TS- 644

Janice Bloom

Everything you wanted to know about MERGE but were afraid to ask

So you have read the documentation in the SAS Language Reference for MERGE and it still does not make sense? Rest assured you are not alone. Some people can read the documentation about MERGE and totally understand the algorithm. Others need illustrations or hands-on experience. This paper targets users wanting MERGE examples that illustrate MERGE basics, as well as examples for the most frequently asked MERGE questions. I will be referencing existing documentation, but will insert examples to illustrate most points. You can make this a hands-on demo by submitting the examples yourself. Grab a piece of paper and draw out the Program Data Vector (PDV), processing the data yourself. Then compare your results to those of SAS. Are they the same?

When you have finished testing the programs, I hope you will have the confidence to know what results you are going to get in a MERGE before you even submit the code. Once you get the basic algorithm, MERGE is a "piece of cake".

Getting Started

When I am asked where one should go to learn about MERGE, I always start with the SAS Language Reference section on BY group processing. MERGE can be performed with a BY statement (match-merge) or without (one-to-one merge). Most often, a MERGE is used to combine two data sets based upon one or more common variables, which means BY group processing will take place!

To understand BY Groups, the SAS Language Reference emphasizes three concepts: BY variable, BY value, and BY group.

BY variable
> a variable named in a BY statement. A BY statement can name one or more variables, and the SAS system expects the data sets to be ordered by the values of the variables named in this statement. When combining data sets, each data set must include the BY variable or variables.

BY value
> the value or formatted value of a BY variable.

BY group
> all observations with the same BY value. If you use more than one variable in a BY statement, a BY group is a group of observations with the same combination of values for those variables (emphasis added).

(SAS Language Reference, Version 6, First Edition p132. For Version 8 documentation, see Chapter 23 beginning on page 303 in the SAS Language Reference, Concepts: Version 8.)

Examples of BY groups are included in the documentation found in either versions of the SAS Language Reference, so are not include here. I encourage you to read these sections for more detailed information. In my opinion, the key concept to understanding how MERGE works is to realize that 'the BY statement modifies the action of the SET, MERGE, or UPDATE statement by controlling when the values in the program data vector are set to missing. During BY-group processing, the SAS System retains the values of variables until it has copied the last observation it finds for that BY group in any of the data sets (SAS Language Reference, Version 6, First Edition p135 or SAS Language Reference, Concepts: Version 8 p309. Keep this point in mind when you step through the examples.

Time to Combine

As stated in the SAS Language Reference, Version 8, on page 328 under 'How to Prepare Your Data Sets', it is advised that before using any of the methods to combine data sets, you should follow these guidelines to ensure that you get the results you want:

Know the structure and the contents of the data sets

- Do you have unique observations in one or more of the data sets?
- Do you have duplicates in one or more of the data sets?
- Are the length and type attributes of the BY variables identical?
- Are there variables that exist in both data sets that are not BY variables?

Ensure that observations are in the correct order, or that they can be retrieved in the correct order

- For a merge with a BY statement, your data needs to be sorted, either by using PROC SORT or having data that is in sorted order already, or indexed using the BY variables

Test your program

- Create subsets of your data and run them through the MERGE. Be sure you include observations that will test the logic in your code. Check the results to see if they are what you expected.

Look at sources of common problems

Understanding the basic algorithm of MERGE will help you understand how the step processes. There are still a few common scenarios whose results sometimes catch users off guard. Here are a few of the most frequent 'gotchas':

1- BY variables have different lengths

It is possible to perform a MERGE when the lengths of the BY variables are different, but if the data set with the shorter version is listed first on the MERGE statement, the shorter length will be used for the length of the BY variable during the merge. Due to this shorter length, truncation occurs and unintended combinations could result. In Version 8, a warning is issued to point out this data integrity risk. The warning will be issued regardless of which data set is listed first:

WARNING: Multiple lengths were specified for the BY variable name by input data sets. This may cause unexpected results.

Truncation can be avoided by naming the data set with the longest length for the BY variable first on the MERGE statement, but the warning message is still issued. To prevent the warning, ensure the BY variables have the same length prior to combining them in the MERGE step with PROC CONTENTS. You can change the variable length with either a LENGTH statement in the merge DATA step prior to the MERGE statement, or by recreating the data sets to have identical lengths for the BY variables.

Example 1.1

```
/* Example illustrating negative results when merge is performed with BY variables of different lengths. */

DATA one;
  INPUT x $;
DATALINES;
cat
catnap
catnip
;

DATA two;
  INPUT x $3. y ;
  DATALINES;
cat 111
dog 222
;

DATA badmerge;
  MERGE two one; /* Shorter length for X listed first */
                 /* to illustrate the truncation      */
  BY x;
RUN;

PROC PRINT;
RUN;

/* RESULTS */

Obs    x      y

 1     cat    111
 2     cat    111
 3     cat    111
 4     dog    222
```

Since the length of X in Data TWO is only 3 bytes, the value of 'catnip' and 'catnap' in Data ONE would be truncated to 'cat', and would therefore combine with the first observation in Data TWO. Reverse the order of the data sets on the MERGE statement, and the two observations would NOT combine. Remember that a warning would be issued in Version 8 because the lengths of the BY variables were different. The following code would successfully combine the data sets with no warnings issued:

```
DATA fixmerge;
  LENGTH x $ 8;
  MERGE two one;  /* Shorter length for X listed first does not        */
                  /* matter now, because LENGTH statement      */
                  /* determined length on PDV                  */
  BY x;
RUN;

PROC PRINT;
RUN;
```

```
/* RESULTS */

Obs    x         y

 1     cat       111
 2     catnap     .
 3     catnip     .
 4     dog       222
```

2- Variables that are common to both data sets, but are not the BY variables. Unique observations in both data sets.

A common variable name will occur only once on the output data set. The data set listed first will populate the PDV first, and if the other data sets have the same BY group values, the value of the other common variables will over-write the current value in the PDV. Here is an example.

Example 1.2

```
DATA one;
  INPUT id $ var;
DATALINES;
a 1
b 2
c 3
;

DATA two;
  INPUT id $ var;
DATALINES;
a 11
c 33
;

DATA common;
  MERGE one two;
  BY id;
RUN;

PROC PRINT;
RUN;

/* RESULTS */

Obs    id    var

1      a     11
2      b      2
3      c     33
```

3- Variables that are common to both data sets, but are not the BY variables. Duplicates in first data set, unique observations in the second data set. Why does on the first match in a BY group have the value from the second data set and not the entire BY group?

Example 1.3

```
/* The example below has the common variable Y.
   It is NOT a BY variable. */

DATA one;
  INPUT x y;
DATALINES;
1 11
1 .
2 22
;

DATA two;
  INPUT x y;
DATALINES;
1 111
2 222
;
```

Let's step through the MERGE DATA step as if we were SAS executing the code:

```
DATA both;
  MERGE one two;
  BY x;
RUN;
```

Read in first observation from ONE into the PDV, fill in values of variables:

```
X   Y
1   11
```

Read in first observation of same BY group from TWO. Values of COMMON variables are overwritten with new values from second data set:

```
X   Y
1   111   ----> output
```

NOTE: Variables are not reinitialized to missing until the BY group changes.

Read in second observation of same BY group from ONE, reassign variable values. Keep in mind that a blank value is still a value. Y's previous value of 111 is overwritten with the new observation's blank value.

```
X   Y
1   .
```

There are no more observations in TWO for this BY group, so the current values are output written to the output data set.

```
X   Y
1   .    ----> output
```

This is the end of the current BY group, variables in the PDV are set to missing, and the next observation is read in from ONE.

X Y
2 22

Read in observation from matching BY group in TWO, value of Y is reassigned and written to the output data set.

X Y
2 222 ----> output

/* RESULTS */

Obs x y

1 1 111
2 1 .
3 2 222

To avoid over-writing common variables during the MERGE, you can either rename the variables involved, or drop the common variable from the data set whose values would be over-written (that is, keep the variable on the data set that has the values you want in the final output data set).

If you wanted to keep the values from Y from both data sets in the example above, you could either rename Y in one data set, like this:

Example 1.4

```
DATA both;
  MERGE one(rename=(y=oldy)) two;
  BY x;
RUN;

PROC PRINT;
RUN;
```

/* RESULTS */

Obs x oldy y

1 1 11 111
2 1 . 111
3 2 22 222

Or drop Y from data set ONE, like this:

Example 1.5

```
DATA both;
  MERGE one (drop=y) two;
  BY x;
RUN;

PROC PRINT;
RUN;
```

/* RESULTS */

Obs	x	y
1	1	111
2	1	111
3	2	222

4- Variables that have the same name but that represent different data

If your data sets have common variable names that represent different data, you want to use the RENAME= data set option so important data is not lost or overwritten.

5- Common variables with the same data but different types

If you try to MERGE data sets whose common variables have different types, SAS stops processing the DATA step and issues an error message:

ERROR: Variable xxxx has been defined as both character and numeric.

To correct this error, you must recreate one of the data sets and use the variable type that is compatible with variable type in the remaining data set(s). (NOTE: Use the PUT function to convert from numeric to character and the INPUT function to convert from character to numeric.)

Example 1.6

```
/* Variable A is defined as character */

DATA char;
  INPUT a $ b;
DATALINES;
1 11
2 22
3 33
;

/* Variable A is defined as numeric */

DATA num;
  INPUT a c;
DATALINES;
1 111
2 222
;

DATA both;
  MERGE char num;
  BY a;
RUN;
```

Submitting this code as is will generate an error:

```
18   DATA both;
19     MERGE char num;
ERROR: Variable a has been defined as both character and numeric.
20     BY a;
21   RUN;
```

To perform the MERGE, we can change CHAR's variable A to numeric with an INPUT function:

Example 1.7

```
/* Changing variable types so Example 1.6 MERGE code will work */

DATA Char;
  SET char (rename=(a=olda));
  A=INPUT(olda,8.);
RUN;
```

Or you can change NUM's variable A to character using the PUT function:

Example 1.8

```
/* Changing variable type so Example 1.6 MERGE code will work */

DATA Num;
  SET Num (rename=(a=foo));
  A=PUT(foo,1.);
RUN;
```

6- Label, Format, and Informat attributes are different between data sets

SAS takes the "attribute from the first data set that contains the variable with that attribute. However, any label, format, or informat that you explicitly specify overrides a default. To ensure that the new output data set has the attributes you prefer, use an ATTRIB statement" (SLR, Version 8, p329).

7- BY variables are not properly sorted.

To successfully MERGE two or more data sets, be sure all data sets are in the same sorted order. This can be done with a PROC SORT on each data set, using identical BY statements. If your data sets are already in sorted order, you do not need to 're-sort' them. In lieu of sorting a data set, you can use an INDEX. SAS will first check to see if an index is available, then check to see if the data is sorted. If the data set is marked as sorted, the index will not be used. If an index is not found, and the descriptor portion of the data set does not indicate the data set is sorted, SAS will still commence the MERGE. If an observation is found to be out of order during the MERGE process, SAS will stop and issue an error:

ERROR: BY variables are not properly sorted on data set XXXX.XXXX.

To verify a data set is sorted, submit a PROC CONTENTS and check the SORTEDBY field.

/* Portion of PROC CONTENTS output */

```
-----Alphabetic List of Variables and Attributes-----
          #    Variable    Type    Len    Pos
          -----------------------------------
       1    x        Num       8      0


          -----Sort Information-----

            Sortedby:      x
            Validated:     YES
            Character Set: ANSI
```

8- NOTE: MERGE statement has more than one data set with repeats of BY values

This note means that SAS has found more than one of your data sets has duplicate values in a BY group. The note is not indicating a problem exists, but is informational. Duplicates in more than one data set can produce results that a user is not expecting if the user is not aware that duplicates do exist in more than one data set.

Here are two data sets with duplicates of the BY variable:

Example 1.9

```
/* Note there are two A's and three C's in ONE */

DATA one;
  INPUT id $ fruit $;
DATALINES;
a apple
a apple
b banana
c coconut
c coconut
c coconut
;

PROC SORT data=one;
  BY id;
RUN;

/* Note there are two B's and two C's in TWO */

DATA two;
  INPUT id $ color $;
DATALINES;
a amber
b brown
b black
c cocoa
c cream
;

PROC SORT data=two;
```

```
  BY id;
RUN;

/* The result is that there will be two observations of the              */
/* B BY group due to duplicates in DATA two. Note the different */
/* values of COLOR for the C BY group. These are the correct      */
/* results but if you did not KNOW that you had duplicates of       */
/* C in both data sets, you might be surprised that COCOA did       */
/* not carry across the BY GROUP.                                          */

DATA test;
  MERGE one two;
  BY id;
RUN;

PROC PRINT;
RUN;

/* RESULTS */

Obs id fruit   color

1   a  apple   amber
2   a  apple   amber
3   b  banana  brown
4   b  banana  black
5   c  coconut cocoa
6   c  coconut cream
7   c  coconut cream
```

Learning More

SAS Language and Procedures, Usage, Chapter 17, Merging SAS Data Sets.

References

SAS Language Reference, Version 6, First Edition, BY-Group Processing,pp 131-137.

SAS Language Reference, Version 6, First Edition, One-to-One Merging, pp 147-151.

SAS Language Reference, Version 6, First Edition, Match-Merging, pp 151-155.

SAS Language Reference: Concepts, Version 8, BY-Group Processing, pp 303-314.

SAS Language Reference: Dictionary, Version 8, One-to-One Merging, pp 340-344.

SAS Language Reference: Dictionary, Version 8, Match-Merging, pp 344-348.