

Which Merge?

Stephen Hanks, Amadeus Software Limited



WHICH MERGE?

Whether we work in Data Warehousing, Data reporting, Statistics or Pharmaceuticals we are continually asked to merge two, or more SAS data sets together in order to get the data we require. This paper compares and contrasts a number of different ways of merging SAS files. We will determine what method should be used when, and what the pros and cons of each method are. Some of the methods we look into are the match merge, using formats as a look up, using SAS/SQL® and using indexed set statements. Finally, this paper looks at how things vary if the data is in an external source or in Scalable Performance Data Server®.

INTRODUCTION

There is no one-way to merge two SAS data sets together, a number of different techniques are available. None of the techniques are right or wrong, however knowing what technique to use when, and not just sticking to one technique will make your code more efficient and simpler to understand by others. This paper will look in detail at four different methods and the advantages and disadvantages of each.

SAS MERGE

The SAS merge is one of the most simple and widely used techniques for merging two files. Firstly, both files must be sorted by the key, or keys. Once the two files are sorted, they are joined together with a merge statement in the data step.

Below is example code to merge tables A and B by the key 'i' only keeping rows (or observations) that are in table A:

```
proc sort data=a;
  by i;
run;

proc sort data=b;
  by i;
run;

data all;
  merge a(in=a) b(in=b);
  by i;
  if a;
run;
```

ADVANTAGES AND DISADVANTAGES

- All combinations of; in both data sets, in either, or in just one data set
- Simple to code
- Pointers are much more efficient than SAS SQL
- Can't do a many to many merge
- Data set needs to be sorted before merging

SQL MERGE

The SQL merge is also a popular way to join to files together, particularly with programmers who have used SQL before, and may not be as familiar with SAS code. This code is very similar to any SQL code, not just SAS SQL. This example joins the same two tables names A, B as the SAS merge code.

Below is example code to merge tables A and B by the key 'i' only keeping rows (or observations) that are in table A:

```
proc sql;
  create table all as
  select a.i, a.x, b.y
  from a left join b
```

```
on a.i=b.i
order i;
quit;
```

ADVANTAGES AND DISADVANTAGES

- All combinations of; in both data sets, in either, or in just one data set
- Can do a many to many merge
- Simple to code particularly if already familiar with SQL
- No need to sort data set before merging
- Can easily join on variables with different names
- Can be inefficient with large data sets

FORMATS TO JOIN FILES

Formats are the SAS equivalent of look up tables. Formats can be created from a SAS data set using proc format. Once a format is created it can be used in a data step with the put statement to create a new variable. With this method only one key can be used and only one new variable can be added to the current table (in this case called A). These disadvantages can be worked around by forming a new key or result as the concatenation of a number of variables.

Below is example code to merge tables A and B by the key 'i' only keeping rows (or observations) that are in table A:

```
data c;
  set b(rename=(y=label i=start)) end=eof;
  retain type 'N' fmtname 'BFMT';
  output;
  if eof then do;
    start=.;
    hlo='0';
    label=9999;
    output;
  end;
run;

proc format cntlin=c;
run;

data all;
  set a;
  x=put(i,bfmt.);
  if x=9999 then delete;
run;
```

ADVANTAGES AND DISADVANTAGES

- No need to sort data set before merging
- Can be efficient particularly if adding details from a small data set to a large data set
- Only observations common to both data sets, or alternatively in one data set can be joined
- Only use a single key, and single results (can be worked around by concatenating fields)
- Must not have duplicates in the data set used to create the format.

INDEXED KEY MERGE

The indexed key merge requires a good understanding of the LPDV. The first step is to index one data set - usually the one to be joined in needs to be indexed on the key. Once this has happened the data step can begin. Initially the first (non indexed) data set is read into the LPDV with a set statement. Hence records in this data set will be read sequentially. The second data set is also read with a set statement, however this is done with the key= option. This options means that instead

of the data set being read sequentially the index is scanned and just the observation with the same value for the key as the present value in the LPDV is read. If there is no corresponding value in the second data set then an error will occur and the internal variable `_iorc_` will not be non-zero. If this occurs, setting the internal variable `_error_` to 0 can turn off the error. The other array code is used to copy the elements from the table B into new variables in the combined table.

Below is example code to merge tables A and B by the key 'i' only keeping rows (or observations) that are in table A:

```
proc datasets lib=work;
  modify b;
  index create i/unique;
quit;

data all;
  set a;
  set b key=i/unique;
  array setmiss (*) y;
  if _iorc_ ne 0 then do;
    _error_=0;
    do i=1 to dim(setmiss);
      setmiss(i)=.;
    end;
  end;
run;
```

ADVANTAGES AND DISADVANTAGES

- No need to sort data set before merging (need to index one data set)
- Can be very efficient
- Need to index one of the data sets
- Needs good understanding of the SAS LPDV
- Only observations common to both data sets, or alternatively in one data set can be joined

RESULTS AND COMPARISON

RESULTS

The table below gives a comparison of the results for joining tables A and B in the same way as the code detailed above. All steps such as sorting and indexing are included in the timings. The system CPU time was the measure used to compare runs. This is just one measure of efficiency; however it is a good indication of how long the process will take. The result timings have been averaged over three runs.

The sizes of the two tables increase as shown in the table below. The results of the merge will have the same number of rows or observations as table A.

Total Time	Size A	Size B						
		10	100	1,000	10,000	100,000	1,000,000	10,000,000
SAS Merge (including 2 sorts)	100	0.02	0.02	0.01	0.02	0.11	0.92	11.12
SAS SQL		0.01	0.01	0.01	0.02	0.09	0.79	11.59
Format (include data step for proc format)		0.03	0.03	0.03	0.07	0.56	N/A	N/A
Index Key Merge		0.01	0.01	0.01	0.02	0.08	0.69	8.08
SAS Merge (including 2 sorts)	1,000	0.01	0.01	0.02	0.03	0.11	0.91	10.82
SAS SQL		0.01	0.01	0.00	0.03	0.10	0.82	10.95
Format (include data step for proc format)		0.03	0.03	0.01	0.09	0.59	N/A	N/A
Index Key Merge		0.02	0.02	0.01	0.02	0.09	0.70	8.01
SAS Merge (including 2 sorts)	10,000	0.03	0.04	0.03	0.03	0.12	0.93	10.90
SAS SQL		0.02	0.03	0.02	0.05	0.11	0.90	10.45
Format (include data step for proc format)		0.02	0.03	0.04	0.09	0.60	N/A	N/A
Index Key Merge		0.01	0.03	0.02	0.03	0.09	0.70	8.13
SAS Merge (including 2 sorts)	100,000	0.16	0.14	0.15	0.15	0.24	1.04	11.06
SAS SQL		0.17	0.16	0.15	0.19	0.37	1.19	11.60
Format (include data step for proc format)		0.07	0.07	0.07	0.15	0.71	N/A	N/A
Index Key Merge		0.06	0.08	0.06	0.08	0.15	0.76	8.55
SAS Merge (including 2 sorts)	1,000,000	1.40	1.26	1.25	1.25	1.37	2.16	12.50
SAS SQL		1.70	1.50	1.48	1.50	1.69	3.88	16.46
Format (include data step for proc format)		0.49	0.56	0.49	0.61	1.13	N/A	N/A
Index Key Merge		0.47	0.57	0.47	0.54	0.57	1.36	8.74
SAS Merge (including 2 sorts)	10,000,000	14.68	14.21	13.93	13.90	14.06	14.85	25.63
SAS SQL		17.22	16.75	16.48	16.28	17.61	20.18	51.87
Format (include data step for proc format)		5.68	5.30	5.47	6.06	6.41	N/A	N/A
Index Key Merge		5.15	6.07	5.84	5.91	5.04	5.84	17.53

It can be seen from the table that the time taken for a **SAS merge** increases as the size of both files increase. There is very little advantage for files being of different size.

SAS SQL reduces the need to sort the data sets however the SQL step takes longer than the SAS merge as the SQL step puts all possible combinations together in memory. This means that SQL is more efficient than a SAS merge for smaller data sets but as the data sets get larger the memory required increases and the time to run a SQL merge increases dramatically. The cross over point where SQL is more efficient

then a SAS merge depends on the memory available in the machine.

Using a **format** to merge data sets is very efficient as it reduces the need to sort either data set. However the format is held in memory and needs to be loaded in one go. Hence the data set that is turned into a format can't get too big or else it will not fit into memory. This is a very effective way to join a large file to a small file where the small file can be made into a format.

The **indexed key merge** regularly appears to be the quickest

way of joining two large files. The main down side of this method is that it is the most complicated, and it is not always best to write the most efficient code if no one can support it. It is worth considering though when speed is of the utmost importance, particularly with files of unequal size.

COMPARISON

While there are no hard and fast rules of what merge to use when, there are a number of things to keep in mind.

With very small data sets, the greatest expense may be programmer time. This may be more important than speed or space, so writing simple maintainable code may be the most important thing.

With larger files, writing the most efficient code possible may be critical particularly is a job must fit within a set time 'window'.

Knowing what is available to the programmer and being able to choose may be as important as the actual choice. Sometimes the most inefficient code may in fact be the best solution as it is the simplest to code and the easiest to maintain but in other circumstances this may be very inappropriate.

Keeping this in mind here are some guidelines of what to use when.

- If you have a many to many merge the only option is SAS SQL.
- If observations that are in either data set are required then SAS SQL or SAS Merge are the better options. If either data set is large then SAS merge is the better option. (Large will depend of the system and particularly the amount of memory available)
- If you want to add variables to a main file or alternatively select observations that are in both data sets then all four options are available to you.
- If you are adding information from a small file to a very large file then using formats is efficient and simple.
- If efficiency is the only criteria then the index key merge is the way to go, particularly if the files are both large but of differing size.
- If both files are small the SQL is usually the simplest and most efficient approach.

SPD SERVER AND SAS/ACCESS

SPD SERVER

When using SPD server there is no need to sort or index the files. SPD server does this for you through the way the files are stored in SPD. This means that the SAS Merge and to a lesser extent the indexed key merge are more efficient.

SAS/ACCESS

When accessing data with SAS/ACCESS there are a number of other things to consider.

We have discovered that with large SAS data sets, a SAS merge with sorts is more efficient then a SAS SQL Merge. But this does not mean that SQL pass-through will be more efficient than copying files to SAS and then merging.

Other issues to be considered with SAS/ACCESS are:

- Network speed between the database and SAS
- The speed of the Database (either machine or priority settings) compared to the speed of SAS (either machine or priority settings).

In some setups it is best to copy the data straight into SAS and then join in SAS. In other set ups it is much more efficient to do

the join with SQL pass-through in the database and then return the results to SAS.

CONCLUSION

Detailed in this report are four different ways of merging two files together. Some are more efficient options, and others are simpler. A good SAS programmer is aware of all the options, keeps them in mind and chooses the best option for the current situation, rather than being restricted to a smaller list of techniques the programmer is comfortable with.

Thinking carefully about what technique to use when, means we are 90% of the way towards choosing the best technique.

CONTACT INFORMATION

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

Author Name	Stephen Hanks
Company:	Amadeus Software Limited
Address:	Orchard Farm, Witney Lane, Leafield, Oxon OX29 9PG
Work Phone:	01993 878287
Fax:	01993 878042
Email:	steve.hanks@amadeus.co.uk
Web:	www.amadeus.co.uk

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.