

Proc Format, a Speedy Alternative to Sort/Merge

Jenine Eason, AUTOTRADER.COM, Atlanta, Georgia

ABSTRACT

Many users of SAS System software, especially those working with large data sets, are often confronted with several challenges. How can one reduce the data set size and reduce the amount of time required retrieving specific data? In this paper, we attempt to do both using a matching method utilizing Proc Format to replace the CPU heavy Sort/Merge. It is ideal for situations when a key from one file is needed to extract data from another file. It is more apparently useful when at least one of the files is quite large. This method has been proven time and again to decrease CPU by 70 - 80%. This paper is intended for the intermediate to advanced SAS user. It is effective on all platforms and uses Base SAS.

INTRODUCTION

Many users of SAS System software, especially those working with large data sets, are often confronted with several challenges. How can one reduce the data set size and reduce the amount of time required retrieving specific data? In this paper, we attempt to do both using a matching method utilizing Proc Format to replace the CPU heavy Sort/Merge. It is ideal for situations when a key from one file is needed to extract data from another file. It is more useful when at least one of the files is quite large. This method has been proven time and again to decrease CPU by 70% - 80%.

First, let's look at the traditional matching routine Sort/Merge.

TRADITIONAL SORT/MERGE

```
Proc Sort data=key (keep=key) nkey=key
  By key;
Run;

Proc Sort data=merge
  By key;
Run;

Data all merge; Merge merge;
  Key=key;
Run;
```

Sort/Merge is used here when key values from one file are needed to extract records from another file with the same key. For a clean merge, both data

sets have to be sorted. Note that both files have to be read twice. If either or both the files are large, CPU time can be considerable.

There are other basic concerns when running any Sort/Merge. Are there any duplicate records in either data set being used in the merge? Is the "king" merge logic being handled properly, so that required data from matching data sets will not be accidentally overlaid? These are no longer concerns with the Proc Format method.

SPEEDY METHOD USING PROC FORMAT

```
Proc Format;
  Define key = start
    Format = $20.
  Value
    1 = 'A'
    2 = 'B'
    3 = 'C'
  Run;

Proc Sort data=key nkey=key
  By start;
Run;

Proc Format outfmt=key;
  Define
    1 = 'A'
    2 = 'B'
    3 = 'C'
  Run;

Data all; Set merge;
  Outfmt=key;
Run;
```

First, a SAS data set needs to be created from the input file with the required format variables LABEL, START, and FMTNAME.

- START is set to the variable assigned as the key.
- FMTNAME becomes the format name to be referenced later. (cannot end in a numeric)
- LABEL becomes the symbol that the desired key values are associated with.

It is very important that the symbol assigned to LABEL will never have an occurrence in the key character string of the master file. Otherwise, an unwanted hit will occur. The asterisk (*) symbol works in most situations.

This pre-format data set needs to be sorted and any duplicates eliminated. Format will not allow duplicate values and will present a "this range is repeated" error. This also holds true for version 8 SAS.

SERENDIPITY

To actually create a working format, execute PROC FORMAT using the CNTLIN= option. It converts the data stored in the pre-format SAS data set to a SAS format.

Example of a format created using this method:

```
data keyfile;
  infile cards;
  input key $8.;
  cards;
Ohio
Georgia
Idaho
Virginia
;
run;
```

~~~~~ code creating format ~~~~~

```
proc format library=work fmtlib;
  select $key;
run;
```

```
+-----+
|FORMAT NAME:$KEY LENGTH: 1 NUMBER OF VALUES: 4|
|MIN LENGTH: 1 MAX LENGTH:40 DEFAULT LENGTH: 1|
+-----+
|START |END | LABEL (VER. 8.230APR2001:14:04:25)|
+-----+
|Georgia|Georgia|*|
|Idaho |Idaho |*|
|Ohio  |Ohio  |*|
|Virginia|Virginia|*|
+-----+
```

This format can be used anywhere in a program for selecting matches to the key. In essence, the assignment statement gives the value of LABEL (in this case \*) to a matching key. This in turn can be used for additional coding. In the above script, it is used to select records matching the key.

Using the Proc Format method, only one file is processed twice and only one variable is needed from it. Sorting of the master file is not required.

## ALTERNATIVE METHODS COMPARING EXACT RESULTS

For the non-believers, the results below show several different methods for comparison. The same input files were used in all examples. All tests were run using the same Unix Sun platform on Version 6 SAS. The key file had 730 observations. The larger file has over 1.5 million records. For these examples, the following CPU times were recorded.

Comparing the Proc Format method to the traditional Sort/Merge; the additional CPU time required to create the pre-format data set and create the format is more than made up by avoiding the pre-sort of the large file. The selection data step instead of the merge is 14.4% more efficient and the results are exactly the same.

|                    | CPU        | %time reduction |
|--------------------|------------|-----------------|
| Proc Format Method | 10.267sec  |                 |
| Sort/Merge         | 242.823sec | 76%             |
| Indexing           | 8.000sec   | 73%             |

## ADDITIONAL USES

### Merging using more than 1 Variable

Frequently, merging requires more than one variable. Two solutions would include concatenation or the use of more than one format. The examples below show the combination of key1 and key2 as the selection criteria.

### 2 Formats Method

```
data keya; set keya; keep=key1;
  start=key1;
  minmax=key1;
  label=key1;
run;

proc sort data=keya; sort=key1;
  by start;
run;

proc format cntlin=keya;
run;

data keyb; set keyb; keep=key2;
  start=key2;
  minmax=key2;
  label=key2;
run;

proc sort data=keyb; sort=key2;
  by start;
run;

proc format cntlin=keyb;
run;

data all; set keya keyb;
  if put(key1,$keya.) = "" and
  put(key2,$keyb.) = ""
run;
```

### Concatenation of 2 Variables Method

```

data key; set keyfile;
start = trim(key1) || trim(key2);
fmlname = $key;
label = "";
run;

proc sort data=key nodupkey;
by start;
run;

proc format cntlin=key;
run;

data all; set bigfile;
if put(trim(key1)||trim(key2), $key) = ""
run;

```

### Using a Numeric Variable as a Key

```

data key; set keyfile;
start = key;
fmlname = $key;
label = "";
run;

proc sort data=key nodupkey;
by start;
run;

proc format cntlin=key;
run;

data all; set bigfile;
if put(key, $key) = ""
run;

```

Many other solutions could be used to perform special processes only for those records matching the key or eliminating records that match the key.

- Make the LABEL more meaningful.

(In first data step)

```
label = 'Match';
```

- Creating more than one format using unique keys.

(In first data step)

```
if key = 'one' then label = 'one';
if key = 'two' then label = 'two';
```

(In last data step)

```
data testone testtwo; set bigfile;
if put(key, $key)='testone' then output
test one; else
if put(key, $key)='testtwo' then output
testtwo; else
delete;
run;

```

## LOGS FOR 3 VERSIONS FOR TIME SAVINGS COMPARISON

### Proc Format Method LOG (Total CPU time 10.267)

```

data key; set key(key=keep=key);
start = key;
fmlname = '$key';
label = "";
keep start fmlname label;
run;

```

NOTE: The data set WORK.KEY has 730 observations and 3 variables.

NOTE: DATA statement used:

|           |               |
|-----------|---------------|
| real time | 0.210 seconds |
| cpu time  | 0.058 seconds |

```

proc sort data=key nodupkey;
by fmlname start;
run;

```

NOTE: 0 observations with duplicate key values were deleted.

NOTE: The data set WORK.KEY has 730 observations and 3 variables.

NOTE: PROCEDURE SORT used:

|           |               |
|-----------|---------------|
| real time | 0.200 seconds |
| cpu time  | 0.043 seconds |

```
proc format cntlin=key;
```

NOTE: Format \$KEY has been output.  
run;

NOTE: PROCEDURE FORMAT used:

|           |               |
|-----------|---------------|
| real time | 0.080 seconds |
| cpu time  | 0.055 seconds |

```

data all; set bigfile;
if put(key, $key) = ""
run;

```

NOTE: The data set WORK.ALL has 13705 observations and 9 variables.

NOTE: DATA statement used:

|           |                |
|-----------|----------------|
| real time | 13.000 seconds |
| cpu time  | 10.053 seconds |

### Sort/Merge method LOG (Total CPU time 42.823)

```

proc sort data=key(keep=key) nodupkey;
by key;
run;

```

NOTE: 0 observations with duplicate key values were deleted.

NOTE: The data set WORK.KEY has 730 observations and 1 variables.

NOTE: PROCEDURE SORT used:

|           |               |
|-----------|---------------|
| real time | 0.500 seconds |
| cpu time  | 0.062 seconds |

```

proc sort data=bigfile;
by key;
run;

```

NOTE: The data set WORK.BIGFILE has 1548721 observations and 9 variables.

## SERENDIPITY

NOTE: PROCEDURE SORT used:

real time 40.480 seconds  
cpu time 31.013 seconds

```
data all; merge bigfile(in=a)
              key(in=b);
```

```
  by key;
  if a and b;
run;
```

NOTE: The data set WORK.ALL has 13705 observations and 9 variables.

NOTE: DATA statement used:

real time 15.890 seconds  
cpu time 11.748 seconds

## AUTHOR CONTACT INFORMATION

Jenine Eason  
Autotrader.com  
5775 Peachtree Dunwoody Road  
Atlanta, Georgia 30342

Phone: (404) 843-7199  
Email: [Jenine.eason@autotrader.com](mailto:Jenine.eason@autotrader.com)  
Home: [Easonconsulting@aol.com](mailto:Easonconsulting@aol.com)

## Indexing Method LOG (Total CPU time 38.000)

```
proc datasets lib=home;
```

-----Directory-----

```
Libref:      HOME
Engine:      V612
Physical Name: /home/jeason
File Name:   /home/jeason
Inode Number: 101001
Access Permission: rwxr-xr-x
Owner Name:  jeason
File Size (bytes): 8192
```

```
# Name      Memtype Indexes
```

```
-----
3 KEY      DATA
```

```
  modify key;
```

```
  index create key;
```

NOTE: Single index KEY defined.

```
run;
```

NOTE: PROCEDURE DATASETS used:

real time 3.550 seconds  
cpu time 0.185 seconds

```
data test1all; set bigfile;
              set home.key key=key;
  if _error_ = 1 then do;
    _error_ = 0;
  end;
  else output;
run;
```

NOTE: The data set WORK.TEST1ALL has 13705 observations and 9 variables.

NOTE: Compressing data set WORK.TEST1ALL decreased size by 0.00 percent.

Compressed is 2 pages; un-compressed would require 2 pages.

NOTE: DATA statement used:

real time 39.330 seconds  
cpu time 37.757 seconds

## REFERENCES

SAS Procedures Guide ver. 6. Page 282

Rick Aster and Rhena Seidman, Professional SAS  
Programming Secrets, Matching pp. 251-258