

Paper CS02

Watch out, a MERGE ahead!

Benjamin Szilagyi, Novartis Pharma AG, Basel, Switzerland

Christof Binder, BIOP AG Basel, Switzerland

ABSTRACT

Merging datasets is part of a SAS® programmer's every day life. However, to really understand this functionality and thus know where the programming traps are, one needs to dig deeper into how SAS processes data and the role the program data vector (PDV) plays here.

The intention of this paper is to highlight some dangerous merging techniques using very simple, standard syntax. It will use two common cases of a MERGE with surprising and unintended results. Each MERGE process will be explained step by step including the PDV functionality to provide a clear picture of what happens and how errors can be prevented.

INTRODUCTION

There are many reasons and ways people start programming SAS and the "Learning by Doing" approach seems to be quite common. Although the programmer will be able to use the software reasonably quickly to do a basic analysis of data, he will sooner or later need to acquire a systematic background of the processes he uses. If not, he risks stumbling over some tools which seem to be the most common such as the MERGE functionality for example.

To merge information from two different datasets can be an easy task with SAS and the syntax has been kept very simple by the developers. However, in this paper we will show that this intuitively usable syntax can also entice the programmer to erroneously think himself safe, while the MERGE produces clearly unintended results.

A basic but systematic knowledge of the program data vector (PDV) and how it works during a merge process will help the programmer to prevent such incidents. We will present two examples of very simple merges and their "surprising" results and explain them step by step with the function of the PDV.

The examples will focus on merges followed by conditional data modification within the same datastep and merges with common variables being outside of the BY GROUP.

FIRST CASE, MIXING MERGE AND CONDITIONAL DATA MODIFICATION IN ONE DATASTEP

In this example we will merge two datasets (dataset ONE and dataset TWO) into dataset THREE and conditionally modify a variable stemming from one dataset, dependent on the value of a variable from the other dataset.

This is dataset ONE :

	patID	y	z
1	01_001	aaa	z33
2	01_001	uuu	z22
3	01_002	iii	z11
4	01_003	uuu	z55
5	01_004	eee	z22
6	01_005	ooo	z44

And this is dataset TWO:

	patID	d	f
1	01_001	da	fi
2	01_002	di	fo
3	01_003	du	fu
4	01_004	de	fa
5	01_005	do	fi

The code for merging and modifying the data is:

```
DATA three;
  MERGE one
        two;
  BY patID;
  IF y = 'aaa' THEN d = 'NONE';
RUN;
```

So dependent on the value of variable “y” coming from dataset ONE, we change the variable “d” coming from dataset TWO. The result for this code looks like this:

	patID	y	z	d	f
1	01_001	aaa	z33	NONE	fi
2	01_001	uuu	z22	NONE	fi
3	01_002	iii	z11	di	fo
4	01_003	uuu	z55	du	fu
5	01_004	eee	z22	de	fa
6	01_005	ooo	z44	do	fi

Surprising isn't it? The first observation `_N_ = 1` clearly provides us with the result that we expected since `y = "aaa"`, but on the second observation, having `y = "uuu"`, it also has been changed to “NONE”.

What went wrong? Actually nothing, SAS behaves absolutely in accordance with the specifications and it's simply the programmer who wrote a code, without taking into account of how the PDV works.

Let's look at this merge process step by step from the PDV's perspective (for more detailed information on the PDV please check our reference list of paper's at the bottom)

Step 1

The BY GROUP defined in the code consists of the variable `patID`. The PDV looks into the first observation `_N_ = 1` of dataset ONE and loads this information:

<code>_N_</code>	<code>_ERROR_</code>	<code>patID</code>	<code>y</code>	<code>z</code>	<code>d</code>	<code>f</code>
1	0	01_001	aaa	z33		

The PDV then searches for the matching BY GROUP in dataset TWO and reads in the first observation, taking the IF condition for `y = 'aaa'` into account (`d = 'NONE'`). It then outputs this observation into dataset THREE as expected.

<code>patID</code>	<code>y</code>	<code>z</code>	<code>d</code>	<code>f</code>
01_001	aaa	z33	NONE	fi

Step 2

The PDV goes to the second observation of dataset ONE within the first BY GROUP. And here is the crucial point: SAS retains the values from `_N_ = 1` for variables “d” and “f”, because SAS reinitializes (i.e. sets to “missing”) at the start of the dataset *only* when the BY GROUP changes! It overwrites variables “y” and “z” with the new values from `_N_ = 2`. The PDV then goes to dataset TWO and looks for the next observation within the same BY GROUP and since there is none, the variables “d” and “f” keep their value from `_N_ = 1` in the PDV.

Consequently, the PDV before the implicit output to dataset THREE looks like this.

<code>_N_</code>	<code>_ERROR_</code>	<code>patID</code>	<code>y</code>	<code>z</code>	<code>d</code>	<code>f</code>
2	0	01_001	uuu	z22	NONE	fi

So what is the safest way to prevent this from happening in your code? We offer you two simple strategies:

- **Keep the merge and data modification separate**

Simply create two datasteps, the first one doing the merge and the second one modifying the data. This way the PDV simply works with the IF condition, without taking any other factors into account.

The advantage of this solution lies within its simplicity. However for big datasets, execution time will increase.

- **Assign the values to a new variable**

Rename the value which needs to be changed at the source dataset and create a new variable having the original name of the variable of your interest:

```
DATA three;
  MERGE one
        two(RENAME d = newvar );
  BY patID;

  IF y = 'aaa' THEN d = 'NONE';
  ELSE d = newvar;
  DROP newvar;
RUN;
```

Obviously only one datastep is needed here, however a junior programmer might not really understand why this is necessary. Also, attributes of variable "d" are lost.

SECOND CASE, MERGING COMMON VARIABLES OUTSIDE OF THE BY GROUP

Common variables in two datasets „in the back of the data“ (e.g. baseline values, treatment group variables etc...) can give surprising results if inconsistencies within the same BY GROUP are present. The general rule „right dataset overwrites left“ doesn't always work the way we would expect!

Again we are merging two datasets (ONE and TWO) into dataset THREE by patID.

This is dataset ONE :

	patID	y
1	01_001	111
2	01_001	222
3	01_001	333
4	01_002	111
5	01_002	222

And this is dataset TWO :

	patID	y
1	01_001	yyy
2	01_002	yyy

The code for the data is:

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

Intuitively one would expect the value for variable "y" on dataset ONE being overwritten by the values from dataset TWO (e.g. all values for "y" in dataset THREE being "yyy").

However, the above merge produces the following dataset THREE:

	patID	y
1	01_001	yyy
2	01_001	222
3	01_001	333
4	01_002	yyy
5	01_002	222

Let's have a look at this merge again from the PDV's perspective.

Step 1

The PDV reads in the first observation of dataset ONE.

N	_ERROR_	patID	y
1	0	01_001	111

The PDV then searches for the matching BY GROUP in dataset TWO and reads in the first observation, overwriting, as expected, the y-value from “111” to “yyy”. This observation gets output to dataset THREE.

patID	y
01_001	yyy

Step 2

In the next iteration, the PDV reads in the second observation within the same BY GROUP patID = “01_001” .

Unlike in the first example, this time SAS replaces the value of “y”, because the variable exists already on dataset ONE and SAS is reading a new record.

N	_ERROR_	patID	y
2	0	01_001	222

The PDV then goes to dataset Right to search for a matching observation within the same BY GROUP. And here is the tricky point: because there is no observation within the same BY GROUP, SAS retains the value y = “222” from dataset ONE and outputs into dataset THREE!

patID	y
01_001	222

In comparison to SQL joins, MERGE will process the code without warning the user in the log that there are common variables in the datasets. However, using the option

OPTIONS msgLevel = i

will create a note in the log file, if common variables are present during a merge. As a general advise, we recommend to only KEEP those variables in the datasets to be merged, which are really needed.

CONCLUSION

The examples we have presented in this paper show how easy it is to fall into traps even with very simple merges. Besides applying the suggested strategies, we consider it essential that professional programmers acquire sufficient knowledge of how the PDV works, as well as the correct use of system options (e.g. msgLevel or mergeNoBy). This not only makes their merges safer but also improves the efficiency of their code in the whole datastep.

ACKNOWLEDGMENTS

The authors would like to thank Dante G. diTommaso for his very valuable input to the paper’s content and for polishing the “Swiss – English” into a more colloquial style!

RECOMMENDED READING

Two excellent paper's on the PDV which should find their place in every SAS programmer's bookshelf:

The Use and Abuse of the Program Data Vector

Jim Johnson, PHARMASUG 2003

<http://www.lexjansen.com/pharmasug/2003/technicaltechniques/tt070.pdf>

The Use and Abuse of the Program Data Vector (The Procs)

Jim Johnson, PHARMASUG 2005

<http://www.lexjansen.com/pharmasug/2005/technicaltechniques/tt03.pdf>

CONTACT INFORMATION

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

Benjamin Szilagyi
Novartis Pharma AG
WSJ-27.5.009
CH-4002 Basel
Switzerland
Tel.: +41 (0) 61 324 23 69
Fax: +41 (0) 61 324 89 22
email: Benjamin.Szilagyi@novartis.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.