Paper 100-2010

Yes, Proc SQL is Great, But I Love Data Step Programming, What's a SAS User To Do?

Mira J. Shapiro, MJS Consultants, Bethesda, MD

ABSTRACT

Like many long-standing SAS users and Data Step programmers, I avoided learning Proc SQL for many years. After making the decision to take the plunge and add Proc SQL to my toolkit of techniques, I went overboard and began using Proc SQL for everything. This paper describes my approach to choosing between Proc SQL and the Data Step for particular tasks. This paper is appropriate for beginning SAS users who have a basic understanding of the Data Step and Proc SQL.

INTRODUCTION

Like many operations in SAS there are multiple ways to achieve the same results. Many SAS users have their favorite techniques and sometimes aren't able to take the time to explore alternatives that may be more appropriate in various situations. Since the most important step in any analysis is data preparation, it is essential that the SAS professional to be able to efficiently and accurately design their analysis data set. Selecting the right combination of techniques including Proc SQL and the Data Step will facilitate not only faster run times, but also result in an approach that is easier to document and share and, is more likely to be reusable, at least in part, for other projects. By exploring the strengths of Proc SQL with respect to a many-to-many merge task typical in epidemiological case control studies, this paper will demonstrate the power of this technique even with a relatively simple task at hand.

DATA STEP AND PROC SQL OVERVIEW

Both the SAS Data Step and Proc SQL will compute variables, allow for renaming variables, subset data sets based on conditions, merge data sets (join tables in SQL terminology), and a host of other operations. In choosing the appropriate technique for a particular operation, understanding the composition of the data sets is essential. The first step in any data management or analysis project should be to run a series of descriptive procedures such as Proc Freq or Proc Means to gain an understanding of the distribution of your data, potential incorrect or missing values and other data management issues.

There are many resources, some listed below in the Reference section, that go into details of Proc SQL syntax and the differences between the Data Step and Proc SQL. This discussion will focus on one example where a many-to-many merge is completed in one simple Proc SQL step that would have required much manipulation and multiple Data Steps to complete.

THE POWER OF PROC SQL FOR A MANY-TO-MANY OPERATION

In this example we have two data sets for a case control study, one with information about our cases and another with information about our controls. We want to find all the potential healthy controls for our cases. We have been told that we must match the controls to the cases by sex and year of birth. We have also been told that there may be more than one control per case and that is desirable since that will give us an opportunity to randomly select one control per case. Furthermore, many of the variables in both of the data sets have the same name: ssn, y (year of birth), date_of_birth, and sex. For this example, we are going to simplify the problem and just match controls to cases on the variable sex.

The first, and most important use of Proc SQL for Data Step die-hards is the merging of two data sets. In particular, the advantage of using Proc SQL for a many-to-many merge (join in SQL terminology) is that it provides a one step solution. The first circumstance that inspired me to add Proc SQL to my repertoire was a situation where I had a many-to-many merge that would have required a series of data steps and several "tried and true" Data Step programming "tricks". The example to follow demonstrates how Proc SQL accomplishes a many-to-many merge in one easy-to-understand and easy-to-document step.

Before beginning the matching process, the data sets were explored with Proc Freq to ascertain the number of male and female cases and how many potential controls are available. As it turns out, there are 9 female cases and 11 male cases. Likewise there are 9 female controls and 11 male controls. Before continuing, it makes sense to calculate the number of observations we expect the resulting data set to contain. Since there are 9 female cases and 9 potential controls for each, we expect the resulting data set to contain 81 observations representing female cases and their possible controls. Likewise the data set will contain 121 observations representing the 11 male cases and the 11 potential controls for each, resulting in an overall total of 202 records.

Given that there is more than one potential control per case what would happen if we tried to sort and match merge the data sets the variable, sex?

Would something like this work?

```
proc sort data=cases2 out=cases2sorted; by sex;
proc sort data=controls2 out=controls2sorted; by sex ;
data potential_controls_data_step;
merge  cases2sorted(rename=(ssn=case_ssn
date_of_birth=case_dob  y=case_year_birth) in=a)
             controls2sorted(rename=(ssn=case_ssn
date_of_birth=case_dob  y=case_year_birth)in=b); by sex ;
if  a and b;
run;
```

Running this code yielded the following NOTES in the SAS Log. Since “they are only” notes and not errors, does it make a difference? Absolutely! In addition to the note about repeated values, notice that the number of observations is only 20, the same number as the original data sets. In this case SAS attempted to do a one to one match merge but we are looking for a many-to-many result.

```
NOTE: MERGE statement has more than one data set with repeats of BY values.
NOTE: There were 20 observations read from the data set WORK.CASES2SORTED.
NOTE: There were 20 observations read from the data set WORK.CONTROLS2SORTED.
NOTE: The data set WORK.POTENTIAL_CONTROLS_DATA_STEP has 20 observations and 6
variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Performing a many to many merge with the SAS Data Step requires multiple Data Steps including using first. logic and counting the number of controls for each case. This technique is described in multiple books and SAS Global Forum / SUGI papers and will not be addressed here.

The more efficient approach to many-to-many merges is to use Proc SQL as shown below. The Proc SQL code is annotated with the function each component performs. This example is meant to just scratch the surface of the available options in Proc SQL and provide a starting point for Data Step users. Proc SQL will allow for much more involved and complicated queries that include combining a number of data sets, multiple selection criteria and performing a host of other tasks.

name the table

create labels for the new variables

```
proc sql;
create table y_potential_controls AS
       Select
         a.ssn as case_ssn                     label='case_ssn'
        ,b.ssn as control_ssn                        label='control_ssn'
        ,a.date_of_birth as case_dob           label='case_dob'
        ,b.date_of_birth as control_dob        label='control_dob'
        ,a.sex as case_sex                     label='case_sex'
        ,b.sex as control_sex                        label='control_sex'
        ,a.year_onset as year_onset
        ,a.y as case_year_birth                label='case_yob'
        ,b.y as control_year_birth                   label='control_yob'
        ,a.case_no as case_no
        from cases2 a, controls2 b
        where
           a.sex=b.sex;
quit;
```

match
on
variable

use a and b to designate the case
and control files in the select statement

rename the variables

The following appears in the SAS log:

NOTE: Table WORK.Y_POTENTIAL_CONTROLS created, with 202 rows and 10 columns

As we predicted from the information we gathered about the number of males and females in our case and control data sets, the resulting data set contains 202 rows or observations. In examining the Proc SQL code clearly we were able to label variables, rename the like-named variables from each file and perform a many-to-many match on the variable sex.

Now that we have completed the match on the variable sex, all that it would take to match on both sex and year of birth as the original problem stated, would be to modify the where section of the select clause as follows:

```
where
a.y=b.y  and a.sex=b.sex;
```

PROC SQL AND DATA STEP: GENERAL RULES OF THUMB

As a statistician, I am often asked a question where the individual wants a simple “yes” or “no” answer. More times than not, I have to start out with an “it depends” response and then dig deeper into the problem at hand. The decision as to which techniques to use for a particular task often requires the same sort of reply. Sometimes both will work equally well and in other circumstances one is the clear winner. With data management tasks, there are multiple factors to consider. Balancing the ease of coding, performance considerations, data set size and constituency and ability to reuse code for subsequent projects is not always a simple or straightforward task. The following table illustrates some of the basic ideas to consider when deciding between techniques. Like with any data management or analysis task, it is essential to explore your data before beginning. In the example above, determining the number of males and females in each file provided the information to understand that a many-to many operation was necessary.

Operation	Proc SQL	Data Step	Usage Notes
Many-to-Many	√		As shown in the example highly recommended to avoid complex data step operations.
One-to-One	√	√	Consider data set size, and if the data set is large, consider whether it is indexed. Both approaches are powerful for creating new data elements, renaming variables and selecting observations based on conditions. For complex selection criteria, Proc SQL provides a more parsimonious approach.
One-to-Many	√	√	
Many-to-One	√	√	

WHAT NEXT...

This discussion was intended to wet your appetite for Proc SQL and take away some of the mystery. The best way to learn more about Proc SQL is to play with it and explore different types and sizes of data sets. There are numerous resources available from the simple to the complex. Some are listed in the reference section below. One of the reasons that many people, including myself, are longtime users of SAS is that there is always something new to explore and learn. Proc SQL is no different. In addition, it is not necessary to learn all the details and features immediately. Start with a simple task such as the example here and build to more complexity as your comfort level and understanding grows.

REFERENCES

Books

Lafler, Kirk Paul. PROC SQL:Beyond the Basics Using SAS®. Cary, NC: SAS Institute Inc.
Prairie, Katherine. 2005. The Essential PROC SQL Handbook for SAS® Users. Cary,
NC: SAS Institute Inc.
SAS Institute Inc., 2004. SAS 9.1 SQL Procedure User’s Guide, Cary, NC: SAS Institute Inc.

SAS Global Forum / SUGI Papers

Lafler,Kirk Paul (2009), “Exploring PROC SQL® Joins and Join Algorithms”**,** *Proceedings of SAS Global Forum 2009.*
Lafler, Kirk Paul (2003), “*Undocumented and Hard-to-find PROC SQL Features*,” *Proceedings of the Eleventh Annual Western Users of SAS Software Conference.*
Dickstein, Craig et. al (2007) “DATA Step vs. PROC SQL: What’s a neophyte to do?,*Proceedings of SAS Global Forum 2007.*
Williams, Christianna (2008), “PROC SQL for DATA Step Die-hards, Hands-on-Workshops, **“***Proceedings of SAS Global Forum 2008”*.

CONTACT INFORMATION

Mira J. Shapiro Email: mira.shapiro@gmail.com