

Paper SBC-114

Joins: How to Put it All Together Janet Stuelpner, Left Hand Computing, Inc.

ABSTRACT

How to make two datasets into one seems like it would be an easy task. There are so many choices. If you are a DATA step bigot then you might want to MERGE. If your favorite step is PROC SQL, then a JOIN might be the way to go. The type of join that you choose depends on the desired results. This paper will try to demystify the mysterious world of joins.

INTRODUCTION

Data warehouses can contain data collected and stored in many different physical forms. These data structures (the physical form of the data) can include flat files, database tables, spreadsheets, and SAS® data sets. Utilization of this “raw” data by an information analyst can require combining two or more of these data structures through the use of a join (merging and joining are synonymous terms referring to the combination of data structures through the use of common variables/fields). One of the strengths of the SAS system is that it provides many different options for joining data values from many different data structures.

JOIN SELECTION CRITERIA

The availability of the proper software joining tool is a decision that is required of the programmer. The focus of this paper will be to show how the PROC SQL will differ and how it will be the same as the MERGE statement that can be used in a DATA step. There are circumstances where a MERGE would be easier, more appropriate and give the desired results. On the other hand, there are also circumstances where a PROC SQL will be the tool that is most appropriate based on the desired result and the status and amount of data. Let’s see what tool is the best to use under the information requirements as shown in the table below.

Informational Requirements	Type of Data Set (Table) Join	SAS Tool
A. All data values from all data sets.	Match-Merge or Full Outer Join	- MERGE statement with BY statement⁴ - PROC SQL²
B. All data values from a single data set (base) and all data values from the other data set(s) that match the data values of the joining variables within the base.	Non-base data set(s) are used as “Look-up” table(s) - Right or Left outer join.	- PROC FORMAT¹ - SET statement with KEYS= option³ - PROC SQL²
C. Only those data values from all data sets that contain the same data values within the joining variables.	Inner Join	- PROC SQL² - MERGE statement with IN= option and BY statement⁴
D. Placement of data sets side by side	One-to-one Merge	MERGE statement ⁴
E. Expansion of data set to include all levels of a non-common variable.	Many-to-many Join	PROC SQL ²

Table 1. The type of data set joins required to construct a specific data set.

Scenarios

Scenario A - Groups of two or more parent data sets are used to build or create a child or output data set. In scenario A all data values from all of the parent data sets are needed to create the child data set (Table 1). The type of joins used to accomplish this are a match-merge or a full join. Missing values are placed within observations that do not occur within all parent data sets. Example 1 illustrates the two SAS tools capable of full joins.

Scenario B - All data values of a base parent data set are kept and only those observations containing matching data values of the common variable(s) are selected from all other parent data sets (Table 1). The terminology right or left join is an indication of which parent data set to use as the base. Example 2 illustrates all three SAS tools available for right or left joins.

Scenario C - An inner join is used when the child data set needs to contain only those data records from the parent data sets where the common variable(s) are identical. Example 3 illustrates an inner join.

Scenario D - One-to-one merging combines all of the parent data sets using a common variable and creates a child data set that is as large as the largest data set within the merge list. The parent data sets are not joined by common variables. Instead parent data sets are placed side by side and each common variable data value is super-imposed by the data values within the last parent data set within the merge. Example 4 illustrates when a one-to-one merge could be used.

Scenario E - Expansion of the child data set occurs when one data set has multiple levels of a non-common variable. An example would be where one data set contains a listing of names by city and the other contains city information. The child data set required contains all of the names for each city plus the city information. In this case the data sets are joined by the common variable city and all names within a city are transferred to the child data set (Example 5).

EXAMPLES

Example 1 - Scenario A examples of both a match-merge and SQL full outer joins.

This is a match-merge. Both datasets have the same variables sorted in the same order. All data values from both datasets are present.

Below are the datasets that are used in this example:

REPS	
REGION	REPNAME
Southeast	Jones
Southeast	Smith
Southeast	Doe
Northeast	Harris
Northeast	James
Northeast	Finch

RGN		
CITY	ST	REGION
Atlanta	GA	Southeast
New York	NY	Northeast
Portland	OR	Northwest

```
PROC SORT DATA=REPS;  
  BY REGION;  
RUN;
```

```
PROC SORT DATA=RGN;  
  BY REGION;  
RUN;
```

```

❶ DATA REPRGN;
    MERGE REPS RGN;
    BY REGION;
    RUN;

```

Below are two examples of Full Outer Joins using PROC SQL (structured query language). These two joins differ because of the order of the variables in the select portion of the SQL. As in the match-merge all values from both data sets are present.

PROC SQL;

```

❷ CREATE TABLE RGNREPSA AS
    SELECT A.REGION,
           B.CITY,
           B.ST,
           A.REPNAME
    FROM REPS A FULL JOIN RGN B
    ON A.REGION=B.REGION;

```

```

❸ CREATE TABLE RGNREPSB AS
    SELECT A.REGION,
           B.CITY,
           B.ST,
           A.REPNAME
    FROM RGN A FULL JOIN REPS B
    ON A.REGION=B.REGION;
QUIT;

```

❶ Match-Merge Results

REGION	REPNAME	CITY	ST
Northeast	Harris	New York	NY
Northeast	James	New York	NY
Northeast	Finch	New York	NY
Northwest		Portland	OR
Southeast	Jones	Atlanta	GA
Southeast	Smith	Atlanta	GA
Southeast	Doe	Atlanta	GA

❷ Full Outer Join Results - A

REGION	CITY	ST	REPNAME
Northeast	New York	NY	Harris
Northeast	New York	NY	James
Northwest	Portland	OR	
Southeast	Atlanta	GA	Jones
Southeast	Atlanta	GA	Smith
Southeast	Atlanta		GA
Northeast	New York	NY	Finch

② Full Outer Join Results - B

REGION	CITY	ST	REPNAME
Northeast	New York	NY	Harris
Northeast	New York	NY	James
Northeast	New York	NY	Finch
Northwest	Portland	OR	
Southeast	Atlanta	GA	Jones
Southeast	Atlanta	GA	Smith
Southeast	Atlanta		GA

Example 2 - Scenario B examples of a “look-up” table, SET statement with KEYS= option and SQL left join.

Creation of “look-up” table using PROC FORMAT:

```

①DATA CALENDAR;
  LENGTH MON $10.;
  WEEK=1;
  DO DAY = '01JAN96' TO '31DEC97'D ;
    IF WEEKDAY(DAY) = 1 THEN
      WEEK= WEEK + 1;
    IF WEEK=53 THEN WEEK=1;
    DATE=LEFT( PUT(DAY, WORDDATE18.) );
    MON=LEFT( SCAN(DATE, 1) );
    MONTH=MONTH(DAY);
    YEAR=YEAR(DAY); FISCYEAR=YEAR;
    IF MONTH=12 AND WEEK=1
      THEN FISCYEAR=YEAR + 1;
    KEEP DATE WEEK MONNAME FISCYEAR;
  OUTPUT;
END;
RUN;

```

Format Creation - The code below creates variables required by PROC FORMAT⁵.

```

DATA WORDDATE;
  SET CALENDAR END=EOF;
  HLO=' ' ;
  START=DATE;
  LABEL=PUT(FISCYEAR,Z4.) || MON ||
        PUT(WEEK,Z2) ;
  FMTNAME='YMW' ;
  TYPE='C' ;
  OUTPUT WORDDATE;
  IF EOF THEN DO;
    HLO='O' ;
    START='OTHER' ;
    LABEL=' ' ;
    OUTPUT WORDDATE;
  END;
RUN;

PROC FORMAT CNTLIN=WORDDATE;
RUN;

```

The following data step uses the format created in the previous PROC FORMAT statement to create two (2) new variables containing data values from the calendar data set.

```

DATA SALESA;
  LENGTH FISCYEAR $4 MONNAME $10
           WEEK $2;

  SET ,SALES;
  FISCYEAR=SUBSTR(PUT (DATE,$YMW.),1,4);
  MONNAME=SUBSTR(PUT (DATE,$YMW.),5,10);
  WEEK=SUBSTR(PUT (DATE,$YMW.),11);
RUN;

```

The following Left Inner Join creates the same data set.

```

PROC SQL;
  CREATE TABLE SALEC AS
  SELECT B.FISCYEAR,
         B.MON,
         B.WEEK,
         A.DATE,
         A.SALES
  FROM SALES A LEFT JOIN
        CALENDAR B
  ON A.DATE=B.DATE;
QUIT;

```

Use of the Set Statement with Key= Option to join the SALES and CALENDAR data sets.

```

PROC DATASETS LIB=WORK;
  MODIFY CALENDAR;
  INDEX CREATE DATE;
QUIT;

DATA SALEB;
  SET SALES;
  SET CALENDAR KEY=DATE;
DROP FISCYEAR;
RUN;

```

Sample of CALENDAR Data Set

MONNAME	WEEK	DATE	MONTH	FISCYEAR
January	1	January 1, 1999	1	1999
January	1	January 2, 1999	1	1999
January	1	January 3, 1999	1	1999
January	1	January 4, 1999	1	1999
January	1	January 5, 1999	1	1999
January	1	January 6, 1999	1	1999
January	1	January 7, 1999	1	1999

Sample of SALES Data Set

DATE	SALES
January 1, 1999	67561.59
January 2, 1999	1162.28
January 3, 1999	52629.84
January 4, 1999	34110.43
January 5, 1999	25372.64
January 6, 1999	15320.03
.....	

Sample of data sets created by the Format and Left Join techniques.

FISCYEAR	MON	WEEK	DATE	SALES
1999	April	14	April 1, 1999	3922.42
1999	April	15	April 10, 1999	2438.21
1999	April	15	April 11, 1999	3194.39
1999	April	15	April 12, 1999	5690.00
1999	April	15	April 13, 1999	1357.40
1999	April	16	April 14, 1999	2648.88
1999	April	16	April 15, 1999	4963.07
1999	April	16	April 16, 1999	3274.21

„Sample of SALEB Data Set.

DATE	SALES	MON	WEEK	MONTH
January 1, 1999	67561.59	January	1	1
January 2, 1999	1162.28	January	1	1
January 3, 1999	52629.84	January	1	1
January 4, 1999	34110.43	January	1	1
January 5, 1999	25372.64	January	1	1
January 6, 1999	15320.03	January	1	1

Example 3 - Scenario C examples of SQL inner joins and MERGE statement with IN= options.
Inner join using PROC SQL.

```
PROC SQL;
CREATE TABLE RGNREP AS
  SELECT A.REGION,
         B.CITY,
         B.STATE,
         A.REPNAME
  FROM REPS A, RGN B
 WHERE A.REGION=B.REGION;
QUIT;
```

Match-merge restricting the data values kept.

```
DATA REPRGN;
MERGE REPS(IN=A) RGN;
  BY REGION;
IF A;
RUN;
```

Results from both the Inner Join and Merge

REGION	CITY	ST	REPNAME
Northeast	New York	NY	Harris
Northeast	York	NY	James
Northeast	New York	NY	Finch
Southeast	Atlanta	GA	Jones
Southeast	Atlanta	GA	Smith
Southeast	Atlanta	GA	Doe

Example 4 - Scenario D example of one-to-one merge.

```
PROC SUMMARY DATA=SALES;  
  VAR SALES;  
  OUTPUT OUT=SALESTOT (DROP=_TYPE_  
                        _FREQ_)  
        SUM=TOTSALES;  
RUN;
```

Output of above summary procedure:

TOTSALES
3199002.11

```
DATA SUMMARY;  
  MERGE REPS SALETOTS;  
  RETAIN SALES;  
  SALES=TOTSALES;  
RUN;
```

Results from One-to-One Merge

REGION	REPNAME	TOTSALES	SALES
Northeast	Harris	3199002.11	3199002.11
Northeast	James	.	3199002.11
Northeast	Finch	.	3199002.11
Southeast	Jones	.	3199002.11
Southeast	Smith	.	3199002.11
Southeast	Doe	.	3199002.11

Note: This type of merge is only useful in adding one or more total variables for use in calculations.

Example 5 - Scenario E example of a many to many join.

REPS

REGION	REPNAME
Southeast	Jones
Southeast	Smith
Southeast	Doe
Northeast	Harris
Northeast	James
Northeast	Finch

RGN

CITY	ST	REGION
Atlanta	GA	Southeast
Miami	FL	Southeast
Boston	MA	Northeast
New York	NY	Northeast

Match-merge of the two (2) data sets.

```
PROC SORT DATA=REPS;
  BY REGION;
RUN;

PROC SORT DATA=RGN;
  BY REGION;
RUN;

DATA REPRGN;
  MERGE REPS RGN;
  BY REGION;
RUN;
```

Many-to-many Join

```
PROC SQL;
  • CREATE TABLE RGNREPS AS
    SELECT    A.REGION,
              A.CITY,
              A.ST,
              B.REPNAME
    FROM      RGN A,
              REPS B
    WHERE A.REGION=B.REGION ;
QUIT;

or

PROC SQL;
  CREATE TABLE RGNREPS AS
  SELECT    A.REGION,
            A.CITY,
            A.ST,
            B.REPNAME
  FROM RGN A LEFT JOIN REPS B
        ON A.REGION=B.REGION ;
QUIT;
```

Match-merge results in a data set that matches rep to a city in each region.

REGION	REPNAME	CITY	ST
Northeast	Harris	Boston	MA
Northeast	James	New York	NY
Northeast	Finch	New York	NY
Southeast	Jones	Atlanta	GA
Southeast	Smith	Miami	FL
Southeast	Doe	Miami	FL

The data set of a many-to-many join contains a record for each rep in each city.

REGION	CITY	ST	REPNAME
Northeast	Boston	MA	Harris
Northeast	Boston	MA	James
Northeast	Boston	MA	Finch
Northeast	New York	NY	Harris
Northeast	New York	NY	James
Northeast	New York	NY	Finch
Southeast	Atlanta	GA	Jones
Southeast	Atlanta	GA	Smith
Southeast	Atlanta	GA	Doe
Southeast	Miami	FL	Jones
Southeast	Miami	FL	Smith
Southeast	Miami	FL	Doe

SAS Tool	Pros	Cons
PROC FORMAT ¹	Creates a “look-up” table using either single variables or multiple variables for the key and label components of the format.	The key values must be unique, no duplicate values can occur within the data set used to create the format. Format can be applied to only one data set at a time.
MERGE statement ⁴	- Used only for one-to-one merges no sorting required. - Two (2) or more data sets can be joined.	The data sets are not joined by any common variables. If there is a common variable between the data sets, then the common variable will contain the values of the common variable in the last data set joined.
MERGE statement with BY statement ⁴	- Two or more data sets can be joined by common variables. - All values of all variables within all data sets will be retained.	All data sets must be sorted by common variable.
MERGE statement with IN= option and BY statement ⁴	- Two (2) or more data sets can be joined by common variables. - All data from each data set will be read before subsetting criteria applied.	All data sets must be sorted by common variable.
PROC SQL ²	- Data sets do not need to be pre-sorted. - Inner joins can occur between two (2) or more data sets.	- All outer joins occur between two data sets only. - Inner join contains only those values of the common variable that match between the data sets joined.
SET statement with KEYS= option ³	Two (2) or more data sets can be joined by common variables.	Data sets need to be indexed by common variables.

Table 2. Pros and cons for using each type of SAS joining tool.

Conclusions

The pros and cons of using the different joins are listed in Table 2.

The selection of a joining tool is dependent on the environment of the data structures, the required contents of the data set and what tool is the most system resource efficient. During the application development process, careful bench-marking of joining tools is required to ensure selection of the correct environment and tool for the job.

REFERENCES

- SAS *Procedures Guide*, Version 8, SAS Publishing, 1999
SAS *Guide to the SQL Procedure: Usage and Reference*, Version 6, First Edition. SAS Publishing, 1989
SAS *Technical Report P-222, Changes and Enhancements to Base SAS Software, Release 6.07*. SAS Publishing, 1991
SAS *Language: Reference*, Version 8, SAS Publishing, 1990.
Bahler, C. and Clos, S. To Format or Merge ... That is the Question. Proceedings of the Southeast SAS Users Group.

ACKNOWLEDGMENTS

I would like to thank my husband Bob for his unending support and tireless proofreading comments. Without his care, fortitude and encouragement, I could not have successfully completed my paper.

AUTHOR CONTACT

Janet Stuelpner
Left Hand Computing, Inc.
326 Old Norwalk Road
New Canaan, CT 06840
(203) 801-4373 voice
(203) 801-0043 fax
lstuelpner@usa.net

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.

Other brand and product names are trademarks of their respective companies.