PharmaSUG 2021 - Paper AP-184

Mashing Two Datasets Together

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 dependent on so many factors including memory and has some sort of index to it.

With the exception of a discussion on a many-to-many merge in PROC SQL, all methods shown here are a one-to-one or one-to-many merge.

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: ADSL
SUBJID   ARMCD
  001      A
  002      A
  003      B
  004      B

Dataset: ADAE
SUBJID   ASTDT     AETERM
  003    04JAN21   FEVER
  002    05JAN21   FRACTURE
  001    04JAN21   HEADACHE
  005    06JAN21   FRACTURE
  003    05JAN21   NAUSEA
```

This data will be used throughout the paper.

It must be noted that SUBJIDs 001 and 004 are not represented in dataset ADAE, SUBJID 003 has multiple ADAE records, and SUBJID 005 is not in dataset ADSL.

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=adsl;
   BY subjid;
PROC SORT DATA=adae;
   BY subjid;
DATA alldata;
   MERGE adsl adae;
   BY subjid;
RUN;
```

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

SUBJID	ARMCD	ASTDT	AETERM
001	A		
002	A	05JAN21	FRACTURE
003	B	04JAN21	FEVER
003	B	05JAN21	NAUSEA
004	B		
005		06JAN21	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 adsl;
      INDEX CREATE subjid /UNIQUE:
   MODIFY adae;
      INDEX CREATE subjid:
QUIT;
DATA alldata;
   MERGE adsl adae;
   BY subjid;
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.armcd
   FROM adae a OUTER UNION JOIN adsl b
   ON a.subjid=b.subjid;
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.;
```

Typically PROC SQL is used to do a many-to-many merge which is efficient but if control is needed using a loop within a loop and the POINT option in the SET statement is possible as the following ADAE dataset is joined with an ADCM dataset in the following example:

```
Dataset: ADCM
SUBJID   ASTDT     CMTRT
  001    04JAN21   Dopamine
  002    04JAN21   Antacid
  003    05JAN21   Sodium bicarbonate
  003    06JAN21   Aspirin
```

In a many-to-many type merge an example of PROC SQL code is:

```
PROC SQL;
   CREATE TABLE _all0 AS
     SELECT a.*, b.cmtrt
     FROM adae a INNER JOIN adcm b
     ON a.subjid=b.subjid AND a.astdt=b.astdt;
   QUIT;
RUN;
```

but using a datastep with a loop within a loop and the POINT option in the SET statement:

```
DATA _all0;
   SET adae;
   DROP _:; ** Drop temporary variables;
   match=0; ** Match flag;
   ** Our loop within a loop -- output if match;
   DO i=1 TO xnobs;  ** Rename the "merging" variables within the ADCM dataset;
      SET adcm (RENAME=(subjid=_subject astdt=_date)) NOBS=xnobs POINT=i;
        ** Have to rename matching variables so that they do not overwrite
           the original values in ADAE;
      IF (subjid=_subject AND astdt=_date) or
         (subjid=_subject AND astdt>. and _date=.)
        THEN DO;
         match=1; ** Yes, there is a match my the "merging" variables;
         OUTPUT;
      END;
   END;
RUN;
```

the output is:

SUBJID	ASTDT	AETERM	CMTERM
001	04JAN21	HEADACHE	Dopamine
002	05JAN21	FRACTURE	Antacid
003	04JAN21	FEVER	
003	05JAN21	NAUSEA.	Sodium bicarbonate

Output 2. Output from data step with a many-to-many join

The code using the datastep is a little more code than the PROC SQL but does give more control with the same results. It is necessary to have the matching variables renamed so these do not overwrite the original values in ADAE. Note also that the variable MATCH is used to flag where a match is made.

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 adsl;
   IF _n_=1 THEN DO;
      DECLARE HASH _h1 (dataset: "ADSL");
      rc=_h1.definekey("SUBJID");
      rc=_h1.definedata("ARMCD");
      rc=_h1.definedone();
      call missing(SUBJID,ARMCD); END;
   SET adae;
   rc=_h1.find();
   IF rc^=0 THEN armcd=" ";
   DROP rc;
RUN;
```

In the example above, the dataset ADSL gets loaded into a hash table, then the ADAE 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 ADSL dataset, and the ARMCD variable as the label, as shown below:

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

In the example a character format TRT_FMT is created from the ADSL dataset, and then this format is used to set the ARMCD variable within the ADAE 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 adae;
   SET adsl KEY=subjid /UNIQUE;
   DO;
      IF _IORC_ THEN DO;
         _ERROR_=0;
         armcd='';
      END;
   END;
RUN;
```

Before this type of merge can work the dataset ADSL 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 ARMCD 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

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

David Franklin
dfranklin@TheProgrammersCabin.com