PharmaSUG 2013 - Paper TF19

SQL, HASH Tables, FORMAT and KEY= — More Than One Way to Merge Two Datasets

David Franklin, TheProgrammersCabin.com, Litchfield, NH

ABSTRACT

The MERGE statement is the most common way to merge one-to-one or one-to-many data. This works very well most of the time but there are other methods that are useful, and sometime more efficient, that should be every SAS® programmers toolbox.

This paper touches on four methods that can be more efficient; quick look at PROC SQL and some of the options that help, HASH tables and some of the considerations for using this format, PROC FORMAT, and the KEY= option in the SET statement.

INTRODUCTION

When two datasets are merged the most common way is using two PROC SORT procedure calls followed by a DATA step with a MERGE statement. This is indeed the most flexible method and there is a lot of control over how the matches are made and what goes out to the output dataset. However, this can type of merge may be slow and efficiency is not the best.

This paper looks at four other methods that can be useful for merging data – the old favorite PROC SQL that has been in existence since SAS v6.06; a relatively new arrival by comparison in the HASH table; the unlikely but still very useful PROC FORMAT; and the sometime forgotten entrant the KEY= option on the SET statement. What this paper does not do is present a “this method is better than that method” as it is dependant on so many factors including memory and has some sort of index to it.

OUR DATA AND LOOKING AT THE MERGE STATEMENT

Before looking at ways to merge data, it is helpful to have some actual data to look at. Below are two datasets that will be used:

```
Dataset: PATDATA
SUBJECT  TRT_CODE
  001       A
  002       A
  003       B
  004       B

Dataset: ADVERSE
SUBJECT  EVENT
  003    FEVER
  002    FRACTURE
  001    HEADACHE
  005    FRACTURE
  003    NAUSEA
```

This data will be used throughout the paper.
It must be noted that SUBJECTs 001 and 004 are not represented in dataset ADVERSE, SUBJECT 003 has multiple ADVERSE records, and SUBJECT 005 is not in dataset PATDATA.
In a typical merge, the data would be combined using the following PROC SORT procedure calls and a MERGE statement call inside a datastep, as shown below:

```
PROC SORT DATA=patdata;
   BY subject;
PROC SORT DATA=adverse;
   BY subject;
DATA alldata;
   MERGE patdata adverse;
   BY subject;
RUN;
```

Using a PROC PRINT call the dataset ALLDATA would have output similar to that in Output 1:

```
SUBJECT  TRT_CODE  ADVERSE
001       A
002       A       FRACTURE
003       B       FEVER
003       B       NAUSEA
004       B
005               FRACTURE
```

Output 1. Output from two PROC SORT calls and the MERGE statement

Before going onto other methods it is worth noting that instead of using PROC SORT before calling the datastep, efficiency can be better is the PROC SORT calls were replaced by creating an index of the data, as shown below:

```
PROC DATASETS LIBRARY=WORK NOLIST NODETAILS;
   MODIFY patdata;
      INDEX CREATE subject /UNIQUE:
   MODIFY adverse;
      INDEX CREATE subject:
QUIT;
DATA alldata;
   MERGE patdata adverse;
   BY subject;
RUN;
```

When running this code the output will be similar to that in Output 1.

PROC SQL

SQL is an standard industry language for database manipulation that has been around in SAS since SAS version 6.06.

To merge the two datasets, the code that would be used is given below:

```
PROC SQL;
   CREATE TABLE alldata AS
   SELECT a.*, b.trt_code
   FROM adverse a OUTER UNION JOIN patdata b
   ON a.subject=b.subject;
QUIT;
RUN;
```

PROC SQL has a few ways that it will join the data and can be shown in the SAS log using the PROC SQL option _METHOD:

_METHOD Code	Description
sqxcrta	Create table as Select
Sqxslct	Select
sqxjsl	Step Loop Join (Cartesian)
sqxjm	Merge Join
sqxjndx	Index Join
Sqxjhsh	Hash Join
Sqxsort	Sort
sqxsrc	Source Rows from table
Sqxfil	Filter Rows
sqxsumg	Summary Statistics (with GROUP BY)
sqxsumn	Summary Statistics (not grouped)
sqxuniq	Distinct rows only

Table 1. _METHOD output codes

The most common of joins is the SQXJM (MERGE) join which will usually sort the data then merge, however there is a limited way that you can tell SQL how to do the join though the undocumented MAGIC= option, as shown below:

```
PROC SQL _METHOD MAGIC=101; * Step loop join, when an equality condition is not
                              specified, a read of the complete contents of the
                              right table is processed for each row in the left
                              table.;
PROC SQL _METHOD MAGIC=102; * Merge join, when the tables specified are already in
                              the desired sort order, resources will not need to be
                              extended to rearranging the tables.;
PROC SQL _METHOD MAGIC=103; * Hash join, when an equality relationship exists, the
                              smaller of the tables is able to fit in memory, no
                              sort operations are required, and each table is read
                              only once.;
```

HASH TABLES

First appearing in SAS version 9.1, and used by database programmers in other languages, this is considered one of the fastest ways to merge data in two datasets. Many papers have been written about this recent feature, how it works, and their use within SAS - references to some notable papers are the Reference section below. The code below does the merge required:

```
DATA alldata0;
   IF _n_=0 THEN SET patdata;
   IF _n_=1 THEN DO;
      DECLARE HASH _h1 (dataset: "PATDATA");
      rc=_h1.definekey("SUBJECT");
      rc=_h1.definedata("TRT_CODE");
      rc=_h1.definedone();
      call missing(SUBJECT,TRT_CODE);
   END;
   SET adverse;
   rc=_h1.find();
   IF rc^=0 THEN trt_code=" ";
   DROP rc;
RUN;
```

In the example above, the dataset PATDATA gets loaded into a hash table, then the ADVERSE dataset is loaded

into the datastep and the match is made using the FIND() method.

PROC FORMAT

It is possible to create a format from the dataset that has unique observations, in this case the PATDATA dataset, and the TRT_CODE variable as the label, as shown below:

```
DATA fmt;
   RETAIN fmtname 'TRT_FMT' type 'C';
   SET patdata;
   RENAME subject=start trt_code=label;
PROC FORMAT CNTLIN=fmt;
DATA alldata0;
   SET adverse;
   ATTRIB trt_code LENGTH=$1 LABEL='Treatment Code';
   trt_code=PUT(subject,$trt_fmt.);
RUN;
```

In the example a character format TRT_FMT is created from the PATDATA dataset, and then this format is used to set the TRT_CODE variable within the ADVERSE dataset. This method is useful as the data does not have to be sorted or indexed beforehand.

MERGE WITH SET-KEY

Many options have been added to the SET statement since it first appeared, one of them being the KEY= option as shown in the following example:

```
DATA alldata0;
   SET adverse;
   SET patdata KEY=subject /UNIQUE;
   DO;
      IF _IORC_ THEN DO;
         _ERROR_=0;
         trt_code='';
      END;
   END;
RUN;
```

Before this type of merge can work the dataset PATDATA must have an index created inside it, using either the INDEX statement inside a DATASETS or SQL procedure, or INDEX option inside a DATA step. It is important to have the DO loop is if no match is found then TRT_CODE will be set to missing - if this is not done then unexpected results may occur.

CONCLUSION

There is no ‘best’ base to merge data but this paper has presented four methods that can be used instead of the MERGE statement to do a one-to-one or one-to-many merge. It is only though trying these different methods at your installation that you will see resource efficiencies between the methods.

CONTACT INFORMATION

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

Name: David Franklin
Enterprise: TheProgrammersCabin.com
Work Phone: 603-275-6809
E-mail: dfranklin@TheProgrammersCabin.com
Web: http://www.TheProgrammersCabin.com
Twitter: ThePgmrsCabin.com