

An OBS Limit with a WHERE Condition

Ian Whitlock, Westat, Rockville, MD

Abstract

In SAS® software the OBS option is incompatible with the WHERE condition. This was a design decision instead of an inherent incompatibility: in other words, a mistake. There are several possible interpretations of how to implement both of these restrictions.

A utility macro, LIMIT, is introduced to help implement a workaround for the interpretation given above. Use of the macro is first illustrated with a PROC PRINT and then in a macro TPRINT providing a more transparent form of solution for this important case.

SAS Version 7 is assumed, since it provides a simple solution to a naming problem, but the code can be easily modified to work in 6.12 at the expense of a little more user effort.

In addition to providing a useful extension to SAS, this example can be used as a learning tool for SAS macro in general and for the design of utility macros as a special case.

Introduction

There is a problem in combining the OBS option and the WHERE statement. The OBS option is a limit on the input, but in fact it is often used to limit the output since limiting input usually results in a corresponding limitation on the output. Perhaps it would be better to introduce an option OUTOBS to limit output with the understanding that input will not be read, once OUTOBS observations have gone out. (Note that PROC SQL does precisely this.) The macro LIMIT given below will work with an OUTOBS parameter and a WHERE parameter to produce limits on the input and output.

The case where one wants this, the most is in PROC PRINT. I need to see 20 cases having a certain property. For example,

```
proc print data = w ( outobs=20
                    where=(r<.3) ) ;
run ;
```

Of course the above is my ideal code, not SAS code. One would also like

```
proc print data = w ( obs=20
                    where=(r<.3) ) ;
run ;
```

In the first case records are processed by the WHERE processor until the end of file or a limit of 20 observations meeting the WHERE condition has been reached. In the second case only 20 records are processed by the WHERE processor and only those observations passing the test appear in the print.

Although the second case is desirable, I will not implement it since it cannot currently be done with WHERE processing. If desired, an equivalent can be obtained by simply replacing the WHERE statement with a subsetting IF statement in the macro LIMIT given below.

The idea is to produce a macro LIMIT that will create a view. In the example above, we write

```
%limit ( data = w ,
        outobs = 20 ,
        where = ( r < .3 ) )
proc print data = _w ;
run ;
```

By default, LIMIT names the view with the data set name prefixed with an underscore.

The Macro LIMIT

LIMIT has the parameters

DATA	input sas data set default last created SAS data set
VIEW	output sas view default _ prepended to input sas data set name, indicated by null
OUTOBS	upper bound on the number of observations. default is MAX
WHERE	where condition default null

For example:

```
data w ;
  do n = 1 to 50 ;
    r = ranuni ( 575741 ) ;
    output ;
  end ;
run ;

%limit ( data = w ,
        outobs = 20 ,
        where = ( r < .3 ) )

proc print data = _w ;
run ;
```

The code contains several tricky points concerning the handling of default parameter values. The automatic macro variable, SYSLAST, is assigned to DATA. This works because parameter evaluation is done at macro execution time, not at macro compile time. To find the default view name it is necessary to break apart the DATA parameter into libref and member. This entails knowing whether a two level name was used. A %IF statement is needed to handle the two different ways that VIEW might be assigned. Finally another %IF is needed in the subsetting view step. When OUTOBS = MAX there should not be any subsetting IF statement; otherwise, there should be. Hence another %IF is needed.

```
%macro limit ( data = &syslast ,
              view = ,
              outobs = max ,
              where =
            ) ;

  %if %quote(&view) = %then
  %do ;
    %if %index(&data,.) %then
      %let view =
%qscan(&data,1,.)_%qscan(&data,2,.);
    %else
      %let view = _&data ;
  %end ;

  data &view / view = &view ;
  %if %upcase(&outobs) ^= MAX
  %then %do ;
    if _n_>&outobs then stop;
  %end ;
  set &data ;
  where &where ;
run ;

%mend limit ;
```

The macro TPRINT

Finally, I would like to wrap it up in a neater package for the print problem. The consumer of %LIMIT should not have to worry about views, where they are located or what they are named. This can be done with a wrapper macro, TPRINT.

```
%macro tprint ( data = &syslast ,
               outobs = max ,
               where =
             ) ;

  %limit ( data = &data ,
          view = __view ,
          outobs = &outobs ,
          where = &where )

  proc print data = __view ;
run ;

%mend tprint ;
```

Conclusion

We have seen how SAS macro might be used to fill a hole in the design of SAS. The solution is not perfect, but it is usable and may help one to understand why the original design decision should be changed.

The author may be contacted by e-mail at

whitloi1@westat.com

or by mail at

Ian Whitlock
1650 Research Boulevard
Rockville, MD 20850

Questions are welcome.

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Perhaps the best is to limit the input to OBS observations, deleting all observations that do not meet the WHERE condition.