A SOLUTION TO UNIQUE MATCH-MERGE SITUATIONS

Terry Lessmann
Mutual of Omaha Insurance Companies

PROBLEM

SAS® software cannot merge two data sets with multiple occurrences of the merge variable in both data sets. For example, a user may have a file containing zip codes and the state and county codes associated with each zip code, and a second file of manager names and their territories based on state and county codes. The user wants to merge the two files by state and county so that the final output will contain all the zip codes in a manager's territory. However, some managers share counties and those shared counties may contain more than one zip code. No matter how the user arranges the merge statement, SAS software cannot successfully merge the data so that each manager gets all the zip codes associated with his/her territory (FIGURES 1 & 2).

SOLUTION

A solution to the problem is to split one of the data sets of multiple occurrences into several single occurrence data sets. These single occurrence data sets are then merged with the remaining multiple occurrence data set. This is accomplished using a combination of FIRST.byvariable and a macro. FIRST.byvariable is employed to strip off the first occurrences of the merge variable in one data set and merge it with all matching occurrences in the second data set. The macro is used to continue this process until all records have been merged.

FIGURE 1

```
DATA FIRST;
  INPUT STNTY $ ZIP $;
  CARDS;
  01001 12345
  01001 12346
  01002 50555
 ;
PROC SORT; BY STNTY;

DATA SECOND;
  INPUT STNTY $ NAME $;
  CARDS;
  01001 JAKE
  01002 JANE
  01003 SNAKE
 ;
PROC SORT; BY STNTY;

DATA X;
  MERGE FIRST (IN=A) SECOND (IN=B);
  BY STNTY;
  IF A AND B;
PROC PRINT; TITLE 'DATA X (FIRST*SECOND)'

DATA Y;
  MERGE SECOND (IN=A) FIRST (IN=B);
  BY STNTY;
  IF A AND B;
PROC PRINT; TITLE 'DATA Y (SECOND*FIRST)'
```

FIGURE 2

```
DATA X (FIRST*SECOND);
  OBS STNTY ZIP NAME
  1 01001 12345  JAKE
  2 01001 12346 SNAKE
  3 01002 50555  JAKE

DATA Y (SECOND*FIRST);
  OBS STNTY NAME ZIP
  1 01001  JAKE  12345
  2 01001  SNAKE 12346
  3 01003  SNAKE  50555

(DESIRE OUTPUT)

OBS STNTY NAME ZIP
  1 01001  JAKE  12345
  2 01001  SNAKE 12346
  3 01002  JAKE  50555
  4 01003  SNAKE 12345
  5 01003  JAKE  50555
```
PROGRAM LOGIC

The following is a description of the logic used in the SAS program named MACRO MERGE PROGRAM. To assist the reader, explanations and program code are presented for analysis:

1.) Insert the merge variable name into the macro variable MERGVAR.

2.) Create the data sets and sort them by the merge variable.

3.) A data step is executed to determine the maximum number of managers who share one county. The number of occurrences is kept in the variable named TOTAL and the highest value of TOTAL is kept in HIGH. After the last record has been processed, the value of HIGH is put into the macro variable NUM (FIGURE 3). NOTE: Data set SECOND was used as input to this data step. It does not matter which data set is used, the final results will be the same.
4.) The macro named MACMERG is executed where each first occurrence of the merge variable is placed into the data set DS1. The remainder of the records are left in data set SECOND. DS1 is then merged with data set FIRST. As the loop continues, first occurrences in the new data set SECOND are put into DS2 and the remainder are left in SECOND. DS2 is merged with data set FIRST. The process continues to DS&NUM (DS3 in our example), which will leave data set SECOND with 0 observations (FIGURE 4).
5.) The macro named COMBINE appends DS1 - DS&NUM so that all records from the original data set SECOND are back together again.

6.) The macro named CLEANUP deletes the individual data sets, since they are no longer needed.

7.) The final output is ready for further processing or printing (FIGURE 5).
CONCLUSION

The MACRO MERGE PROGRAM employs several macros and data steps resulting in less programmer intervention. For example, the %LET macro function allows the user to insert the merge variable name in one statement and SAS automatically inserts the name throughout the program. However, reading programs containing macros can be intimidating. There are two rules to remember when reading or writing programs with macros: (1) macros generate SAS code and (2) always include OPTIONS MPRINT in the SAS code. MPRINT shows the SAS code that the macro generates in the SASLOG. The SASLOG for the MACRO MERGE PROGRAM should be examined to see all of the SAS code that the macros create. All the macros do in the MACRO MERGE PROGRAM is generate repetitive code, allowing the programmer to create several single occurrence data sets that are merged with one multiple occurrence data set. This technique permits the programmer to overcome certain unique match-merge situations.

Questions or comments regarding this paper should be directed to:

Terry Lessmann
Mutual of Omaha
Mutual of Omaha Plaza
Omaha, NE 68175
(402) 342-7600

SAS is a registered trademark of SAS Institute Inc., Cary, NC, USA.

*** The complete program can be found on the following pages ***
DATA FIRST;
    INPUT STCITY $ ZP $;
    CARS;
    01001 12345
    01002 56789
    01002 66666
    02002 66666
    ;
PROC SORT; BY EMERGVAR;
DATA SECOND;
    INPUT STCITY $ NAME $;
    CARS;
    01001 JAKI
    01002 JAKI
    01003 JAKI
    01009 JAKI
    01002 CURLY
    01002 MOE
    01003 SCROODY
    02002 FISHHEAD
    ;
PROC SORT; BY EMERGVAR;

******************************************************************************
**** Detect maximum number of data sets needed: 2 ****
**** Total containing maximum number within each ****
**** Value occurrence of Emergvar: highest value ****
**** of 'Total' is kept in 'High'. When last ****
**** record is processed, the value of 'High' is ****
**** put into macro variable 'num'. 'num' is the ****
**** maximum number of data sets that will be ****
**** needed in later macros. ****
******************************************************************************
DATA NULL;
SET SECOND END=LASTREC;
BY EMERGVAR;
RETAIN HIGH;
IF W_1 = 1 THEN HIGH = 0;
IF 01001 EMERGVAR THEN TOTAL = 0;
TOTAL = 1;
IF LAST. EMERGVAR AND TOTAL > HIGH THEN 
    HIGH = TOTAL;
IF LASTREC THEN 
    CALL SYMPUT ('num', 'LEF'T(PUT(HIGH, 2, .))); 
RUN;

PUT maximum number of data sets needed: 'num';

******************************************************************************

*** Macro merge ***
DATA DS1 SECOND;
    SET SECOND;
    BY EMERGVAR;
    IF FIRST. EMERGVAR THEN OUTPUT DS1;
    ELSE OUTPUT SECOND;
    RUN;
DATA DS8;
    MERGE DS1 (IN=A) FIRST (IN=B);
    BY EMERGVAR;
    IF A AND B;
    RUN;
SEND;
SEND;
****
**** Macmerge ***
******************************************************************************

DATA DS1 SECOND;
    SET SECOND;
    BY EMERGVAR;
    IF FIRST. EMERGVAR THEN OUTPUT DS1;
    ELSE OUTPUT SECOND;
    RUN;
DATA DS8;
    MERGE DS1 (IN=A) FIRST (IN=B);
    BY EMERGVAR;
    IF A AND B;
    RUN;
SEND;
SEND;
****
**** Maccombine ***
******************************************************************************

DATA FINAL;
    SET DS2 x-1 INTO SRNM;
    DS2X;
    ;
    RUN;
SEND;
****
**** Mccombine ***
******************************************************************************

*** Macro cleanup deletes EST-DSNUM since data ****
*** set FINAL is created and DS1-DSNUM are no ****
*** longer needed. ****
******************************************************************************

*** Macro cleanup ***
PROC DATASETS LIBRARY=WORK;
DELETE
    ZDO X-1 INTO SRNM;
    DS2;
    ;
    RUN;
SEND;
****
**** Mccleanup ***
PROC SORT; BY EMERGVAR;
PROC PRINT;
TITLE 'FINAL OUTPUT OF MERGED DATA SETS';