

Cr8_In_List - Easy Table Driven Programming

Steve Sanders, Regions Bank, Birmingham, AL

ABSTRACT

We all use the "IN" logical operator in "IF" and "WHERE" conditions, but what a maintenance problem it can create when the lists of comparison values need to be changed within a library of SAS® programs. What if you could get the benefit of a subquery within DATA step language? Cr8_In_List is a macro that allows you to do this by creating the "IN" operator's comparison list based on data in other SAS data sets and lookup tables.

INTRODUCTION

This paper presents a utility process that retrieves information from a SAS dataset and stores it as a formatted string in a macro symbol. There are some formatting options that will allow the returned string to be used in a variety of ways, but the primary intended purpose is to use it with an "IN" function. This allows the IN function to be dynamically linked to a lookup table of values instead of a hard-coded list of literals that would require maintenance.

TABLE DRIVEN PROGRAMMING

The IN function { if variable IN ("value1" "value2") then ... } is undoubtedly one of the most basic and common functions used in many programming languages. It is an extremely useful tool, but it has one critical weakness. Using this function results in a hard-coded list of values that is likely to be forgotten and become inaccurate as changes within your data system occur.

So, we have a fundamental programming tool that easily leads to stale code that would then produce inaccurate results. So, how do we solve this problem?

Before we get started there are some terms that need to be defined for the context of this paper:

- Lookup tables are data tables that define data contained within a data system. An example would be SASHelp.ZipCode, which contains U.S. Postal Service ZIP codes and a variety of related information.
- "Table driven programming" is the use of lookup tables to make programming tasks easier to write and require less maintenance while maximizing accuracy and adaptability.

The only requirement for table driven programming is that you must have data that contains values that are critical to processes and defined in well maintained lookup tables. The good news is that most data systems meet this criterion.

The Cr8_In_List macro that is included in this paper will read the lookup table, select records using a filter, capture data from the selected records, format it, and store it in a SAS Macro Symbol (macrovariable). This macrovariable is then used to replace the hard-coded list of values in the IN Function. Since this macro executes every time the program runs, any updates that are made to the source lookup table will automatically update the "In List" created by this macro. The statement containing the IN Function is now table driven and automatically maintained.

SAMPLE USE OF CR8_IN_LIST

For demonstration purposes, let us assume that a program is needed that will select address records from a very large mailing list that have ZIP codes that are defined to be in Jefferson County, Alabama. The entire nationwide mailing list could be sorted by zip code and merged, but that would require a very time consuming sort process. The IN Function will select the records in random order, but the April 2004 ZipCode table that shipped with SAS 9.1.3 lists 105 ZIP codes in Jefferson County, Alabama. No one wants to type a list of 105 values into their code. The Cr8_In_List macro call to create this list is as follows:

```
%Cr8_In_List(infile=SASHelp.ZIPCode,  
             variable=ZIP,  
             macvar=JeffCo_AL_ZIP,  
             test=StateCode = "AL" and  
               uppercase(CountyNm) = "JEFFERSON");
```

The output from the macro is:

```
JeffCo_AL_ZIP = ("35005" "35006" "35015" "35020" "35021" "35022" "35023"
"35036" "35041" "35048" "35060" "35061" "35062" "35064" "35068" "35071"
"35073" "35091" "35094" "35111" "35116" "35117" "35118" "35119" "35123"
"35126" "35127" "35139" "35142" "35173" "35180" "35181" "35201" "35202"
"35203" "35204" "35205" "35206" "35207" "35208" "35209" "35210" "35211"
"35212" "35213" "35214" "35215" "35216" "35217" "35218" "35219" "35220"
"35221" "35222" "35223" "35224" "35225" "35226" "35228" "35229" "35230"
"35231" "35232" "35233" "35234" "35235" "35236" "35237" "35238" "35240"
"35243" "35244" "35245" "35246" "35249" "35253" "35254" "35255" "35259"
"35260" "35261" "35263" "35266" "35277" "35278" "35279" "35280" "35281"
"35282" "35283" "35285" "35286" "35287" "35288" "35289" "35290" "35291"
"35292" "35293" "35294" "35295" "35296" "35297" "35298" "35299")
```

The immediate benefit from the macro is that the code required to execute the macro required 192 characters of code (including the spaces for formatting) but the in list that was produced is 775 characters. So, 583 keystrokes were eliminated.

The future benefit is seen when you look at the second quarter 2009 updated SASHelp.ZIPCode table that is available from the SAS website << <http://support.sas.com/rnd/datavisualization/maponline/html/misc.html>>>. When the same code above is run with the second quarter 2009 table as the source data, the resulting list only has 93 values.

```
JeffCo_AL_ZIP = ("35005" "35006" "35015" "35020" "35021" "35022" "35023"
"35036" "35041" "35048" "35060" "35061" "35062" "35064" "35068" "35071"
"35073" "35091" "35094" "35111" "35116" "35117" "35118" "35119" "35123"
"35126" "35127" "35139" "35142" "35172" "35173" "35180" "35181" "35201"
"35202" "35203" "35204" "35205" "35206" "35207" "35208" "35209" "35210"
"35211" "35212" "35213" "35214" "35215" "35216" "35217" "35218" "35219"
"35220" "35221" "35222" "35223" "35224" "35226" "35228" "35229" "35231"
"35232" "35233" "35234" "35235" "35236" "35237" "35238" "35243" "35244"
"35246" "35249" "35253" "35254" "35255" "35259" "35260" "35261" "35266"
"35282" "35283" "35285" "35287" "35288" "35290" "35291" "35292" "35293"
"35294" "35295" "35296" "35297" "35298")
```

The in list is integrated into standard code as follows:

```
data Work.Jeff_Co_Mailing_List;
  set Whse.Master_Mailing_List;
  where ZIP in &JeffCo_AL_Zip;
run;
```

It really is that simple.

OPTIONS AVAILABLE IN CR8_IN_LIST

The basic operation of the macro is to create a list, and the default behavior is to format the list as demonstrated above. However, there are times when you will want a list formatted differently, so options exist to provide format flexibility.

If the ZIP variable in the Whse.Master_Mailing_List dataset was numeric, you would not want quotation marks around each value in the in list. If you include Quotes = N when you execute the macro, then the quotation marks will not be added.

If you need to suppress the parentheses, then include Enclose_In_Parentheses = N.

If you need commas between each value, then include Comma = Y.

CONCLUSION

This utility macro was originally written specifically to create a formatted list for use with the IN function, but over the years, it has become a much more flexible and useful tool. Whenever you have a program that contains a list of values that requires maintenance and could be created from data in a maintained table, you have an opportunity to use this macro. I hope you find it as useful as I have.

CR8_IN_LIST MACRO CODE

```
/* *****
* This macro will read any file (infile) testing the data
* with a where condition and build a concatenated string of
* values in the "variable" field. The resulting list can be
* used in an "IN" condition or other statements depending on
* the formatting options that are selected.
* Steve Sanders (Steve.G.Sanders@gmail.com) 7/9/2009
* *****/
%macro Cr8_In_List(infile=,
                  variable=,
                  macvar=,
                  Quotes=Y,
                  Comma=N,
                  Enclose_In_Parentheses=Y,
                  test=);

/* Since this is a macro with declared options, a local symbol
   table will be created by default and any macrovariables that
   are created within the macro will be added to the local symbol
   table. This will cause the formatted list to be created but
   deleted when the macro completes and before it can be used.
   The '%Global' statement is used to force the macro to the
   global symbol table so it will be retained.*/
%Global &MacVar;
/* Initialize the macrovariable to be created. If there is no
   data to match the conditions specified, this empty value
   will remain. Code below addresses any empty values.*/
%let &MacVar =;

/* This will only add the new value to the In List
   if the list does not already contain this value.*/
%macro Add_A_Value;
  %if %index(&&&MacVar,&Curr_Value) = 0 %then %do;
    %let &MacVar=&&&MacVar &Curr_Value;
  %end;
%mend Add_A_Value;

/* If the "comma" option is set to Y, this will
   insert commas between each value in the list.*/
%macro Add_A_Comma;
  %let &MacVar=&&&MacVar , ;
%mend Add_A_Comma;

data _null_;
  set &infile end=eof;
  where &test;
  Item_Count + 1;
  %if &Quotes = Y %then
    call symput("Curr_Value",quote(trim(left(&Variable))));
  %if &Quotes = N %then
    call symput("Curr_Value",trim(left(&Variable))));
  if "&Comma" = "N" or
    ("&Comma" = "Y" and
     Item_Count = 1) then
    call execute('%Add_A_Value');
  if "&Comma" = "Y" and
    Item_Count gt 1 then
```

```

        call execute('%Add_A_Comma');
        call execute('%Add_A_Value');
run;

%if &&&MacVar = %then %do;
    %put NOTE: Cr8_In_List returned no values;
    %if &Enclose_In_Parentheses = Y %then
        %let &MacVar = ("");
    %if &Enclose_In_Parentheses ne Y %then
        %let &MacVar = " " ;
%end;
%else
    /* Conditionally insert parentheses.*/
    %if &Enclose_In_Parentheses = Y %then
        %let &MacVar = (&&&MacVar.);
    /* This will trim trailing spaces.*/
    %if &Enclose_In_Parentheses ne Y %then
        %let &MacVar = &&&MacVar.;

    /* This displays the list that has been created in the log window.*/
    %put &MacVar = &&&MacVar;
%mend Cr8_In_List;

```

ACKNOWLEDGMENTS

Ron Richardson and Daniel Whitman provided inspiration for the creation of this macro, and SAS Technical Support provided assistance on more than one occasion to make it work.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Steve Sanders
 Regions Bank
 2050 Parkway Office Circle
 Birmingham, AL 35244
 Work Phone: (205) 560-7231
 Fax: (205) 560-3593
 Work E-mail: Steve.G.Sanders@Regions.com
 Personal E-mail: Steve.G.Sanders@gmail.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
 Other brand and product names are trademarks of their respective companies.