Paper 45-25

Building an Audit and Tracking System Using SAS/AF® and SCL

Hung X Phan, U.S. Census Bureau

ABSTRACT

This paper describes how to build an audit and tracking system using SAS/AF and SCL. There will be a master data set which contains the original information and several transaction data sets which contain the new information.

The audit system will take information from the transaction data sets and apply changes to the master data set. An audit data set will be used to keep track of all changes.

The tracking system will display the master data set in the first data table on the top half of the screen and the audit data set in the second data table on the bottom half.

To track a record the user clicks on a row within a given data table. The selected record is then highlighted and displayed in the top row of the two data tables where the user can easily compare and/or view changes made to the record.

Buttons on a toolbar, FIND, SORT, HIDE/UNHIDE, TOP, BOTTOM, RESET, are utilized to help the user navigate throughout the system. This system is currently running in a multi-user windows environment on a SASV8/Windows95 Novell server at the U.S. Census Bureau.

INTRODUCTION

The Alphabetical Index of Industries and Occupations was developed primarily for use in classifying a respondent's industry (employer's kind of business) and occupation (employee's kind of work) as reported in the 1990 Census and other surveys conducted by the U.S. Census Bureau.

There are about 33,000 records in the occupation master file and 22,000 records in the industry master file. In preparation for the 2000 decennial census, our task is to bring and keep these files up-to-date for press releases and publications.

There is a staff of survey analysts whose job is to classify occupations and industries by assigning each occupation and each industry reported a numeric code. Each analyst is responsible for a set of transaction data sets. The analysts may add new records, modify existing records, or delete outdated records.

We built the audit system to store changes made to the master records and the tracking system to retrieve a history of changes to these records. As a demonstration, in this paper we will only work with the industry master file.

BUILDING THE DATA ACCESS INTERFACE

The Audit and Tracking System is a windows interactive application where the user is presented with a Graphical User Interface (GUI).

We build the Data Access screen as shown in Figure 1, for the Audit and Tracking system by using the BUILD procedure that SAS/AF software provides. We use Screen Control Language (SCL), a powerful programming language that SAS/AF software also provides to manage and control activities on the screens.

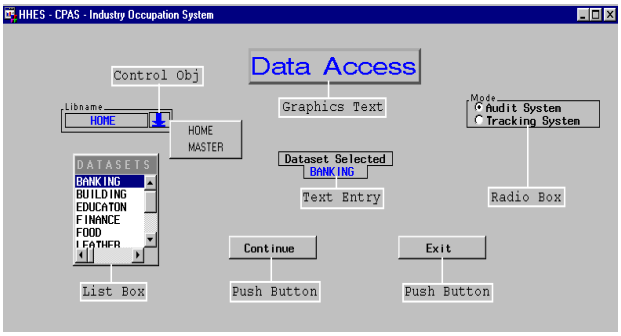

Figure 1: The Data Access Interface

First, we create a blank frame called DTACCESS by submitting the following SCL code.

```
libname mylib 'c:\myapp';
proc build c=mylib.mycat.DTACCESS.frame;
run;
```

Second, we create a list box called DATASETS as shown in Figure 2, to display the available data sets within the selected library. The user uses the available transaction data sets to apply changes to the master data sets. We place the list box on the left side of the screen as seen in Figure 1.

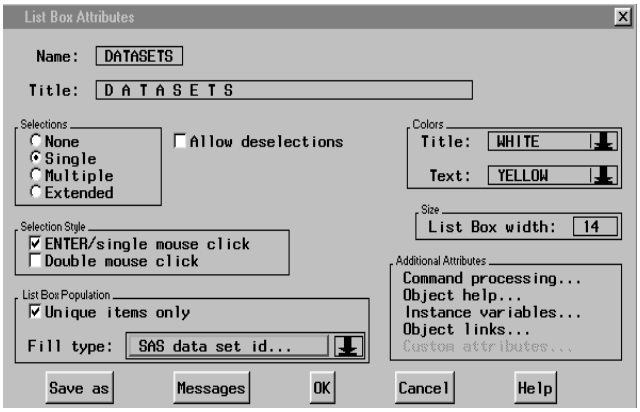

Figure 2: List Box Attributes Windows

Next, we create a control object LIB_SEL as shown in Figure 3 below, which contains a down arrow within a box and, when selected, displays the available library references, i.e. HOME and MASTER. If the user clicks on the HOME library reference, it displays the available transaction data sets whereas if the user clicks on the MASTER library reference, it displays the available master data sets and the audit data set. We place the control object above the data sets list box as seen in Figure 1.

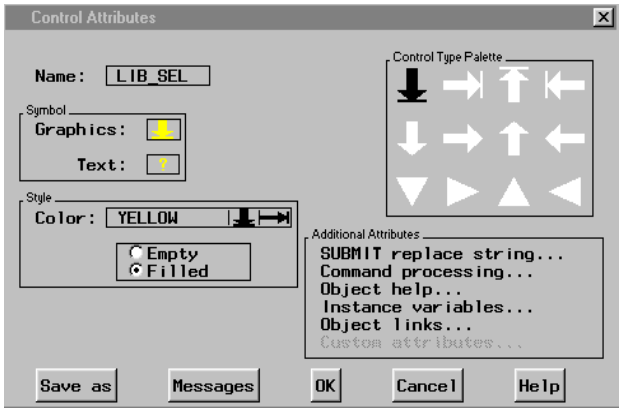

Figure 3: Control Object Attributes Windows

Next we add a graphics text to display the heading "Data Access" for the application. We place the graphics text at the top center of the frame as shown in Figure 1.

We then add a text entry to display the data set selected from the list box. We place the text entry right below the graphics text "Data Access".

Next we add a radio box to display two options, Audit System and Tracking System, from which the user can make a single selection. We place the radio box on the right side of the Data Access graphics text.

Lastly we add the two push buttons labeled Continue and Exit, which are set up to execute an action when clicked on by the user. We place the push buttons at the bottom center of the Data Access screen.

THE AUDIT AND TRACKING PROCESS

The user is initially presented with the Data Access screen, Figure 1, where the user clicks and picks a transaction data set from the list box. The user subsequently selects an option from the Radio Box, the Audit System or the Tracking System. To perform the desired action on the selected data set the user clicks on the Continue button.

THE MASTER AND THE TRANSACTION DATA SETS

The master data set referred to in this paper is called INDUSTRY and contains the original data. The transaction data sets, BANKING, BUILDING, FINANCE, FOOD, LEATHER, contain the new updated data. Our task is to update data in the master data set using the data from the transaction data sets and keep a history of changes to a file called AUDIT_DS.

The master data set and the transaction data sets have the same number of variables as shown below.

```
Variable  Type  Len Label
-------------------------------------------
REC_NUM   Char  5   RECORD NUMBER
USERNAME  Char  8   USERNAME
MODIFIED  Num   8   DATE MODIFIED
STATUS    Char  1   STATUS FLAG
IND_1990  Char  3   INDUSTRY CODE 1990
IND_WORK  Char  5   INDUSTRY CODE 2000
OCC_1990  Char  3   OCCUPATION CODE 1990
OCC_WORK  Char  5   OCCUPATION CODE 2000
DESCRIPT  Char  200 DESCRIPTION
```

REC_NUM contains the record identification number. USERNAME contains the name of the survey analysts who made the last modification to the record. MODIFIED contains the date when the record was last modified. STATUS contains the status flag on the record (A for add, E for edited, D for deleted and R for revised). OCC_1990 contains the occupation code as reported in the 1990 Census. OCC_WORK contains the occupation code for the 2000 Census. IND_1990 contains the industry code as reported in the 1990 Census. IND_WORK contains the industry code for the 2000 Census. DESCRIPT contains the description for the associated industry.

THE AUDIT SYSTEM

As mentioned in the introduction, each survey analyst is responsible for a set of transaction data sets. The analysts may make changes to the records, that is add, modify, or delete via a Data Entry and Correction System. For more information about such a system, please read "Building a Data Entry and Correction System by Synchronizing the Data Table and the Data Form", by Hung X Phan, SUGI24 Conference Proceedings.

The audit data set contains 17 variables as shown below.

```
Variable  Type  Len Label
-------------------------------------------
REC_NUM   Char  5   RECORD NUMBER
OLD_USER  Char  8   OLD USERNAME
NEW_USER  Char  8   NEW USERNAME
OLD_DATE  Num   8   OLD MODIFIED DATE
NEW_DATE  Num   8   NEW MODIFIED DATE
OLD_STAT  Char  1   OLD STATUS FLAG
NEW_STAT  Char  1   NEW STATUS FLAG
OLD_IN90  Char  3   OLD IND CODE 1990
NEW_IN90  Char  3   NEW IND CODE 1990
OLD_INWK  Char  5   OLD IND CODE 2000
NEW_INWK  Char  5   NEW IND CODE 2000
OLD_OC90  Char  3   OLD OCC CODE 1990
NEW_OC90  Char  3   NEW OCC CODE 1990
OLD_OCWK  Char  5   OLD OCC CODE 2000
NEW_OCWK  Char  5   NEW OCC CODE 2000
OLD_DESC  Char  200 OLD DESCRIPTION
NEW_DESC  Char  200 NEW DESCRIPTION
```

AUDIT SYSTEM SCL CODE

Listing 1 contains the SCL code which is used to implement the Audit System. The logic used here is pretty straightforward. We fetch one observation at a time from the transaction data set BANKING. For each record found in the BANKING data set, we find its matching record in the master data set INDUSTRY via the record identifier REC_NUM.

We then compare the corresponding fields in each record and if there is a difference in values, we set the audit_flg to 1, change the master record to the values on the transaction record and write an entry log to the audit data set AUDIT_DS. We also set up a counter to keep track of the number of transaction records which are used to update the master records.

```
/**** Listing 1. Audit System SCL code ****/
INIT:
/* initialize and assign libref */
    trans_ds = 'HOME.BANKING';
    mas_ds   = 'MASTER.INDUSTRY';
    audit_ds = 'MASTER.AUDIT_DS';
/* open transaction ds, BANKING */
    dsid=open(trans_ds,'i');
return;
MAIN:
/* execute the DO_AUDIT module */
   link DO_AUDIT;
return;
TERM:
/* close transaction ds, BANKING */
   rc=close(dsid);
return;

DO_AUDIT:
/* This module is called by the LINK DO_AUDIT
statement in the MAIN section.*/

/*fetch one rec at a time fr BANKING ds*/
   rc=fetch(dsid);
   auditflg = 0;
   trans_rc = getvarc(dsid,rec_num);
   occ_1990 = getvarc(dsid,occ_1990);
   occ_work = getvarc(dsid,occ_work);
   ind_1990 = getvarc(dsid,ind_1990);
   ind_work = getvarc(dsid,ind_work);
   descript = getvarc(dsid,descript);

/*fetch the matching rec fr INDUSTRY ds*/
   SUBMIT SQL CONTINUE;
     SELECT
            rec_num, username, modified,
            status, occ_1990, occ_work,
            ind_1990, ind_work, descript
     INTO
            :mas_rec,  :mas_user, :mas_mod,
            :mas_stat, :mas_oc90, :mas_ocwk,
            :mas_in90, :mas_inwk, :mas_desc
     FROM
               &mas_ds
     WHERE
               rec_num = &trans_rc;
    ENDSUBMIT;

/*compare each field in the selected record*/
if (occ_1990 ne mas_oc90)
    then
     do;
        auditflg = 1;
        old_oc90 = mas_oc90;
        new_oc90 = occ_1990;
     end;

if (occ_work ne mas_ocwk)
    then
     do;
        auditflg = 1;
        old_ocwk = mas_ocwk;
        new_ocwk = occ_work;
    end;

if (ind_1990 ne mas_in90)
    then
     do;
        auditflg = 1;
        old_in90 = mas_in90;
        new_in90 = ind_1990;
     end;

if (ind_work ne mas_inwk)
    then
     do;
        auditflg = 1;
        old_inwk = mas_inwk;
        new_inwk = ind_work;
     end;
if (descript ne mas_desc)
    then
     do;
        auditflg = 1;
        old_desc = mas_desc;
        new_desc = descript;
     end;

/* if auditflg =1 then
     there is a change in the selected record.
  => update  the master record. */
if (auditflg)
  then
    do;  /* do update */

     SUBMIT SQL CONTINUE;
      RESET NOPRINT;
      UPDATE
          &mas_ds
      SET
         REC_NUM  = "&rec_num", /*character*/
         USERNAME = "&tran_usr",
         MODIFIED =  &modified, /*numeric*/
         STATUS   = "&status",
         OCC_1990 = "&occ_1990",
         OCC_WORK = "&occ_work",
         IND_1990 = "&ind_1990",
         IND_WORK = "&ind_work",
         DESCRIPT = "&descript"
      WHERE
           rec_num = "&trans_rc";
     ENDSUBMIT;

/* Increment update counter */
     upd_cnt = upd_cnt + 1;
/* Insert an entry log to the AUDIT_DS */
    SUBMIT SQL CONTINUE;
     RESET NOPRINT;
     INSERT INTO
           &audit_ds
       SET
           REC_NUM  = "&rec_num",
           NEW_USER = "&new_user",
           OLD_USER = "&old_user",
           OLD_DATE =  &mas_mod,
           NEW_DATE =  &modified,
           OLD_STAT = "&old_stat",
           NEW_STAT = "&new_stat",
           OLD_OC90 = "&old_oc90",
           NEW_OC90 = "&new_oc90",
           OLD_OCWK = "&old_ocwk",
           NEW_OCWK = "&new_ocwk",
           OLD_IN90 = "&old_in90",
           NEW_IN90 = "&new_in90",
           OLD_INWK = "&old_inwk",
           NEW_INWK = "&new_inwk",
           OLD_DESC = "&old_desc",
           NEW_DESC = "&new_desc";
    ENDSUBMIT;

  end;  /* end update */
return;
```

/*End Listing 1. Audit System SCL code**/**

Figure 4 shows the Audit System in its completed state.

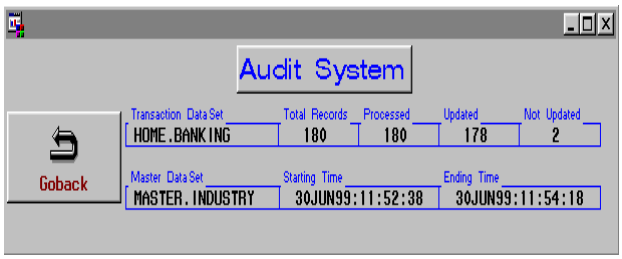

Figure 4: The Audit System in its completed state.

Notice that in Figure 4 there are 180 total records in the transaction data set BANKING of which 178 records have been updated on the master data set INDUSTRY. There are 2 records in the BANKING data set that did not get updated because the values for these records were unchanged. In other words, the new values for these 2 records in the BANKING data set were the same as the original values in the INDUSTRY data set. Hence, there was no audit entry log written out to the AUDIT_DS for these 2 records.

THE TRACKING SYSTEM

The tracking system allows the user to trace an audit history on a specified record. The windows screen we build for the tracking system allows the user to view two different data sets side-by-side as shown in Figure 5.

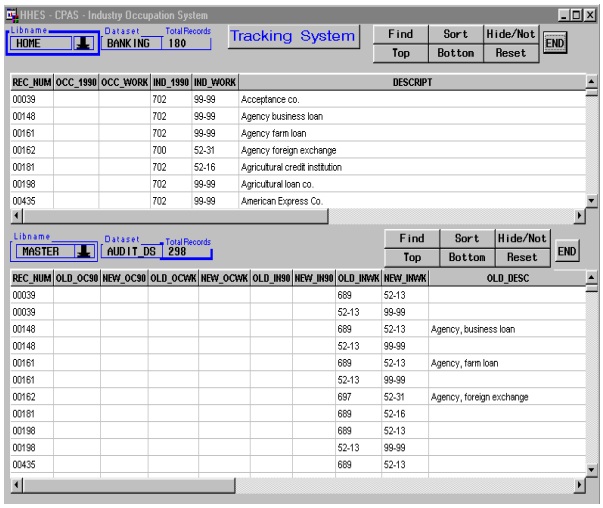

Figure 5: The Tracking System

TRACKING SYSTEM SCL CODE

Listing 2 contains the SCL code which is used to implement the Tracking system. The logic used in this module is very simple. We display the transaction data set, BANKING, in the first data table and the audit data set, AUDIT_DS, in the second data table as seen in Figure 5.

The user can navigate throughout the tracking system using the associated tool bars located in the top right corner above the data tables.

To track an audit history for a particular record in the BANKING data set, the user clicks on the corresponding row in the first data table, and the tracking system will find the matching record in the AUDIT_DS, in the second data table. The Tracking system will then highlight and display the selected record on the top row of the two data tables, where the user can easily review and examine the changes.

/*Listing 2. Tracking System SCL code***/**

```
INIT:
/* initialize and assign libref */
   trans_ds  = 'HOME.BANKING';
   audit_ds  = 'MASTER.AUDIT_DS';
return;

MAIN:
/* execute the following modules */
   link DISPTAB1;
   link DISPTAB2;
return;

TERM:
  rc=close(dsid);
return;

DISPTAB1:
/* get the widget identifier for data table 1 */
  call notify('.','_get_widget_','table1',t_id1);
/* open the transaction ds in data table 1 */
  call notify('table1','_set_dataset_',trans_ds,
                'BRONLY','NOADD','NODEL');
return;

DISPTAB2:
/* get the widget identifier for data table 2 */
  call notify('.','_get_widget_','table2',t_id2);
/* open the audit ds in data table 2 */
  call notify('table2','_set_dataset_',audit_ds,
                'BRONLY','NOADD','NODEL');
return;

TABLE1:
/* This module is invoked when the user clicks on a row
within the data table 1. */

/* make  rowlist & collist coordinate for
     selected cell clicked on by the user */
 rowlist=makelist();
 collist=makelist();
/* unselect cell where  previously selected */
 call send(t_id1,'_clear_select_');
 call send(t_id2,'_clear_select_');
/* get coordinate of the current selected cell */
 call  send (t_id1,'_get_active_cell_',rowlist,collist);
/* get the corresponding row fr. the selected cell */
 rownum=getitemn(rowlist,1);
/* highlight the selected row */
 call send(t_id1,'_select_row_',rowlist);
```

```
/* get the current selected row number */
  call send(t_id1,'_get_current_row_number_',currow);
/* display the selected row on the top of data table 1 */
  call send(t_id1,'_goto_absolute_row_',currow);
/* get the value of rec_num that was selected */
     call send(t_id1,'_get_column_text_','rec_num',rec_num);
/* make find_lst  that contains the find request*/
    find_lst=makelist();
/* build the srch_row string */
    srch_row = 'rec_num='||quote(rec_num);
/* insert the srch_row string into the find list */
  find_lst=insertc(find_lst,srch_row);
/* return the row that meets the find request list*/
   call send(t_id2,'_find_row_',find_lst,find_row);
/* no long need find_lst list  -  so delete it */
    rc=dellist(find_lst);
/* find_row > 0 => we find the srch_row */
    if (find_row > 0)
      then
       do;
/*display a msg - matching rec found in table2 */
     _msg_= 'NOTE: Find a match for '||srch_row;
/* make row_lst  that contains the srch_row*/
       row_lst=makelist();
/*  insert the srch_row string into  row_lst list */
          row_lst=insertn(row_lst,find_row);

/* highlights the srch_row found in table 2 */
          call send(t_id2,'_select_row_',row_lst);
/* no long need row_lst list  -  so delete it */
          rc=dellist(row_lst);
        end;
     else
/*find_row =0 => we did not find the srch_row*/
        do;
         alarm;
_msg_= 'NOTE: End of file reached without a match.';
        end;
  end;

/* no longer need these lists  -  so delete them */
   rc=dellist(rowlist);
   rc=dellist(collist);
 return;
```

/*End Listing 2. Tracking System SCL code*/

CONCLUSION

In this paper, we have presented a technique to build an Audit System which updates a MASTER data set using data from multiple TRANSACTION data sets and keeps an audit history of changes in an AUDIT data set.

We have also demonstrated a technique to build a Tracking System to trace an audit history of changes on a specified record and display the information on the screen.

This system was built on a very basic concept of auditing and tracking and can be implemented to work with any SAS data sets.

ACKNOWLEDGMENTS

The author wishes to thank Mr. Richard A Denby and his staff who have provided encouragement and technical advice. The author would also like to thank Alice Phan for help editing this paper.

REFERENCES

U.S. Bureau of the Census, January, 1992,
1990 Census of Population and Housing,
Alphabetical Index of Industries and Occupations.

SAS Institute Inc.,1994,
SAS Screen Control Language
Reference, Version 6, Second Edition

Destiny Corporation, 1997, Building Frame Entry Applications

The author can be contacted at:

Mr. Hung X Phan
U.S. Bureau of Census
1483/3 CPAS/HHES
Washington, D.C. 20233
Tel: 301-457-3204
E-mail: hphan@census.gov

DISCLAIMER:

This paper reports the results of a research and development project undertaken by the author. It is not an official report of the U.S. Census Bureau and does not necessarily reflect the views of the U.S. government. The author is solely responsible for any errors and/or inaccuracies in the contents of this paper.