Audit Trails for SAS Data Sets

Minh Duong
Texas Institute for Measurement, Evaluation, and Statistics
University of Houston, Houston, TX

ABSTRACT

SAS data sets are now more accessible than ever. They are commonly stored on network shared folders and accessed by many different users. Tracking when modifications were made, who made the modifications and why is an increasing challenge. Without a system to log modification you might be compromising the quality of your data. Audit trails can be used to help solve this problem.

INTRODUCTION

Audit trails track modifications and gives you the ability to restore records. The Audit Trail is an optional SAS file that can be created to log modifications to a SAS data file. Every time a record is added, deleted, or updated, information is written to the audit trail about who made the modification, what was modified, and when. Audit trails contain all the changes to the data set, this allows you the ability to fix unintended changes and restore to an earlier version of the data. Since historical information about the data is maintained, you can also develop statistics and patterns.

Audit trails will also store observations from failed append operations that were rejected by integrity constraints. The audit trail also enables you to write a DATA step to extract the failed or rejected observation and then reapply the observation to the data file.

The audit trail is created by the default Base SAS engine and has the same member name as the data file but set type of AUDIT. It replicates the variables in the data set and adds _AT*_ variables and user variables. Below are the variables in the audit trail and code values.

_AT*_ variables are the table below.

ATDATETIME	Stores the date and time of a modification
ATUSERID	Stores the logon user ID that is associated with a modification
ATOBSNO	Stores the observation number that is affected by the modification, except when REUSE=YES (because the observation number is always 0)
ATRETURNCODE	Stores the event return code
ATMESSAGE	Stores the SAS log message at the time of the modification
ATOPCODE	Stores a code that describes the type of modification

ATOPCODE code values are below

AL	Auditing is resumed
AS	Auditing is suspended
DA	Added data record image
DD	Deleted data record image
DR	Before-update record image
DW	After-update record image
EA	Observation add failed
ED	Observation delete failed
EU	Observation update failed

INITIATE AUDIT TRAIL

Let's do an example. First we create a libname to a temporary directory and copy the sashelp.class data set to the temporary location. Then we create an audit trail for the SAS data set using the code below. An optional user variable _reason_change_ is added to the audit trail. User variables can be used to capture reasons for a modifications or any other information.

```
libname mydata "c:\era\";

data mydata.class;
  set sashelp.class;
run;


proc datasets lib=mydata;
 audit class;
 initiate;
 user_var _reason_change_ $30;
quit;
```

Suspend, resume, and terminate are other options for the audit statement. The suspend option stops the event logging but does not delete the audit trail. Resume resumes the event logging after the audit trail was suspended. Terminate stops the event logging and deletes the audit trail. We can see the status of the data set by running proc contents. Figure 1 shows the output of the proc contents.

```
proc contents data=mydata.class(type=audit);
run;
```

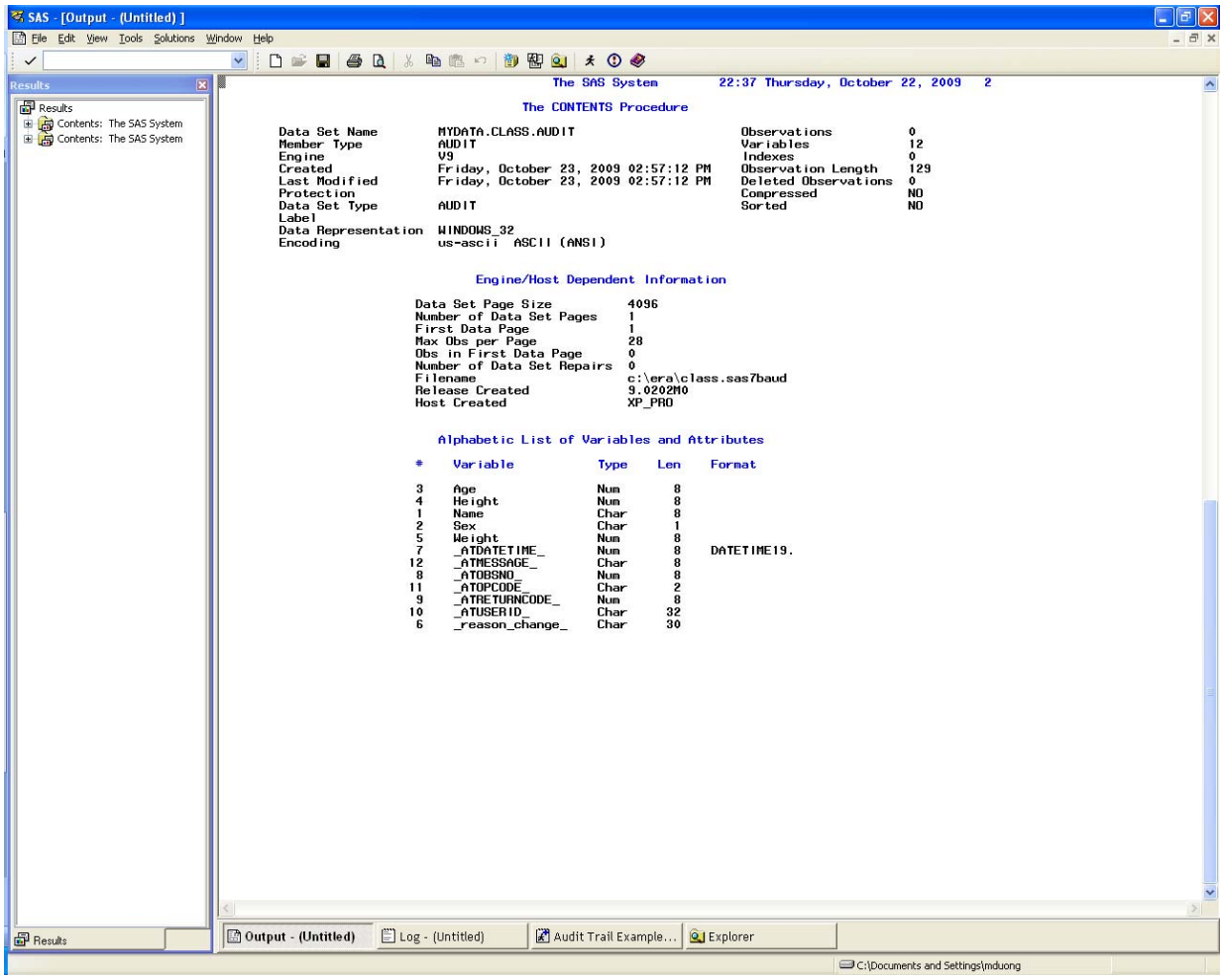


Figure 1. Output of proc contents of audit trail.

Now let's modify the data set and see the audit trail in action. We run the code below to modify the data set. The first data step will modify John's age to 22. The next will insert a new record into the data set and the third will delete a record from the data set. We set _reason_change_ to the reason for the modification. This will be retained in the audit trail and can be referenced later by viewing the audit table.

```
data mydata.class;
  modify mydata.class;
  where name = "John";
  age = "22";
  _reason_change_ = "correct John's age";
run;

proc sql;
 insert into mydata.class
 set name = "Minh",
  sex = "M",
  Age = 30,
```

```
  Weight = 175,
  Height = 66,
 _reason_change_ = "Moved to school";
quit;

data mydata.class;
  modify mydata.class;
  where name = "William";
  remove;
run;
```

Now that the modifications have been made we can see what the audit trail looks like. We run the code below to create a data set from the audit trail. The 'type=audit' is required to refer to the audit trail.

```
data classAudit;
  set mydata.class(type=audit);
run;
```

When we open the classAudit data set in viewtable we see that several records were created in the audit trail. Each record in the audit trail contains information about who modified the record, date & time, and type of modification. The modification type is stored in the _ATOPCODE_ variable with a code. The first record has a _ATOPCODE_ value of DR. The DR code means that this record is the record image before an update has occurred. The second record contains the _ATOPCODE_ value of DW. The DW means that this is the record after the update. This record also contains the _reason_change_ we set in the modify data step. The third record contains a _ATOPCODE_ value of DA. The DA code means that this record was added to the data set. The forth record has a _ATOPCODE_ value of DD. The DD code means that this record was deleted from the data set. Figure 2 shows what work.classaudit looks like.

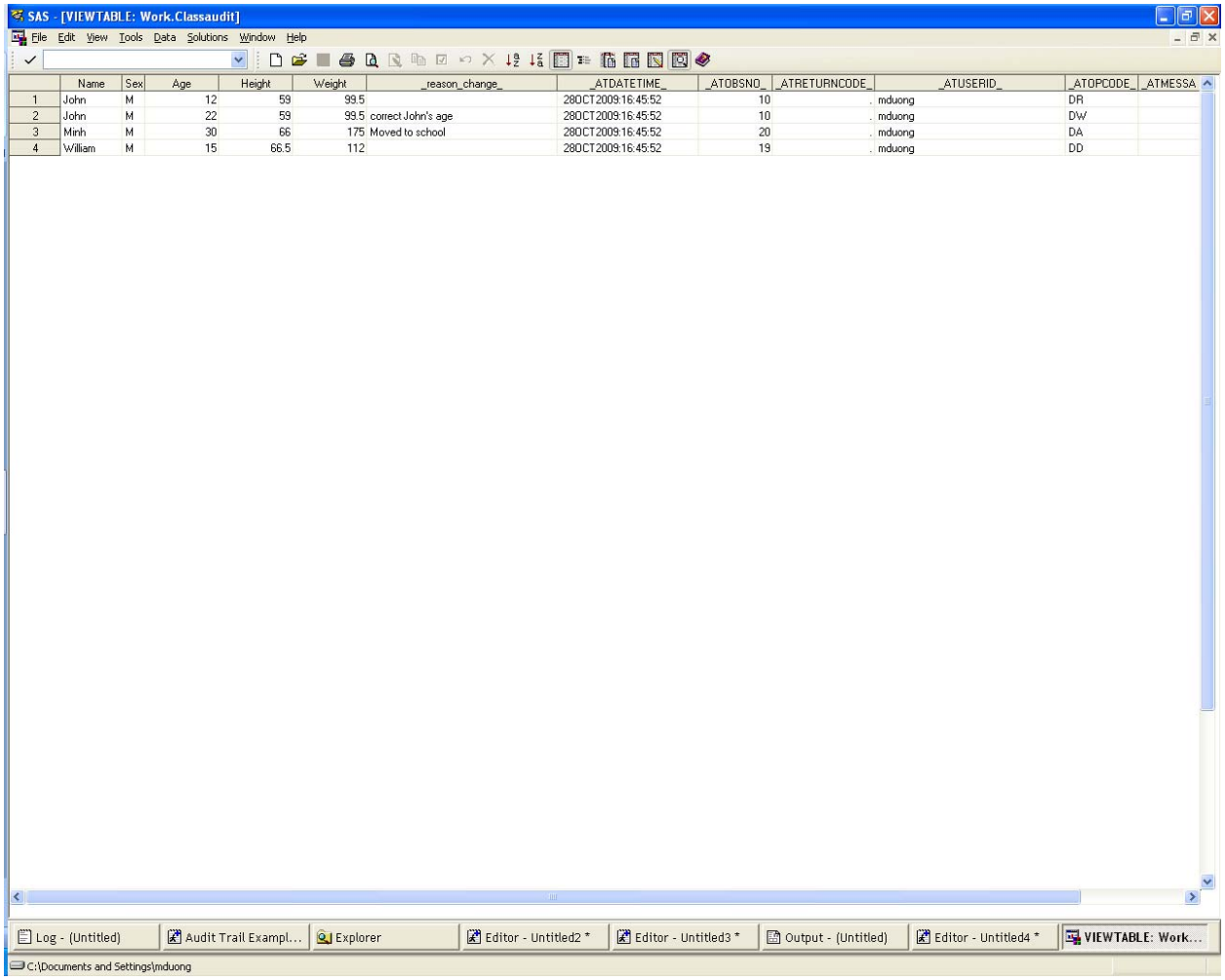


Figure 2: Viewtable: Work.classaudit

RECOVER RECORDS

Normally, the only way to recover records that have been changed or deleted would be to restore a data set from backups or shadow copy. Backups may not be available if the error is detected weeks or months after the modification was made. With audit trails you can easily restore the original records back without having to contact your backup administrator. This is how it can be done.

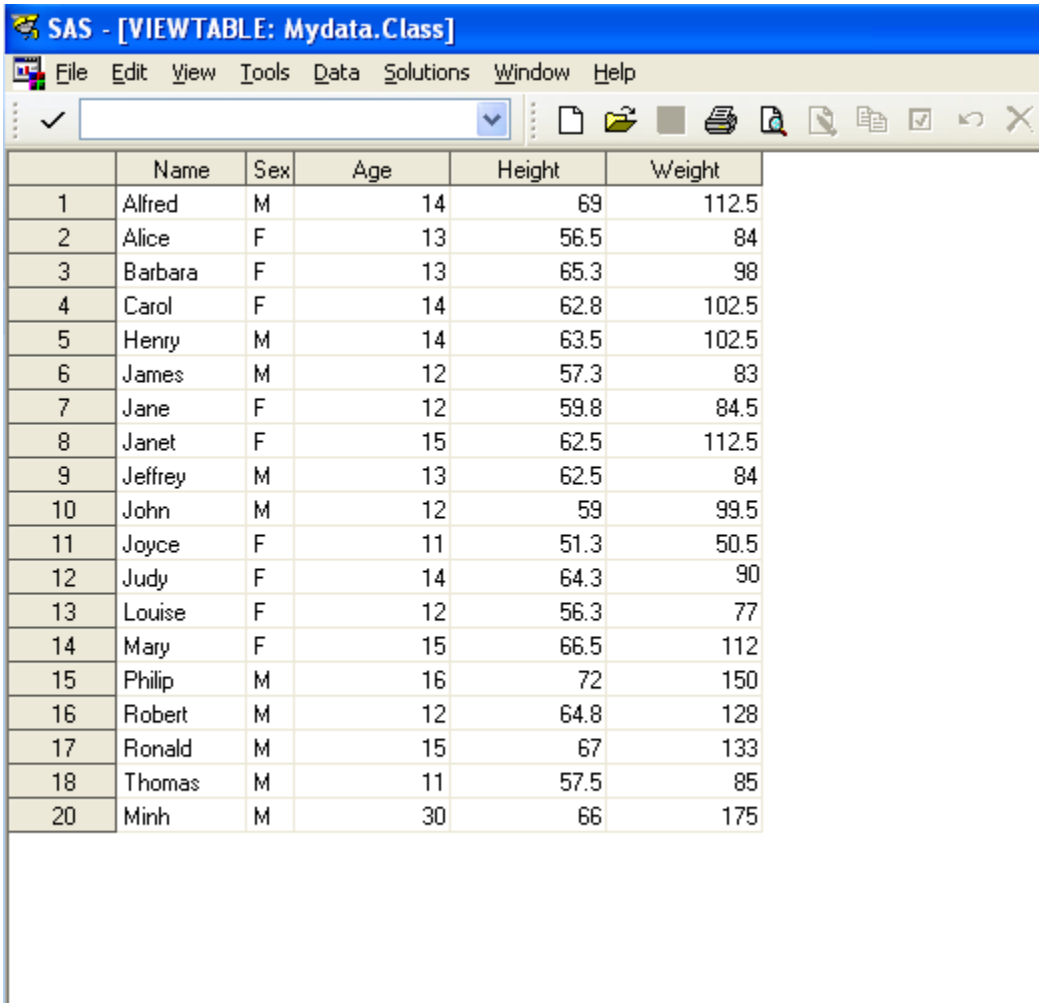
In the previous example we already modified John's age from 12 to 22. Now we can restore his original record back from the audit trail. We first create a transaction data set named rollbackupdate from the audit trail by running the code below. The rollbackupdate data set will contain the record image before it was updated.

```
data rollbackupdate;
  set classAudit;
  where name = "John" and _ATOPCODE_ = "DR" and _ATUSERID_= "&sysuserid";
  drop _: ;
run;
```

Now we are ready to modify the master data set (mydata.class) with the transaction data set (rollbackupdate) we just created. We run the code below to modify the master data set with transaction data set.

```
data mydata.class;
  modify mydata.class rollbackupdate;
  by name;
run;
```

We now can open mydata.class with viewtable and see that John's age has been modified back to 12. Figure 3 shows the viewtable of mydata.class.



	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Alice	F	13	56.5	84
3	Barbara	F	13	65.3	98
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84
10	John	M	12	59	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90
13	Louise	F	12	56.3	77
14	Mary	F	15	66.5	112
15	Philip	M	16	72	150
16	Robert	M	12	64.8	128
17	Ronald	M	15	67	133
18	Thomas	M	11	57.5	85
20	Minh	M	30	66	175

Figure 3: Viewtable mydata.class

ERROR IN SAS PROGRAM

A SAS program normally contains many data steps and procedures. It is common to submit a program several times before the program runs without errors. A syntax error in the program or a system error will cause some data steps to finish successfully while some will not. Going thru the logs and figuring

out which data steps need to resubmitted can be a tedious and a time consuming task. An alternative is to use an audit trail and rollback records if an error occurs in the SAS program.

We first create the macro rollbackerror to restore records during an error. The datasetName parameter is the name of the data set we recovering records from. The rollbackerror macro will look for records that were updated, deleted and restore the records back to the data set.

```
%macro rollbackerror(datasetName);

data &datasetName.Audit;
  set &datasetName(type=audit);
run;

data rollbackupdate;
  set &datasetName.Audit;
  where _ATOPCODE_ ="DR" and _ATUSERID_= "&sysuserid" and
datepart(_ATDATETIME_) = today();
  drop _: ;
run;

data rollbackdelete;
  set &datasetName.Audit;
  where _ATOPCODE_ ="DD" and _ATUSERID_= "&sysuserid" and
datepart(_ATDATETIME_) = today();
  drop _:;
run;

data rollbackinsert;
  set &datasetName.Audit;
  where _ATOPCODE_ ="DA" and _ATUSERID_= "&sysuserid" and
datepart(_ATDATETIME_) = today();
  drop _:;
run;

data &datasetName;
  modify &datasetName rollbackupdate;
  by name;
run;

proc append base=&datasetName data=rollbackdelete force; run;

data &datasetName;
  modify &datasetName rollbackinsert;
  by name;
  remove;
run;
%mend;
```

We now create a SAS program to modify a SAS data set. We initiate the audit trail for mydata.class and make several modifications to the data set. We then intentionally cause an error by adding an extra ‘s’ to mydata.class. This will set the automatic macro variable syserr greater than 0 and the macro rollbackerror will be invoked. The rollbackerror macro will recover the records before they were updated and restore the deleted records. It will also delete records that were added to the data set.

```
%macro dosomething;

libname mydata "c:\era\";

data mydata.class;
  set sashelp.class;
run;

proc datasets lib=mydata;
 audit class;
 initiate;
 user_var _reason_change_ $30;
quit;

data mydata.class;
  modify mydata.class;
  where name = "John";
  age = "22";
  _reason_change_ = "correct John's age";
run;

proc sql;
 insert into mydata.class
 set name = "Minh",
  sex = "M",
  Age = 30,
  Weight = 175,
  Height = 66,
 _reason_change_ = "Moved to school";
quit;

data mydata.class;
  modify mydata.class;
  where name = "William";
  remove;
run;

data mydata.class;
  modify mydata.classs; *THIS WILL CAUSE AN ERROR;
  where name = "Jeff";
  remove;
run;

%if &syserr > 0 %then %do;
  %rollbackerror(mydata.class);
%end;

proc datasets lib=mydata;
 audit class;
 terminate;
quit;

%mend;
%dosomething;
```

We can run proc compare to verify that sashelp.class and mydata.class have the same values. Figure 4 shows the output of the proc compare.

```
proc compare base=sashelp.class compare =mydata.class; run;
```

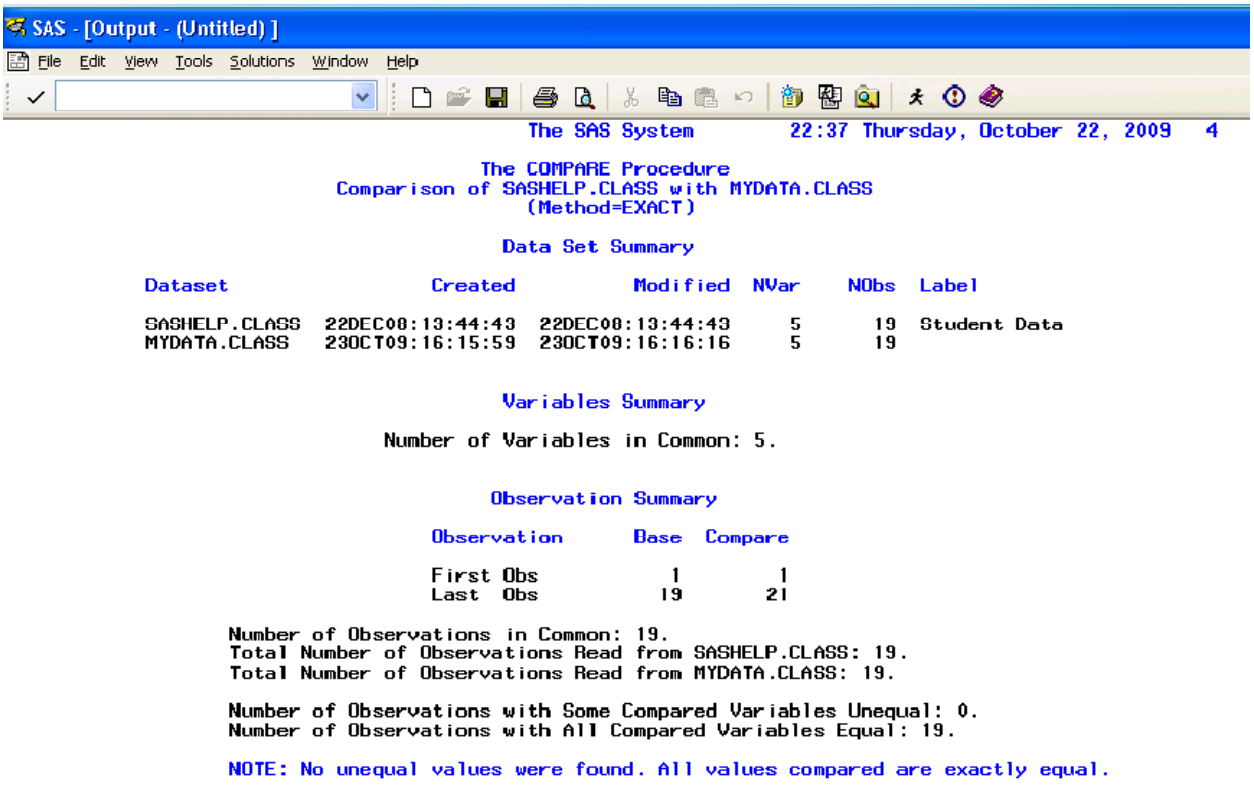


Figure 4: proc compare of sashelp.class and mydata.class

CONS OF AUDIT TRAIL

There are disadvantages with using audit trails. Audit trails can become large quickly because it contains all the data and modifications. This can affect the performance of the SAS system when you are working with the data set. The audit trail also only applies to data changes made using a modify statement in the data step, UPDATE, INSERT, and DELETE statements with PROC SQL, or PROC APPEND. The audit trail will be deleted if the data set is rebuilt using the DATA step or SQL procedure.

CONCLUSION

Audit trails are a great tool for tracking modifications to SAS data sets. Not only does it allow you monitor changes but it also allows you to rollback to a previous version. I would encourage everyone to use audit trails in their environment.

REFERENCES

(1) SAS 9.2 Language Reference Concepts
http://support.sas.com/documentation/cdl/en/lrcon/61722/PDF/default/lrcon.pdf

(2) Maria Y. Reis, "Audit Trails of SAS Data Set Changes An Overview"
http://www.nesug.org/Proceedings/nesug03/ph/ph006.pdf

CONTACT INFORMATION

Minh Duong
minh.duong@times.uh.edu