CLASS)

The SQL Column Expression Builder class (QWCLEXPR.CLASS) provides seven methods: EDIT_ADD_EXPR, GET_TEXT, GET_TEXT_LIST, GET_VALUE_LIST, SUBMIT_EXPR, GET_ATTRIB, and CLEAN_UP. These methods provide access to the lists that control the behavior of this class. A brief description of each of these methods follows, plus full descriptions provided by the SAS online help facility are available by selecting the Index tab from the Help Topics: SAS System Help window, and typing QWCLEXPR.

The EDIT_ADD_EXPR method makes an SCL list of all the columns available for creating a new column. It also opens the frame interface, or window, used by the end user to select these columns and define the new column. Once a new column has been defined, this method adds the new column to the master list for the QWCLEXPR class, so that the new column may be used to create other new columns.

The GET_TEXT method uses a text reference array to contain the text string that is the new column definition. See the section Using The GET_TEXT method for a fuller description on using this method.

The GET_TEXT_LIST method is very similar to the GET_TEXT method, except that the results are returned to an SCL list, and thus the 200 character limit does not apply. Both the GET_TEXT and GET_TEXT_LIST methods optionally return a text string reporting the status of the text string built in the BUILD A COLUMN EXPRESION window.

The GET_VALUE_LIST method is a shorthand version of the GET_TEXT_LIST method. This method returns just the text string of the new expression, without the format, label, and expression name returned by the GET_TEXT_LIST method.

The SUBMIT_EXPR method sends the new column expression to the SAS supervisor for processing.

The GET_ATTRIB method returns information about the new column expression. This method will return the new column type (character or numeric), the format applied to the new column (if any), and the label assigned to the new column (if any).

The CLEAN_UP method performs the housekeeping chores for the QWWHERE.CLASS. It will recursively delete the SCL lists used by the QWCLEXPR.CLASS, thus freeing up system resources used by the execution of this class. This method can be used to clean up individual new column expressions, or it can be used to clean up all of the new column expression by using the keyword _ALL_ for the second argument.

Using the QWCLEXPR.CLASS

The QWCLEXPR.CLASS can be made available to an application by loading the class, and then instantiating it. These two steps are often nested into one SCL statement in the following manner:

```
QW_ID=instance(loadclass('SASHELP.SQL.QWCLEXPR.CLA
SS'));
```

Where QW_ID is the identifier of this instance of the QWCLEXPR.CLASS. Now, a CALL SEND statement may reference this identifier to execute any of the methods available to this class.

Using The EDIT_ADD_EXPR Method

The EDIT_ADD_EXPR method performs two functions. It builds a list of the available columns for creating a new expression, plus it displays the Build A Column Expression window. This is the interface the user will use to create an SQL expression creating a new column. This method accepts six arguments, and the syntax is:

```
CALL SEND
(QW_ID,'EDIT_ADD_EXPR',dsname,exprmast,exprname,exprla
bel,exprformt,expmsg);
```

The only required argument is the second, exprmast. The rest are optional. If dsname is omitted, then a valid value for exprname must be provided for the Build A Column Expression window to be presented. A valid value would be the name of an existing new column expression.

DSNAME is the name of the data set or data sets used to supply the columns from which the new column expression is built. If one data set name is provided, the list of available columns will show the one level name of the columns, i.e. SALES. If two or more data set names are provided, then the list of available columns will show the two level name of the columns, i.e. RETAIL.SALES.

EXPRMAST is the master list for the QWCLEXPR. CLASS. It contains a sublist for each new column expression. Each of these lists are assigned the name of the column they represent, and they contain all the sublists and list items used by the QWCLEXPR. CLASS. See Figure 1 for a simple example of EXPRMAST containing one new column expression. item. If no label exists for the column, then the column name is used. See Figure 2 for an example.

EXPRNAME is the name of the expression, or the name to be assigned to the new column. If more than one expression is to be built, it is useful to provide a default name here, and then change it in the Build A Column Expression window using the column attributes. This will allow multiple new column expressions to be created in one session.

EXPRLABEL is the label to be assigned to the expression. Usually this is left blank, so that the label may be assigned using the column attributes in the Build A Column Expression window.

EXPRFORMT is the format for the expression. It may be any valid SAS format for the expression. Like the EXPRLABEL, it is usually left blank, to be assigned using the column attributes in the Build A Column Expression window.

EXPRMSG is a text string returned by the EDIT_ADD_EXPR method which returns the status of the method call.

Using The GET_TEXT Method

The GET_TEXT method uses an array to contain the text string that is the new column definition. This text string is limited to 200 characters for each element of the array. To use this method, a reference array must be initialized with an explicitly defined dimension , and the length set to 200. It is important to understand how a reference array works with the internal array for the GET_TEXT method. A reference array does not use system resources like a normal array. It is simply a pointer to an array, in this case an array internal to the GET_TEXT method. Because of this, an asterisk (*) can not be used to define the dimension for the reference array, the dimension must be explicitly set. Additionally, the internal array of the GET_TEXT method is defined with a length of 200 for the elements of the array. If the reference array is defined with a length shorter than 200, then each element will be truncated as the internal array tries to place 200 characters into the shorter element of the reference array.

Saving a Column Expression

Since the QWCLEXPR.CLASS uses SCL lists to manipulate and display column expressions, a column expression built by the QWCLEXPR.CLASS may be saved just like any other SCL list. In the case of the column expressions, the master list is saved using the SCL SAVELIST function. In practice, a scheme must be devised to create unique names or directories for each saved master list, so that previously saved column expression master lists are not over written. There are any number of valid naming schemes, and it is up to the developer to determine one that will work in any given situation. (See Example 1: SCL, section SAVE:).

Loading an Existing Column Expression

Loading the QWCLEXPR.CLASS with existing lists built and/or modified in a previous session is simply the opposite of the save technique. By using the SCL function FILLIST, the master list can be recreated. Then the column expression master list may be passed to the QWWHERE.CLASS, so that they can be reviewed or modified. Note, it is not necessary to provide the name(s) of the data set(s) used when the expression was built. This information is contained in the column expression master list, and therefore does not need to be provided. However, if a data set name is passed, then only the columns from the passed data set are available in the Build A Column Expression window.

Additionally, if the master list contains more than one column expression, the name of the column expression will need to be passed so the proper column expression can be presented to the user. This allows the end user to proceed with the building of a column expression just as if the end user had created it from scratch. This can be especially useful for building a library of standard queries that can be modified by an end user to meet his/her specific needs at a later date.

In practice, an interface is needed by the end user to select from the library of stored queries. The discussion of developing this interface is beyond the scope of this paper.

Viewing the Results of the Where Clause Builder

When the SQL statements are created in a SUBMIT block of SCL code, they are stored in the SAS Preview buffer. This buffer is available to be viewed until the buffer is cleared by either a direct CLEAR command, or by sending the code to the SAS supervisor through a SUBMIT CONTINUE block. Therefore, the SQL statements and where clause may be viewed through the use of the SCL Preview command, which grants access to the SAS preview buffer. (See Example 1: SCL, section VIEW:)

A Simple Example

Example 1 is a simple example of the use of the QWCLEXPR.CLASS in a Frame entry. The Frame entry has four command push buttons, Build New Column, Save Column Expr, Load Column Expr, and View Column Expr. The SCL entry has the labeled sections NEW, SAVE, LOAD, and VIEW respectively. This example is far to simple to be robust enough for a practical application, it is intended to simply show the basics of how to use the QWCLEXPR.CLASS.

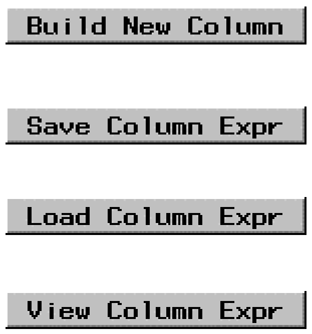
Conclusion

SAS Institute has spend a great deal of effort in creating very elegant object classes to support it’s various modules. SASHELP.SQL.QWCLEXPR. CLASS is just one of these classes, and it is a great advantage for the SAS developer to be able to use this class. Using this existing class greatly reduces the amount of time necessary to develop an application needing this functionality. Testing is much simpler since SAS has already performed a great deal of testing on this class. And finally, the end users have likely been using this class for some time and thus end user acceptance of the application is made much easier since they are already familiar with this class.

Author Contact Information

Christopher A. Roper
Qualex Consulting Services, Inc.
Rt. 2 Box 224C
Hillsville, Va. 24343
Chris.Roper@QLX.COM
http://www.qlx.com

Example 1: Frame



Example 1: SCL

```
array textarray {5} $200
      ;

INIT:
 /* Prevent compile time warnings */
 rc = rc;

 /* Load and instantiate the COLUMN EXPRESSION BUILDER
class */
 qw_id =
instance(loadclass('SASHELP.SQL.QWCLEXPR.CLASS'));

 /* Create the master list used by the QWCLEXPR.CLASS */
 exprmast = makelist();

 /* Create the sublist used by the methods GET_TEXT_LIST
and GET_VALUE_LIST */
 stringlst = makelist();

 /* Save indention of PROC SQL step in Preview window */
 control asis;

return; /* INIT */


NEW:

 /* Start with a default name for the expression */
 exprname = 'Default';

 /* Invoke the Column Expression Builder */
 call send(qw_id,'EDIT_ADD_EXPR','SASHELP.RETAIL',
exprmast,exprname);

return;  /* NEW */


LOAD:

 /* Start with a clean list */
 rc = clearlist(exprmast,'Y');

 /* Repopulate the master list EXPRMAST */
 rc =
fillist('CATALOG','Q.QWCLEXPR.QUERY.SLIST',
exprmast);

 /* Pass the rebuilt lists to the Column
Expression Builder */
 call
send(qw_id,'EDIT_ADD_EXPR','',exprmast);

return;  /* LOAD */

SAVE:

 /* Save the master list to a SAS catalog
*/
 rc =
savelist('CATALOG','Q.QWCLEXPR.QUERY.SLIST'
, exprmast);
return;  /* SAVE */
VIEW:

 /* Build the SQL in a submit block */
 submit;

 proc sql;

    create  table NEW as
    select
 endsubmit;

 /* Create a list to contain the built
column expression */
 if not stringlst then stringlst =
makelist();

 /* Retrieve the build column list */
 call
send(QW_ID,'GET_TEXT_LIST',exprmast,stringl
st,exprname);

 /* The column expression must be parsed
from STRINGLST   */

 colexp = getitemc(stringlst,1);

 submit;
           &colexp
 endsubmit;

 /* Complete the Proc SQL step */
 submit;
    from    SASHELP.RETAIL ;

 quit;
 endsubmit;

 /* Invoke the preview window to see the
SQL statements */
 rc = preview('BROWSE','Build A Column
Expression Preview');

 /* Send the code to the SAS Supervisor */

 submit continue;
    /* Demo of QWCLEXPR.CLASS */
 endsubmit;
```

```
return;
```

Figure 1: The master list EXPRMAST

```
( RET100=( AVCOLLST=( *='* RETAIL * <all columns>'
                      RETAIL='RETAIL.SALES
                      RETAIL='RETAIL.DATE
                      RETAIL='RETAIL.YEAR
                      RETAIL='RETAIL.MONTH
                      RETAIL='RETAIL.DAY
                      RET100='Ret100'
                     )[1959]
           AVCLBLST=( *='* RETAIL * <all columns>'
                      RETAIL.SALES='Retail sales in
millions of $'
                      RETAIL.DATE='DATE'
                      RETAIL.YEAR='YEAR'
                      RETAIL.MONTH='MONTH'
                      RETAIL.DAY='DAY'
                      RET100='Ret100'
                     )[1961]
           SLCOLLST=( RETAIL='RETAIL.DATE
                      RETAIL='RETAIL.YEAR
                      RETAIL='RETAIL.MONTH
                      RETAIL='RETAIL.DAY
                      RET100='RETAIL.SALES/100 as Ret100

                     )[2007]
           SLCLBLST=( RETAIL='RETAIL.DATE
label=DATE
                      RETAIL='RETAIL.YEAR
label=YEAR
                      RETAIL='RETAIL.MONTH
label=MONTH
                      RETAIL='RETAIL.DAY
label=DAY
                      RET100='Retail sales in millions
of $/100 as Ret100 label="Retail Sales/100"'
                     )[2009]
           SLTABLST=( SASHELP.RETAIL='SASHELP.RETAIL'
                     )[1973]
           EXMEMLST=( ='Ret100'
                     )[1971]
           EXASLIST=( RET100=' as Ret100'
                     )[1979]
           EXFMTLST=()[1969]
           EXLBLLST=( RET100=' label="Retail Sales/100"'
                     )[1963]
           CLDEXLST=( OPR='Retail sales in millions of
$'
                      OTH='/'
                      CONSTANT='100'
                     )[1965]
           CEPRMLST=()[1975]
           CINFOLST=( RETAIL.SALES='N'
                      RETAIL.DATE='N'
                      RETAIL.YEAR='N'
                      RETAIL.MONTH='N'
                      RETAIL.DAY='N'
                      RET100='N'
                     )[2023]
           EXNAMLST=( RET100='COLUMNS'
                     )[2025]
           COLEXLST=( OPR='RETAIL.SALES'
                      OTH='/'
```

```
                        CONSTANT='100'
                       )[2027]
            COLWHLST=( CONSTANT='<CONSTANT
  enter value>'
                        'RETAIL.SALES
  '
                        'RETAIL.DATE
label= Retail sales in millions of $  '
label= DATE             'RETAIL.YEAR'
label= YEAR                          '
label= MONTH            'RETAIL.MONTH
label= DAY                           '
                        'RETAIL.DAY
  '
                       )[2029]
            RPTMDLST=()[1987]
            EXPRDSN='SASHELP.RETAIL'
            COLEXNO=2
           )[1967]
   )[1947]



label=DATE'
label=YEAR'
label=MONTH'
label=DAY'

label="Retail Sales/100"'
```