Advanced MATCH-MERGING: Techniques, Tricks, and Traps

Malachy J. Foley

University of North Carolina at Chapel Hill, NC

ABSTRACT

Match-merging, or BY merging, is the most common merging technique used in SAS**. Yet, how it works is not always obvious. This tutorial shows many of the technique's nuances and subtleties, gives examples of merges where even experienced programmers have been tripped up, and demonstrates defensive programming strategies.

INTRODUCTION

Several times a year you hear it. It might be in a SAS users group, at the office, or in the SAS-L. What you hear goes like this: "The SAS merge is a subtle thing"; or "Merge should be made to work better in this situation"; or "Be very careful with using merge in this case because you get unexpected results!"

This paper is meant to clear up some of the mystery that surrounds the match-merge. It shows some interesting examples of merges where even experienced programmers have been tripped up. It also covers the basics. By the end of this paper the reader should have a strong foundation in match-merging and know how to avoid most of the basic problems programmers typically have with merge.

WHAT IS A MERGE?

There are many definitions for merge. All of them talk about taking two or more sorted input files and combining them into one output file. Of course, sorted means that the input files have a common key and that the records in each file are ordered according to the key field(s). Consider the following two input files, as an example.

```
 FILE ONE          FILE TWO
---------        -------------
ID   NAME        ID   AGE  SEX
---------        -------------
A01  SUE         A01  58   F
A02  TOM         A02  20   M
A05  KAY         A04  47   F
A10  JIM         A10  11   M
```

These two files have a common key field called ID. The records in both files are sorted by ID. A Match-merge in SAS means that records from the one file will be matched up with the records of the second file that have the same ID. The information in the matched records is combined to form one output record. Match-merging the above two input files would give the following result.

```
     FILE ONE_TWO
------------------
ID   NAME  AGE  SEX
------------------
A01  SUE   58   F
A02  TOM   20   M
A04        47   F
A05  KAY
A10  JIM   11   M
```

Even with this simple example, there is already a hint of problems. Observe that the records A05 (file ONE) and A04 (file TWO) did not have a matching record. As a result there is missing data for records A04 and A05 in the output file called ONE_TWO. This case is rather obvious and intuitive. But the cases that we will examine become increasingly more complex.

A COMMON MISTAKE

The SAS code needed to merge the files as described above is:

```
PROC SORT DATA=ONE; BY ID; RUN;
PROC SORT DATA=TWO; BY ID; RUN;
DATA ONE_TWO;
  MERGE ONE TWO;
  BY ID;
RUN;
```

This code is simple enough, but it could be made even simpler. If one knows, for sure, that the input data sets are already sorted then the SAS code could be reduced to:

```
DATA ONE_TWO;
  MERGE ONE TWO;
  BY ID;
RUN;
```

Now, what could be simpler? Well, not much. But this is probably one of the traps: the match-merge is deceivingly simple.

For example, a common error made in coding a match-merge is to inadvertently forget to include the BY statement. To do so will beget the following unexpected results with no warning or error messages.

```
---------------          FILE ONE_TWO
DATA ONE_TWO;         -------------------
  MERGE ONE TWO;      ID   NAME  AGE  SEX
RUN;                  -------------------
---------------       A01  SUE   58   F
                      A02  TOM   20   M
                      A04  KAY   47   F
                      A10  JIM   11   M
```

Notice that the A05 ID is lost in this merge and the Name Kay is moved from ID=A05 to ID=A04, and one does not even get a note to say that something is wrong with the code. All one gets is a bad output file. The reason for this is that SAS recognizes the above code as a valid one-to-one merge rather than the intended match-merge.

The one-to-one merge is another type of a SAS merge process. It is a DATA step which never has a BY statement. The match-merge, on the other hand, is a DATA step that always has a BY statement. This is the reason a match-merge is also called a BY merge.

BY VARIABLES AND ASSOCIATED TRAPS

In the SAS match-merge, the matching process is controlled by the BY variables. BY variables are the variables listed in the BY statement. As we have just seen, one can not have a match-merge without a BY statement. There can be one or many BY variables in the BY Statement. To perform a match-merge, the input files must be sorted on the BY variables.

BY variables should be key variables. Key variables are either character or numeric variables that uniquely identify or label the records or observations within the input data sets. Typically, there is only one key variable and it identifies the unit of data the record is associated with, such as a patient, account number, transaction number, household, interviewee, etc.. So, good BY variables are key variables that EXACTLY identify to whom or what the record belongs.

BY variables must be chosen and controlled carefully to have a successful merge. Programmers have been know to spend hours trying to figure out what was wrong with their SAS code, only to find out that they had faulty BY variables.

There are at least two kinds of traps associated with BY variables: (1) the BY variable does not uniquely identify the record (ambiguous identification); and (2) the BY variables in the input files have significantly different characteristics.

AMBIGUOUS BY VARIABLE

Data items such as name, time, and date are ambiguous, or difficult to obtain correctly. As such, these types of items do not reliably identify records for matching and therefore should not be used as BY variables.

Dates, for example, are notoriously bad identifiers. As an illustration, let's say in a clinical trial, patients need to come into a clinic for a blood draw on a particular date. An intuitive and logical choice for a key variable in this situation would be the visit date.

However, in practice, using the date as the key variable to identify the patient's visit does not work very well. The nurse filling out the form can forget exactly what date it is and record a different day. Even if a computer date is used, such dates have been known to be mis-set. The patient might be unable to come in on the specified day and actually come in the next day. Of course, someone can always transcribe a date incorrectly. How often do people forget exactly what year it is during January? And the list goes on.

Now imagine trying to match two records, one from the lab and another the clinic, when the visit dates are reported differently for the same visit.

Suffice it to say that dates are unreliable key variables. In our particular illustration, a blood draw number or visit number with a check digit would have been a much more reliable choice as a key variable and the resulting merge would have a better chance of matching the records from a lab and a clinic.

BY VARIABLE CHARACTERISTICS

Another pitfall associated with BY variables is having different characteristics connected to the same BY variable in the different input data sets. Some of the characteristics to be on the lookout for are:

- Manipulation History
- Type (numeric or character)
- Justification (for character variables only, usually left)
- Stability
- Length

The following sections will explore each of these characteristics and how they can cause the demise of a match-merge.

BY VARIABLE HISTORY

Each BY variable has a processing history, i.e., how it was created, initialized, and later manipulated before it gets to be a BY variable in an input file to a match-merge.

BY variables are key variables, and key variables are essentially labels. Labels, as a rule, are attached to their product and are never changed. And so it should be with record or observation labels. In other words, the ideal history of a BY variable is short and sweet.

A variable which has been derived numerically, manipulated, or changed, in any way, is often a poor candidate as a BY variable. The reason is that to match input records, the BY variables must have EXACTLY identical values in all the input files. Approximately the same value will not match the record (unless you are doing a fuzzy merge which is neither discussed here nor recommended).

Because the values must be EXACTLY the same, we suggest that, like all good labels, a BY value be attached once to each record and never altered. If this is impossible, and only if it is impossible, we suggest they have EXACTLY the same history of manipulation. For example, if one file has a numerical BY variable, say

IDx=11, and it is multiplied by 10, then it should be multiplied by 10 in the other file. Multiplying by 10 in one file, and dividing by 10 and multiplying by 100 in the other file is mathematically identical, but in practice 11*10 is not equal to (11/10)*100. This difference has to do with decimal-to-binary conversion and the characteristics of fractions. But the moral to the story is to use exactly the same manipulation on the BY variables of all files, or expect disaster. To put it another way, the BY variables in each input file should have identical histories of manipulation (including numeric manipulation and LENGTH manipulation).

BY VARIABLE TYPE, JUSTIFICATION, AND STABILITY

SAS is not designed to handle different-typed key variables. Each BY variable must be of the same type (numeric or character) in all the input data sets. Thus, if a specific BY variable, say ID3, is numeric in one input data set, it must be numeric in all the data sets. If this is not the case SAS will give an error message like "ERROR: Variable ID3 has been defined as both character and numeric."

This problem can be corrected by converting a given key variable to the same type in all the input files before the MERGE is attempted. However, before trying such a conversion, it is wise to find out why the same-named variable has a different TYPE in different files. While they have the same name, they obviously do not have the same manipulation history and may not even be the same variable. In other words, they could be two different types of data which, by coincidence, were named the same.

SAS will not give any message for character BY variables with different justifications (left or right) in the input data sets. Nor will SAS be able to reliably match the records from input data sets based on such a BY variable. If a BY variable called ACCT_NUM has a length of 5, for example, SAS will not match the left-justified value of "A05 " from one data set with the right-justified value of " A05" from a second data set.

The solution to a justification mismatch problem is to similarly justify all character BY variables before they are merged using the LEFT and RIGHT functions. However, once again, one would be well advised to find out why same-named variables have different justifications before attempting a merge.

And a word about stability. Decimal fractions have a way of changing their values ever so slightly without a programmer being aware of the change. Only a very knowledgeable and careful programmer can use BY variables with decimal fraction values and know, with certainty, that they will match properly. It is recommended that only integers and character variables be used as BY variables. Any variable whose value is a fraction, or derived from fractions, is inherently unstable and should not be used as BY variables.

BY VARIABLE LENGTH

Another characteristic (and attribute) of BY variables is LENGTH. One should make sure that the LENGTHs of the BY variables are the same in all the input files and that the manipulation history of the LENGTHs, if they have changed, are the same. Look at the following example.

```
      CODE                          OUTPUT
----------------------          ------------
DATA SDS_1;
INPUT ID $ 1-3 V_8 6-7;
CARDS ;
A22  12
A38  88
A51  33
;
RUN;

DATA SDS_2;
INPUT ID $ 1-4 V_9 6-7;
CARDS ;
A22  72
A38  37
A41  11
A511 58                         FILE SDS_1_2
;                               ------------
RUN;                            ID  V_8  V_9
                                ------------
DATA SDS_1_2;                   A22  12   72
  MERGE SDS_1 SDS_2;            A38  88   37
    BY ID;                      A41   .   11
RUN;                            A51  33   58
----------------------          ------------
```

In this example, the variable, ID, has a LENGTH=3 in the first file and a LENGTH=4 in the second file. At compile time, the program data vector, for the output file, the attributes of each variable is determined by the first input data set where they appear. Thus, in this case, the after first file in the merge statement is scanned, the data vector is (ID $3, V_8). Then, the second file is scanned and new variables added to the vector, so that the final output data is (ID$3 V_8 V_9). Since the ID has a LENGTH=3 in the data vector, the value of ID=511 in the second file is clipped to A51 and matched with the record A51 from the first file. This is an example of how, when the LENGTHs are different, one can get undesired results.

When the data files are reversed in the merge statement, the desired results are obtained!

```
--------------------            FILE SDS_2_1
DATA SDS_2_1;                   ------------
  MERGE SDS_2 SDS_1;            ID  V_9  V_8
    BY ID;                      ------------
RUN;                            A22  72   12
--------------------            A38  37   88
                                A41  11    .
                                A51   .   33
                                A511 58    .
```

This last example, shows how reversing the order of the data sets in the merge statement can sometimes change the values and records in the output file.

MATCH-MERGE TYPES

The whole idea of a match-merge is to match records from two or more input files. Records are matched based on the key variables. In SAS the key variables are identified in the BY statement. To understand the subtleties of SAS's match-merge, it is crucial to know three terms that are related to each other. These terms are the BY Variable (which was examined in a previous section), the BY Value, and the BY Group.

The BY variables are the variables named in the BY statement of the merge. There can be one or many BY variables in the BY Statement. The input files must be sorted on the BY variables.

The BY value is the value a BY variable has in a particular record or observation. The BY group is all the records in a sorted data set that have the same BY values for all the BY variables. The following two SAS data sets, named ALPHA and BETA, will be used to illustrate the different types of BY groups.

ALPHA		BETA	
ID	V_1	ID	V_4
A01	23	A01	58
A02	99	A02	20
A05	56	A04	47
A10	88	A10	11
A25	24	A25	4
A25	22	A25	91
A32	91	A32	1
A55	83	A32	22
A55	19	A32	61
A55	42	A55	88
A92	70	A92	14
A92	46	A92	72
A96	90	A92	7
A96	25	A96	37
A96	93	A96	28

Notice that in the above two input files, there are many different values for the BY variable called ID. The first value for ID is A01. The second value for ID is A02. For each value of the BY variable in the above two files, there is a correspond set of records. Each set of records is called a BY group. Hence, there are two records for ID=A01, one record from the ALPHA file and one record from the BETA file.

The next table is a copy of the above two input files with a line separating each BY group of records.

INPUT FILE					
ALPHA		BETA			NUM
ID	V_1	ID	V_4	MATCH-MERGE TYPE	OUT REC
A01	23	A01	58	one-to-one	1
A02	99	A02	20	one-to-one	1
		A04	47	zero-to-one	1
A05	56			one-to-zero	1
A10	88	A10	11	one-to-one	1
A25	24	A25	4	many-to-many	2
A25	22	A25	91		
A32	91	A32	1	one-to-many	3
		A32	22		
		A32	61		
A55	83	A55	88	many-to-one	3
A55	19				
A55	42				
A92	70	A92	14	few-to-many	3
A92	46	A92	72		
		A92	7		
A96	90	A96	37	many-to few	3
A96	25	A96	28		
A96	93				

Match-merging is accomplished one BY group at a time. The above table also names eight types of match-merging. Each type is associated with a BY group. The 8 different types of merges illustrated in the previous table are:

1) zero-to-one
2) one-to-zero
3) one-to-one
4) one-to-many
5) many-to-one
6) few-to-many
7) many-to-few
8) many-to-many

The way SAS matches records depends upon how many records are in each of the input files for a given BY group. For this reason, it is very important to be able to recognize BY groups and the corresponding type of match-merge.

In theory, all types of match-merges can coexist in the input files, and they can be arranged in any order and combination. In practice, one usually only finds one or several types of match-merges appearing in the input files.

The number of output records from a match-merge is determined by looking at each of the input BY groups. The previous table shows the number of output records (see the column entitled NUM OUT REC) that would result from match-merging the ALPHA and BETA files for each BY group. The table implies that the number of output records for a given BY group is equal to the largest number of records in any of the corresponding input files for that BY group. For example, look at the group of records for ID=A92 in the above table. For ID=A92, there are 2 input records in the ALPHA data set and 3 input records in the BETA data set, thus, the number of output records for a merge of the two data sets is 3.

BY groups are at the very heart of the SAS match-merge process. The number of output records depends on the characteristics of the BY group. Variables are initialized and retained by BY groups. The value of the IN= data option can be easily ascertained by observing the BY groups. The automatic variables FIRST.variable and LAST.variable are controlled by BY groups.

Furthermore, the match-merge process itself looks at the input records on a BY group by BY group basis, i.e., SAS looks at the first BY group and processes it. Then, it looks at the second BY group and processes it. And, so on.

For each BY group there is a specific type of match-merge.

The following sections will examine how each type of match-merge is processed and how the corresponding output records are formed.

1-TO-1, 0-TO-1, AND 1-TO-0 MATCH-MERGE

The workings of one-to-one, zero-to-one, and one-to-zero match-merge are fairly intuitive and have already been examined in the first example given in this paper.

(IMPORTANT: Please note that this paper distinguishes between a one-to-one merge and a one-to-one Match-merge. The one-to-one merge is a SAS DATA step with *no* BY statement. Whereas the match-merge is a DATA step WITH a BY statement. A one-to-one match-merge is a name given to that part of a match-merge which corresponds to a BY group with only one record in each of the input data sets.)

MANY-TO-MANY MATCH-MERGE

The many-to-many type of match-merge occurs when, for a given BY group, there are the same number of records in all the input data sets. A 2-to-2 merge, a 3-to-3 merge, a 6-to-6-to-6 merge, or a 10-to-10 merge are all examples of the many-to-many merge. In this type of merge, the 1st record from each input data set is combined with the first record from each of the other input data sets to form a single output record. Then all the second records within the BY group are combined. This process continues until all the last records in the BY group are combined into one last output record. The following is an example of a many-to-many match-merge for one BY group.

```
------------------
DATA ALPH_BET;
  MERGE ALPHA BETA;
  BY ID;
RUN;
------------------
```

```
----------      ---------       -------------
FILE ALPHA      FILE BETA       FILE ALPH_BET
----------      ---------       -------------
 ID   V_1        ID  V_4        ID   V_1  V_4
--------         -------        -------------
A25   24        A25   4         A25   24    4
A25   22        A25  91         A25   22   91
A25   76        A25  38         A25   76   38
```

The many-to-many match-merge is essentially a one-to-one (non-BY, non-match) merge and has the same drawbacks and dangers. Specifically, one has very little control over the actual order of the records within the BY group for each of the input data sets.

For example, how does one know that the first value of V_1 (24) is supposed to be matched with the first value of V_4 (4). Why shouldn't V_1=24 be matched with V_4=91 (the second value of V_4)? If great care is not taken, a many-to-many merge can result in random matching of variable values.

A many-to-many match-merge is DANGEROUS and often unreliable. Having this type of BY group in an input data set often points to a key variable that does not sufficiently identify/label/distinguish the different records in at least one of the input data sets. Perhaps additional BY variables are required to make a proper match, or perhaps the BY variables themselves are faulty (see the BY VARIABLE and associated traps section for a discussion of faulty BY variables).

FEW-TO-MANY MATCH-MERGE

The few-to-many and the many-to-few merges are essentially the same type of merge. Both types of merges are akin to a many-to-many merge and both are DANGEROUS.

The few-to-many type of match-merge occurs when for a given BY group, there is more than one record in the first input data set, and the second input data set has more records than the first. A 2-to-3 merge, a 3-to-5 merge, a 8-to-10, or a x-to-y (where y>x>1) merge are all examples of the few-to-many match-merge.

In this type of merge the "few" (=x) records are matched using one-to-one correspondence. For all practical purposes, few-to-many (or x-to-y merge) match-merge is the same as a many-to-many merge (or a 1-to-1 non-match non-BY merge) for the first "few" records. For the last y-x+1 records in the BY group, the match-merge acts like a one-to-many match-merge which is described later. An example of a few-to-many match-merge follows. (This merge is performed using the code given in the MANY-TO-MANY MATCH-MERGE section.)

```
----------      ---------       -------------
FILE ALPHA      FILE BETA       FILE ALPH_BET
----------      ---------       -------------
 ID   V_1        ID  V_4        ID   V_1  V_4
--------         -------        -------------
A92   70        A92  14         A92   70   14
A92   46        A92  72         A92   46   72
                A92   7         A92   46    7
```

Since essentially the first few records in the BY group of a few-to-many perform a many-to-many match-merge, the few-to-many has the same dangers and the many-to-many match-merge. Both merges are dangerous for the same reasons. Please see the Many-To-Many Match-merge section for a discussion of the dangers and possible solutions.

Additionally, since the last part of a few-to-many is a 1-to-many match-merge, the few-to-many has the danger that the order of the data sets in the MERGE statement can be significant. See the next section for a discussion of this danger.

ONE-TO-MANY MATCH-MERGE

The simplest, and most useful, merge after the one-to-one match-merge is the one-to-many match-merge. Please consider the following example.

```
------------------
DATA ALPH_BET;
  MERGE ALPHA BETA;
  BY ID;
RUN;
------------------
```

```
----------     ---------      -------------
FILE ALPHA     FILE BETA      FILE ALPH_BET
----------     ---------      -------------
ID    V_1      ID   V_4       ID   V_1  V_4
--------       -------        -------------
A32     5      A32   15       A32   5    15
A35     3      A32   22       A32   5    22
               A32   61       A32   5    61
                              A35   3     .
```

In this merge there are two BY groups. The first output record is the same as in a one-to-one match-merge. But for the second record in the BETA file there is no corresponding ALPHA record, so SAS retains the V_1 value from the first ALPHA record and passes it to the second output record.

It is as if there were a RETAIN statement for all the variables in the input data sets. This automatic retain stays in effect during the BY group. When a new BY group is started the value of all the merged variables is set to missing.

The fact that all the values of all the merge variables are retained throughout the execution of the BY group is important and makes the one-to-many match-merge a little tricky. Consider the same example with one additional line in the DATA step.

```
------------------
DATA ALPH_BET;
  MERGE ALPHA BETA;
  BY ID;
  V_1=V_1*2;
RUN;
------------------
```

```
----------     ---------      -------------
FILE ALPHA     FILE BETA      FILE ALPH_BET
----------     ---------      -------------
ID    V_1      ID   V_4       ID   V_1  V_4
--------       -------        -------------
A32     5      A32   15       A32   10   15
A35     3      A32   22       A32   20   22
               A32   61       A32   40   61
                              A35    6    .
```

See how the V_1 variable is changing in value for ID=A32! This may be counter-intuitive, but is exactly what is supposed to happen when a value of a variable is retained.

If the intention is for V_1 to have the same value (in this case, 10) for the whole BY group, the following code would provide such a result, provided that there is always exactly one record in the ALPHA data set for each of the BY groups.

```
------------------
DATA ALPH_BET;
  MERGE ALPHA BETA;
  BY ID;
  IF FIRSTS THEN V_1=V_1*2;
RUN;
------------------
```

When writing code predicated on some assumption about the data, it is always good programming practice to write additional code to check that the assumption is actually true. All too frequently the actual data one receives in a file does not conform with the file specifications.

We illustrate this principle by applying it to the previous example. There, a line of code (IF FIRSTS THEN V_1=V_1*2;) was added to force V_1 to behave as desired. However, that line of code was added based on the assumption that all the BY groups had exactly one record in the first input file. Thus, the code would execute properly on a 1-to-0 and a 1-to-1 BY group as well as a 1-to-many (missing values for V_1 will be accepted). The following code adds a statement that checks that there is one, and only one, ALPHA record in each BY group:

```
------------------
DATA ALPH_BET;
  MERGE ALPHA(IN=INA) BETA;
  BY ID;
  IF FIRSTS THEN V_1=V_1*2;
  IF INA=0 & FIRSTS   ¬OR
     IMAM   & first.ID=0  THEN
     PUT "Tiff than 1 Alpha Rec for By group";
RUN;
------------------
```

OVERLAPPING VARIABLES: A ONE-TO-MANY MATCH-MERGE

The classical idea of a match-merge is to take some variables from one file and ADD them (or tack them on) to a second file. But with SAS, it is possible to have the same variables in both input files. When two input files have some non-key variables in common, those variables are called overlapping variables.

Let's look at another example of a one-to-many merge. This example is the same as the first one-to-many merge we saw. The only difference is this has one overlapping variable, V_4.

```
------------------
DATA ALPH_BET;
  MERGE ALPHA BETA;
  BY ID;
RUN;
------------------
```

```
------------     ---------      -------------
 FILE ALPHA      FILE BETA      FILE ALPH_BET
------------     ---------      -------------
ID  V_1  V_4      ID  V_4       ID  V_1  V_4
------------      -------       -------------
A32   5   10      A32  15       A32   5   15
A35   3    6      A32  22       A32   5   22
                  A32  61       A32   5   61
                                A35   3    6
```

Please note how the value of V_4 in the second merge file overwrites the value of V_4 in the first merge file. Even a missing value can overwrite a valid value.

Since overlapping variables cause SAS to overwrite values, the merit of the overlapping variable is often dubious. Obviously, it is a potentially dangerous situation to have values overwriting each other.

If two input files legitimately have several variables in common, in most situations they would also have the same values in both files. If these two files were to be merged, a good defensive strategy would be to check that the two files do indeed have the same values during the merge. This can easily be accomplished by renaming the common variables in one file, comparing the two values during the merge, and dropping the renamed variables.

Sometimes a variable in one file inadvertently has the same name as in a second file, and the two variables have different information in them. When this happens one gets inadvertent overlapping and overwriting variables. A simple way to avoid this situation is to compare the number of input variables with the number of output variables. If there are two files and no overlapping variables, the number of output variables (O) should be the sum of number of input variables (I1+I2) less the number of BY variables (B). In formula form, O=(I1+I2)-B. If there are n files and no overlapping variables, then O=(I1+...In)-B*(n-1).

ORDER OF DATA SETS IN MERGE STATEMENT

In the section BY VARIABLE LENGTH, we have already seen how the order of the data sets in the merge statement altered the results when the characteristics of the key variables were different.

Let's look at another example of a one-to-many merge to examine another case of where the order of the data sets in the MERGE statement alters the results. This example is similar the to the last example. The only difference is that the order of the data sets in the MERGE statement is reversed.

```
------------------
DATA BET_ALPH;
  MERGE BETA ALPHA;
  BY ID;
RUN;
------------------
```

```
---------      ------------     ------------
FILE BETA       FILE ALPHA      FILE BET_ALPH
---------      ------------     ------------
 ID  V_4       ID  V_1  V_4     ID  V_4   V_1
-------        ------------     ------------
A32  15        A32   5   10     A32   10    5
A32  22        A35   3    6     A32   22    5
A32  61                         A32   61    5
                                A35    6    3
```

Note that the value of V_4 is different for the first output record! This puts to rest the common misconception that the order of the data sets in the MERGE statement has no effect on the value of the resulting variables. As can be seen from this example the order can have a lot to do with the values of the output data set. Whenever there is variable overlap and at least one record in each data set for a BY group, the order of the files in the MERGE statement is significant.

There is a second, and less important, effect of the order of the data sets in the MERGE statement. Specifically, the position of the variables in the output data set is determined by the order of the data sets in the MERGE statement. Observe that in last example the output variables are positioned (in the description part of the data set) as ID V_4 V_1, where as in the second last example, the output variables were positioned as ID V_1 V_4.

THE BIGGEST TRICK OF ALL

This paper's title offers tricks on SAS Merging. The biggest trick of all is to know the SAS Algorithm for the match-merge. Whoever studies the algorithm and gets it down cold, comes a long way in avoiding ALL the traps in the Merge.

The match-merge is just a special type of a DATA Step. So if one knows the DATA Step algorithm, it is fairly easy to take the DATA Step algorithm and embellish it a little to have the MERGE algorithm.

There are many ways to describe the algorithm. One description is given on page 151 of SAS Language: Reference (Version 6, First Edition), under the title "DATA Step Processing during Match-Merging".

This paper has examined most of the more tricky aspects of the algorithm. The next section summarizes what to look out for.

CONCLUSION

Sometimes the way the match-merge works is counter-intuitive.

To insure a successful merge , it is always good to know what types of match-merges are going to be involved and anticipate how records in those kinds of merges are handled. Few-to-many, many-to-few, and many-to-many match-merges should be avoided. To avoid unexpected types of merges, code should be added to the match-merge DATA step to check that only the expected types are occurring in the actual merge.

Overlapping variables should be avoided. Whenever there is variable overlap and at least one record in each data set for a BY group, the order of the files in the MERGE statement is significant. Inadvertent overlapping variables (including inadvertently same-named variables) can be checked for by comparing the variable counts in the input and output data sets.

The order of the data sets in the merge statement will change the position of the variables in the output data set.

The attributes (ex: LENGTH, position) of overlapping variables (including the key variables) are determined by the first input data set in which they appear. Different attributes and characteristics (ex:justification) in the overlapping variables can cause unexpected matching of records. Thus, the attributes and characteristics of all overlapping (including key) variables in all of the input files should be checked to be sure that they are the same.

BY variables should be designed and chosen carefully, so that they truly and unambiguously identify, label, and distinguish each record in a data set.

Finally, always check that the match-merge DATA step has a BY statement.

REFERENCE

SAS Institute, Inc. (1990), SAS Language: Reference, Version 6, First Edition, Carry, NC: SAS Institute Inc.

TRADEMARKS

AUTHOR CONTACT

The author welcomes comments, questions, corrections and suggestions.

Malachy J. Foley
2502 Foxwood Dr.
Chapel Hill, NC 27514

Email: FOLEY@unc.edu