Paper 165-28

New Ways and Means to Summarize Files

Curtis A. Smith, Defense Contract Audit Agency, La Mirada, CA

ABSTRACT

Frequently, when creating files for a data warehouse or to be used within a SAS program, it makes sense to create smaller files to save space and reduce processing time. Sometimes creating summary files is the solution. Of course, the SUMMARY procedure is the tool to use to create a summarized SAS data set. While this procedure has been available for several version of SAS, version 8 of SAS gives us some significant new features that we can use to get more information in our summarized files and to optimize the summary process. In this paper the author will discuss the techniques and programming code for creating summary files, where the SAS programmer collapses a data file into fewer observations and variables. The author will give special attention to the new version 8 features.

INTRODUCTION

You will probably want to build efficiencies into your data warehouse and SAS jobs by summarizing SAS data sets where possible. When you summarize a file you collapse the file by eliminating unneeded variables and aggregating the numeric values. Compared to the original detailed file, a summarized file requires less storage space, less processing time when used, and less I/O operations when used. In this paper, I will discuss some basic techniques to summarize a SAS data set using the SUMMARY procedure. But I will concentrate on new SAS version 8 features in the SUMMARY procedure that give you more control over the output and can reduce the processing time needed to create a summarized SAS data set.

PURPOSE OF SUMMARIZE A SAS DATA SET

Summarizing a SAS data set makes sense whether you do it in a job stream or for permanent storage in your data warehouse. Summarized SAS data sets are smaller, reducing storage space requirements, reduce I/O operations, and require less time to process. Sometimes you may need to convert a large data file into a lookup table with only one occurrence of a combination of character variables. Summarizing your file is a way to accomplish such a task. Consider the following sample master SAS data set:

NAME	DIV	ACCOUNT	JOURNAL	DEPT	AMOUNT
Al Abaster	A1	3101	AAA	H100	100.00
Ivan Aikenback	A1	3101	BBB	H100	550.00
Robin Steele	A1	3225	BBB	H200	300.00
Sharon B. Good	A1	3225	BBB	H300	250.00
Patty O'Furniture	A1	3225	CCC	H300	110.00
Peter Pants	B2	3101	AAA	H100	450.00
Drew Blanks	B2	3225	BBB	H100	325.00
Chester Drawers	B2	3225	BBB	H200	50.00
Hoosier Taylor	C3	3225	AAA	H100	25.00
Bea Hemoth	C3	3225	AAA	H300	1050.00

In this master SAS data set there are four pieces of accounting data (variables or columns): the division (DIV), account (ACCOUNT), journal (JOURNAL), and department (DEPT). And, there is an amount (AMOUNT) and a name (NAME) associated with each transaction. Our master SAS data set contains ten observations (rows). Notice that each row is unique, by the combination of the four pieces of accounting data.

If you are building a data warehouse for your users but your users never need the transaction detail for the journal, department, or name, you could drop these variables.

The resulting SAS data sets would look like the what you see in the following box.

DIV	ACCOUNT	AMOUNT
A1	3101	100.00
A1	3101	550.00
A1	3225	300.00
A1	3225	250.00
A1	3225	110.00
B2	3101	450.00
B2	3225	325.00
B2	3225	50.00
C3	3225	25.00
C3	3225	1050.00

But notice that each row is no longer unique, considering the remaining two pieces of accounting data. The first and second row are both recorded to division A1 and account 3101. The third and fourth row are both recorded to division A1 and account 3225. Similarly, the seventh and eighth rows are recorded to division B2 and account 3225. And the ninth and tenth rows are recorded to division C3 and account 3225. This circumstance provides you the opportunity to summarize (collapse) these records together while accumulating the amounts (thus preventing any numeric value loss). Once done, your resulting SAS data sets will look like this:

DIV	ACCOUNT	AMOUNT
A1	3101	650.00
A1	3225	660.00
B2	3101	450.00
B2	3225	375.00
C3	3225	1075.00

Eliminating variables will not always result in rows that are no longer unique. So, it is possible to drop variables but not be able to benefit from summarizing the SAS data sets. But when you can summarize your SAS data sets, you can save a substantial amount of storage space and greatly reduce processing time when using the summarized SAS data sets.

HOW TO SUMMARIZE

The SUMMARY procedure creates a new SAS data set containing summary statistics on numeric variables from an existing SAS data set. Usually, this procedure is used to summarize a SAS data set into a smaller SAS data set having only one record for each occurrence of specified variable(s). Summarizing a SAS data set is very useful when you expect to use a SAS data set more than once and don't need the original level of detail. Having the data summarized into a smaller data set reduces the amount of processing time and associated cost for each report or other procedure and reduces the cost of storage. Also, often original SAS data sets are too large to store on disk, but a summarized SAS data set can be stored on disk.

PROCEDURE SYNTAX

If you look in the SAS documentation for information about the SUMMARY procedure, you will find yourself re-directed to the MEANS procedure. The MEANS and SUMMARY procedure are the same, except that the SUMMARY procedure creates an output SAS data set, which is what you want when you need to create a summary file. Following is the syntax for the SUMMARY procedure. We will examine each statement below and look at some examples of code, log, and output.

```
PROC SUMMARY DATA=libref.filename options;
  BY (or CLASS) variable-list;
  ID variable-list;
  VAR variable-list;
  TYPES variable-list;
  WAYS n;
  OUTPUT OUT=libref.filename(options)
      output statistic=variable-list/options;
RUN;
```

SPECIFICATIONS

We will begin discussing some old tried and true capabilities within the SUMMARY procedure. Then, we will explore the new features available in version 8. Consider the following examples:

```
PROC SUMMARY DATA=TAPE.BIGGER;
  BY DIV ACCOUNT JOURNAL;
  ID DESCR;
  VAR AMOUNT HOURS;
  OUTPUT OUT=DISK.SMALLER SUM=AMOUNT HOURS;
RUN;

PROC SUMMARY DATA=TAPE.BIGGER;
  CLASS DIV ACCOUNT JOURNAL;
  VAR AMOUNT;
  OUTPUT OUT=DISK.SMALLER(DROP= TYPE )
    SUM=AMOUNT;
RUN;
```

BY AND CLASS STATEMENT

These statements are used to control the order or grouping of selected variables within the SAS data set. In the examples above, the input libref.filename is 'TAPE.BIGGER' and output libref.filename is 'DISK.SMALLER'. The output file is summarized on the combination of DIV, ACCOUNT, and JOURNAL. When you use the BY statement to identify the variables on which you want to summarize, the input file must already be sorted or indexed by the same variables. The CLASS statement can be used in place of the BY statement and will sort and summarize the input file, eliminating the need to pre-sort or index the input file. However, the CLASS statement does require more memory to function than does the BY statement. Therefore, if the input file is large (hundreds of thousands of records or more) or if you are including many variables in your CLASS statement, the SUMMARY procedure may fail for lack of memory. Also, when the CLASS statement is used, the SUMMARY procedure will exclude any rows with a missing value in any of the variables in the CLASS list. This could have disastrous results. However, losing such rows will be prevented if you use the MISSING option (more on this later). All variables identified in a BY or CLASS statement should be character variables, unless they are numeric variables with few, discrete values.

With the CLASS statement the SUMMARY procedure creates summary records for each possible combination of variables in the CLASS list. In the example above the SUMMARY procedure would produce a summary row for each unique combination of DIV, ACCOUNT, and JOURNAL, and a summary row for each unique combination of DIV and ACCOUNT, and a summary row for each unique combination of DIV and JOURNAL, and so forth. When all you want to do is to collapsed your input data set by a combination of variables you want only one row for each unique combination of all variables in the CLASS list (the highest level of interaction). The other rows the SUMMARY procedure creates just get in your way. If you CLASS statement and want rows for only the highest level of interaction between the variables, there is an option to do this (more on this later). If you use the BY statement the SUMMARY procedure will produce rows for only the highest level of interaction between the variables in the BY list.

ID STATEMENT

The ID statement is used to specify any character variables not specified in the BY or CLASS variable list that you want to retain in the output SAS data set. Variables you specify with the ID statement are not summarized. Rather, the value of the ID variable on the last row summarized is retained. This is useful when a variable must be retained in the output file, but whose value is the same for each combination of the BY or CLASS variable list. Variables listed with an ID statement could be added to the BY or CLASS variable list to produce the same result. However, doing so will create a more complicated summarization sequence that will slow the summarization process.

VAR STATEMENT

The VAR statement is used to specify the numeric variables that will be aggregated. Any variable in the input data set not specified with either a BY, CLASS, ID, or VAR statement will not be included in the output data set.

OUTPUT STATEMENT

The OUTPUT statement is used to specify the output SAS data set to be created and to specify the output statistic to aggregate the numeric variables identified in the VAR list. Within the OUTPUT statement you specify the output libref.filename with the OUT= option. Specify the output statistic, followed by an "=" and followed by the variables in the VAR list. If your goal is to collapse the input SAS data set into a less detailed output SAS data set, you will use the SUM output statistic (see the SAS Online Documentation for other output statistics).

AUTOMATIC VARIABLES

The SUMMARY procedure creates two variables automatically: _FREQ_ and _TYPE_. A numeric count of the number of rows from the input SAS data set summarized into each single row in the output file is stored in the _FREQ_ variable. The _TYPE_ variable contains a numeric value identifying the level of interaction between the variables in the CLASS list. When a BY statement is used, the _TYPE_ variable will always equal 0. When the CLASS statement is used, the _TYPE_ variable will contain 0 for a grand total row and values of 1 through n for various levels of interaction between the variables in the CLASS list.

The SUMMARY procedure creates two automatic variables: _FREQ_ and _TYPE_

What Do We Have So Far?

Using the statements and options presented so far, let's summarize a SAS data set. First, let's summarize our sugi.master file using a BY statement. This, of course, requires we first sort the unsorted file on the same variables on which we plan to summarize. Consider the following code, SAS log, and results.

```
proc sort data=sugi.master out=work.master;
    by div account;
run;
proc summary data=work.master ;
    By div account;
    var amount;
    output out=sugi.summary sum=AMOUNT;
run;
```

In this example, you will notice first a SORT procedure and then a SUMMARY procedure using the BY statement. First, notice that in both the SORT and SUMMARY procedures the BY statements are identical. Then, notice in the log below that the SAS data set *work.master* contains 10 rows and the SAS data set *SUGI.summary* contains 5 rows.

```
209  proc sort data=sugi.master out=work.master;
210       by div account;
211  run;
NOTE: There were 10 observations read from the data
set SUGI.MASTER.
NOTE: The data set WORK.MASTER has 10 observations
and 5 variables.
223  proc summary data=work.master;
224       by div account;
225       var amount;
226       output out=sugi.summary_by sum=AMOUNT;
227  run;

NOTE: There were 10 observations read from the data
set WORK.MASTER.
NOTE: The data set SUGI.SUMMARY_BY has 5
```

Now, take a look at those 5 rows in the output file.

Summary with BY

Obs	DIV	ACCOUNT	_TYPE_	_FREQ_	AMOUNT
1	A1	3101	0	2	650
2	A1	3225	0	3	660
3	B2	3101	0	1	450
4	B2	3225	0	2	375
5	C3	3225	0	2	1075

Notice the frequency counts under the automatic variable _FREQ_.

Also, notice the automatic variable _TYPE_ and the values of "0" for each row. Harkening back to the early portion of this paper, the output summary rows are the results we expected. Let's summarize our file again using the CLASS statement. Notice this time we do not need to pre-sort our unsorted file because the CLASS statement takes care of sorting. Consider the following code, SAS log, and results.

```
proc summary data=work.master ;
    class div account;
    var amount;
    output out=sugi.summary sum=AMOUNT;
run;
```

```
213  proc summary data=sugi.master ;
214       class div account;
215       var amount;
216       output out=sugi.summary sum=AMOUNT;
217  run;

NOTE: There were 10 observations read from the data
set SUGI.MASTER.
NOTE: The data set SUGI.SUMMARY has 11 observations
```

Notice in the log that we now have 11 rows! In this case, that's more rows than in the input SAS data set. How could that be? Let's look at those 11 rows in detail.

The CLASS statement with the NWAY option produces a summary for only the highest interaction between the variables in the CLASS list.

Summary with CLASS Only

Obs	DIV	ACCOUNT	_TYPE_	_FREQ_	AMOUNT
1			0	10	2160
2		3101	1	3	1100
3		3225	1	7	1060
4	A1		2	5	1310
5	B2		2	3	825
6	C3		2	2	25
7	A1	3101	3	2	650
8	A1	3225	3	3	660
9	B2	3101	3	1	450
10	B2	3225	3	2	375
11	C3	3225	3	2	1075

Look at what the CLASS statement provides for you. You get a set of rows for every combination of variables in the CLASS list. In this example, that means you get summary rows for every unique DIV, and every unique ACCOUNT, and every unique combination of DIV and ACCOUNT, and a grand total. Notice that each combination has a different _TYPE_ value. For example, each combination of ACCOUNT is _TYPE_ = 1 and each combination of DIV is _TYPE_ = 2. The SUMMARY procedure behaves this way to provide an output file with one row for every unique combination of all class variables. In a later data step or procedure if, for example, you wanted to analyze the summary file for only every unique combination of DIV and ACCOUNT, you could use a WHERE statement for _TYPE_ = 3.

What More Can We Do?

There are two very important options available to alter the way the SUMMARY procedure behaves when we use the CLASS statement.

OPTIONS

There are two important SUMMARY procedure options: MISSING and NWAY. The MISSING option instructs the SUMMARY procedure to considers missing values in a class variable when creating summary rows. If you omit the MISSING option, the SUMMARY procedure excludes any rows with a missing value in a CLASS variable from the resulting output SAS data set. The MISSING option can either be placed on the PROC SUMMARY statement or following a "/" at the end of the CLASS statement. The NWAY options instructs the SUMMARY procedure to only create rows with a combination of all class variables. These options are available only when used with the CLASS statement, not with the BY statement. When the SUMMARY procedure is used with the BY statement it will produce the same output file as when used with the CLASS statement combined with the NWAY option.

Let's take a look at the same code as our previous example but adding the NWAY and MISSING options and notice the difference to the log and output.

```
proc summary data=work.master nway;
    class div account /missing;
    var amount;
    output out=sugi.summary_nway sum=AMOUNT;
run;
```

In the log you will see that the SAS data set *work.master* contains 10 rows and the SAS data set *sugi.summary* contains 5 rows.

```
218  proc summary data=work.master nway;
219      class div account /missing;
220      var amount;
221      output out=sugi.summary_nway sum=AMOUNT;
222  run;

NOTE: There were 10 observations read from the data
set WORK.MASTER.
NOTE: The data set SUGI.SUMMARY_NWAY has 5
observations and 5 variables.
```

Now, take a look at those 5 rows.

Summary with CLASS and NWAY Option

Obs	DIV	ACCOUNT	_TYPE_	_FREQ_	AMOUNT
1	A1	3101	3	2	650
2	A1	3225	3	3	660
3	B2	3101	3	1	450
4	B2	3225	3	2	375
5	C3	3225	3	2	1075

These results look like the results we want - only 5 summarized rows. In fact, at first glance, it looks like the same results as when we used the BY statement. However, a closer look will reveal that when we used the BY statement the _TYPE_ values were always "0". But in this case, the _TYPE_ values are all "3". Why? Looking at our first example using the CLASS statement (but without the NWAY option), the rows with all possible combinations of DIV and ACCOUNT the _TYPE_ value is "3". Thus, the NWAY options causes the SUMMARY procedure to exclude all _TYPE_ combinations other than the _TYPE_ that results from the combination of all class variables. In the case of using the BY statement, the _TYPE_ variable is just filled with "0" as there are no class variables.

When you use the CLASS statement with the NWAY option to keep only rows with the highest level of interaction between the class variables, the _TYPE_ variable will always contain the same value. Therefore, I don't find the _TYPE_ variable useful when I use the NWAY option (or, when using the BY statement). Therefore, I use the (DROP=_TYPE_) data set option to eliminate the unneeded _TYPE_ variable from the output file.

Version 8 Goodies

As advertised, there are new features to the SUMMARY procedure in version 8. Let's take a look.

OPTIONAL VARIABLES

The LEVELS and WAYS options can be added to the OUTPUT statement to cause the SUMMARY procedure to include in the output the _LEVEL_ and _WAY_ variables. The _LEVEL_ variable contains a value from 1 to *n* that indicates the combination of class variables. The _WAY_ variable contains a value from 1 to the maximum number of class variables that indicates how many class variables the SUMMARY procedure combines to create a row in the output SAS data set. To use these options, add them to the OUTPUT statement after a "/". (Look for these in all of the upcoming code and output examples.)

WAYS STATEMENT

The WAYS statement is used to specify the number of ways to make combinations of class variables. This causes the SUMMARY procedure to create summary rows for only the combinations specified. The WAYS statement does not work with the BY statement. You use this statement by specifying one or more integers that define the number of class variables to combine. For example, if you want only the row combination of DIV and ACCOUNT then use the following WAYS statement:

```
WAYS 2;
```

You can also request multiple ways. For example, if you want all the rows representing a combination of two class variables and you want the row representing the grand total (no combination of any class variables), you can use the following WAYS statement:

```
WAYS 0 2;
```

Let's take a look at SAS code, log, and output. Here we will run three SUMMARY procedures: first with WAYS 0, then WAYS 1, and then WAYS 2. (Only the example with WAYS 0 is shown in the code example.)

```
proc summary data=work.master missing;
    class div account;
    var amount;
    ways 0;
    output out=sugi.summary_ways0 sum=AMOUNT
       /LEVELS WAYS;
run;
```

```
228  proc summary data=work.master missing;
229      class div account;
230      var amount;
231      ways 0;
232      output out=SUGI.summary_ways0 sum=AMOUNT
     /LEVELS WAYS;
233  run;
NOTE: There were 10 observations read from the data
set WORK.MASTER.
NOTE: The data set SUGI.SUMMARY_WAYS0 has 1
observations and 7 variables.
234  proc summary data=work.master missing;
235      class div account;
236      var amount;
237      ways 1;
238      output out=SUGI.summary_ways1 sum=AMOUNT
      /LEVELS WAYS;
239  run;
NOTE: There were 10 observations read from the data
set WORK.MASTER.
NOTE: The data set SUGI.SUMMARY_WAYS1 has 5
observations and 7 variables.
240  proc summary data=work.master missing;
241      class div account;
242      var amount;
243      ways 2;
244      output out=SUGI.summary_ways2 sum=AMOUNT
      /LEVELS WAYS;
245  run;
NOTE: There were 10 observations read from the data
set WORK.MASTER.
NOTE: The data set SUGI.SUMMARY_WAYS2 has 5
observations and 7 variables.
```

Now, take a look at the results in the three output files. For comparison, refer to the output earlier when we used the SUMMARY procedure with the CLASS statement only, without using the NWAY option.

Summary with CLASS and WAYS 0

Obs	DIV	ACCOUNT	_WAY_	_TYPE_	_LEVEL_	_FREQ_	AMOUNT
1			0	0	1	10	3210

Notice in the example above with WAY 0 (the _WAY_ variable will always contain "0"), we got the same row as the _TYPE_ = 0 because we asked for the combination of none of the class

variables.

Summary with CLASS and WAYS 1

Obs	DIV	ACCOUNT	_WAY_	_TYPE_	_LEVEL_	_FREQ_	AMOUNT
1		3101	1	1	1	3	1100
2		3225	1	1	2	7	1060
3	A1		1	2	1	5	1310
4	B2		1	2	2	3	825
5	C3		1	2	3	2	1075

In the case of WAYS 1 above (the _WAY_ variable will always contain "1"), we got the rows for combinations of any one class variable. This equates, in our example, to _TYPE_ = 1 and 2. Notice what the _LEVEL_ variable is doing. Each row within each _TYPE_ is successively incremented.

Summary with CLASS and WAYS 2

Obs	DIV	ACCOUNT	_WAY_	_TYPE_	_LEVEL_	_FREQ_	AMOUNT
1	A1	3101	2	3	1	2	650
2	A1	3225	2	3	2	3	660
3	B2	3101	2	3	3	1	450
4	B2	3225	2	3	4	2	375
5	C3	3225	2	3	5	2	1075

In the case of WAYS 2 above (the _WAY_ variable will always contain "2"), we got the rows for combinations of any two class variables. This equates, in our example, to _TYPE_ = 3. Notice what the _LEVEL_ variable is doing. In this case all of the rows have the same _TYPE_ value, so the _LEVEL_ values are simply 1 through 5.

TYPES STATEMENT

The TYPES statement creates summary rows for combinations of specified class variables. The TYPES statement does not work with the BY statement. You use this statement by specifying each combination of class variables you want included in the summary process by stating the class variables separated by an asterisk. For example, if you want only the row combination of DIV and ACCOUNT, and you want the rows for DIV only, then use the following TYPES statement:

```
TYPES DIV*ACCOUNT DIV;
```

If you want the grand total row, use the syntax TYPES (). Sound confusing? Let's take a look at SAS code, log, and output. Here we will run three SUMMARY procedures: first with TYPES DIV, then TYPES ACCOUNT, and then TYPES DIV*ACCOUNT.

```
proc summary data=work.master missing;
    class div account;
    var amount;
    types div;
    output out=SUGI.summary_types1 sum=AMOUNT
       /LEVELS WAYS;
run;
proc summary data=work.master missing;
    class div account;
    var amount;
    types account;
    output out=SUGI.summary_types2 sum=AMOUNT
       /LEVELS WAYS;
run;
proc summary data=work.master missing;
    class div account;
    var amount;
    types div*account;
    output out=SUGI.summary_types3 sum=AMOUNT
      /LEVELS WAYS;
run;
```

```
246  proc summary data=work.master missing;
247       class div account;
248       var amount;
249       types div;
250       output out=SUGI.summary_types1 sum=AMOUNT
/LEVELS WAYS;
251  run;
NOTE: There were 10 observations read from the data
set WORK.MASTER.
NOTE: The data set SUGI.SUMMARY_TYPES1 has 3
observations and 7 variables.
252  proc summary data=work.master missing;
253       class div account;
254       var amount;
255       types account;
256       output out=SUGI.summary_types2 sum=AMOUNT
/LEVELS WAYS;
257  run;
NOTE: There were 10 observations read from the data
set WORK.MASTER.
NOTE: The data set SUGI.SUMMARY_TYPES2 has 2
observations and 7 variables.
258  proc summary data=work.master missing;
259       class div account;
260       var amount;
261       types div*account;
262       output out=SUGI.summary_types3 sum=AMOUNT
/LEVELS WAYS;
263  run;
NOTE: There were 10 observations read from the data
```

Take a look at the results in the three output files. For comparison, refer to the output earlier when we used the SUMMARY procedure with the CLASS statement only, without using the NWAY option.

Summary with CLASS and TYPES on DIV Variable

Obs	DIV	ACCOUNT	_WAY_	_TYPE_	_LEVEL_	_FREQ_	AMOUNT
1	A1		1	2	1	5	1310
2	B2		1	2	2	3	825
3	C3		1	2	3	2	1075

In the example above we asked for TYPES on DIV only. We got one row for each value of DIV. Notice the _WAY_ value is "1" because only variable is combined into the output row.

Summary with CLASS and TYPES on ACCOUNT Variable

Obs	DIV	ACCOUNT	_WAY_	_TYPE_	_LEVEL_	_FREQ_	AMOUNT
1		3101	1	1	1	3	1100
2		3225	1	1	2	7	2110

In the example above we asked for TYPES on ACCOUNT only. We got one row for each value of ACCOUNT. Notice the _WAY_ value is "1" because only variable is combined into the output row.

Summary with CLASS and TYPES on DIV*ACCOUNT Variable

Obs	DIV	ACCOUNT	_WAY_	_TYPE_	_LEVEL_	_FREQ_	AMOUNT
1	A1	3101	2	3	1	2	650
2	A1	3225	2	3	2	3	660
3	B2	3101	2	3	3	1	450
4	B2	3225	2	3	4	2	375
5	C3	3225	2	3	5	2	1075

In the example above we asked for TYPES on the combination of DIV and ACCOUNT. We got one row for each combination of the value of DIV and ACCOUNT. Notice the _WAY_ value is "2" because two variables are combined into the output row.

What's the Difference?
A careful review of the results using the NWAY option, not using the NWAY option, using the WAYS statement, and using the TYPES statement reveal that you can use these in some instances to achieve the same results. For example, using the CLASS statement and the NWAY option will result in the same output if you use the CLASS statement and the WAYS statement specifying the same number of variables that are in the class list. Likewise, you will get the same results if you use the CLASS statement with the TYPES statement while listing the same variables as in the class list each separated by an "*". The following three approaches will produce the same results when wanted just the highest level of interaction.

```
PROC SUMMARY DATA=SUGI.MASTER NWAY;
CLASS DIV ACCOUNT JOURNAL;
more code...

PROC SUMMARY DATA=SUGI.MASTER;
CLASS DIV ACCOUNT JOURNAL;
WAYS 3;
more code...

PROC SUMMARY DATA=SUGI.MASTER;
CLASS DIV ACCOUNT JOURNAL;
TYPES DIV*ACCOUNT*JOURNAL;
more code...
```

Similarly, the following three approaches will produce the same results when wanting all levels of interaction.

```
PROC SUMMARY DATA=SUGI.MASTER;
CLASS DIV ACCOUNT JOURNAL;
more code...

PROC SUMMARY DATA=SUGI.MASTER;
CLASS DIV ACCOUNT JOURNAL;
WAYS 0 1 2 3;
more code...

PROC SUMMARY DATA=SUGI.MASTER;
CLASS DIV ACCOUNT JOURNAL;
TYPES () DIV ACCOUNT JOURNAL DIV*ACCOUNT
      DIV*JOURNAL ACCOUNT*JOURNAL
      DIV*ACCOUNT*JOURNAL;
more code...
```

So, why bother with the WAYS and TYPES statements? Well, SAS must have added them for some reason, right? Well first of all, you might want some of the combination of rows created by the CLASS statement without the NWAY option, but not all of the combinations. The WAYS and TYPES statements let you select specific combinations to create. Another reason is performance. The SAS documentation indicates that when the WAYS and TYPES statements are used they instruct the SUMMARY procedure to create only the specified combinations of variables. But, apparently, when the CLASS statement is used without a WAYS or TYPES statement and is used with the NWAY option the SUMMARY procedure will create all combinations but only keep the rows for the highest level of interaction. As a result, using the WAYS or TYPES statement with the CLASS statement should result in less processing time and less memory usage. Let's take a look at some benchmark results.

I processed a SAS data set containing 996,204 rows (one of my smaller files) on a Windows 2000 PC running SAS 8.2 with a Pentium 4 800 MHz chip and 255 MB of RAM. I wanted to summarize the file with an output of only 1 row for each combination of 5 character variables and use the SUM statistic on 2 numeric variables. In try #3 I used WAYS 5 and in try #4 I used TYPES with the same variables as in the class list, each separated by an "*".

```
Try #1   Proc Sort                    2.57 seconds
         Proc Summary w/BY           21.81 seconds
         Total                       24.38 seconds
Try #2   Proc Summary w/CLASS
           w/NWAY                    15.61 seconds
Try #3   Proc Summary w/CLASS
           w/WAYS                     5.65 seconds
Try #4   Proc Summary w/CLASS
           w/TYPES                    5.64 seconds
```

In each case, the resulting output SAS data set contains 22,229 rows. While the WAYS and TYPES methods only trimmed about 10 seconds off the time of the CLASS statement with the NWAY option, it's only about ⅓ of the time. If using the WAYS or the TYPES statement will reduce the time needed for the SUMMARY procedure to work its magic by about ⅔, then these statements are well worth using.

CONCLUSION

Reducing the size of your temporary and permanent SAS data sets can have a tremendous efficiency benefit on storage and processing. Any time you do not need all the character variables in a SAS data set, you may be able to collapse your file into a smaller SAS data set. The SUMMARY procedure is one procedure you should use often. SAS version 8 enhancements to the SUMMARY procedure provide new ways and means to more efficiently and effectively summarize your SAS data sets.

REFERENCES

SAS Online Documentation, version 8, Base SAS Software, SAS Procedures Guide, Procedures, The MEANS Procedure

TRADEMARK CITATION

CONTACT INFORMATION

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

Curtis A. Smith
Defense Contract Audit Agency
P.O. Box 20044
Fountain Valley, CA 92728-0044
Work Phone: 714-896-4277
Fax: 714-896-6915
Email: casmith@mindspring.com