The Missing Semicolon™

Volume 10, Number 2 Summer 2008

What's New in SAS 9.2

With each new release of SAS, I embark on a treasure hunt through the "What's New" documentation, finding many shiny new features to add to my trove. In this article, I share some of my bounty, focusing on the Base SAS software, functions, and macros. I encourage you to set out on a hunt of your own, because there is much more to be found with a little digging!

About SAS 9.2

The SAS Institute announced the release of SAS version 9.2 at the 2008 SAS Global Forum. At SSC, we have installed 9.2 and are very happy with it. There are several major enhancements and hundreds of small improvements that make our jobs easier.

The SAS Institute is delivering SAS 9.2 in phases. The "Classic SAS" system, including Base SAS, SAS/GRAPH, SAS/STAT, and other basic SAS products, is currently available. The managed products, such as SAS/BI, will be released later this year. SAS 9.2 can now be downloaded via the Internet through a new Electronic Software Delivery process. The new SAS Deployment Wizard streamlines installation and configuration.

Base SAS: Major Updates

Some of the new features included in Base SAS 9.2 are:

- A new SAS Code Analyzer procedure, SCAPROC. This procedure collects metadata about the job step, input, output and use of macro symbols as a program is running. The metadata is sent to a file in a location you specify.
- A new option, STEPCHKPT | NOSTEPCHKPT. In checkpoint mode, SAS stores data in a checkpoint restart library as a job executes. If the job terminates prior to completion, you can restart the job beginning with the step that was executing at termination.

Continued on page 3

Free Webinar

Advantages of Dimensional Data Modeling

Thursday, July 17th
2:00 pm - 3:00 pm

Learn what dimensional data modeling is and how it can help you, a SAS Programmer or Analyst, get the most out of your data.

Part 1 (2:00-2:30)

- Introduction to data modeling
- Overview of benefits of the dimensional model

Part 2 (2:30 - 3:00)

- Dimensional modeling basics: facts and dimensions
- Drill-down and drill-across queries
- Surrogate keys and slowly changing dimensions

FREE registration at: www.sys-seminar.com/publications_tms.php

Space is limited—register today

SYSTEMS SEMINAR CONSULTANTS, INC.
2997 Yarmouth Greenway Drive
Madison, WI 53711
www.sys-seminar.com
train@sys-seminar.com
1-800-997-7081

Online, Instructor Based SAS Training Schedule, p. 9

In This Issue

Letter From the President

Dear SAS User:

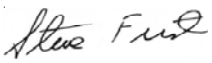
This spring I attended my 25th International SAS User Group. The conference was held in San Antonio, right on the Riverwalk. It was exciting to not only attend the conference but also to spend time in this historic neighborhood of San Antonio. I had many wonderful meals on the Riverwalk and even found a few free minutes for their boat tour and a visit to The Alamo.

It was an honor to be an invited speaker at SAS Global Forum again this year. I presented *The SAS File and Infile Statements*. The ideas for this paper came from a conversion project I was working on. Even after over 30 years of programming in SAS, I am still learning new things about the software's capabilities. This paper overviews the ways I have used the file and infile statements for decades and the new ways I am using it now. For a copy of my paper and presentation, visit http://www.sys-seminar.com/presentations.php.

The overall theme of the conference was predictive analytics. This is where SAS business applications become really innovative. We can look not only at what our company and our customers did in the past (although that's very important), but also leverage our data to help us make informed business decisions going forward!

Another exciting piece of the conference was the introduction of SAS 9.2. This new release offers the same great product with improved and enhanced functionality. For more details on SAS 9.2, please see *What's New in SAS 9.2* in this issue of the newsletter.

As always, I enjoyed being involved in the SAS community. The conference was a great time to see many of you – reconnecting with old acquaintances and meeting many new people. I was eager to share my knowledge and learn from many of you. And, of course, I always look forward to learning about the direction of the SAS software. It has been a superior tool for decades, and it's ever evolving functionality will continue to provide us with the support to make the best business decisions based on our data.



Sascommunity.org

The SAS community has recently come out with an exciting new tool to connect the community and share issues and knowledge.

I am sure that we are all familiar with the website wikipedia.org. This is just one example of a new web phenomenon called wikis. A wiki is a collection of web pages designed to enable anyone who accesses it to contribute or modify content. So, wikis are basically user driven web pages. This means that everyone can contribute their ideas and knowledge. Of course, a certain amount of caution should be used when using information from a wiki since anyone can contribute. This drawback is mostly mitigated by the fact that other users can question or modify entries which don't seem completely accurate, and if it a controversial topic, there is also a discussion area for everyone's ideas to be hashed out.

Sascommunity.org is a wiki which is built by SAS users and supported by The SAS Institute. It was introduced at the 2007 SAS Global Forum. This site contains everything from blogs to presentations to forums to SAS opportunities and more. This site is the ideal place to share your SAS knowledge and take advantage of the knowledge of others.

Community is very important in an industry such as ours. It allows us to draw upon the expertise of others (if someone else has already figured out a solution to our problem, why not borrow from that!), share our knowledge with the hope of helping others, and commiserate and celebrate about our shared failures and successes.

I encourage each of you to visit sascommunity.org and explore which pieces may be useful to you. While you are there, consider what you can contribute to the site. It could be as much as a series of articles or as little as a one word correction or addition. We are all in this together, and our mutual support and cooperation will benefit everyone.

What's New in SAS 9.2

CONTINUED FROM PAGE 1

- Ability to write your own functions. A new procedure, PROC FCMP, allows you tocode functions in the SAS language. Alternatively, functions written in C or C++ can be registered using PROC PROTO.

- Performance enhancements galore. PROC SQL, PROC SORT, PROC REPORT and other procedures sport new enhancements. Additionally, the IN operator is more efficient and dash-list and colon-list syntax is now available to read like-named SAS data sets.

Base SAS: Some Specifics

Next, let's zoom in on a few new features in Base SAS that are especially useful.

- A new option to the SET statement: INDSNAME= variable. This option creates and names a variable that stores the name of the SAS data set from which the current observation is read.

- Two new options to PROC PRINT: SUMLABEL and BLANKLINE. SUMLABEL displays the label of the BY variable on the summary line. BLANKLINE inserts a blank line after every *n* observations.

The following example illustrates these options. Suppose you have three regional sales data sets. You want to combine the data sets in a single report, but still separate the data by region. The three data sets, the code, and the results are shown below.

Data:

```
MidWest
   Channel      Sales
   DirectMail   25
   TV           55
   Web          110


South
   Channel      Sales
   DirectMail   55
   TV           175
   Web          25

East
   Channel      Sales
   DirectMail   175
   TV           75
   Web          85
```

Code:

```
data Regions;
 /* Create variable dsn which stores name    */
 /* of the dataset that the record was read */
 /* from.                                    */
 set Midwest East South indsname=dsn;
   by channel;
 /* scan dataset name, select 2nd word       */
 /*   delimited by .                         */
 /* which excludes the libref work,          */
 /* creating Region                          */
 Region = scan (dsn, 2, '.=); run;

options nobyline nodate pageno=1;

 /* sumlabel allows channel to be displayed */
 /* on summary line, also put a blank line  */
 /* after every single observation.         */

proc print data=Regions noobs sumlabel
        blankline=1;
  by channel;
  var Region Sales;
  sum Sales;
  label channel='Media Channel';
 /* put value of by variable into the title.*/
  title 'Totals for #byval(channel) Sales';
run;
```

Results:

```
 Totals for DirectMail Sales            1

Region           Sales
MidWest          25
East             175
South            55
-------------    -----
Media Channel    255

 Totals for TV Sales                     2

Region           Sales
MidWest          55
East             75
South            175
-------------    -----
Media Channel    305

 Totals for Web Sales                    3

Region           Sales
MidWest          110
East             85
South            25
-------------    -----
Media Channel    220
                 =====
                 780
```

Continued on next page

Functions

Several new functions make their debut in SAS 9.2. My favorites follow, along with examples illustrating their use.

Sorting functions

- CALL SORTC sorts the values of character arguments.
- CALL SORTN sorts the values of numeric arguments.

The following code sorts the character variable in the array in ascending order.

Code:

```
data _null_;
   array x(8) $10
    ('Janice' 'Joyce' 'Jerome' 'Jackie'
       'Joleen' 'Justin' 'Jerry' 'JoAnn');
     call sortc(of x(*));
     put +3 x(*);
run;
```

SAS writes the following output to the log:

```
Jackie Janice Jerome Jerry JoAnn Joleen Joyce
Justin
```

Character functions

- CHAR returns a single character from a specified position in a character string.
- FIRST returns the first character in a character string.

In the following example, the CHAR function returns the character at positions -1 through 4. The FIRST function also makes an appearance.

Code:

```
data test;
    retain string "abc"=;
    do position = -1 to 4;
        resultChar =char(string, position);
        resultFirst=first(string);
       output;
    end;
run;

proc print noobs data=test;
run;
```

Results:

string	position	result Char	result First
abc	-1		a
abc	0		a
abc	1	a	a
abc	2	b	a
abc	3	c	a
abc	4		a

Search functions

- FINDW searches a character string for a word.
- WHICHC searches for a character value equal to the first argument, and returns the index of the first matching value.
- WHICHN searches for a numeric value equal to the first argument, and returns the index of the first matching value.

In the following example, the FINDW function returns the position of the beginning of the word "she."

Code:

```
data _null_;
    whereisshe=findw('She sells sea shells? Yes,
    she does.','she');
    put whereisshe=;
run;
```

SAS writes the following output to the log:

```
whereisshe=28
```

The following code searches the array for the first argument and returns the index of the first matching value.

Code:

```
data _null_;
    array state (*) $12 state1-state3
    ('wisconsin'          'iowa' 'california');
    array statenum (*) statenum1-statenum3 (048
    024 104);

    california=whichc('california', of
    state[*]);
    wisconsin=whichc('wisconsin', of state[*]);
    illinois=whichc('illinois', of state[*]);
    _048=whichn(48, of statenum[*]);
    _104=whichn(104, of statenum[*]);
    _024=whichn(24, of statenum[*]);

    put california= / wisconsin= / illinois=;
    put _048= / _104= / _024=;
run;
```

Continued on next page

SAS writes the following output to the log:

```
california=3
wisconsin=1
illinois=0
    _048=1
    _104=3
    _024=2
```

Holiday Function

- HOLIDAY('*holiday*', *year*) returns a SAS date value for the holiday and year specified.

The following code finds the date of Father's Day 2008.

Code:

```
Data _Null_;
   fathers = holiday('fathers', 2008);
   format fathers date9.;
          put fathers;
       run;
```

SAS writes the following output to the log:

```
15JUN2008
```

Macros

Improvements to the macro language include new macro variables and new options.

New automatic macro variables

- The &SYSERRORTEXT automatic macro variable contains the last error message formatted for display on the SAS log.
- The &SYSWARNINGTEXT automatic macro variable contains the last warning message formatted for display on the SAS log.

Code:

```
data NULL;
    Set doesnotexist;
run;
%put &syserrortext;
%put &syswarningtext;
```

Log:

```
24    data NULL;
25           set doesnotexist;
ERROR: File WORK.DOESNOTEXIST.DATA does not
exist.
26    run;

NOTE: The SAS System stopped processing this step
because of errors.
WARNING: The data set WORK.NULL may be
incomplete.   When this step was stopped there
were 0observations and 0 variables.
WARNING: Data set WORK.NULL was not replaced
because this step was stopped.
NOTE: DATA statement used (Total process time):
      real time            0.01 seconds
      cpu time             0.01 seconds

27    %put &syserrortext;
File WORK.DOESNOTEXIST.DATA does not exist.
28    %put &syswarningtext;
Data set WORK.NULL was not replaced because this
step was stopped.
```

New Options

- The MEXECNOTE system option displays the macro execution information in the SAS log at macro invocation.

The MEXECNOTE option is turned on in the following code.

Code:

```
options mexecnote;
%macro test;
   %do i = 1 %to 3;
      %put &i;
   %end;
%mend test;
%test
```

Log:

```
36    options mexecnote;
37    %macro test;
38       %do i = 1 %to 3;
39          %put &i;
40       %end;
41    %mend test;
42    %test
NOTE: The macro TEST is executing from memory.
```

- The SECURE option enables you to write secure macros that protect intellectual property contained in stored compiled macros. This option causes the contents of a macro to be encrypted when stored in a stored compiled macro library. The SECURE option can only be used in conjunction with the STORE option.

When the following code is executed, the compiled macro is encrypted and its contents cannot be viewed.

Code:

```
Libname macrolib 'c:\mymaclib';
options mstored sasmstore= macrolib;
%macro calcFinancials/store secure;
   /* macro code goes here */
%mend calcFinancials;
```

Other Products

Several other SAS products boast exciting new features, procedures and enhancements. Read the "What's New in SAS 9.2" documentation for details. Happy hunting!

REFERENCES

Tyndall, Russ, "Give Your Macro Code an Extreme Makeover: *Tips for even the most seasoned macro programmer*" *Your SAS Technology Report, October 2005* Available http://www.sas.com/news/newsletter/tech/2005_10_11.pdf

SAS Institute Inc. 2008. *What's New in SAS® 9.2*. Cary, NC: SAS Institute Inc., Available http://support.sas.com/documentation/cdl/en/whatsnew/61982/PDF/default/whatsnew.pdf

Advantages of Dimensional Data Modeling

Dimensional Data Modeling for SAS Users

In this two-part series, we discuss the why's and how's of the dimensional data model, with a concentration on how data modeling and the dimensional model work with SAS software. We will explain several advantages of the dimensional data model. Part 1, in this issue of The Missing Semicolon, gives you an overview of what a data model is and why you should care, and introduces the dimensional model. In the next issue, part 2 discusses implementation and use of the dimensional model in more detail. Stay tuned.

Data Modeling

Consciously or not, we all use data modeling. Whenever you impose a logical structure on your data you have a "data model." Ideally, the logical structure helps you use your data efficiently. When working with a large volume of data, data with complex interdependencies, or data that must be presented to users with varying levels of technical and business knowledge, the data model you choose is critically important. In this article, we discuss three common data models: normalized, de-normalized, and dimensional. We will focus on how data modeling can be used with SAS software.

Note that a data model is not necessarily tied to any database management system or product. Just about any relational data system, including SAS, can be used to build the data models discussed here, including the dimensional model.

De-Normalized Data

For SAS users, it is common to use a model in which all relevant attributes of an "observation" are stored with the observed measurement in a single table row. This is a de-normalized data model. To many SAS users this model is second nature because most SAS procedures are designed to work with this type of data.

Sales Table
Transaction number
Customer name
Customer street address
Customer city
Customer state
Customer zip
Multi-state region
Prouduct category
Product number
Product name
Calendar day
Day of week
Month
Year
Season
Annual product cycle number
Sale quantity
Sale dollar amount

Figure 1: De-normalized Data Model

Free Webinar

Advantages of Dimensional Data Modeling

July 17, 2008

See the front page of this newsletter for details.

The sales table shown in Figure 1 is an example of a de-normalized data model. There is one row for each sale. All the customer and date attributes of our observed Sale Quantity and Sale Dollar Amount exist in that row. With PROC MEANS, these attributes could be CLASS variables and the Sale Quantity and Sale Dollar Amount could be VAR variables.

There are drawbacks to this data model. For example, the Product Category is directly correlated to the uniquely identifying Product Number, so we don't need to store the Product Category if we know the Product Number. Storing both uses additional disk space and negatively affects performance. More importantly, if a Product Category name changes or a product is shifted from one category to another, updating this table is cumbersome since all rows with the affected product must be changed. In addition, if a Product Category is changed, there is no natural way to maintain the change history or report both past and present values.

Normalized Data

An alternative to the de-normalized model is the normalized model. Figure 2 shows the sales data from figure 1 restructured with a normalized data model. Technically, there are degrees of normalization, but here we'll just call this "normalized."

Normalization reduces redundancy in the data and makes it easier to update. For example, if Product Category changes only the Product table need be updated instead of every row in the sale Transaction table. However, this approach comes with its own problems. Query performance is compromised because the joins required to bring the data together are complicated and often require multi-table bridging. In the example above, a simple total of sales by state would require joining four tables: Sales, Transaction, Customer, and Zip.

The Dimensional Data Model

You can think of the dimensional data model as falling between the normalized and de-normalized models. Data is partially normalized to avoid redundancy, but de-normalized data is kept where it simplifies the overall scheme, and facilities query performance and understandability. In fact, query performance and understandably are two key reasons to use the dimensional model.

Figure 3 shows the sales data arranged as a dimensional model. Note that our observed measures, Sale Dollar Amount and Sale Quantity, appear in the center table. These are continuously valued, high cardinality variables, meaning they have many unique values.

Continued on next page

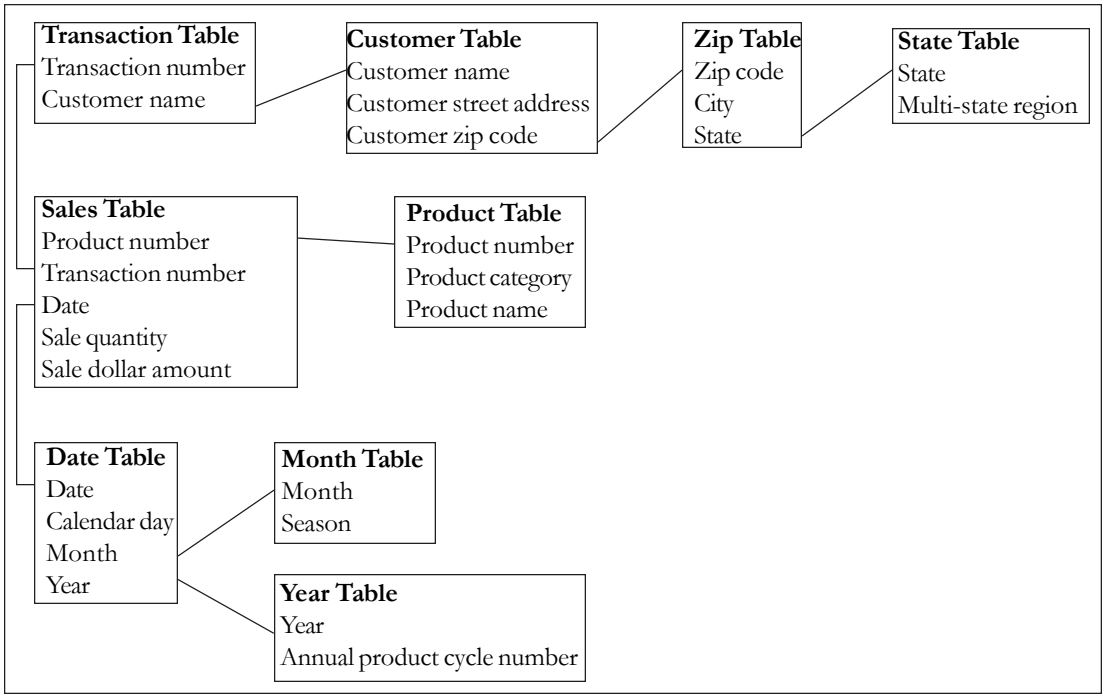


Figure 2: Normalized Data Model

Variables with lower cardinality (fewer unique values), such as Customer State and Year, appear in the tables surrounding the Sales Fact table. The overall pattern resembles a star and this type of model of table relationships is called a star schema.

Moving the high cardinality variables to a separate table reduces redundancy because we no longer have to repeat values for attributes such as Product Category for each sale, as with the de-normalized schema.

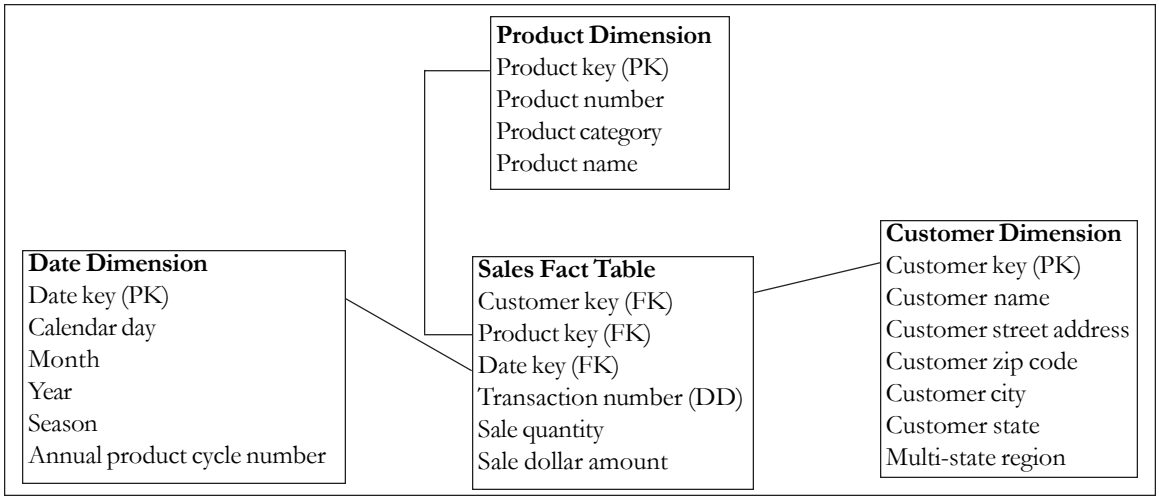
Table relationships are only one level deep, meaning you don't have to bridge through intermediate tables to associate an attribute with a sale. In contrast to the normalized schema above, sales total by state involves a join of only two tables: Sales Fact and Customer. The simplified structure leads to a more user-friendly database.

Because the dimensional model is designed to make queries fast and easy to construct, it may be your best choice for any system that will be queried or reported in ways that you cannot predict in advance, or systems subject to ad hoc queries. The dimensional model is also appropriate for systems in which low cardinality attributes, such as Product Category in our example, must be changed in a predictable and maintainable manner.

The dimensional model is often associated with data warehouse systems, but dimensional modeling is useful in many other situations, including POR (plain old reporting) systems.

Part 2 in this series, in the next issue of the Missing Semicolon, discusses implementing a dimensional model including a step-by-step process for converting a de-normalized table. In the meantime, check out our upcoming Dimensional Modeling webinar. Details about the webinar appear elsewhere in this issue.

Figure 3: Dimensional Model (star schema)



Resume Tips

1. **Mention your accomplishments**-Hiring managers want to see how you contributed in your past roles.
2. **Use action verbs**-Use action verbs to describe your responsibilities. Try words like performed, compiled, developed, generated, increased, and analyzed.
3. **Be consistent**-Be consistent with verb tense, numbers, dates, and abbreviations.
4. **Avoid personal pronouns**-Replace the personal pronouns "I" and "We" with action words.
5. **Include bullet points**-A well written resume will include bullet points that highlight your most relevant skills. Use of bullets helps avoid personal pronouns and use of sentences.
6. **Keep it brief, but not too brief**-There is a misconception that resumes should be only one page. It is true that you want to keep it brief, but not at the cost of excluding important skills. Use your judgment (and that of others). If you feel that your resume is too long, then it probably is.
7. **Provide contact information**-Include your personal phone number and email. Avoid using your work phone number or work email. In most cases, you should not discuss new opportunities while you are at work. So, to avoid an uncomfortable situation, omit your work contact information.
8. **Check spelling and grammar**-Reread your resume and use spell check. Keep in mind you may need to do this several times to pick up any errors. There are some hiring managers who will weed out candidates based on poor spelling and grammar. Your skills could be a perfect match, but this oversight could cost you the job.
9. **Enlist a second set of eyes**-Using another pair of eyes to review your resume is crucial. People tend to overlook their own spelling/grammar errors.
10. **Create customized versions**-It is good to have multiple versions of your resume. You want the reader to easily see how your experience relates to that particular position or industry.

QUICK TIP

There are times when you come across a program that uses a large number of macro variables. Frequently, when there is a problem, you want to display their values in order to validate the values at a particular point in time.

You are probably familiar with the %PUT and its various options:

```
%PUT <text| _ALL_ |_AUTOMATIC_ | _GLOBAL_
| _LOCAL_ | _USER_ >;
```

The problem is that the list of macro variables that displays in the log with this statement does not appear to be in any particular order. This makes it hard to look for the specific ones that you need.

What you may not be aware of is that an automatic SAS dataset exists, sashelp.vmacro, that contains all known macro variables and their values. You can use this dataset like any other SAS dataset.

Select the scope of macro that you want and sort by name:

```
proc sql;
    select
       name,
       value
    from    sashelp.vmacro
    where scope = 'GLOBAL'
    order by name;
quit;
```

Working with Audit Trail Files

When modifying or updating a SAS data set, it may be necessary to restore the data set to a prior view. One tool that helps accomplish this is an audit trail file. An audit trail file is a SAS file that logs modifications to a SAS data file, recording additions, deletions and updates.

Creating an Audit TrailFile

An audit trail file must be initiated by an AUDIT statement in the DATASETS procedure:

```
proc datasets lib=saslib;
   audit softsale;
     initiate;
quit;
```

The audit trail file has the same member name and library reference as the SAS data file, but a data set type of AUDIT instead of DATA. It contains the same variables as the SAS data file, plus two types of audit variables: automatic _AT* variables and user variables.

Automatic Variables

The following _AT* variables automatically store modification data:

- _ATDATETIME_ - The date and time of the modification
- _ATUSERID_ - The user id associated with the modification
- _ATOBSNO_ - The observation number affected by the modification
- _ATRETURNCODE_ - The event return code
- _ATMESSAGE_ The SAS log message at the time of the modification
- _ATOPCODE_ - The operation code that describes the type of modification

User Variables

The user may define optional user variables that store additional modification data. User variables associate data values with the SAS data file but are stored within the audit trail file instead of the SAS data set. The user variables are updated like any other data variable within the step to modify the SAS data file. User variables are defined by the USER_VAR statement when the audit trail is initiated.

```
proc datasets lib=saslib;
   audit softsale;
     initiate;
     user_var msgCode $ 10;
quit;
```

An Example

The following step will modify an observation and insert a new observation into the SAS data file softsale:

```
proc sql;
   insert into saslib.softsale
     set name='Laura'
       , division='S'
       , years=5
       , sales=3985.36
       , expense=628.18
       , state='IL'
       , msgCode='New Sales';
     update saslib.softsale
       set state='WI'
          , msgCode='transfer'
       where name = 'SARAH';
quit;
```

Every time observations are added, deleted, or modified in the SAS data file an observation is added to the audit trail file. After the insert and update are applied to the data set, a print of the audit trail shows that the following three observations have been recorded:

```
Obs Name    Division  Years  Sales    Expense   State

1   Laura       S       5    3985.36   628.18   IL
2   SARAH       S       6    301.21     65.17   MN
3   SARAH       S       6    301.21     65.17   WI

Obs msgCode       _ATDATETIME_       _ATOBSNO_
1   New Sales   03MAY2008:10:54:31       16
2                                         3
3   transfer    03MAY2008:10:55:51        3

Obs _ATRETURNCODE_  _ATUSERID_    _ATOPCODE_
_ATMESSAGE_
1          .         tschudrowitz     DA
2          .         tschudrowitz     DR
3          .         tschudrowitz     DW
```

Type Indicator

The value of _ATOPCODE_ is an indicator of the type of modifications made to the observation in the data file. _ATOPCODE_ has the following possible values:

- AL – Audit is resumed
- AS – Audit is suspended
- DA – Record image of inserted observation
- DD – Record image of deleted observation
- DR – Before update record image of updated observation
- DW – After update record image of updated observation
- EA – Observation add failed
- ED – Observation delete failed
- EU – Observation update failed

Continued on next page

Log Statement Options

The type of audit trail records recorded is controlled by the options specified on the LOG statement when the audit trail is initiated. If the LOG statement is omitted, then the default is to log all images of data changes. The possible options include the following:

- ADMIN_IMAGE=YES|NO – Controls the A operation codes
- BEFORE_IMAGE=YES|NO – Controls the DR operation code
- DATA_IMAGE=YES|NO – Controls D operation codes, except DR operation code
- ERROR_IMAGE=YES|NO – Controls E operation codes

Suppose only the modified image of updated observations is required. Then the DR operation code may be turned off.

```
proc datasets lib=saslib;
   audit softsale;
     initiate;
      log before_image=no;
     user_var msgCode $ 10;
quit;
```

If BEFORE_IMAGE were on, both the before update record image, with _ATOPCODE_ of DR, and the after update record image, with _ATOPCODE_ of DW, would be written to the audit trail. With BEFORE_IMAGE turned off, only the after update record image is written to the audit trail.

Displaying the Audit Trail

The audit trail is read-only. To refer to the audit trail the TYPE= data set option is required. A display of the audit trail data may be reported as follows:

```
proc print data=saslib.softsale (type=audit);
   title 'Audit Trail for saslib.softsale';
run;
```

Suspension and Termination

Once initiated, observations are written to the audit trail unless auditing is suspended or terminated. Suspension or termination of the audit trail is controlled by the AUDIT statement in the DATASETS procedure.

```
proc datasets lib=saslib;
   audit softsale;
     suspend;
quit;
```

After suspension, changes to the SAS data file are no longer tracked, but the audit trail file still exists. Logging of changes is resumed by the AUDIT statement in the DATASETS procedure as follows:

```
proc datasets lib=saslib;
   audit softsale;
     resume;
quit;
```

An audit trail is terminated as follows:

```
proc datasets lib=saslib;
   audit softsale;
     terminate;
quit;
```

When an audit trail is terminated, logging of changes stops and the audit file is deleted.

QUICK TIP

PROC IMPORT is a very useful tool for converting "text" files to SAS® data sets. However, if the imported file doesn't have consistent cell formats in each column, SAS may get confused regarding the type of data it is actually trying to read. By default, SAS scans the first 20 rows to determine variable attributes such as field length and data type when reading delimited text (CSV, TAB, and DLM) files. If the first 20 rows are numeric data and the next rows are mixed data, the default behavior will create missing values from row 21 forward. For example, a character value may have a length of 3 in the first 20 rows but more than 3 in subsequent rows. In this case, the character value of rows 21 and following would be truncated to 3.

Example:

Input	Result	Input	Result
1005	1005	cat	cat
2	2	rat	rat
345	345	dog	dog
.	.	.	.
. through	row 20 or	beyond then	
100A	.	elephant	ele
G222	.	giraffe	gir
3R5T	.	coyote	coy

Starting in Version 9.1, by using the statement GUESSINGROWS= one can tell Proc Import how many rows to scan.

GUESSINGROWS = 1 to 3276;
scans data for its data type from row 1 to the row number that is specified.

The following code tells SAS to scan the first 12,000 records and reads the CSV file correctly.

```
Proc Import out=myDataIn datafile = '/mydir/
       myData1.csv' dbms=csv;
    getnames=yes;
    guessingrows=12000;
    datarow=2;
run;
```

Why use an Audit Trail file?

There are many uses for audit trail files. Some possibilities are:

- Track data changes for security reasons
- Retain historical information about the data
- Mantain ability to track data from the time it enters the data file to the point it leaves
- Peverse changes when necessary.

System Performance

Because each update to the data file is also written to the audit file, the audit trail may negatively impact system performance. Suspending the audit trail for large, regularly scheduled batch updates will avoid this problem.

Live Web SAS Training

Learn from the Comfort of Your Office...

25 Years of SAS Training Experience

Complimentary Follow Up Help Desk

Our Trainers Are Also Expert Consultants

SAS® Report Writing Series	July 21-23
Tips, Tricks, & SAS® Techniques	July 24-25
Advanced SAS®	August 11-13
Exploiting SAS® ODS	August 14-15
SAS® Macros	August 25-26
Advanced SAS® Macros	August 27
The SAS® SQL Procedure	August 28
SAS® Efficiencies	August 29
SAS® Enterprise Guide	September 8-9
Introduction to SAS®	September 10-12
The SAS® SQL Procedure	September 22
SAS® Macros	September 23-24
Advanced SAS® Macros	September 25
SAS® Efficiencies	September 26
Advanced SAS®	October 6-8
Tips, Tricks, and SAS® Techniques	October 9-10
SAS® Report Writing	October 20-21
Introduction to PROC Report	October 22
Introduction to SAS®	November 3-5
The SAS® SQL Procedure	November 6

For questions or registration, contact
Our Training Coordinator at 1-800-997-7081, ext. 306
or visit www.sys-seminar.com

Steve First
President

Kathleen Nosal
Consultant
Editor

Teresa Schudrowitz
Project Manager

Rosalind Gusinow
Senior Consultant/Trainer

Kelly Kieler
Senior Consultant

Thomas Miron
Senior Consultant

Jennifer First
Office Manager

Erin Ward
Placement Coordinator

Ines Bowers
Administrative Assistant
Editor