Paper 95

What Sort of Input Should You Input to a Sort?

Michael A. Raithel, WESTAT

Abstract

Because the SAS System makes sorting so simple, some programmers sort an entire SAS data set without giving it a second thought. But, efficiency-minded programmers often pause to consider what portion of a SAS data set actually needs to be input to PROC SORT. What if only a specific subset of a SAS data set's variables are required in the final sorted data set? Is it really necessary for the entire data set be sorted, or is there a more efficient alternative?

This paper discusses different methods of inputting a SAS data set to PROC SORT. It compares the efficiency of sorting an entire SAS data set with three methods of subsetting the data set's variables and inputting them to PROC SORT. This practical information can be used by SAS programmers to improve the efficiency of their sorts.

Introduction

Consider a situation in which you need a sorted SAS data set as input to a SAS program. You know that the sort will incur computer overhead in the form of expended CPU time and EXCP's (I/O's). If the SAS data set is very large, and if the observations are very long, then the expense of the sort will be very high. You may consider that the expenditure is acceptable, since the data set *must* be sorted. But, you should stop and ask yourself if it is really necessary to sort the *entire* SAS data set. Will you be using all of the variables in the SAS data set past the point of the sort? If not, will you be using a relatively small subset of the variables?

If the answer to the two questions, above, is "yes" then you will want to subset the number of variables that are input to the sort. But, what is the most efficient way to do this? The following sections provide an answer to this question. They detail three separate methods of subsetting a SAS data set's variables as it is input to a sort. The final section compares the relative merits of these methods and provides some concluding remarks.

Sorting the Entire SAS Data Set

The easiest thing to do is to sort the entire SAS data set. This might even be the "right" thing to do if the data set has relatively few observations and the observation length is small. But, if the opposite is true, then this will not be an efficient way to go when only a small number of variables are needed after the sort. Here is an example of sorting an entire SAS data set:

```
proc sort data=proddata out=sortprod;
          by state city cdid;
run;
```

1. Sort With a KEEP= Statement on the DATA= Option

The first alternative method to a full SAS data set sort is to use the KEEP= statement on the DATA= option of the SORT statement. The KEEP= statement directs the SAS System to keep specific variables. By using the KEEP= statement, you can limit the variables that are input to the sort. Here is an example of utilizing this method:

```
proc sort data=proddata(keep=state city cdid numsold
notsold totsales);
       by state city cdid;
run;
```

2. Utilizing a Subsetting DATA Step Before the Sort

A second alternative is to use a DATA step to subset the variables of the SAS data set. The resulting SAS data set is then input to the sort. This method ensures that the sort receives a "slimmed down" version of the original SAS data set. Here is an example of how this method could be coded:

```
data  subdata;
set proddata(keep=state city cdid numsold notsold
totsales);
run;

proc sort data=subdata;
       by state city cdid;
run;
```

3. Utilizing a Subsetting DATA Step VIEW Before the Sort

A third alternative method to a full data set sort is to use a DATA Step VIEW to subset the SAS data set's variables before the sort. The VIEW, with its fewer variables, would be input to the sort. Thus, the SORT procedure

would materialize the VIEW and use its "slimmer" version of the original SAS data set as input. Here is an example of this method:

```
data  subdata / view=subdata;
set proddata(keep=state city cdid numsold notsold
totsales);
run;

proc sort data=subdata out=sortdata;
        by state city cdid;
run;
```

Note that in the SORT procedure, above, the OUT= option is used. This is necessary since a SAS data set VIEW can not be sorted. The data set VIEW can be a sort input, but the sort output must be directed to a new SAS data set. Thus, SUBDATA—the data set VIEW—is the input, and SORTDATA—a new SAS data set—is the output.

Conclusions

The three methods, above, were used to subset the variables of a SAS data set with 210 variables and 100,000 observations. Of the 210 variables, only ten were needed past the sort, so they were the ones that were kept. Three of the ten variables were sort BY variables. The table at the end of the paper, below, summarizes the total CPU time and EXCP Count utilized by each of the three methods that were discussed.

It should come as no surprise that the most expensive sort was the sort of the entire SAS data set. The least expensive method, in terms of CPU time, was to first subset the number of variables with a DATA step. Similarly, the leaset expensive method, in terms of EXCP Count was to add a subsetting DATA step VIEW. It is interesting to note that adding a DATA step or a DATA step VIEW *actually reduced* the CPU time and the EXCP Count of the sort. Yet it did! This seems counter-intuitive to the lack-luster method of using a KEEP= on the SORT procedure's DATA= option.

So, what sort of input should you input to a sort? Well, if you have a very large SAS data set with long observations, and you only need a small subset of the variables, then you should input either: A SAS data set that has subset the number of variables, or A SAS data set VIEW that has subset the number of variables.

Method	**CPU Time (secs)**	**EXCP Count**
Sort of Entire Data Set	2.85	856
1. KEEP= on DATA= Option	2.49	536
2. Subsetting DATA Step	1.79	501
3. Subsetting Data Step VIEW	1.86	440

Trademarks

References

Raithel, Michael A.
Tuning SAS Applications in the MVS Environment
Cary, NC:SAS Institute Inc.,1995
303pp.

Contact Information

Michael A. Raithel
WESTAT
1650 Research Boulevard
Rockville, Md. 20850-3129
E-mail: maraithel@erols.com