Speedy Data Retrieval and Display in a SAS/AF® System: A Case Study

Louise Weiler, Pacific Gas and Electric Company, San Francisco CA
Standish Sibley, Pacific Gas and Electric Company, San Francisco CA
Tim Berryhill, Programming Consultant, San Francisco CA

ABSTRACT

Developers of SAS/AF systems must strive to minimize how long users wait for each screen display. Short screen waits are particularly challenging when displaying data that must first be retrieved from a very large data base. This paper will show how, using features new to Version 6.07, a real life SAS/AF system went from impulse power to warp drive. This system was developed on MVS but the concepts and capabilities are applicable to all platforms.

Introduction

The electric utility industry plans for the future by studying the electric consumption patterns of its customers. To this end, special meters, called load profile recorders, are installed at the homes and businesses of a sample of the total customers of the utility. These special meters provide data that contains:

CHAN – Meter Identification variable

DATE – Date

KW1-KW48 – Electric consumption for each half hour in the day

This database was born in 1985, includes several thousand meters, spans several tapes and several naming conventions, and continues to grow every week. To make extraction from these voluminous files easier, the authors of this paper have written a SAS/AF system so that users can specify which CHAN/DATE combinations to extract. The system then constructs a batch job that performs the extraction. Users also have an option to ask whether the CHAN/DATE combinations they want to extract are actually available in the database before performing an extract. This option, called DATAMAP, provides speedy interactive displays. This paper will discuss how such speed is achieved via efficient techniques for storage, retrieval and display.

The DATAMAP Screen

Every day in a single year can be represented on a standard VDT terminal as a matrix where there are twelve rows representing the months and 31 columns representing the possible number of days in the month. A '+' sign in the matrix indicates that data is available on that day. A blank indicates that data is not available on that day. A '/' sign indicates that the day itself does not exist (like June 31). For a particular CHAN and year, the basic DATAMAP display looks like:

```
                                 1111111111222222222233
           12345678901234567890123456789012345678901
January:                    ++++++++++++++++++
February:  ++++++++++++++++++++++++++++++///
March:     +++++++++++++++++++++++++++++++
April:     ++++++++++++++++       ++++++++/
May:       +++++++++++++++++++++++++++++++
June:      +++++++++++++++++++++++++++++++/
July:      +++++++++++++++++++++++++++++++
August:    +++++++++++++++++++++++++++++++
September: +++++++++++++++++++++++++++++++/
October:   +++++++++++++++++++++++++++++++
November:  +++++++++++++++++++++++++++++++/
December:  +++++++++++++++++++++++++++++++
```

Not shown here is the title of the display, containing the year, CHAN and other customer characteristics. Also not shown is a legend describing how to interpret the display, and a footnote with instructions to the user on how to see the next or previous screen. In this particular display, data is not available on January 1-13 and April 17-22.

DATAMAP Storage

The first secret of speedy retrieval is having a file that already contains the information needed to appear on the screen. Whenever the CHAN/DATE observations described in the Introduction are updated in weekly batch production runs, a summary file is created that contains CHAN/YEAR observations. This is called the DATAMAP file. The DATAMAP observations contain:

CHAN – Meter identification variable

YEAR – Year

MONTH1-MONTH12

where each MONTHnn variable is a bit string containing indicators for the presence or absence of data on a given day of the month. Four bytes of data contains enough bits (32) to handle a bit for each day of the month. For example, if you want to see whether data is present on January 1 in a given year, you would check to see if the left-most bit has been set in MONTH1, i.e. the value of MONTH1 is equal to '1..... ....'B. January 2 would have a bit set in the next to left-most position, i.e. the value of MONTH1 would be equal to '1..... ....'B. January 31 would have the 31st from the left-most bit set, i.e. the value of MONTH1 would be equal to '..... ....1.'B. The right-most bit would never be set, because the 32nd of January (or any month) is non-existent. Note: Bit testing syntax allows blanks in the string to make it easier to count.

For ease in processing, however, the MONTHnn variables are stored as numeric variables of length 5 rather than as character variables of length 4. Since the SAS System stores all numeric variables in floating point, the extra byte is needed for an exponent (see SAS Institute Inc. (1990), *SAS Language Reference, Version 6, First Edition*, p.86-88).

This DATAMAP summary file is very small compared to the original CHAN/DATE data files and can be stored on disk, rather than tape. The DATAMAP summary file is indexed on a composite of YEAR and CHAN for speedy retrieval during an interactive session. The code to create the composite index is as follows:

```
PROC DATASETS LIBRARY=MASTER;
  MODIFY DATAMAP;
    INDEX CREATE YEARCHAN=(YEAR CHAN);
RUN;
```

The code that updates the DATAMAP file is in the Appendix.

DATAMAP Retrieval

When users specify which CHAN/YEAR combinations they want to view on screen, the selections are placed in a work file called TARGET. The TARGET file must now be matched to the DATAMAP file within an SCL SUBMIT block. Because the DATAMAP file is indexed on the composite of YEAR and CHAN, it is *not* necessary to use a MERGE statement to perform the matching. Instead, we can use the SET statement with a KEY= option. The SET statement with KEY= option enables you to access observations non-sequentially in a SAS data set according to a value. This is analogous to the POINT= option which allows you to access observations nonsequentially according to the observation number (see page 43 of SAS Technical Report P-222). The SET statement with KEY= option, available since Version 6.07, often makes for speedier matching than the MERGE statement (see Berryhill, Tim, *Fast Keyed Access to SAS Datasets*) .

For example, here is the slowest way to match the TARGET file to the DATAMAP file:

```
DATA DATAMAP;
  MERGE WORK.TARGET (IN=INTARGET)
        MASTER.DATAMAP (IN=INMAP);
  BY YEAR CHAN;
  IF INTARGET AND INMAP;
RUN;
```

The faster way (but not entirely accurate way) to match the TARGET file to the DATAMAP file is:

```
DATA DATAMAP;
  SET TARGET;
  SET MASTER.DATAMAP KEY=YEARCHAN;
RUN;
```

The above method could be inaccurate if a user requests a CHAN/YEAR combination that does not exist in the DATAMAP file. The variables read in via a SET statement with KEY= option are retained — they are not set to missing either at the top of the DATA step or before the keyed SET. If the key is not found, the variables retain their values from the most recent successful read (see Ben Conner, *Keys to the Data, Using the KEY= Option on the SET Statement*).

The fastest accurate way to match the TARGET file to the DATAMAP file is:

```
DATA DATAMAP;
  SET TARGET;
  SET MASTER.DATAMAP KEY=YEARCHAN;
  IF _IORC_ EQ 0 THEN
    OUTPUT;
  ELSE
    _ERROR_ = 0;
RUN;
```

The _IORC_ variable is an automatic variable that is set to a return code showing the status of the most recent I/O operation. When _IORC_ is zero, a key has been found. If a key is not found (this can happen legitimately in our system), an error condition is set in the variable _ERROR_. We restore _ERROR_ to zero to keep the system running smoothly. Also, note that you can only specify one index, single or composite, in the KEY= option. The statement

```
SET MASTER.DATAMAP KEY=YEAR CHAN;
```

would result in an error.

DATAMAP Display

Now that the DATAMAP system has retrieved the CHAN/YEAR observations that are to be displayed on screen, PROC FSEDIT in SAS/FSP® performs the display. The bit strings in variables MONTH1-MONTH12 must somehow appear as plus signs, blanks, and slashes. To accomplish this we create 12 new character variables of length 31, CM_1-CM_12. The FSEDIT screen contains 12 fields defined to display these 12 new variables.

```
                          11111111112222222222233
                 123456789012345678901234567890l
January:     _____________________________
February:    _____________________________
March:       _____________________________
April:       _____________________________
May:         _____________________________
June:        _____________________________
July:        _____________________________
August:      _____________________________
September:   _____________________________
October:     _____________________________
November:    _____________________________
December:    _____________________________
```

The new variables are created as follows. First, the bit string is converted to a string of '1's and '0's via the BINARY31 format. Then, using the TRANSLATE function, the '1's and '0's are converted to the '+'s and blanks and '/'s that will appear on the screen.

Within an SCL program, the following code is needed:

```
SUBMIT CONTINUE;
DATA SCRNFILE;
   SET DATAMAP;
   ARRAY BITMON (12) MONTH1-MONTH12;
   ARRAY CHARMON(12) $31 CM_1-CM_12;
   ARRAY MLENGTH(12) 2 MLEN1-MLEN12
          (31,28,31,30,31,30,31,31,30,31,30,31);
   IF MOD(YEAR,4) EQ 0 AND YEAR NE 2000 THEN
       MLEN2 = 29; * FEBRUARY IN A LEAP YEAR;
   DO M=1 TO 12;
      CHARMON(M) = SUBSTR(PUT BITMON(M),
           BINARY31.),1,MLENGTH(M));
      CHARMON(M) = TRANSLATE
           (CHARMON(M),"+ /*, '10 ');
   END;
RUN;
ENDSUBMIT;

CALL FSEDIT ('WORK.SCRNFILE',
             'DBN.SCREENS.DATAMAP.SCREEN',
             'BROWSE');
```

CONCLUSION

When you need to retrieve and display data quickly during a SAS/AF session, you should consider how you will store the data that will be retrieved. We found tremendous gains in speed by storing data in bit-string variables within files indexed by the variables that users specify during the SAS/AF session. The index allows you to retrieve the data using the SET statement with the KEY= option.

APPENDIX

Code to Update the Master DATAMAP File

The code to update the MONTHnn variable for a given DATE can be written in a few DATA steps.

First, at the CHAN/DATE level, create a variable BITVAL where a bit has been set in the appropriate position:

```
DATA DAYBIT;
   SET NEWDATA;
   LENGTH YEAR $4 MONTH 2 BITVAL 5;
   YEAR = PUT(YEAR(DATE),4.0);
   MONTH = MONTH(DATE);
   BITVAL = 2**(32-DAY(DATE));
RUN;
```

If, for example, the date January 25 were present, the statement that assigns the value of BITVAL would put a bit into the 8th position from the right which is the 25th position from the left.

2**(32-25) =
2**7=
128 which converts to '10000000'B

The next step is to summarize the variable BITVAL to the CHAN/YEAR/MONTH level.

```
PROC MEANS DATA=DAYBIT NOPRINT MISSING;
   CLASS CHAN YEAR MONTH;
   VAR BITVAL;
   OUTPUT OUT=MONTHBIT SUM= ;
RUN;
```

Then reshape the data to the CHAN/YEAR level.

```
DATA YEARBIT (KEEP=CHAN YEAR M_1-M_12);
   SET MONTHBIT (KEEP=CHAN YEAR MONTH BITVAL)
   BY CHAN YEAR;
   LENGTH M_1-M_12 5;
   RETAIN M_1-M_12;
   ARRAY MON (12) M_1-M_12;
   IF FIRST.YEAR THEN
      DO ERASE = 1 TO 12;
         MON(ERASE) = 0;
      END;
   MON(MONTH) = BITVAL;
   IF LAST.YEAR THEN
      OUTPUT;
RUN;
```

Finally, update the master file, making use of the BOR function (see SAS Technical Report P-222, page 49). The BOR function returns the bitwise logical OR of two arguments.

```
PROC SORT DATA = YEARBIT;
   BY YEAR CHAN;
RUN;

DATA MASTER.DATAMAP(KEEP=CHAN YEAR
                         MONTH1-MONTH12);
   MERGE MASTER.DATAMAP (IN=INOLD)
         YEARBIT (IN=INNEW);
   BY YEAR CHAN;
   ARRAY PERMMON (12) MONTH1-MONTH12;
   ARRAY TEMPMON (12) M_1-M_12;
   IF NOT INNEW THEN
      DO;
         * WRITE OUT OLD RECORD, NOT UPDATED;
         OUTPUT;
      END;
    ELSE IF NOT INOLD THEN
       DO I=1 TO 12;
          * WRITE OUT COMPLETELY NEW CHAN/YEAR;
          PERMMON(I) = TEMPMON(I)
          OUTPUT;
       END;
   ELSE
      DO I=1 TO 12;
         * NEW DATA UPDATES EXISTING DATA;
         PERMMON(I)=BOR(PERMMON(I),TEMPMON(I));
         OUTPUT;
      END;
RUN;
```

References

Berryhill, Tim, *Fast Keyed Access to SAS Datasets,* Western Users of the SAS System 1994 Proceedings

Conner, Ben, *Keys to the Data, Using the KEY= Option on the SET Statement,* Proceedings of the Nineteenth Annual SAS Users Group International Conference

SAS Institute Inc. (1990), *SAS Language Reference, Version 6, First Edition,* Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1991), *SAS Technical Report P-222, Changes and Enhancements to Base SAS Software, Release 6.07,* Cary, NC: SAS Institute Inc.

Acknowledgments

The authors of a batch precursor of DATAMAP include Ron Provine, Peter Hsieh and Peter Kretzman. Danine Cozzens provided production assistance. SAS-L provided inspiration.

If you have any questions, comments or suggestions, the authors can be contacted at:

Mail Pacific Gas and Electric Company
Mail Code B10B
P.O. Box 770000
San Francisco CA 94177

Internet LAW4@pge.com
SWS3@pge.com
TWB2@pge.com

SAS, SAS/AF, and SAS/FSP are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.