

Parsing a Macro Variable String -- Rob Krajcik

Have you ever received a macro variable string, and wanted to use it in a WHERE clause, such as Where <varname> in(<macro variable string>); ?

Here is some sample data:

```
data testsmpl ;
  infile cards;
  input labtest $ @@;
cards4;
ALP ALT K PLAT RBC GLUC ETOH BARB PCP CANN
;;;;
NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
NOTE: The data set WORK.TESTSMPL has 10 observations and 1 variables.
```

And here is the macro variable string:

```
%let list = ALP ALT GRANC AST CREAT HB PLAT TBI LI WBC CANN;
```

Before you can use the above macro string in a WHERE clause, you need to enclose each macro variable in either single or double quotes. What is the best way to do that?

One solution that comes to mind involves writing a macro function and returning the result:

```
%macro _quote(list = );
/* Parse the &list macro variable into a macro array &d&num */
/* Build quoted macro string &in_str from &list macro variable */
/* Written here as a macro function -- returns %unquote(&in_str) */
/* also returns &num */
%global num;
%let list = %upcase(&list);
%let num=1;
%let in_str=;
%do %while(%scan(&list, &num, %str( )) ne);
  %let d&num = %scan(&list, &num, %str( ));
  %let in_str = &in_str&d&num;
  %let num = %eval (&num+1);
  %if %scan(&list, &num, %str( )) ne
  %then %let in_str = &in_str%str('%' %');
%end;
%let num = %eval (&num-1);
%str('%')&in_str%str('%')
%mend _quote;
```

Let's test it:

```
%put %_quote(list=&list);
'ALP' 'ALT' 'GRANC' 'AST' 'CREAT' 'HB' 'PLAT' 'TBI LI' 'WBC' 'CANN'
```

Parsing a Macro Variable String -- Rob Krajcik

Now let's see if we put it to use:

```
data labselect;
    set testmpl;
    where labtest in("%_quote(list = &LIST)");
run;

MPRINT(_QUOTE):  "'ALP' 'ALT' 'GRANC' 'AST' 'CREAT' 'HB' 'PLAT' 'TBI LI' 'WBC' 'CANN'"

NOTE: There were 0 observations read from the data set WORK.TESTSMPL.
      WHERE 0 /* an obviously FALSE where clause */ ;
NOTE: The data set WORK.LABELCT has 0 observations and 1 variables.
```

We enclose the macro function in double quotes so the the macro compiler will run the macro function for us. Obviously, plugging this directly into the IN clause like this doesn't quite work -- we have a mix of double and single quotes here.

What if we use %SYSFUNC and the DEQUOTE function to strip off the double quotes?

```
data labselect;
    set testmpl;
    where labtest in(%sysfunc(dequote("%_quote(list = &LIST))));
run;

NOTE: There were 4 observations read from the data set WORK.TESTSMPL.
      WHERE labtest in ('ALP', 'ALT', 'AST', 'CANN', 'CREAT', 'GRANC', 'HB', 'PLAT', 'TBI LI', 'WBC');
NOTE: The data set WORK.LABELCT has 4 observations and 1 variables.
```

Or how about if we simply enclose our macro function within another?

```
data labselect;
    set testmpl;
    where labtest in(%unquote(%_quote(list = &list)));
run;

NOTE: There were 4 observations read from the data set WORK.TESTSMPL.
      WHERE labtest in ('ALP', 'ALT', 'AST', 'CANN', 'CREAT', 'GRANC', 'HB', 'PLAT', 'TBI LI', 'WBC');
NOTE: The data set WORK.LABELCT has 4 observations and 1 variables.
```

Did we really need to write that macro function in the first place?

What if we used %SYSFUNC and the TRANWRD and COMPBL functions?

```
data labselect;
    set testmpl;
    where labtest in(%unquote(%str(%')%qsysfunc(tranwr(%sysfunc(compbl (&list)),
%str( ), %str(%' %')))%str(%')));
run;
```

Parsing a Macro Variable String -- Rob Krajcik

NOTE: There were 4 observations read from the data set WORK.TESTSMPL.

```
WHERE labtest in ('ALP', 'ALT', 'AST', 'CANN', 'CREAT', 'GRANC', 'HB', 'PLAT', 'TBILI', 'WBC');
```

NOTE: The data set WORK.LABELCT has 4 observations and 1 variables.

The coding may look a bit tedious, but it is shorter than writing a macro function. Remember that when adding single (unmatched) quotes to a macro variable, you need to escape with with a percent (%) sign.

But is there another way to do this?

How about this:

```
data labelct;
  set testsmpl;
  where input(resolve('%index(&list, '||trim(labtest)||')'), 5.) gt 0;
run;
```

NOTE: There were 4 observations read from the data set WORK.TESTSMPL.

```
WHERE INPUT(RESOLVE('%INDEX(&LIST, '||TRIM(LABTEST)||')'), F5.)>0;
```

NOTE: The data set WORK.LABELCT has 4 observations and 1 variables.

The RESOLVE function can be used to resolve macro variables and macro functions.

Using the %INDEX function this way will return a non-zero value if the value of the labtest variable we just read from the SET statement is in the macro list &list.

This last solution seems the most concise of all the ones presented.

And we didn't need to enclose the macro variable string in quotes in order to use it in a WHERE clause!