Database Update with Incremental Audit Trail and Rollback Using the FSEDIT Procedure and Screen Control Language

William R. Fehiner, SAS Institute (Canada) Inc.

ABSTRACT

A critical component of many database update applications is the ability to maintain a complete audit trail, even when update is interrupted. This paper describes the implementation of an incremental audit trail within the FSEDIT Procedure using Screen Control Language (SCL). This application first without change on several platforms under Release 6.05 of the SAS System. Audit data are as current as each updated observation, allowing recovery from unexpected interruptions to be complete. Audit data are stored in a form which makes possible automatic rollback to earlier states of the database. The SCL code that generates audit data is written to be independent of other SCL functions that may be required by the FSEDIT application. Dependence on the data structure of the edited data view or data file is limited to a small segment of SCL.

This paper also describes programs that automatically generate the SCL code from PROC CONTENTS output, automate rollback, and generate a printed form of audit trail report for tracing changes.

INTRODUCTION

An important element in many database update applications is the ability to maintain a complete audit trail. If this audit trail is in a machine-readable form, then it can also be used to reverse a set of updates, either completely or for a subset of changes. This paper describes the implementation of an incremental audit trail using SAS/FSP software. Database engines now available in Release 6.05 of SAS/ACCESS software provide for direct editing of database tables with PROC FSEDIT. This form of audit trail described here can be used to document, back up, and reverse updates to database tables as well as data sets in SAS libraries.

John Rinehart (1989) proposed using the SAS System to generate an audit trail with a DATA step. In his scheme, the current version of a data set is compared to a backup stored before an update session. Some difficulties with this approach come from the expiration time between making an update and recording it. For example, this approach fails to identify individual changes, either by time or account. When several accounts simultaneously update a data set using SAS/SHARE software, it is important to associate updates with accounts. These difficulties are overcome here by using code to collect audit trail data at the same time that an observation is changed.

This implementation has several other desirable characteristics. The trail is maintained independently and synchronously with the file being updated. The edit and update operations are not slowed down by the audit trail code. You are not required to develop any skills beyond those required by the update application before adding an audit trail. For your ease in adding this function to your FSEDIT screens, the mechanism which creates and maintains the audit trail is designed to be independent of any other operations such as field validation. As well, very little of the audit trail code depends on the specific structure of the data set being updated.

Rollback can be done at various times and to various degrees. During editing, you may wish to reverse one or more changes to a particular observation by using a REVERSE command. A database administrator may want to reverse changes to the database in any of the following ways:

* back to a particular point in time,
* only for certain observations,
* only for changes made by a particular operator.

A rollback data set can also serve as an incremental backup. You can recreate an earlier version of the database from the rollback data set instead of saving multiple copies of the database on a regular basis.

This paper is written for database administrators, application developers, and maintenance programmers. The incremental audit trail provides an added benefit when you choose PROC FSEDIT to develop interfaces to a database. EXAMPLE OF OUTPUT illustrates the type of information provided by the rollback data set for a typical update session.

You can include an audit trail in your application by copying the prewritten code segments into your own SCL code and recopying.

INSTALLATION outlines for a maintenance programmer how to install the SCL components. The programs to generate backups and audit trail reports from the rollback data set are also described.

GENERAL DESIGN OBJECTIVES describes, for the application programmer, the general design objectives of this project. FUNCTION OF SCL PHASES summarizes the function of each SCL phase in meeting these objectives. These two sections highlight the features of SCL used in the application code.

EXAMPLE OF OUTPUT

The following example illustrates the key features of the output produced by this application. This example is intentionally simple in order to limit the size of the figures. CONCLUSION provides details on the type and range of testing used to validate the SCL code.

Figure 1 shows a sample data set before editing and Figure 2 is the same data set after an editing session. Figure 3 gives the formatted audit trail report corresponding to this editing session. Three observations were modified, two were added, one was deleted, and the changes to one of the observations were undone. The rollback data set created during this session is displayed in Figure 4 in chronological order. Note how the changes that were undone are flagged by reversing the sign of OBS_ID. Values in the rollback data set reflect the state of an observation before a change. In the special case when a Value changes from missing to nonmissing, the recorded value is a special missing value.

Although the process for building and maintaining the rollback data set is transparent to you, its existence in the background provides you with a convenient REVERSE command. A standard FSEDIT CANCEL command cancels all changes made to the current observation. The REVERSE command allows you to return to an observation and undo one or more sets of changes even though the changes had been saved. When CANCEL is used, no observation is written to the rollback data set since no changes are made to the master data set. On the other hand, REVERSE reverses earlier changes to the master data set, adds an observation to the rollback data set, and marks the rollback observation corresponding to the earlier change as undone.

It is not necessary for you to remember the number or type of changes made to an observation in order to use the REVERSE command. Whenever an observation is newly displayed, or after a
CANCEL command has been issued, the REVERSE command is available. Each time a REVERSE command is issued, the rollback dataset is searched in reverse order for a matching OBS_ID. If it is found, it is used to reverse the earlier change and a message indicating the date and time of the earlier change is displayed. On the other hand, if no further changes are outstanding for the current observation, an appropriate message is displayed.

The formatted audit trail in Figure 3 is generated by interleaving the rollback dataset with the new version of the master dataset. Only observations which have been edited are included in the audit trail.

INSTALLATION

There are several stages in adding audit trail capability to an FSEDIT screen. The first stage is to add a unique observation ID variable to the master dataset. This sequence number is maintained by the audit trail code and provides the link between the master dataset and any rollback datasets which are produced.

The second stage creates the structure for a rollback dataset that will be used with the master dataset. This structure contains each of the variables in the master dataset plus a date/time stamp and an indicator for the type of modification: add, delete, change or reverse. OBS_ID holds the date/time stamp for each modification, while CHANGE indicates the type of change. Thus, the second stage is to run a SAS program that creates a history data set that will be used to archive rollback data at the end of each editing session.

The majority of SCL code that builds an observation in the rollback dataset is independent of the structure of the dataset being updated. A small segment (the INT phase) however does depend on the number of character or numeric variables in the master data set. The same SAS program that creates a history data set also writes this structure dependent segment of SCL code.

The third stage is to merge the two SCL segments that implement the audit trail with any other SCL segments that are needed by the application. A series of include commands followed by a compile is sufficient to accomplish this. A special naming convention for nonwindow SCL variables used in the audit trail code avoids any conflicts with variables used in other segments.

The final stage is to set up a policy and procedure for naming and archiving each rollback data set as it is produced. Although this can be a manual procedure, you can also automate it in the PREPARE phase of the SCL.

When these four stages are complete, the application is ready to be used. The code segments that implement the audit trail do not have to be modified when other changes and enhancements are made to the FSEDIT screen.

GENERAL DESIGN OBJECTIVES

There are many ways to implement an incremental audit trail within
Screen Control Language. Several criteria have been used to decide on the particular design presented here:

* Record-level capture of audit data. In other words, a rollback observation should be written at the same time as each modified master observation is being saved.

* Minimal resources needed to collect audit data.

* Audit trail SCL independent of other SCL.

* Automatic generation of audit trail SCL from the master data set structure.

* Minimal amount of code explicitly dependent on the master data set structure.

Features of the design which help to meet these criteria include the following:

* All nonscreen variable names begin with two underscores (_-) to avoid collision with variable names in the master data set.

* Arrays are used to limit all references to master data set structure to the FSEINIT phase.

* Special missing values are used in rollback observations so a restore can be implemented with a standard update statement.

* The SET function is used to simplify and speed up data set access.

* The ability to open multiple data sets simultaneously and the same data set more than once is key to record-level capture of rollback observations.

FUNCTIONS OF SCL PHASES

A copy of the SCL for an FSEDIT screen to be used with a data set with two numeric and three character variables is displayed in the Appendix. The general contribution of each SCL phase to audit trail collection is described in this section. Comments in the SCL code itself provide additional documentation. As noted in the MAIN phase, you can add other components of SCL to this base set to provide additional functionality in your FSEDIT screen, for example, cross validation of entry fields.

* FSEINIT
  This is the only phase which refers to the structure of the master data set explicitly. Arrays and nonscreen variables are set up to reflect this structure. This phase creates and opens an empty data set for new rollback observations.

* INIT
  This phase establishes a reference point for identifying changes to screen variables during the TERM phase.

* MAIN
  This component implements the REVERSE command. It searches for a match to the current observation among rollback observations. If a match is found then its values are applied to the current observation. The REVERSE command can also be used to restore a previously deleted observation.

* TERM
  This phase verifies that modifications have been made to the current observation. If they have, it constructs a corresponding rollback observation, complete with datatime stamp, to document the changes.

* FSETERM
  This phase appends rollback observations to the history data set prior to leaving the application.

Each time you modify an observation in the master data set, an observation is added to the rollback data set. The variable _CHANGE is set to C and the old value of each variable is stored. As mentioned above, when a value changes from missing to non-missing, the recorded value is a special missing value. When you add a new observation to the master data set, _CHANGE is set to A and the rollback variables contain a special missing value. When you delete an observation, _CHANGE is set to D and a complete copy of the deleted observation is made. All of these operations are transparent to you. These choices for the rollback data set make it easy to undo a change while editing, create an audit trail report, and rollback the master data set when the edit session is complete.

CONCLUSION

You do not need to learn any special skills to use this incremental audit trail. Rollback observations are added to the rollback data set automatically and pointers to the master data set are maintained by the SCL. The REVERSE command adds significant functionality by expanding the scope of the CANCEL command to any observation that has changed during this session. The REVERSE command is capable of restoring a deleted observation as well.

The incremental audit trail adds no perceptible increase in panel response time when running an FSEDIT application on MVS, CMS, CMS*5, UNIX* or OS/2® operating systems. On the MS-DOS operating system, an increase in panel response time is noticeable on slower machines. It is only on MS-DOS that you need to consider the tradeoff between response time and the added functionality it buys.

Development of the code was done partly with Release 6.04 of the SAS System on the MS-DOS operating system and partly with Release 6.06 of the SAS System on the CMS operating system.
The final version of the code runs without change under either release.

The SAS program to automatically generate the FSEINIT component of SCL from the master data set is fast and effective. It handles two special cases where the master data set lacks either numeric or character variables. Although only 14 statements in the FSEINIT phase depend on the master data set structure, using the SAS program to generate the code avoids editing errors.

The incremental audit trail was tested with a test data set which included all possible combinations of one or two modifications of numeric or character variables or both. Changes to or from missing values were included. All combinations of add, duplicate, delete, change end cancel were tested. The REVERSE command was tested after each of these operations. As well, the accuracy of a batch rollback was verified. Finally, the audit trail report generated from the rollback data set was checked for each of the above combinations.

Data set compression is available in Release 6.06. If disk storage is at a premium, compression of the rollback data can be used to minimize disk storage requirements. Even without compression, the use of an incremental audit trail may be justified by the savings obtained from replacing incremental backups by the rollback data set.

There is a myth that only specialized database management systems have the capability of providing immediate collection of audit trail data. This paper may help to dispel that myth. Screen Control Language has all the components required to collect audit trail data simultaneously with updates and efficiently use resources both for collection and storage of this data.

REFERENCES


For further information, please contact:

William R. Fehlner
SAS Institute Canada Inc.
225 Duncan Mill Road
North York, Ontario M3B 3K9

SAS, SAS/FSF, and SAS/ACCESS are registered trademarks of SAS Institute Inc., Cary, NC, USA. MS-DOS is a registered trademark of Microsoft Corp. OS/2 is a registered trademark of International Business Machines Corp. UNIX is a registered trademark of AT&T. VMS is a registered trademark of Digital Equipment Corp.