

Processing VSAM Files Using SAS

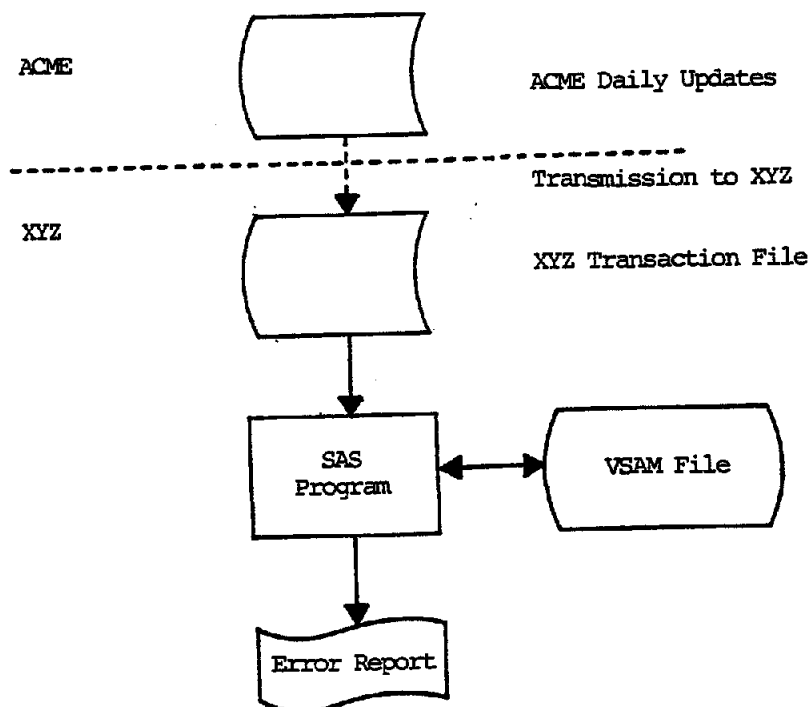
Jim Nipper - Electronic Data Systems Corporation

Abstract

This paper will demonstrate the ease and flexibility which SAS provides in processing VSAM files. In particular, it will illustrate how a batch program written in SAS applies transaction updates to a Key Sequenced Data Set (KSDS) file. The transaction record types include adds, changes and deletes. In addition, error reporting and update verification is also illustrated.

Problem

The ACME Company has a customer file which is stored in a KSDS VSAM file and is updated daily. XYZ Incorporated has purchased this file, which it houses independently in its own data center on a KSDS VSAM file. Each night, ACME transmits its daily updates to XYZ. In addition, ACME sends a summary record describing the adds, changes and deletes that occurred during the day and a total count of records on the file at day's end. XYZ's requirement is to apply the transactions to its copy of the VSAM file, report any errors encountered during the update process, and verify the total number of records on its file after applying updates.



The customer file at ACME is a fixed length file with a total record layout of 350 bytes. The key is 13 bytes long, beginning in position 1. The attributes and layout of the customer file at XYZ are the same. The daily updates at ACME are stored in a file which has a length of 351 bytes. The extra byte of data is an update code attached to the front of each record signifying that the record is an add (A), change (C) or delete (D). The last record on the transaction file is a totals record (T), indicating the total number of adds, changes and deletes. The totals record also indicates the number of records which should be present on the customer file after the transactions have been applied.

```

                                     3
                                1  5
                           5  0  0
-----|-----|-----
0000123456789
      key

```

```

                                     3
                                1  5
                           5  0  0
-----|-----|-----|
a0000123456789
      key

```

```

                                     3
                                1  5
                           5  0  0
-----|-----|-----
0000123456789
      key

```

About the Program

The first statement in the program is an OPTIONS statement. This will override any VSAM options which may have been set during installation of the SAS System.

```
OPTIONS VSAMUPDATE VSAMREAD;
```

Next, we will read records from the transaction file (sent from ACME) and check the indicator to determine what the record layout should be. Two SAS data sets will be created. Any errors encountered in the add, change or delete logic will generate a record which will be written to TRANERR. The totals record (IND = 'T') will be output to TRANCHEK.

```
DATA TRANERR TRANCHEK;
  LENGTH    CODE    2.
    ACT      4.
    CCT      4.
    DCT      4.
    TCT      4.
    ERRORSW  2.
    EFLAG           $1
    EKEY          $13.
    MSG           $35.;
  MSG = 'NO ERRORS';
  INFILE TRAN;
  INPUT      @1  IND      $CHAR1. @;
  IF IND IN ('A','C','D') THEN DO;
    INPUT@2  CUSTKEY  $CHAR13.
           @15 REC1   $CHAR200.
           @215 REC2  $CHAR137.;
  END;
  IF IND = 'T' THEN DO;
    INPUT@2  ACOUNT  7.
           @9  CCOUNT  7.
           @16 DCOUNT 7.
           @23 TCOUNT 7.;
  END;
  KEYVAR = CUSTKEY;
```

Evaluate the indicator (IND) to determine whether the transaction is an add, change, delete or totals record. If the indicator is an 'A', write the record to the XYZ VSAM file. We will use the FEEDBACK = option to check for VSAM logical errors, and if any are detected we will write the offending record to the error file TRANERR and continue processing with the next record. To

continue processing, we must reset the FEEDBACK variable to 0 and the automatic SAS variable `_ERROR_` to 0.

RULE: To add records to a VSAM file in SAS programs, you must have a pair of INFILE and FILE statements for the file. Specify the VSAM type option in both the INFILE and the FILE statements. Specify all other options in the INFILE statement, which must precede the FILE statement. When adding records to an existing KSDS, you do not have to include an INPUT statement with the associated INFILE statement. An INPUT statement is unnecessary in this case because you do not have to read a record to add a new record.

```
IF IND='A' THEN DO;
  INFILE XYZFILE VSAM FEEDBACK=FDBK;
  FILE XYZFILE VSAM;
  PUT  @1  CUSTKEY   $CHAR13.
        @14 REC1     $CHAR200.
        @214 REC2    $CHAR137.;
  IF FDBK^=0 THEN DO;
    ERRORSW + 1;
    CODE=FDBK;
    EKEY=CUSTKEY;
    MSG='ERROR ATTEMPTING TO ADD RECORD';
    FDBK=0;
    _ERROR_=0;
    OUTPUT TRANERR;
    RETURN;
  END;
  ACT + 1;
END;
```

If the indicator is a 'C', over-write the record on the XYZ VSAM file with the new data. Once again we will use the FEEDBACK= option to check for VSAM logical errors, and if any are detected we will write the offending record to the error file TRANERR and continue processing with the next record. Note the INFILE options KEYPOS and KEY. KEY indicates that keyed direct access is to be used to retrieve the record, and the variable KEYVAR (initialized previously in the data step with the variable CUSTKEY) will provide the key of the record to be read. KEYPOS specifies a numeric SAS variable that the SAS System sets to the position of the key field.

RULE: To update records in a VSAM file in SAS programs, you must have a pair of INFILE and FILE statements for the file. Specify the VSAM type option in both the INFILE and the FILE statements. Specify all other options in the INFILE statement, which must precede the FILE statement. When

updating KSDS records, you must first retrieve a record you want to update; that is, the record must be read with an INPUT statement before it can be updated.

```
IF IND = 'C' THEN DO;
  INFILE XYZFILE VSAM FEEDBACK = FDBK KEYPOS = POSKEY
    KEY = KEYVAR;
  INPUT;
  IF FDBK^ = 0 THEN DO;
    ERRORSW + 1;
    CODE = FDBK;
    EKEY = CUSTKEY;
    MSG = 'ERROR ATTEMPTING TO CHANGE RECORD';
    FDBK = 0;
    _ERROR_ = 0;
    OUTPUT TRANERR;
    RETURN;
  END;
  FILE XYZFILE VSAM;
  PUT  @POSKEY  KEYVAR      $CHAR13.
      @14      REC1       $CHAR200.
      @214     REC2       $CHAR137.;
  CCT + 1;
END;
```

If the indicator is a 'D', delete the record on the XYZ VSAM file. If an error is detected we will write the offending record to an error file and continue processing. Note the INFILE option ERASE. ERASE defines a numeric SAS variable that you must set when you want to erase (delete) a VSAM record. The record is erased when you set the ERASE= variable to 1 before execution of a PUT statment for the output data set. After the record is erased, the ERASE= variable is automatically reset to 0. You must reset the ERASE= variable to 1 in order to erase another record.

RULE: To delete records from a VSAM file in SAS programs, you must have a pair of INFILE and FILE statements for the file. Specify the VSAM type option in both the INFILE and the FILE statements. Specify all other options in the INFILE statement, which must precede the FILE statement. When deleting KSDS records, you must first retrieve the record you want to delete; that is, the record must be read with an INPUT statement before it can be deleted.

```
IF IND = 'D' THEN DO;
  INFILE XYZFILE VSAM FEEDBACK = FDBK ERASE = ERASEVAR
    KEY = KEVAR;
```

```

INPUT;
IF FDBK^=0 THEN DO;
    ERRORSW + 1;
    CODE = FDBK;
    EKEY = CUSTKEY;
    MSG = 'ERROR ATTEMPTING TO DELETE RECORD';
    FDBK = 0;
    _ERROR_ = 0;
    OUTPUT TRANERR;
    RETURN;
END;
FILE XYZFILE VSAM;
ERASEVAR = 1;
PUT _INFILE_;
DCT + 1;
END;

```

If the indicator is a 'T' (totals record), output the record to the data set TRANCHEK.

```

IF IND = 'T' THEN DO;
    OUTPUT TRANCHEK;
END;

```

The next step will verify the updates. Set TRANCHEK into itself. This will give us access to the counts for the totals record. Next, we will open the XYZ VSAM file using the RECORDS= option to obtain the number of logical records. Once obtained, compare the counts generated by the program with the counts supplied by the transaction file totals record. If any of the counts are in error, read the TRANERR data set to obtain the records which may have caused the counts to be in error. Output any records read and set the abend switch (ABND SW) to 'Y'. If all counts are verified, output the totals record.

```

DATA TRANCHEK;
    LENGTH    ABND SW    $1.;
    SET TRANCHEK;
        INFILE XYZFILE VSAM OBS = 1 RECORDS = CNT;
        INPUT;
        TCT = CNT;
        EFLAG = 'N';
        ABND SW = 'N';
        IF ACT^ = ACOUNT THEN EFLAG = 'Y';
        IF CCT^ = CCOUNT THEN EFLAG = 'Y';
        IF DCT^ = DCOUNT THEN EFLAG = 'Y';

```

```

IF TCT^ = TCOUNT THEN EFLAG = 'Y';
IF EFLAG = 'Y' THEN DO OBSNUM = 1 TO LAST;
    SET TRANERR POINT = OBSNUM NOBS = LAST;
    ABNDSW = 'Y';
    OUTPUT;
    END;
ELSE DO;
    OUTPUT;
    END;

```

Print the error report.

```

PROC PRINT DATA = TRANCHEK NOOBS UNIFORM SPLIT = '*';
    VAR EKEY CODE MSG;
    LABEL      EKEY = 'RECORD*KEY'
              CODE = 'FEEDBACK*CODE'
              MSG  = 'MESSAGE';
    TITLE1 'LIST OF ERRORS ENCOUNTERED';
    TITLE2 'DURING VSAM UPDATE PROCESS';

```

The last step is used to notify the operator or programmer that errors were encountered during the update process. If the abend switch is set to 'Y' (see previous data step), SAS will issue a return code of 16 on the job step.

```

DATA TRANCHEK;
    SET TRANCHEK;
    IF ABNDSW = 'Y' THEN ABORT RETURN 16;

```

JCL and Expanded Program

```

//JOBNAME JOB (ABCDEF,1234),'UPDATE CUST FILE',
//      CLASS=X,MSGCLASS=A,NOTIFY=USERID
//JS010 EXEC SAS
//TRAN DD DSN=TRAN.FILE.NAME,
//      DISP=(SHR,KEEP,KEEP)
//XYZFILE DD DSN=XYZ.VSAM.FILE,
//      DISP=(OLD,KEEP,KEEP),
//      AMP='BUFND=??,BUFNI=??'
//FT12F001 DD DSN=ERROR.RPT.FILE,
//      DISP=(NEW,CATLG,DELETE),
//      UNIT=PERMDA,
//      SPACE=(CYL,(5,1),RLSE),
//      DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)

```

//SYSIN DD *

OPTIONS VSAMUPDATE VSAMREAD;

DATA TRANERR TRANCHEK;

LENGTH	CODE	2.
--------	------	----

ACT	4.
-----	----

CCT	4.
-----	----

DCT	4.
-----	----

TCT	4.
-----	----

ERRORSW	2.
---------	----

EFLAG	\$1
-------	-----

EKEY	\$13.
------	-------

MSG	\$35.;
-----	--------

MSG = 'NO ERRORS';

INFILE TRAN;

INPUT @1 IND \$CHAR1. @;

IF IND IN ('A','C','D') THEN DO;

INPUT@2	CUSTKEY	\$CHAR13.
---------	---------	-----------

@15	REC1	\$CHAR200.
-----	------	------------

@215	REC2	\$CHAR137.;
------	------	-------------

END;

IF IND = 'T' THEN DO;

INPUT@2	ACOUNT	7.
---------	--------	----

@9	CCOUNT	7.
----	--------	----

@16	DCOUNT	7.
-----	--------	----

@23	TCOUNT	7.;
-----	--------	-----

END;

KEYVAR = CUSTKEY;

IF IND = 'A' THEN DO;

INFILE XYZFILE VSAM FEEDBACK = FDBK;

FILE XYZFILE VSAM;

PUT @1	CUSTKEY	\$CHAR13.
--------	---------	-----------

@14	REC1	\$CHAR200.
-----	------	------------

@214	REC2	\$CHAR137.;
------	------	-------------

IF FDBK^=0 THEN DO;

ERRORSW + 1;

CODE = FDBK;

EKEY = CUSTKEY;

MSG = 'ERROR ATTEMPTING TO ADD RECORD';

FDBK = 0;

ERROR = 0;

OUTPUT TRANERR;

RETURN;

```

        END;
    ACT + 1;
    END;

IF IND = 'C' THEN DO;
    INFILE XYZFILE VSAM FEEDBACK = FDBK KEYPOS = POSKEY
        KEY = KEYVAR;
    INPUT;
    IF FDBK^ = 0 THEN DO;
        ERRORSW + 1;
        CODE = FDBK;
        EKEY = CUSTKEY;
        MSG = 'ERROR ATTEMPTING TO CHANGE RECORD';
        FDBK = 0;
        _ERROR_ = 0;
        OUTPUT TRANERR;
        RETURN;
    END;
    FILE XYZFILE VSAM;
    PUT  @POSKEY  KEYVAR      $CHAR13.
        @14      REC1       $CHAR200.
        @214     REC2       $CHAR137.;
    CCT + 1;
    END;

IF IND = 'D' THEN DO;
    INFILE XYZFILE VSAM FEEDBACK = FDBK ERASE = ERASEVAR
        KEY = KEVAR;
    INPUT;
    IF FDBK^ = 0 THEN DO;
        ERRORSW + 1;
        CODE = FDBK;
        EKEY = CUSTKEY;
        MSG = 'ERROR ATTEMPTING TO DELETE RECORD';
        FDBK = 0;
        _ERROR_ = 0;
        OUTPUT TRANERR;
        RETURN;
    END;
    FILE XYZFILE VSAM;
    ERASEVAR = 1;
    PUT _INFILE_;
    DCT + 1;
    END;

```

```

IF IND = 'T' THEN DO;
    OUTPUT TRANCHEK;
END;

```

```

DATA TRANCHEK;
    LENGTH    ABND SW    $1.;
    SET TRANCHEK;
        INFILE XYZFILE VSAM OBS = 1 RECORDS = CNT;
        INPUT;
        TCT = CNT;
        EFLAG = 'N';
        ABND SW = 'N';
        IF ACT^ = ACOUNT THEN EFLAG = 'Y';
        IF CCT^ = CCOUNT THEN EFLAG = 'Y';
        IF DCT^ = DCOUNT THEN EFLAG = 'Y';
        IF TCT^ = TCOUNT THEN EFLAG = 'Y';
        IF EFLAG = 'Y' THEN DO OBSNUM = 1 TO LAST;
            SET TRANERR POINT = OBSNUM NOBS = LAST;
            ABND SW = 'Y';
            OUTPUT;
            END;
        ELSE DO;
            OUTPUT;
            END;

```

```

PROC PRINT DATA = TRANCHEK NOOBS UNIFORM SPLIT = '*';
    VAR EKEY CODE MSG;
    LABEL      EKEY = 'RECORD*KEY'
            CODE = 'FEEDBACK*CODE'
            MSG = 'MESSAGE';
    TITLE1 'LIST OF ERRORS ENCOUNTERED';
    TITLE2 'DURING VSAM UPDATE PROCESS';

```

```

DATA TRANCHEK;
    SET TRANCHEK;
    IF ABND SW = 'Y' THEN ABORT RETURN 16;

```

Conclusion

Despite the proliferation in recent years of distributed processing, relational databases, graphical user interfaces (GUI's), etc., mainframe based systems

remain a fact-of-life for much of corporate and academic America. With many of these mainframe based systems dependent on "older" technologies (such as VSAM), the SAS System offers a quick and viable alternative to languages such as COBOL or PL/I when processing VSAM files in a batch environment. This paper has attempted to demonstrate the SAS Systems' update capability with respect to KSDS VSAM files in a batch environment. Any questions or comments are welcomed and may be directed to the author at:

Jim Nipper
Electronic Data Systems Corporation
5400 Legacy Dr.
C3-3D-35
Plano, TX 75023
(214) 604-6000.

For more information about processing VSAM files using SAS (including ESDS, KSDS and RRDS), consult the SAS Guide to VSAM Processing, Version 6 Edition.

SAS and the SAS System are registered trademarks or trademarks of the SAS Institute Inc. in the USA and other countries.

Other brand and product names are registered trademarks or trademarks of their respective companies.