Paper T09-2009

Where Does This Where Go?

Scott Davis, COMSYS, Portage, MI

ABSTRACT

The clever folks at SAS® have given the user community a variety of ways to accomplish tasks. Being the creative bunch that we are, SAS programmers have yet to tire of discovering and using different paths to get to the same end. In that spirit, this paper focuses on a very popular SAS statement, the WHERE statement, and areas that can pose a problem when it is combined with PROC SORT.

In a typical clinical study (read: ALL) there will be adverse events. Many times programmers will find themselves doing validation on an adverse event table that either they created or was created by someone else. We all know the power of the NODUPKEY option in PROC SORT, and countless experts have forewarned about the dangers that come with its use. For those of us who have heeded those warnings and proceeded with care, it turns out that another danger awaits when using a subsetting WHERE clause in your PROC SORT.

This paper explores the placement of a WHERE statement in a PROC SORT during input and output and how very different the results can be depending on your data.

INTRODUCTION

Programmers are required to check their code for a variety of reasons. How the checking is performed, and who performs it can vary greatly. Often times we find ourselves writing independent code to check the validity of an existing program.

In the clinical world, the task is usually related to number validation, i.e., are the values being produced for the table accurate? When checking the numbers, we usually don't have to worry about the appearance of the output since the validation program can often use 'quick and dirty' techniques to generate the output. A quick sort of the data, a frequency run, and ta-da, numbers are all generated. It usually is that easy, but sometimes…

I encountered some unexpected results when combining PROC SORT with a WHERE clause and the NODUPKEY option. I was checking the numbers for an adverse event table and I thought it was going to be easy. It's important to note here that, at the time, the table I was checking contained incorrect results. Had the numbers been correct I may never have followed the rabbit down this hole.

PROBLEM

I've often said that a programmer's knowledge of SAS is what's key, and the data is just data. I've used this "theory" in discussion (at times successfully, and other times not so much) to suggest that if your SAS knowledge is strong then you should be able to transfer that working knowledge to virtually any industry, be it pharmaceutical, financial, retail, etc. While I firmly believe that, I also believe that you need to become familiar with the data set(s) involved in your work.

When working with data, it is imperative to have a deep enough understanding so that you are able to process it and get to the desired end. Beyond knowing the variables, their types, lengths and whatnot, it is also very helpful to know how key variables are related and what they mean. In doing so, you are well prepared to define what you expect as the source programmer, or as the validation programmer, to be able to verify if the source programmer reached the desired end. You can then correctly identify issues and work to understand how to fix them. Sometimes (never if your boss is reading this) there may be a syntax error or other misuse that the programmer has introduced to the code, other times (almost always unless you are in data management) there is a problem with the data.

ANATOMY OF THE PROC SORT

The data set I am using to demonstrate the problem consists of a handful of variables from a general adverse event data set. Here's an example:

```
data testae;
   infile datalines dsd;
   input trt pt serious : $char1. aesoc : $char30. aept : $char25.
         aestart : date7. aestop : date7.;
```

```
   format aestart aestop date7.;
   datalines;
1,1001,Y,Gastrointestinal disorders,Nausea,01Jun08,03Jun08
1,1001,N,Gastrointestinal disorders,Nausea,04Jun08,04Jun08
1,1002,N,Gastrointestinal disorders,Stomach pain,01Jun08,02Jun08
1,1002,N,Gastrointestinal disorders,Stomach pain,06Jun08,08Jun08
1,1002,Y,Gastrointestinal disorders,Nausea,01Jun08,02Jun08
1,1002,N,Gastrointestinal disorders,Stomach pain,04Jun08,05Jun08
1,1002,N,Gastrointestinal disorders,Stomach pain,06Jun08,07Jun08
1,1003,N,Gastrointestinal disorders,Nausea,21Jun08,21Jun08
1,1003,N,Gastrointestinal disorders,Nausea,24Jun08,25Jun08
1,1003,N,Gastrointestinal disorders,Nausea,26Jun08,26Jun08
1,1003,Y,Gastrointestinal disorders,Nausea,27Jun08,29Jun08
1,1003,N,Gastrointestinal disorders,Nausea,30Jun08,30Jun08
1,1004,Y,Gastrointestinal disorders,Stomach pain,11Jun08,15Jun08
1,1004,N,Gastrointestinal disorders,Stomach pain,21Jun08,22Jun08
;
run;
```

Here's the listing of the testae dataset:

Obs	trt	pt	serious	aesoc	aept	aestart	aestop
1	1	1001	Y	Gastrointestinal disorders	Nausea	01JUN08	03JUN08
2	1	1001	N	Gastrointestinal disorders	Nausea	04JUN08	04JUN08
3	1	1002	N	Gastrointestinal disorders	Stomach pain	01JUN08	02JUN08
4	1	1002	N	Gastrointestinal disorders	Stomach pain	06JUN08	08JUN08
5	1	1002	Y	Gastrointestinal disorders	Nausea	01JUN08	02JUN08
6	1	1002	N	Gastrointestinal disorders	Stomach pain	04JUN08	05JUN08
7	1	1002	N	Gastrointestinal disorders	Stomach pain	06JUN08	07JUN08
8	1	1003	N	Gastrointestinal disorders	Nausea	21JUN08	21JUN08
9	1	1003	N	Gastrointestinal disorders	Nausea	24JUN08	25JUN08
10	1	1003	N	Gastrointestinal disorders	Nausea	26JUN08	26JUN08
11	1	1003	Y	Gastrointestinal disorders	Nausea	27JUN08	29JUN08
12	1	1003	N	Gastrointestinal disorders	Nausea	30JUN08	30JUN08
13	1	1004	Y	Gastrointestinal disorders	Stomach pain	11JUN08	15JUN08
14	1	1004	N	Gastrointestinal disorders	Stomach pain	21JUN08	22JUN08

Table 1. Testae dataset

I used the DATA=, OUT=, and NODUPKEY options along with the required BY statement. Since I was interested in a particular subset of the data, I also chose to add a WHERE statement to my PROC SORT. I placed the WHERE= option on the input data set.

```
proc sort data=testae(where=(serious='Y')) out=testae_input nodupkey;
   by trt pt aesoc;
run;
```

The data looks like this with the above WHERE clause applied:

Obs	trt	pt	serious	aesoc	aept	aestart	aestop
1	1	1001	Y	Gastrointestinal disorders	Nausea	01JUN08	03JUN08
2	1	1002	Y	Gastrointestinal disorders	Nausea	01JUN08	02JUN08
3	1	1003	Y	Gastrointestinal disorders	Nausea	27JUN08	29JUN08
4	1	1004	Y	Gastrointestinal disorders	Stomach pain	11JUN08	15JUN08

Table 2. Testae dataset with WHERE applied to data=

A frequency on this data produced these results:

Table of aesoc by trt		
aesoc	trt	
	1	Total
Gastrointestinal disorders	4	4
Total	4	4

Table 3. Frequency of testae dataset with WHERE applied to data=

Since these results did not match to the table that I was trying to validate, I found myself slicing and dicing the data differently to see what the issue might be. One approach I decided to take was to place the WHERE option on the output data set.

```
proc sort data=testae out=testae_output(where=(serious='Y')) nodupkey;
   by trt pt aesoc;
run;
```

The data looks like this with the above WHERE clause applied:

Obs	trt	pt	serious	aesoc	aept	aestart	aestop
1	1	1001	Y	Gastrointestinal disorders	Nausea	01JUN08	03JUN08
2	1	1004	Y	Gastrointestinal disorders	Stomach pain	11JUN08	15JUN08

Table 4. Frequency of testae dataset with WHERE applied to data=

A frequency on this data produced these results:

Table of aesoc by trt		
aesoc	trt	
	1	Total
Gastrointestinal disorders	2	2
Total	2	2

Table 5. Frequency of testae dataset with WHERE applied to out=

What's going on here? The location of the where clause causes SAS to produce different results..

CONCLUSION

When the WHERE clause is on the DATA= option, SAS will apply the WHERE first and then do the sort with the NODUPKEY.

When the WHERE clause is on the OUT= option, SAS will do the sort with the NODUPKEY and then apply the WHERE clause.

This seems painfully obvious when presented in this step-by-step illustration of a small sample data set, but when you're knee-deep in a problem working with a large volume of data, the solution isn't always as clear as we'd like it to be. Sometimes when you are struggling to validate someone else's work, it can become quite challenging to wade through your own mistakes. It's important to be careful and trust your instincts.

ACKNOWLEDGMENTS

I would like to thank Rosie Grzebyk for her support in reviewing this paper.

RECOMMENDED READING

Britta Kelsey. “The Mystery of the PROC SORT Options NODUPRECS and NODUPKEY Revealed”. http://www2.sas.com/proceedings/sugi30/037-30.pdf

CONTACT INFORMATION

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

Scott Davis
COMSYS
5220 Lovers Lane, Suite 200
Portage, MI 49002
Work Phone: 269-553-5122
E-mail: sdavis@comsys.com