

An Iterative Loop of SQL Code

Ian Whitlock, Westat

Abstract

Some problems with good PROC SQL solutions require an iterative step. A good intuitively simple illustrative example was presented as a question in the first quarter of 1997 on SAS®-L. This paper restates the problem and gives a simple SQL solution using very simple macro code for the iterative loop.

The Problem

Suppose we have a data set, W, containing the variables NAME and GROUP.

NAME	GROUP
A	1
B	1
B	2
C	2
D	2
D	3
E	3
F	4
G	4
G	5
H	5

We want to collapse groups to the lowest level. For example, since A and B belong to group 1 and B and C belong to group 2, then all members of group 2 are part of group 1 because the groups have the common member B. Once this is seen one can add group 3 to the new group 1 because of the common member D. Thus group 1 covers A, B, C, D and E. Similarly F, G, and H ultimately belong to group 4. More formally, two groups are in the same chain if there is a sequence of groups containing the given groups such that each consecutive pair of groups contains a common name. Using this definition the data set consists of disjoint chains. The problem is to write a program identifying each chain by the minimum group number in the chain.

The intuitive argument given in the previous paragraph uses two kinds of minimization.

1. Find the minimum group (call it MINGROUP) for all names having the same value (e.g. NAME = 'B' has MINGROUP = 1).

2. Find the minimum of all MINGROUP values for all names in a common group (e.g. GROUP = 2 has MINGROUP = 1).

PROC SQL is very suitable to both types of minimization. In the first case we might have

```
create table t as
select name, group,
       min (group) as mingroup
from dataset
group by name ;
```

In the second case we have

```
create table t as
select name, group,
       min (mingroup) as mingroup
from t
group by group ;
```

These two operations must be repeated over and over until no new minimums are found, since each new extension of a group may mean further collapsing. To express the iteration of this code to an arbitrary level, we need a macro %DO-loop.

The Macro GROUPIT

For generality we make parameters to name the input and output data sets, and the variables represented by NAME, GROUP, and MINGROUP. The parameter MAX is added to insure that the macro does not execute for an excessively long time. (Since the algorithm does converge one could do away with this parameter or set it to the number of observations.)

```
%macro groupit
( data = &syslast,      /* input data */
  out = _DATA_,        /* resulting data */
  name = name,         /* name variable */
  group = group,       /* group variable */
  mingroup = mingroup, /* minimum */
  max = 20             /* limit #iterations */
) ;

/* minimize group on name and then group */
/* repeat until max iterations or done */

%local i done ;
```

```
proc sql ;

/* -----
initial set up - get first
minimums, start numbered sequence
of data sets
----- */

create table __t0 as
select &name , &group ,
      min ( &group ) as &mingroup
from &data
group by &name ;

create table __t0 as
select &name , &group ,
      min ( &mingroup ) as &mingroup
from __t0
group by &group ;

/* -----
iterate until done or too many
iterations
----- */

%do %until ( &done or &i > &max ) ;
  %let i = %eval ( &i + 1 ) ;

  create table __t&i as
  select &name , &group ,
        min ( &mingroup ) as &mingroup
  from __t%eval(&i-1)
  group by &name ;

  create table __t&i as
  select &name , &group ,
        min ( &mingroup ) as &mingroup
  from __t&i
  group by &group ;

/* are we finished? */

reset noprint ;
select w1.&name
  from __t%eval(&i-1) as w1,
       __t&i as w2
  where w1.&name=w2.&name and
        &group=w2.&group and
        w1.&mingroup ^=
          w2.&mingroup
;
```

```
%let done =
      %eval ( not &sqlobs ) ;
reset print ;
drop table __t%eval(&i-1) ;
%end ;

%if not &done %then
  %put WARNING(GROUPIT):Process
    stopped by condition MAX=&max;
%else
%do ;
  create table &out as
  select &name , &group ,
        &mingroup
  from __t&i
  order by &name , &group
  ;
  drop table __t&i ;
%end ;

quit ;
%mend groupit ;

%groupit ( data = w , name = name1 ,
          group = group1 , out = w2 )

proc print data = w2 ;
run ;
```

Conclusion

Macro iteration combined with PROC SQL and its ability to make macro variables provides a very powerful pattern for the SAS programmers tool box.

The author can be contacted by mail at:

Westat Inc.
1650 Research Boulevard
Rockville, MD 20850-3129

or by e-mail at:

whitloi1@westat.com

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