PharmaSUG China 2017 - Paper 24

Automatic Retention with MERGE Statement

Yuni Chen, Roche(China) Holding Ltd., Shanghai, China

ABSTRACT

As a SAS programmer, we may get very much familiar with MERGE statement, which will be used by quite a few of us whenever we need to combine some variables. Sometimes, however, we may get UNEXPECTED result when doing data manipulation to the variable joined from MERGE statement. In this paper, the author will use examples to demonstrate the AUTOMATIC RETENTION with MERGE statement and how to resolve the possible unexpected result. It is always a good way to learn from other's mistakes!

INTRODUCTION

As SAS® documentation describes the DATA step MERGE: "The MERGE statement Joins observations from two or more SAS data sets into a single observation"[1], we know that MERGE statement is one of the most frequently used ways to join observation in SAS. And sometimes within the same DATA step, we need to do further data manipulations after MERGE statement. Have you ever met situations when the data manipulation was for the variable joined from MERGE statement and the result was not what you expected? The reason probably lies in that, variables read by the MERGE statements are automatically retained between iterations of the DATA step. Below in this article, the author will use the real case together with some simple example to demonstrate how the unexpected results come along with the automatic retention and what we need do to avoid the undesirable outcome.

EXAMPLE 1 – REAL CASE IN CLINICAL STUDY

Let's firstly take a look at one real example. The author got unexpected result in this real example when resetting the value of the variable read by MERGE statement.

DATA SETS

Below are the two data sets we are going to merge. Those data sets have been modified a bit for the sake of data confidentiality and for better illustrating the point in this paper. Additionally, the column row here is not an original variable in the data sets, added for the purpose of distinguishing the data by audiences.

row	usubjid	paramcd	ontrtfl	avisit	gradeh
1	1301	ALT	N	BASELINE	1
2	1301	ALT	Y	WEEK 12	1
3	1301	ALT	Y	WEEK 24	3
4	1301	AST	N	BASELINE	2
5	1301	AST	Y	WEEK 12	1
6	1301	AST	Y	WEEK 24	2
7	1302	ALT	N	BASELINE	0
8	1302	ALT	Y	WEEK 12	1
9	1302	ALT	Y	WEEK 24	0
10	1302	AST	N	BASELINE	1
11	1302	AST	Y	WEEK 12	0
12	1302	AST	Y	WEEK 24	1

Table 1. Content of Data Set ONE

row	usubjid	paramcd	gradeh	grhpt
1	1301	ALT	3	Y
2	1301	AST	2	Y
3	1302	ALT	1	Y
4	1302	AST	1	Y

Table 2. Content of Data Set TWO

SAS CODE

Note that, there are three common variables usubjid, paramcd, and gradeh in data set one and two. We would like to merge those two by the three common variables:

```
proc sort data=one; by usubjid paramcd gradeh; run;
data final;
    merge one two;
    by usubjid paramcd gradeh;
    if ontrtfl ne 'Y' then grhpt='';
run;
```

DESIRED RESULT

There is an IF statement after MERGE and BY in the above DATA step. What we expect to see is as below:

row	usubjid	paramcd	ontrtfl	avisit	gradeh	grhpt
1	1301	ALT	N	BASELINE	1	
2	1301	ALT	Y	WEEK 12	1	
3	1301	ALT	Y	WEEK 24	3	Y
4	1301	AST	Y	WEEK 12	1	
5	1301	AST	**N**	BASELINE	2	
6	1301	AST	Y	WEEK 24	2	Y
7	1302	ALT	N	BASELINE	0	
8	1302	ALT	Y	WEEK 24	0	
9	1302	ALT	Y	WEEK 12	1	Y
10	1302	AST	Y	WEEK 12	0	
11	1302	AST	**N**	BASELINE	1	
12	1302	AST	Y	WEEK 24	1	Y

Table 3. Table of Desired Result

All we would like to do with the IF statement in the same DATA step is to change values of grhpt on row 5 and 11 which correspond to ontrtfl of "N" (highlighted in red in **Table 3**) to missing, leaving everything else as is after MERGE statement.

ACTUAL OUTCOME

However, the final outcome turns out to be:

row	usubjid	paramcd	ontrtfl	avisit	gradeh	grhpt
1	1301	ALT	N	BASELINE	1	
2	1301	ALT	Y	WEEK 12	1	
3	1301	ALT	Y	WEEK 24	3	Y
4	1301	AST	Y	WEEK 12	1	
5	1301	AST	**N**	BASELINE	2	
6	1301	AST	Y	WEEK 24	2	??
7	1302	ALT	N	BASELINE	0	
8	1302	ALT	Y	WEEK 24	0	
9	1302	ALT	Y	WEEK 12	1	Y
10	1302	AST	Y	WEEK 12	0	
11	1302	AST	**N**	BASELINE	1	
12	1302	AST	Y	WEEK 24	1	??

Table 4. Content of Data Set FINAL

As it can be seen from **Table 4**, values of grhpt on row 5 and 11 are missing now. However, those of grhpt on row 6 and 12 are missing as well, which we really need to be originally as “Y”. Why? The following paragraphs will be talking about how SAS® processes the data and result in this undesired outcome.

EXAMPLE 2 – DETAILED ILLUSTRATION OF DATA PROCESSING

As we know, a key operational component of SAS is the Program Data Vector(PDV). With knowledge of how PDV functions, programmers can better understand how SAS works. Let’s see how data is being processed based on PDV. Two much more simple data sets scores and levels are being used here.

DATA SETS

name	test	score
Alice	1	84
Alice	2	93
Alice	3	100
Alice	4	65

Table 5. Content of Data Set SCORES

name	level
Alice	new

Table 6. Content of Data Set LEVELS

SAS CODE

We need to read the level variable from data set levels and create a new variable grade in the final data. The SAS code is as below:

```
data results;
    merge scores levels;
    by name;

    if score > 92 then do;
           level = 'scholar';
           grade = 'A';
    end;
run;
```

DATA PROCESSING

As you read through what follows, you'll want to refer to both the DATA step code and the two data sets. As always, at compilation time, SAS creates the program data vector. In this case, it contains the automatic variables (_N_ and _Err_, _Err here is abbreviated for _ERROR_ for sake of space), the three variables in the scores data set (name, test, and score), a variable in the levels data set (level), one variable defined within the DATA step (grade), and as a result of the BY statement, a first.name, a last.name. Here's what the PDV looks like at the beginning of the first iteration of the DATA step, note that initial values for first.name and last.name are both 1:

_N_	_Err_	name	test	score	level	grade	first.name	last.name
1	0		.	.			1	1

Table 7. Content of PDV When Beginning the First Iteration

Now, SAS reads the first observation from the scores data set. there is an observation in the levels data set for which name = Alice. The level variable is read from the levels data set, getting a value of new. The observation is the first in the group of name, therefore first.name is assigned the value of 1 and last.name is assigned a value of 0. Because the condition in IF statement is not met, the variable level and grade remain to be new and missing, respectively. Here's what the program data vector looks like now:

_N_	_Err_	name	test	score	level	grade	first.name	last.name
1	0	Alice	1	84	new		1	0

Table 8. Content of PDV When Reading the First Observation

As we've reached the end of the first iteration of the DATA step, SAS writes the contents of the PDV to the output results data set. In so doing, the internal variables (_N_, _Err_, first.name, last.name) are dropped. Here's what the final data set looks like after the first iteration of the DATA step:

name	test	score	level	grade
Alice	1	84	new	

Table 9. Content of the Results Data Set When the First Iteration is complete

Right before it begins the second iteration, SAS keeps values of each variable as is in the first iteration except that of the newly created variable grade. The grade variable is initialized to missing in the beginning of each iteration:

_N_	_Err_	name	test	score	level	grade	first.name	last.name
2	0	Alice	1	84	new		1	0

Table 10. Content of PDV When Beginning the Second Iteration

Note here the value of the level variable is automatically retained as new. Now, SAS reads the second observation from the scores data set. Again, there is an observation in the levels data set for which name = Alice. The observation is neither the first nor the last in the group of name, therefore first.name and last.name are both assigned a value of 0. Differently from the first iteration, the condition in IF statement is met this time. Hence the value of level is changed to scholar and that of grade is set to A. Here's what the PDV looks like now:

_N_	_Err_	name	test	score	level	grade	first.name	last.name
2	0	Alice	2	93	scholar	A	0	0

Table 11. Content of PDV When Reading the Second Observation

As we've reached the end of the second iteration of the DATA step, SAS writes the contents of the program data vector to the output results data set, dropping appropriate variables along the way. Here's what the final data set looks like after the second iteration of the DATA step:

name	test	score	level	grade
Alice	1	84	new	
Alice	2	93	scholar	A

Table 12. Content of the Results Data Set When the Second Iteration is complete

Now SAS starts a new iteration, similar as the first one, it keeps values of variables from the previous iteration except for that of grade:

_N_	_Err_	name	test	score	level	grade	first.name	last.name
3	0	Alice	2	93	scholar		0	0

Table 13. Content of PDV When Beginning the Third Iteration

Again, the value of the level variable is automatically retained from the previous iteration. Now, SAS reads the third observation from the scores data set. Same as before, an observation is read from the levels data set for which name = Alice and first.name and last.name are both assigned a value of 0 as the observation is neither the first nor the last in the group of name variable. The condition in IF statement is met again this time, therefore, the value of level variable is scholar and that of grade variable is A. Here's what the program data vector looks like now:

_N_	_Err_	name	test	score	level	grade	first.name	last.name
3	0	Alice	3	100	scholar	A	0	0

Table 14. Content of PDV When Reading the Third Observation

Because we've reached the end of the third iteration of the DATA step, SAS writes the contents of the program data vector to the output results data set, dropping appropriate variables along the way. Here's what the final data set looks like after the third iteration of the DATA step:

name	test	score	level	grade
Alice	1	84	new	
Alice	2	93	scholar	A
Alice	3	100	scholar	A

Table 15. Content of the Results Data Set When the Third Iteration is complete

As SAS starts the next iteration, it keeps values of variables other than that of variable grade from the previous iteration:

_N_	_Err_	name	test	score	level	grade	first.name	last.name
4	0	Alice	3	100	scholar		0	0

Table 16. Content of PDV When Beginning the Fourth Iteration

Again, the value of the level variable is automatically retained from the previous iteration. Now, SAS reads the fourth observation from the scores data set. There is an observation read from the levels data set for which name = Alice. The observation is the last in the group of name variable, therefore *last.name* is assigned the value of 1 and *first.name* is assigned a value of 0. Attention here, as the condition in IF statement is not satisfied (score variable has a value of 65, which is less than 92), nothing will be done with this statement. Therefore, the level variable remains the value of scholar as retained and the grade variable is missing. Here's what the PDV looks like now:

_N_	_Err_	name	test	score	level	grade	first.name	last.name
4	0	Alice	4	65	scholar		0	1

Table 17. Content of PDV When Reading the Fourth Observation

Because we've reached the end of the fourth iteration of the DATA step, SAS writes the contents of PDV to the output results data set, dropping appropriate variables along the way. Here's what the final data set looks like after the fourth iteration of the DATA step:

name	test	score	level	grade
Alice	1	84	new	
Alice	2	93	scholar	A
Alice	3	100	scholar	A
Alice	4	65	scholar	

Table 18. Content of the Results Data Set When the Fourth Iteration is complete

Obviously, it is not something we expected, which is thought to be:

name	test	score	level	grade
Alice	1	84	new	
Alice	2	93	scholar	A
Alice	3	100	scholar	A
Alice	4	65	new	

Table 19. Content of Expected Results for Results Data Set

RESOLUTION

Based on the above examples, we see that the level variable is automatically retained between iterations of the data step. However, the grade variable is not falling the same as it is newly created in the DATA step. To fix this problem and get what we want, we may use the rationale that newly created variables are not automatically retained. By simply creating a new variable level2 here and renaming it to level, we will get desired result as showing in **Table 19**.

```
data results(rename = (level2 = level));
  merge scores levels;
  by name;

  if score > 92 then do;
   level2 = 'scholar';
   grade = 'A';
  end;
  if level2 = '' then level2 = level;

  drop level;
run;
```

Similarly, we may resolve it in the real example which is showed at the beginning of this article by creating a new variable grhpt2 in the DATA step. The SAS code is as below:

```
proc sort data=one; by usubjid paramcd gradeh; run;
data final(rename=(grhpt2 = grhpt));
    merge one two;
    by usubjid paramcd gradeh;
    if ontrtfl ne 'Y' then grhpt2='';
    else grhpt2 = grhpt;

    drop grhpt;
run;
```

CONCLUSION

Variables that are read by the MERGE statement are automatically retained between iterations of the DATA step. Variables that are created in the DATA step are not automatically retained. To avoid this automatic retention leading to unexpected result, we may use the rationale that newly created variables are not automatically retained. By simply creating a new variable, doing the manipulation based on some criteria, and rename it to the original one read by merge statement, we will get the expected outcome. Please do not forget the attribute for this new variable if there were some for the original one.

NOTES

[1] refer to the "SAS® 9.2 Language Reference Dictionary", Fourth Edition, Page1679, MERGE Statement

REFERENCES

SAS SUPPORT, SAMPLES & SAS NOTES, http://support.sas.com/kb/48/147.html

ESSENTIALS of the PROGRAM DATA VECTOR (PDV), http://support.sas.com/resources/papers/proceedings13/125-2013.pdf

CONTACT INFORMATION <HEADING 1>

Your comments and questions are valued and encouraged. Contact the author at:

Name: Yuni Chen
Enterprise: Roche(China) Holding Ltd,.
Address: Floor 4, Building 11, 1100 Longdong Avenue
City, State ZIP: Shanghai, 201203
Work Phone: +862128923169
E-mail: yuni.chen@roche.com