THE JOYS OF MERGING: ALTERNATIVE TECHNIQUES

Joan Keesey, RAND, Santa Monica, CA

ABSTRACT

The SAS System™ provides the capability of performing one-to-one and many-to-one data set merges easily and efficiently via the MERGE statement. Many-to-many merges require alternative techniques but are often very useful, particularly when there are errors and inconsistencies in the merge variables or when there is a "best" match based on some criteria, i.e., the record with the highest score or the record which best satisfies some formula or requirement. Using a many-to-many merge technique provides the user with increased flexibility in searching any number of potential matches for the "best" match. The implementation of a many-to-many merge can eliminate repeated merge steps and make more efficient use of the matching information.

This paper examines three alternative methods of programming many-to-many data set merges: (1) merging data sets by directly accessing one of the files using the POINT option; (2) utiizing the transformation capabilities of the TRANSPOSE procedure; and (3) using the SQL procedure for the merge.

Two additional merge techniques will also be discussed: (1) merging data sets using PROC FORMAT; and (2) combining data sets by directly accessing one of the files using the KEY option.

INTRODUCTION

The SAS System provides the capability of performing one-to-one and many-to-one data set merges easily and efficiently via the MERGE statement. Many-to-many merges require alternative techniques but are often very useful, particularly when there are errors and inconsistencies in the merge variables or when there is a "best" match based on some criteria, i.e., the record with the highest score or the record which best satisfies some formula or requirement.

This paper explores three possible solutions to programming many-to many data set merges: (1) merging the data sets by directly accessing one of the files using the POINT option; (2) utilizing the transforming capabilities of PROC TRANSPOSE and then reading the file directly using the KEY option; and (3) using the SQL procedure for the merge.

The example I am going to use is a problem which was part of a study of the survival and health costs of very low birthweight infants. For each 1987 California birth we were interested in identifying the nearest intermediate or tertiary care (ITC) facility in a specific geographic region. To do this it was necessary to: (1) match the 1987 California birth certificate (CBC) data to a file of ITC facility data by region; (2) compute the distance to each ITC facility in the region; and (3) determine the nearest facility. The counties in California were grouped into 11 geographic regions. There were approximately 500,000 births in California in 1987--many thousands in each of the eleven regions. The number of ITC facilities in each region varied from 1 to 21.

Using a many-to-many merge technique is one approach to this problem. There were many thousands of births and one or more ITC facilities in each region. A simple merge by region would not do the trick. In addition, there is no perfect match. We were interested in finding the "best solution", which in this case was the "closest ICT facility".

DATA

The 1987 CBC data include one record for each live birth or fetal death in California for the year 1987. The birth certifcate record contains the zipcode of the mother's residence. This file was appended with the latitude and longitude of the zipcode centroid which was then used to calculate the distance to each ITC facility in the region. Figure 1 shows the birth certificate data with multiple records per region and the latitude and longitude of the zipcode centriod.

Fig. 1. 1987 California Birth Certificate Data

REC_ID	REGION	ZIP	LATCENT	LONGCENT
1	7	90016	34.03	118.35
2	9	92102	32.71	117.11
3	3	94608	37.83	122.29
4	3	94563	37.89	122.21

Figure 2 is a listing of the ITC facilities. These facilities are grouped by region. The associated latitude and longitude define the location. To use a file as a direct access file the file must have a variable equal to the observation number; in this example the variable is OBSNUM. In addition, the file cannot have been compressed using the SAS COMPRESS option.

Fig. 2. Intermediate/Tertiary Care Facilities

REGION	OBSNUM	ITC_ID	H_LAT	H_LONG
1	1	0521	37.75	122.41
1	2	0525	37.75	122.41
1	3	0326	37.95	122.54
1	4	0605	38.45	122.79
1	5	0070	40.76	124.13
1	6	0519	37.79	122.44
1	7	0510	37.78	122.46
1	8	0527	37.76	122.46
2	9	0536	37.89	121.30
2	10	0589	40.54	122.43
2	11	0443	38.55	121.46
2	12	0441	38.57	121.44
3	13	0043	37.90	121.96
3	14	0003	37.86	122.24
3	15	0005	37.83	122.26

IMPLEMENTATION

The programs for the following work were written in SAS/BASE™ (version 6.09) and executed on a Sun™ SPARCstation™ 2 using a UNIX™ operating system.

METHOD 1: DIRECT ACCESS USING THE POINT OPTION

Each birth certificate was matched to each ITC facility in the region of the birth. PROC FORMAT was used to construct a table of the first and last observation numbers defining the ITC facility records in a given region. The SAS code used to construct this table is shown in Figure 3; the table, in Figure 4.

Fig. 3. Create Table of the First and Last Observation Numbers Corresponding to a Given Region

```
DATA MKFRMT (KEEP=START END LABEL TYPE FMTNAME);

SET SUGI.FACILITY;  BY REGION;              *** ITC FACILITY DATA;

LENGTH BEGOBS ENDOBS $3 LABEL $6
RETAIN BEGOBS ENDOBS;

IF FIRST.REGION THEN BEGOBS=_N_;            *** 1ST REC.;

IF LAST.REGION THEN DO;                     *** LAST REC;
FMTNAME="FREGION";                          *** FORMAT NAME;
TYPE='N';                                   *** FORMAT TYPE;
START=REGION;                               *** RANGE;
END  =REGION;
ENDOBS=_N_;                                 *** LAST REC.NO.;
LABEL=BEGOBS || ENDOBS;                     *** LABEL;
OUTPUT MKFRMT;
END;

PROC FORMAT LIBRARY=LIBRARY CNTLIN=MKFRMT;
```

Fig. 4. Table of First and Last Observation Numbers Corresponding to a Given Region

START	END	LABEL (VER. 6.09
1	1	1 1 8
2	2	2 9 12
3	3	3 13 15
4	4	4 16 18
5	5	5 19 22
6	6	6 23 25
7	7	7 26 46

Accessing the Relevant ITC Records Using PROC FORMAT

The table in Figure 4 was used to define the set of ITC facilities in a given region. The PUT statement was used to access this table. The code is shown in Figure 5. The file of ITC facility data was then read as a direct access file using the POINT option. Figure 6 shows the SAS code which reads the appropriate records from the ITC facility file and calculates the distance between the ITC facility and the residence of birth. The MACRO, GETDIST, used to calculate the distance from the zipcode centroid to the ITC facility is shown in Figure 7. This is the Pythagorean Theorem adjusted for the fact that a degree of longitude becomes smaller as you move North or South from the equator (69.09 miles at the equator). The calculated distance is in miles.

Fig. 5. Determining the First and Last Observation Numbers for a Given Region Using PROC FORMAT

```
DATA CHKDIST  BRTHCRT ;

SET SUGI.BRTHCRT (IN=INBC);          **** BIRTH CERT.DATA ;
*************************************************************
USE PROC FORMAT TO DEFINE ITC RECORDS FOR REGION
*************************************************************;
LENGTH BEG_END $6;
BEG_END=PUT(REGION,FREGION.);    **** ACCESS FORMAT;

BEGOBS=SUBSTR(BEG_END,1,3);      **** FIRST OBS.NO.;
ENDOBS=SUBSTR(BEG_END,4,3);      **** LAST OBS. NO.;
```

Fig. 6. Accessing the ITC Data Using the POINT Option

```
*************************************************************
READ ITC FACILITY FILE AS DIRECT ACCESS FILE
COMPUTE & SAVE DISTANCE TO NEAREST ITC FACILITY
*************************************************************;
SAV_DIST=.;                          **** INITIALIZE DISTANCE  ;

IF (BEGOBS>0) THEN DO OBSNUM=BEGOBS TO ENDOBS;
SET SUGI.FACILITY POINT=OBSNUM;  **** ITC FACILITY DATA ;
%GETDIST(H_LAT,H_LONG,ITC_ID);   **** GET DISTANCE;
OUTPUT CHKDIST;                  **** CHECK FILE;
END;

IF INBC THEN OUTPUT BRTHCRT;     **** FINAL OUTPUT FILE ;
```

Fig. 7. MACRO to Compute Distance from Zipcode Centroid to ITC Facility

```
%MACRO GETDIST (H_LAT,H_LONG,ITC_ID);

RETAIN SAV_DIST SAV_HOSP;

LATR=&H_LAT*3.14159/180;             **** CONVERT TO RADIANS;

TRM1=(&H_LAT - LATCENT);
TRM2=(&H_LONG - LONGCENT)*COS(LATR); *** ADJUST LONG.   ;

DIST=69.09*SQRT(TRM1**2 + TRM2**2); *** PYTHAGOREAN DIST.  ;
DIST=ROUND(DIST,.1);                  *** DISTANCE IN MILES    ;

IF (DIST<SAV_DIST) | (SAV_DIST=.) THEN DO;
SAV_DIST=DIST;                       **** SAVE SHORTEST DIST.;
SAV_HOSP=&ITC_ID;                    **** SAVE FACILITY ID;
END;

%MEND;
```

The output file (CHKDIST) from the code in Figures 5-7 is shown in Figure 8. This is a report for checking purposes; it shows each pair of matched CBC and ITC facility records, the computed distance (DIST) to the ITC facility, along with the identifier (SAV_HOSP) of and distance (SAV_DIST) to the nearest ITC facility as of that record. For space reasons I have used the second and third observations in the CBC file (REC_ID=2,3). The final output data set, BRTHCRT, is shown in Figure 9. It contains one record for each birth certificate with the distance to and identifier of the closest ITC facility.

Fig. 8. Merged CBC and ITC Facility Data

REC_ID	REGION	ITC_ID	DIST	SAV_HOSP	SAV_DIST
2	9	0492	7.3	0492	7.3
2	9	0498	3.9	0498	3.9
2	9	0078	87.0	0498	3.9
2	9	0508	4.0	0498	3.9
REC_ID	REGION	ITC_ID	DIST	SAV_HOSP	SAV_DIST
3	3	0043	18.7	0043	18.7
3	3	0003	3.0	0003	3.0
3	3	0005	1.3	0005	1.3

Fig. 9. Final Data Set--One record per Birth Certificate with Distance to and Identifier of Nearest ITC Facility

OBS	ZIPCODE	REC_ID	REGION	SAV_DIST	SAV_HOSP
1	90016	1	7	3.2	0150
2	92102	2	9	3.9	0498
3	94608	3	3	1.3	0005
4	94563	4	3	2.9	0003

METHOD 2: PROC TRANSPOSE & THE KEY OPTION

PROC TRANSPOSE reads a SAS data set and creates an output data set in which the rows of the original data set become the columns of the new data set. One record is generated for each value of the BY variable and for each variable selected for the output file. The code used to transform the ITC facility file is shown in Figure 10. Figure 11 shows the output--three records for each value of the BY variable, REGION.

Fig. 10 .PROC TRANSPOSE - Multiple Variables

```
*************************************************************************
USE PROC TRANSPOSE TO COLLAPSE DATA SET BY REGION
*************************************************************************;

PROC TRANSPOSE DATA=SUGI.FACILITY OUT=LATF;
VAR H_LAT H_LONG ITC_ID;
BY REGION;
```

Fig. 11 Transposed ITC Facility Data

REGION=1

NAME	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8
H_LAT	37.75	37.75	37.95	38.45	40.76	37.79	37.78	37.76
H_LONG	122.41	122.41	122.54	122.79	124.13	122.44	122.46	122.48
ITC_ID	0521	0525	0326	0605	0070	0519	0510	0527

REGION=2

NAME	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8
H_LAT	37.89	40.54	38.55	38.57	.	.	.	.
H_LONG	121.30	122.43	121.46	121.44	.	.	.	.
ITC_ID	0536	0589	0443	0441				

The output file shown in Figure 11 is not exactly what was desired, so I decided to use the PREFIX option and transpose the file three times--once for each of the three variables. These transposed files can easily be merged to produce a file with one record for each value of REGION containing an array of latitudes (LAT1-LATn), longitudes (LNG1-LNGn), and ITC facility identifiers (ITC1-ITCn). These variables correspond to the n observations for a specific value of REGION. To generalize the routine I created a MACRO variable, &MAXR, equal to the maximum number of ITC facilities in any given REGION. Figure 12 shows the code required to transpose the ITC facility file and create one record per region; Figure 13, the PROC TRANSPOSE output; and Figure 14, the code used to initialize the macro variable, &MAXR.

Fig. 12. Transpose ITC Facility Data and Merge Transposed Files

```
*************************************************************************
USE PROC TRANSPOSE TO COLLAPSE DATA SET BY REGION
*************************************************************************;

PROC TRANSPOSE DATA=SUGI.FACILITY OUT=LATF PREFIX=LAT;
VAR H_LAT;
BY REGION;

PROC TRANSPOSE DATA=SUGI.FACILITY OUT=LNGF PREFIX=LNG;
VAR H_LONG;
BY REGION;

PROC TRANSPOSE DATA=SUGI.FACILITY OUT=HSPF PREFIX=HSP;
VAR ITC_ID;
BY REGION;

*************************************************************************
MERGE TRANSPOSED FILES
*************************************************************************;

DATA SUGI.TRNFACIL ;

MERGE LATF (KEEP=REGION LAT1-LAT&MAXR)
      LNGF (KEEP=REGION LNG1-LNT&MAXR)
      HSPF (KEEP=REGION HSP1-HSP&MAXR) ;
BY REGION;
```

Fig. 13. Transposed ITC Data

TRANSPOSED DATA SET - LATITUDES

REGION	_NAME_	LAT1	LAT2	LAT3	LAT4	LAT5	LAT6	LAT7	LAT8
1	H_LAT	37.75	37.75	37.95	38.45	40.76	37.79	37.78	37.76
2	H_LAT	37.89	40.54	38.55	38.57	.	.	.	.
3	H_LAT	37.90	37.86	37.83	.	.	.	.	.

TRANSPOSED DATA SET - LONGITUDES

REGION	_NAME_	LNG1	LNG2	LNG3	LNG4	LNG5	LNG6	LNG7	LNG8
1	H_LONG	122.4	122.4	122.5	122.8	124.1	122.4	122.5	122.5
2	H_LONG	121.3	122.4	121.5	121.4	.	.	.	.
3	H_LONG	122.0	122.2	122.3	.	.	.	.	.

TRANSPOSED DATA SET - ITC FACILITY IDS

REGION	_NAME_	HSP1	HSP2	HSP3	HSP4	HSP5	HSP6	HSP7	HSP8
1	ITC_ID	0521	0525	0326	0605	0070	0519	0510	0527
2	ITC_ID	0536	0589	0443	0441				
3	ITC_ID	0043	0003	0005					

Fig. 14. Determine MACRO Variable, &MAXR

```
*************************************************************************
DETERMINE MAXIMUM NO. OF ITC FACILITIES IN A REGION
*************************************************************************;

DATA _NULL_;
SET SUGI.FACILITY;  BY REGION;
RETAIN MAXR CNTREC;
IF FIRST.REGION THEN CNTREC=0;
CNTREC=CNTREC+1;                          *** COUNT ITC FACILITIES;
IF LAST.REGION & CNTREC>MAXR THEN DO; *** SAVE MAXIMUM   ;
MAXR=CNTREC;
CALL SYMPUT ('MAXR',LEFT(PUT(MAXR,BEST.)));
END;
```

This now becomes a one-to-many merge problem. Because the CBC file was quite large and was not sorted by REGION, I decided to access the new ITC Facility file directly using the KEY option. I could as easily have sorted the Birth Certificate file by REGION and merged directly with the transposed ITC file. To use a file directly with the KEY option it is necessary to create an INDEX into the data set using PROC DATASETS. The code for this is shown in Figure 15.

Figure 16 illustrates the code used to merge the two files using the KEY option. When the KEY option is used, an automatic variable, _IORC_, is set to zero if a successful match is found. Also, note that I have used the UNIQUE option on the SET statement, so that the SET statement functions as an extended lookup, always returning the first value for a given KEY. Without this option the SET statement will automatically advance to the next record and no match will be found for multiple consecutive records with the same value of the KEY variable. Variables brought in by the SET statement are automatically retained; therefore, it is critical to check the value of _IORC_.

The DO loop set up in method 1 was modified to loop through the arrays of latitudes and longitudes corresponding to a specified REGION and calculate the distance using the MACRO, GETDIST. The output from the code in Figure 17 is the same as that produced by method 1 and shown in Figures 8 and 9.

Fig. 15. Create INDEX into ITC Facility File Using PROC DATASETS

```
*************************************************************************
CREATE INDEX INTO ITC FACILITY FILE
*************************************************************************;

PROC DATASETS LIBRARY=SUGI;
MODIFY TRNFACIL;
INDEX CREATE REGION;
```

Fig. 16. Merge CBC and ITC Facility Files Using the KEY Option

```
*************************************************************************
READ ITC FACILITY DATA USING KEY OPTION
*************************************************************************;

DATA GETDIST
  BRTHCRT (KEEP=REC_ID ZIPCODE REGION SAV_DIST SAV_HOSP);
SET SUGI.BRTHCRT (IN=INBC);                  ***** BIRTH CERT.FILE   ;
SET SUGI.TRNFACIL KEY=REGION / UNIQUE; *** ITC FACILITY FILE   ;
```

Fig. 17 Calculate Distance to Nearest ITC Facility Using ARRAYS

```
*************************************************************************
CALCULATE DISTANCE TO NEAREST ITC FACILITY USING ARRAYS
*************************************************************************;

ARRAY LAT_(&MAXR) LAT1-LAT&MAXR;        **** ARRAY OF ITC LAT;
ARRAY LNG_(&MAXR) LNG1-LNG&MAXR;        **** ARRAY -  ITC LONG;
ARRAY HSP_(&MAXR) HSP1-HSP&MAXR;        **** ARRAY OF ITC IDS;

SAV_DIST=.;                             **** INITIALIZE DIST    ;

IF _IORC_ =0 THEN DO I=1 TO &MAXR;      **** CHECK _IORC_    ;
  IF (H_LAT NE .) & (H_LONG NE .) THEN DO;
  %GETDIST (LAT_(I),LNG_(I),HSP_(I));   **** FIND SHORT. DIST;
  OUTPUT GETDIST;
  END;
END;

DROP LNG1-LNG&MAXR LAT1-LAT&MAXR HSP1-HSP&MAXR;
IF INBC THEN OUTPUT BRTHCRT;
```

METHOD 3: PROC SQL

PROC SQL provides the capability of merging data sets with multiple records of the BY variables. The general form of the PROC SQL step is:

```
PROC SQL;
    CREATE TABLE/VIEW table/view
     SELECT columns
         FROM table/view
         WHERE expression
         ORDER BY columns;
```

SELECT *columns* identifies the variables to retrieve.
FROM *table* identifies the file to read.
WHERE *expression* subsets the rows.
ORDER BY *columns* returns the rows in sorted order.

One record is generated for each combination of records satisfying the criteria in the WHERE statement. It is necessary to specify the input and output data sets in a FROM statement. All of the required variables must be explicitly defined in a SELECT statement. The output data set is defined in the CREATE statement.

Figure 18 shows the code required to accomplish the many-to-many merge for this example.The output for observations 2 and 3 of the CBC file is shown in Figure 19. Once the many-to-many merge has been executed, it is a very straight forward programming task to read the records using FIRST.REC_ID and LAST.REC_ID and calculate the distance using the MACRO, GETDIST. In Figure 20 we see the code required to determine the closest ITC facility. This code produces the same output generated by methods 1 and 2 and illustrated in Figures 8 and 9.

Fig. 18. Merge CBC and ITC Files Using PROC SQL

```
PROC SQL;
 CREATE TABLE GETDIST AS
 SELECT FACILITY.REGION, FACILITY.H_LAT, FACILITY.H_LONG,
        FACILITY.ITC_ID,
        BRTHCRT.REC_ID, BRTHCRT.REGION,
        BRTHCRT.LATCENT, BRTHCRT.LONGCENT
 FROM SUGI.BRTHCRT, SUGI.FACILITY
 WHERE BRTHCRT.REGION=FACILITY.REGION;
```

Fig. 19. PROC SQL Output

REC_ID=2

REGION	H_LAT	H_LONG	ITC_ID	LATCENT	LONGCENT
9	32.76	117.00	0492	32.71	117.11
9	32.75	117.17	0498	32.71	117.11
9	32.77	115.62	0078	32.71	117.11
9	32.75	117.17	0508	32.71	117.11

REC_ID=3

REGION	H_LAT	H_LONG	ITC_ID	LATCENT	LONGCENT
3	37.90	121.96	0043	37.83	122.29
3	37.86	122.24	0003	37.83	122.29
3	37.83	122.26	0005	37.83	122.29

Fig. 20. Method 3: Determine Closest ITC Facility

```
DATA BRTHCRT (KEEP=REGION REC_ID SAV_DIST SAV_HOSP);

SET GETDIST; BY REC_ID;

IF FIRST.REC_ID THEN SAV_DIST=.;

%GETDIST (H_LAT,H_LONG,ITC_ID);

IF LAST.REC_ID THEN OUTPUT BRTHCRT;
```

DISCUSSION: WHICH METHOD IS BEST?

Deciding which method to use depends upon the relative sizes of the data sets, the number of variables, and the space constraints. Method 3, PROC SQL, is probably the easiest to use. The code is simple and straight forward. There are several disadvantages, however. The user must explicitly define all variables within a SELECT statement. Other variables associated with a data set are not included by default. This can easily be solved by defining only those variables required for the match algorithm and then merging the output from the DATA step with the original data set by observation number. There is also the potential for generating a very large data set in the PROC SQL statement depending upon the relative sizes of the data sets. In the example used in this paper approximately 260,000 records were generated in the PROC SQL statement for 50,000 CBC input records-- an average of 5 records for each CBC record.

In Method 1, direct access using the POINT option, there is no space problem since the "best" record is determined during the merge step--only one record is generated for each input record. While more code is required to set-up and correctly read the direct access file, the routine is extremely flexible and no explicit reference is required for unique variables in the respective data sets.

Method 2, PROC TRANSPOSE with the KEY option, is probably the least flexible and offers no clear advantage over Methods 1 and 3. The potential for generating a very large record in the TRANSPOSE step could be a problem depending upon the contraints on record size. Knowing the maximum number of records for any value of the BY variables is essential if this technique is to be used successfully. The limitations of PROC TRANSPOSE--one record per variable for each value of the BY variable--also makes this method more cumbersome to use.

I executed the three methods using 50,000 CBC records and found that there was little difference in the overall time. The data step in method 3 required about half the time used in methods 1 and 2; however, once the time for the SQL step was taken into account, the times were essentially equivalent.

CONCLUSION

The SAS System provides the capability of performing many-to-many data set merges. Using a many-to-many merge technique provides the user with increased flexibility in searching any number of potential matches for

the "best" match. The implementation of a many-to-many merge can eliminate repeated merge steps and make more efficient use of the matching information.

This paper explores three possible techniques: (1) merging data sets by directly accessing one of the files using the POINT option; (2) utiizing the transformation capabilities of the TRANSPOSE procedure, in conjunction with the KEY option; and (3) using the SQL procedure for the merge. In my opinion method 1, while requiring more effort to program, offers the greatest flexibility and minimizes possible problems due to space constraints.

REFERENCES

Prior, Mark, "Techniques for Merging Non-Indexed Data Sets with Duplicate Values of the BY Variable," *Proceedings of the Eighteenth Annual SAS Users' Group International Conference, 1993, pp. 162-167.*

Bell, B., J. Keesey, and T. Richards, "The Urge to Merge: A Computational Method for Linking Data Sets with No Unique Identifier," *Proceedings of the Eighteenth Annual SAS Users' Group International Conference, 1993, pp. 49-54.*

Conner, Ben, "Keys to the Data: Using the KEY= OPTION on the SET Statement," *Proceedings of the Inaugural Conference of the Western Users' of SAS Software, 1993, pp. 11-13.*

SAS Institute Inc., *Making the Transition to Release 6.06 of the SAS System*, Course Notes, 1990.

ACKNOWLEDGEMENTS