USING THE SAS® SYSTEM WITH ANOTHER PROGRAMMING LANGUAGE

Hank Leddon and Frank R. Fry
Board of Governors of the Federal Reserve System

Abstract

An occasion may arise where data reside in a data base that is not easily accessable to base SAS software. One solution to this difficulty is to create a file using a language that can readily access the data base and then use base SAS software to process the data in the file. This paper will discuss how this two step process can be accomplished. We illustrate with a program used at the Federal Reserve Board to generate an alphabetical listing of depository institutions. The program uses an in-house programming language to retrieve data from an IBM IMS data base and then uses Version 5 of the SAS language to process the data. It is run on an IBM OS system.

Introduction

At the Federal Reserve Board, data for every depository institution in the country are stored in an IBM IMS data base. This data base is comprised of roughly 80,000 records, with each record containing sixty-six fields of nonfinancial data such as institution identification number, name, city, Federal Reserve district, entity type, and membership status. Frequently, a request is made by a user to retrieve a subset of data from the data base and then generate a listing of the data, formatted to the user's specifications. Data are retrieved from the data base using SMART,[1/] an in-house programming language which was in use prior to the Board's installation of the SAS System. Written in FORTRAN, SMART enables a user to retrieve selected data for a selected date from the IMS data base, create a sequential file and then write the file out to a disk pack or tape in integer binary format. The continued use of SMART instead of SAS IMS is due to unfamiliarity with that software and the lack of time needed to become conversant with it.

After the selected data are written to a disk or tape, the data are then processed using base SAS software. SMART could be used to process the data after the file is created, but it is cumbersome to use and time consuming. Because it is easy to use and has excellent data handling features, SAS is the much preferred language.

Program Description

To illustrate our SMART/SAS combination, we will examine a program called BANKTYPE which produces an alphabetical listing of depository institutions grouped by entity type. Entity type is a two-digit code which identifies the type of institution, such as a commercial bank, credit union or savings and loan association.

Appendix A contains a copy of the program code, listing the appropriate SMART and SAS statements, as well as the JCL needed to run the program. Appendix B contains a copy of the SAS portion of the program after the program has been executed. Appendix C contains a copy of the first few lines of the output listing.

In the first part of the program, denoted by STEP1 in the JCL (Appendix A), SMART is used to retrieve from the IMS data base selected data for four different entity types which were active on December 31, 1985. The data are then placed in a sequential file, created by the SMART command SUBFILE: (USER1,FD,INTEGER), which is then written to a disk pack. This portion of the program took 81.40 CPU seconds or about 1 1/2 minutes to execute.

In STEP2 of the program, base SAS software is used to read the newly created file and process the data. To read an external or non-SAS file, the INFILE and INPUT statements must be used. The INFILE statement identifies the external file to be read with the INPUT statement. The INPUT statement describes the order and format of the input record and assigns values to the corresponding SAS variables used in the data step. Depending on the type of data to be described, column, list, formatted or named input may be used in the INPUT statement. Standard form data, stored with one digit or character per byte, can be read with all four input types. Nonstandard form data, such as binary data, can be read only with formatted input.

In our example, the INFILE statement identifies the external file to be read as FT18F001 (Appendix B). Our data being nonstandard, we used grouped format lists in the INPUT statement to format the data. The first format list names the variables to be read, the second list assigns the corresponding informats. The $CHARw.d informat is used to input character data, since our character strings contain trailing blanks, and the IBw.d informat, indicating integer binary data, is used to input numeric data. The n modifier specifies that the informat is to be repeated n times, or nine times in this program.

After all the data are input into SAS data set BNK-INFO, consisting of 18,784 observations and thirteen variables, PROC SORT is used to sort the data. The data are sorted first by entity type, then within entity type, by institution name, then by city and finally by state abbreviation.

After the data are sorted, PROC PRINT is used to print and count the number of observations by entity type. The N option is used to count the number of observations in each entity type group and the NOOBS option is used to suppress the printing of observation numbers. The SPLIT= option and the LABEL statement are used to assign descriptive labels to variable names. The VAR statement names the variables to be printed and the BY and PAGEBY statements instruct SAS to print the data by entity type and to start a new page when the entity type

changes. The TITLE statements provide descriptive report titles.

The desired output listing was generated in only 13.72 CPU seconds using the SAS System. By comparison, it would take 17.97 CPU seconds to generate the same listing using SMART. Although that is only about five seconds more in execution time, more programming time would be required since title, column heading, and detail lines in SMART, like FORTRAN, must be formatted. Thus, this two step process of retrieving data with SMART and then processing it with base SAS software enables us to easily and quickly fulfill a user's data request.

The process of using the SAS language with another language can be modified to fit different applications. At the Board, for example, the SMART/SAS combination has been incorporated into a user friendly CLIST, from which the program BANKTYPE was created and submitted. The CLIST enables any user to retrieve, sort and print data for up to eight different entity types and to generate up to nine copies of the output listing. The CLIST can be executed any number of times.

Conclusion

We have illustrated the ease with which the SAS System can be used with another programming language and the possible savings in programming and execution time. Used in conjunction with another programming language when neccessary or desired, the SAS System enables a user to easily and efficiently process data regardless of the format of the data base in which the data reside.

Footnote

1/ When executed, SMART calls another routine which then executes the DL1 calls to the IMS data base.

Acknowledgment

SAS® is the registered trademark of SAS Institute Inc., Cary, North Carolina, USA.

This paper does not necessarily reflect the views of the Board of Governors of the Federal Reserve System; therefore, no official endorsement should be inferred.

References

SAS User's Guide: Basics, Version 5 Edition, SAS Institute Inc., Cary, NC 1985.

Author Contacts

Hank Leddon
Frank R. Fry
Micro Statistics Section
Division of Research and Statistics
Board of Governors of the Federal Reserve System
Washington, D.C. 20551
(202) 452-2950

Appendix A - JCL and Program Statements

```
//BANKTYPE JOB (D585,4411,DK),'QLEDDON ABINS',REGION=2096K,
//         MSGCLASS=A,NOTIFY=M1MAL00,TIME=(1,59)
/*JOBPARM LINES=30,COPIES=1
//STEP1  EXEC SMART
//FT18F001 DD DSN=&BANKS,UNIT=SYSDA,SPACE=(CYL,(5,1),RLSE),
//            DCB=(RECFM=VBS,LRECL=100,BLKSIZE=19069),DISP=(NEW,PASS)
//SYSIN DD *
*************************************************************
* SMART IS USED TO RETRIEVE STRUCTURE INFO FOR THE SELECTED ENTITY *
* TYPES & WRITE THE DATA TO A TEMP DS. SAS IS THEN USED TO SORT &  *
* PRINT THE DATA ALPHABETICALLY BY ENTITY TYPE.                    *
*************************************************************
SETIN: 851231
NULL: ZEROES
SELECT: (BANK9999.EQ.851231.AND.BANK9550.LT.02)
   IF (BANK9331.EQ.01) GO TO PX
   IF (BANK9331.EQ.10) GO TO PX
   IF (BANK9331.EQ.16) GO TO PX
   IF (BANK9331.EQ.31) ACCEPT
PX:
ITEMS: &&
   BANK9000 BANK9010 BANK9130 BANK9200 BANK9347 BANK9350 BANK9421 &&
   BANK9422 BANK9423 BANK9424 BANK9550 BANK9331 BANK9420 (IMS,851231)
COUNT:
SUBFILE: (USER1,FD,INTEGER)
PROCESS: (CONTINUE)
DONE:
/*
//STEP2   EXEC SAS
//FT18F001 DD  DSN=&BANKS,DISP=(OLD,DELETE)
//SYSIN DD *

DATA BNK_INFO;
   INFILE FT18F001;
   INPUT
      (DSB NAME CITY ST HI_HC DIR_HC CHR_TYPE MEM_ST
       BNK_TYPE INS_ST OFF_ST ENT_TYPE CHR)
      (IB4. $CHAR28. $CHAR20. $CHAR4. 9*IB4.);

PROC SORT DATA=BNK_INFO;
   BY ENT_TYPE NAME CITY ST;

PROC PRINT DATA=BNK_INFO N NOOBS SPLIT='*';
   BY ENT_TYPE;
   PAGEBY ENT_TYPE;
   VAR
      DSB NAME CITY ST HI_HC DIR_HC CHR_TYPE MEM_ST
      BNK_TYPE INS_ST OFF_ST CHR;
   LABEL
      ENT_TYPE='ENTITY TYPE'    HI_HC='HIGH*HC'
      DIR_HC='DIR*HC'           CHR_TYPE='CHR/LIC*TYPE'
      MEM_ST='MEM*STAT'         BNK_TYPE='BNK*TYPE'
      INS_ST='INSUR*STAT'       OFF_ST='OFFICE*STAT'
      CHR='CHR*CLASS';
   TITLE1 'ALPHABETIC LISTING OF INSTITUTIONS BY ENTITY TYPE';
   TITLE2 '(LISTING AS OF 851231)';
```

Appendix B - SAS Code and SAS Notes

```
1          SAS(R) LOG   OS SAS 5.08           MVS/XA JOB BANKTYPE STEP STEP2    PROC SAS

NOTE: COPYRIGHT (C) 1984 SAS INSTITUTE INC., CARY, N.C.  27511, U.S.A.
NOTE: THE JOB BANKTYPE HAS BEEN RUN UNDER RELEASE 5.08 OF SAS AT FEDERAL RESERVE BOARD OF
      GOVERNORS (0146100l).
NOTE: CPUID    VERSION = 03  SERIAL = 010826  MODEL = 5890 .
      CPUID    VERSION = 03  SERIAL = 210826  MODEL = 5890 .

NOTE: NO OPTIONS SPECIFIED.

1
2          DATA BNK_INFO;
3             INFILE FT18F001;
4             INPUT
5                (DSB NAME CITY ST HI_HC DIR_HC CHR_TYPE MEM_ST
6                 BNK_TYPE INS_ST OFF_ST ENT_TYPE CHR)
7                (IB4. $CHAR28. $CHAR20. $CHAR4. 9*IB4.);
8


NOTE: INFILE FT18F001 IS:
      DSNAME=SYS87016.T074133.RA000.BANKTYPE.BANKS,
      UNIT=DISK,VOL=SER=WORK05,DISP=OLD,
      DCB=(BLKSIZE=19069,LRECL=100,RECFM=VBS)

NOTE: 18784 LINES WERE READ FROM INFILE FT18F001.
      THE MINIMUM LINE LENGTH IS 92.
      THE MAXIMUM LINE LENGTH IS 92.
NOTE: DATA SET WORK.BNK_INFO HAS 18784 OBSERVATIONS AND 13 VARIABLES. 344 OBS/TRK.

9          PROC SORT DATA=BNK_INFO;
10            BY ENT_TYPE NAME CITY ST;
11

NOTE: DATA SET WORK.BNK_INFO HAS 18784 OBSERVATIONS AND 13 VARIABLES. 344 OBS/TRK.

12         PROC PRINT DATA=BNK_INFO N NOOBS SPLIT='*';
13            BY ENT_TYPE;
14            PAGEBY ENT_TYPE;
15            VAR
16               DSB NAME CITY ST HI_HC DIR_HC CHR_TYPE MEM_ST
17               BNK_TYPE INS_ST OFF_ST CHR;
18            LABEL
19               ENT_TYPE='ENTITY TYPE'    HI_HC='HIGH*HC'
20               DIR_HC='DIR*HC'           CHR_TYPE='CHR/LIC*TYPE'
21               MEM_ST='MEM*STAT'         BNK_TYPE='BNK*TYPE'
22               INS_ST='INSUR*STAT'       OFF_ST='OFFICE*STAT'
23               CHR='CHR*CLASS';
24            TITLE1 'ALPHABETIC LISTING OF INSTITUTIONS BY ENTITY TYPE';
25            TITLE2 '(LISTING AS OF 851231)';
26

NOTE: THE PROCEDURE PRINT PRINTED PAGES 1 TO 363.

NOTE: SAS INSTITUTE INC.
      SAS CIRCLE
      PO BOX 8000
      CARY, N.C.  27511-8000
```

Appendix C - Example of Output

ALPHABETIC LISTING OF INSTITUTIONS BY ENTITY TYPE 7:44 FRIDAY, JANUARY 16, 1987 1
(LISTING AS OF 851231)

ENTITY TYPE =1

DSB	NAME	CITY	ST	HIGH HC	DIR HC	CHR/LIC TYPE	MEM STAT	BNK TYPE	INSUR STAT	OFFICE STAT	CHR CLASS
7550015	ABBOTSFORD ST BK	ABBOTSFORD	WI	0	0	3	0	1	1	0	21
10200010	ABILENE FIRST NB	ABILENE	KS	2334	2334	1	1	1	1	0	3
7170010	ABINGDON B&TC	ABINGDON	IL	7432	7432	3	0	1	1	0	21
1250010	ABINGTON NB	ABINGTON	MA	0	0	1	1	1	1	0	3
11481961	ABRAMS CENTRE NB	DALLAS	TX	0	0	1	1	1	1	0	3
6221385	ACADIAN BK	THIBODAUX	LA	6533	6533	3	0	1	1	0	21
6220627	ACADIANA NB	LAFAYETTE	LA	0	0	1	1	1	1	0	3
7190010	ACKLEY ST BK	ACKLEY	IA	920	920	3	0	1	1	0	21
9270010	ADA NB	ADA	MN	3484	3484	1	1	1	1	0	3
10400025	ADAIR ST BK	ADAIR	OK	0	0	3	0	1	1	0	21
8210010	ADAIRVILLE BKG CO	ADAIRVILLE	KY	4612	4612	3	0	1	1	0	21
10312000	ADAMS COUNTY BK	KENESAW	NE	1726	1726	3	0	1	1	0	21
5421685	ADAMS COUNTY NB	CUMBERLAND TWP	PA	6546	6546	1	1	1	1	0	3
10310010	ADAMS ST BK	ADAMS	NE	0	0	3	0	1	1	0	21
11480047	ADDISON NB	ADDISON	TX	0	0	1	1	1	1	0	3
7170020	ADDISON ST BK	ADDISON	IL	0	0	3	0	1	1	0	21
6130040	ADEL BKG CO	ADEL	GA	8792	8792	3	0	1	1	0	21
7550030	ADELL ST BK	ADELL	WI	0	0	3	0	1	1	0	21
2366910	ADIRONDACK TC	SARATOGA SPRINGS	NY	350	350	3	1	1	1	0	13
10201648	ADMIRE B&TC	EMPORIA	KS	2706	2706	3	0	1	1	0	21
10290010	ADRIAN BK	ADRIAN	MO	0	0	3	0	1	1	0	21
7260020	ADRIAN ST BK	ADRIAN	MI	0	0	3	0	1	1	0	21
9270040	ADRIAN ST BK	ADRIAN	MN	3609	3609	3	0	1	1	0	21
7171470	AETNA BK	CHICAGO	IL	3792	3792	3	0	1	1	0	21