Paper S1-05

Things Dr Johnson Did Not Tell Me: An Introduction to SAS® Dictionary Tables

Peter Eberhardt, Fernwood Consulting Group Inc., Toronto Canada

ABSTRACT

SAS maintains a wealth of information about the active SAS session, including information on libraries, tables, files and system options; this information is contained in the Dictionary Tables. Understanding and using these tables will help you build interactive and dynamic applications. Unfortunately, Dictionary Tables are often considered an 'Advanced' topic to SAS programmers. This paper will help novice and intermediate SAS programmers get started with their mastery of the Dictionary tables.

Ever needed a list of the tables (datasets) in a library? How about the columns (variables) in a table? Need to make sure you reset any titles after you run a report? Got some pesky warning messages in your SAS log you would like to clean up? Sure, you can look them up in the table and column properties in the explorer window. Or you can run a Proc Contents and check the listing. And of course you can ignore the warnings and errors in the SAS log since they 'almost always appear'. Or you can go to the Dictionary Tables and have your programme find out what libraries are allocated or what columns are available. So, what are Dictionary Tables and where do we access them?

Before we proceed, let's come to some common ground with terminology. In this paper we will talk about tables; for SAS programmers a table is the same as a dataset. Where a dataset has observations a table has rows. Where a dataset has variables a table has columns. You ask "Why do we use this terminology?". And we try to answer "Because Relational Database Management Systems (RDBMS) use this terminology, and they have always had their own Dictionary Tables. The SAS Dictionary Tables are documented in Proc SQL, so we assume this is why SAS uses the SQL/RDBMS terminology."

WHAT ARE DICTIONARY TABLES?

What happens when you start a SAS session? Ever right clicked on a table in the SAS explorer and looked at its properties? Would you like to put basic information about an input file into a report footer? Have you been asked to change one of your macros so all of the SAS options are set to the same values they had before you macro was invoked? Ever wonder how you can access some of this information in a programme? The first step is using this information is to understand what this information represents; simply, this information represents ***metadata***. In general we think of metadata as "data about data". That is, whereas the data in a table might represent patient records, the metadata tell us attributes of the table; attributes such as column names, column type (numeric, character), table location etc.. The metadata SAS makes available goes beyond the traditional "data about data". The metadata SAS makes available also includes data about the operating environment. The metadata SAS makes available can be found in the SAS Dictionary Tables.

SAS Dictionary Tables are ***read only*** tables created and maintained by SAS; they contain a wealth of information about the current SAS session.; the Dictionary Tables are stored in a SAS library called ***DICTIONARY***. Although SAS Dictionary tables are read only, they are ***dynamic*** in that virtually everything you do in the SAS session causes a change in the Dictionary Tables. When you create a new table, whether in a DATA Step, SQL or another SAS Proc, several Dictionary tables are immediately updated. When you set a title or footnote a Dictionary Table is immediately updated. When you create a new macro variable a Dictionary Table is immediately updated. Set a system option… that's right, a Dictionary Table is updated. The tables are dynamically updated and always available; you have to do nothing to keep them up-to-date. Now that we have a basic understanding of what the SAS Dictionary Tables represent, metadata, let's look at the tables that are available.

SAS introduced Dictionary Tables in version 6. In SAS v8 the number of Dictionary Tables grew to eleven; these were augmented in SAS v9.1.3 and grew to twenty-two. As noted above, the Dictionary Tables cover virtually every aspect of the SAS session. The tables below enumerate the Dictionary Tables, first showing the v8 tables then the new tables in v9.

SAS V8 DICTIONARY TABLES

Table	Description
CATALOGS	Contains information about SAS Catalogs
COLUMNS	Contains information about variables/columns
EXTFILES	Contains information about external files
INDEXES	Contains information about columns participating in indexes
MACROS	Contains information specific to macros
MEMBERS	Contains information about all data types (tables, views and catalogs)
OPTIONS	Current session options
STYLES	ODS styles
TABLES	Contains information about tables/datasets
TITLES	Contains information about titles and footnotes
VIEWS	Contains information about views

ADDITIONAL SAS V9.1.3 DICTIONARY TABLES

Table	Description
CHECK_CONSTRAINTS	Contains information about Check constraints
CONSTRAINT_COLUMN_USAGE	Contains information about Constraint column usage
CONSTRAINT_TABLE_USAGE	Constraint table usage
DICTIONARIES	DICTIONARY tables and their columns
ENGINES	Available engines
FORMATS	Available formats
GOPTIONS	SAS/Graph options
LIBNAMES	LIBNAME information
REFERENTIAL_CONSTRAINTS	Referential constraints
REMEMBER	Remembered information
TABLE_CONSTRAINTS	Table constraints

In v6 and v8 you had to go to the SAS documentation to discover which Dictionary Tables were available. In v9 a new table was added, ***DICTIONARIES***, which has table and column information on all of the Dictionary Tables. Now, in order to discover what Dictionary Tables are available you simply have to query the ***DICTIONARIES*** table; the following SQL snippet shows how to get a list of all available Dictionary Tables:

```
Select distinct memname
from dictionary.dictionaries;
```

In Figure 1 below (Courtesy of CodeCrafters Inc) we can see the tables and how they are related and organized.

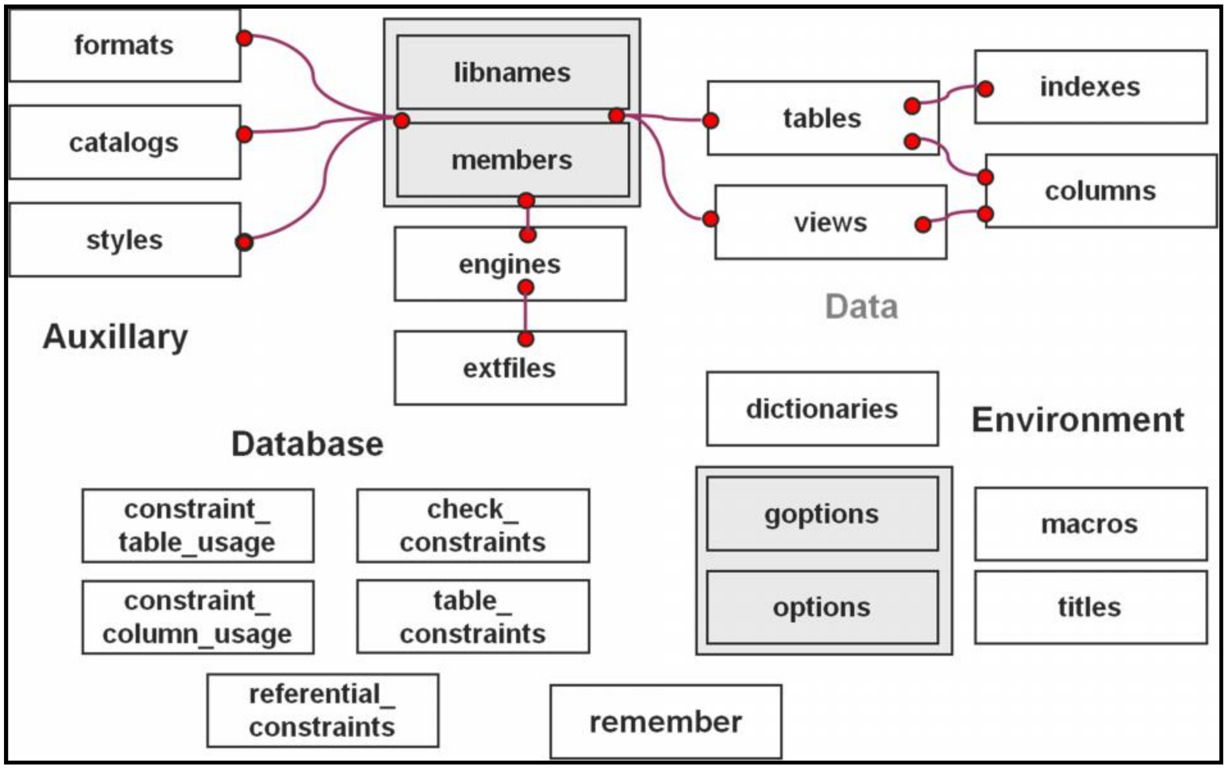


FIGURE 1 – DICTIONARY TABLE RELATIONSHIPS – COURTESY OF CODE CRAFTERS INC.

This diagram gives a concise picture of the tables and how they fit into the SAS environment. As we can see, many of the these dictionary tables contain the metadata about our data (e.g. ***libnames, members, tables, columns***) , but as SAS has evolved and added more RDBMS type capacity to its data management strengths we can see this reflected in new tables being added to the dictionary (e.g. ***constraint_table_usage, check_constraints***). In addition, SAS has provided us with tables with metadata about session environment (e.g. ***options, goptions, macros***) as well as auxiliary metadata (e.g. ***styles, formats***). Before we look at how we can use these tables, let us look at how we can determine the structure and content of the tables.

So, how can we see the structure of these tables; that is, what columns are in the table? Proc SQL has a DESCRIBE TABLE command that will display the SQL that was used to create the table; the table structure is displayed in the log window. For example Listing 1 shows the SAS code submitted, SAS log notes and the table structure of DICTIONARY.TABLES as they appear in the log window:

LISTING 1 – GETTING THE STRUCTURE OF A DICTIONARY TABLE

```
27   proc sql;
28   describe table dictionary.tables;
NOTE: SQL table DICTIONARY.TABLES was created like:

create table DICTIONARY.TABLES
  (
   libname char(8) label='Library Name',
   memname char(32) label='Member Name',
   memtype char(8) label='Member Type',
   dbms_memtype char(32) label='DBMS Member Type',
   memlabel char(256) label='Dataset Label',
   typemem char(8) label='Dataset Type',
   crdate num format=DATETIME informat=DATETIME label='Date Created',
    modate num format=DATETIME informat=DATETIME label='Date Modified',
    nobs num label='Number of Physical Observations',
    obslen num label='Observation Length',
    nvar num label='Number of Variables',
    protect char(3) label='Type of Password Protection',
    compress char(8) label='Compression Routine',
    encrypt char(8) label='Encryption',
    npage num label='Number of Pages',
    filesize num label='Size of File',
    pcompress num label='Percent Compression',
    reuse char(3) label='Reuse Space',
    bufsize num label='Bufsize',
    delobs num label='Number of Deleted Observations',
    nlobs num label='Number of Logical Observations',
    maxvar num label='Longest variable name',
    maxlabel num label='Longest label',
    maxgen num label='Maximum number of generations',
    gen num label='Generation number',
    attr char(3) label='Dataset Attributes',
    indxtype char(9) label='Type of Indexes',
    datarep char(32) label='Data Representation',
    sortname char(8) label='Name of Collating Sequence',
    sorttype char(4) label='Sorting Type',
    sortchar char(8) label='Charset Sorted By',
    reqvector char(24) format=$HEX48 informat=$HEX48 label='Requirements Vector',
    datarepname char(170) label='Data Representation Name',
    encoding char(256) label='Data Encoding',
    audit char(3) label='Audit Trail Active?',
    audit_before char(3) label='Audit Before Image?',
    audit_admin char(3) label='Audit Admin Image?',
    audit_error char(3) label='Audit Error Image?',
    audit_data char(3) label='Audit Data Image?'
   );

29   quit;
```

HOW DO I ACCESS DICTIONARY TABLES?

Now that we know how to find table and column names, how do we access them? There is an automatic library called DICTIONARY, so we access the tables the same way we access any SAS table using SQL. For example, to access the MEMBERS table we would do something like:

```
PROC SQL;
SELECT *
FROM dictionary.members;
QUIT;
```

A VIEW OF THE DICTIONARY

One thing that jumps out at experienced DATA step programmers is the library name of DICTIONARY. SAS libname definitions are limited to 8 characters yet DICTIONARY has 10 characters. This means that the Dictionary Tables cannot be directly accessed by the DATA step or by SAS PROCs, aside from PROC SQL. To access the Dictionary Tables directly you must use PROC SQL. However, that does not mean the metadata are not available to the DATA step or to other SAS PROCs; SAS also provides ***views*** in the SASHELP library that can surface the metadata in the DATA step and/or other PROCs. The Dictionary Tables are only directly accessible through PROC SQL whereas, the views are accessible from any SAS proc (including SQL), data step as well as the SAS explorer window. One of the simplest ways to become familiar with the Dictionary Tables is to open the equivalent SASHELP view in the SAS explorer window. In this paper I use the term Dictionary Table even though some of the examples use the equivalent views.

The tables below list the SASHELP views along with the SQL that is used to create the view. The first table has the views that are identical to the underlying Dictionary Table, and the second table has those views that provide a subset of the underlying tables.

VIEWS THAT ARE IDENTICAL TO THE UNDERLYING TABLE

SASHELP VIEWS	Source
SASHELP.VOPTION	select * from dictionary.OPTIONS
SASHELP.VGOPT	select * from dictionary.GOPTIONS
SASHELP.VTITLE	select * from dictionary.TITLES
SASHELP.VMACRO	select * from dictionary.MACROS
SASHELP.VLIBNAM	select * from dictionary.LIBNAMES
SASHELP.VENGINE	select * from dictionary.ENGINES
SASHELP.VEXTFL	select * from dictionary.EXTFILES
SASHELP.VMEMBER	select * from dictionary.MEMBERS
SASHELP.VTABLE	select * from dictionary.TABLES
SASHELP.VVIEW	select * from dictionary.VIEWS
SASHELP.VCOLUMN	select * from dictionary.COLUMNS
SASHELP.VDCTNRY	select * from dictionary.DICTIONARIES
SASHELP.VINDEX	select * from dictionary.INDEXES
SASHELP.VCATALG	select * from dictionary.CATALOGS
SASHELP.VFORMAT	select * from dictionary.FORMATS
SASHELP.VSTYLE	select * from dictionary.STYLES
SASHELP.VCHKCON	select * from dictionary.CHECK_CONSTRAINTS
SASHELP.VREFCON	select * from dictionary.REFERENTIAL_CONSTRAINTS
SASHELP.VTABCON	select * from dictionary.TABLE_CONSTRAINTS
SASHELP.VCNTABU	select * from dictionary.CONSTRAINT_TABLE_USAGE
SASHELP.VCNCOLU	select * from dictionary.CONSTRAINT_COLUMN_USAGE
SASHELP.VREMEMB	select * from dictionary.REMEMBER

VIEWS THAT ARE A SUBSET OF THE UNDERLYING TABLE

View	Contents	Source
SASHELP.VALLOPT	Options and Goptions	select * from DICTIONARY.OPTIONS union select * from DICTIONARY.GOPTIONS;

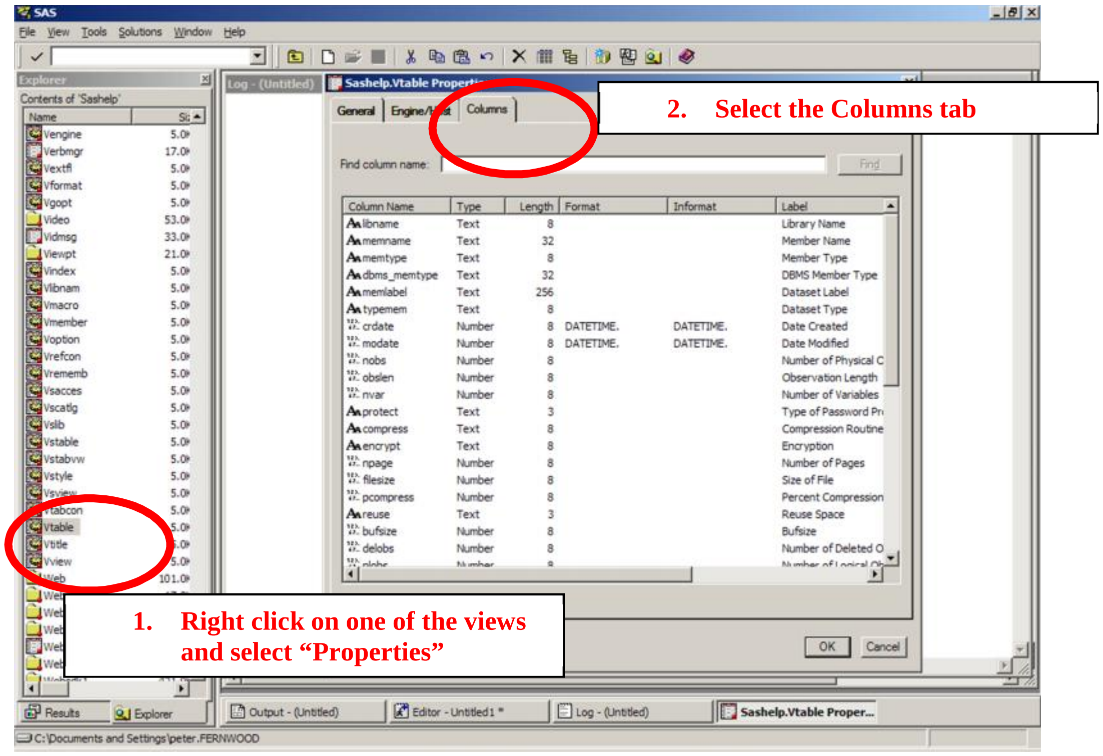
SASHELP.VCFORMAT	Character Formats	select fmtname
		 from DICTIONARY.FORMATS
		 where source='C';

View	Contents	Source
SASHELP.VSACCES	SAS/ACCESS Views	select libname, memname from dictionary.members where memtype = ‘ACCESS’ order by libname, memname;
SASHELP.VSCATLG	SAS CATALOGS	select libname, memname from dictionary.members where memtype = ‘CATALOG’ order by libname, memname;

SASHELP.VSLIB	SAS Libraries	select distinct(libname), path
		from dictionary.members
		order by libname;

SASHELP.VSTABLE	SAS Data Tables	select libname, memname from dictionary.members where memtype = ‘DATA’ order by libname, memname;
SASHELP.VSTABVW	SAS Data Tables and View	select libname, memname, memtype from dictionary.members where memtype =’VIEW’ or memtype =’DATA’ order by libname, memname;
SASHELP.VSVIEW	SAS Views	select libname, memname from dictionary.members where memtype = ‘VIEW’ order by libname, memname;

If you open the SASHelp library in the explorer window, right click on one of the SASHelp views you can also see the structure of the underlying data. The following figure shows the properties window for SASHelp.vmember; note that the columns are the same as we saw in the results of the DESCRIBE TABLE above:



Name	Si
Vengine	5.0
Verbmgr	17.0
Vextfl	5.0
Vformat	5.0
Vgopt	5.0
Video	53.0
Vidmsg	33.0
Viewpt	21.0
Vindex	5.0
Vlibnam	5.0
Vmacro	5.0
Vmember	5.0
Voption	5.0
Vrefcon	5.0
Vrememb	5.0
Vsacces	5.0
Vscatlg	5.0
Vslib	5.0
Vstable	5.0
Vstabvw	5.0
Vstyle	5.0
Vsview	5.0
Vtabcon	5.0
Vtable	5.0
Vtitle	5.0
Vview	5.0
Web	101.0

Column Name	Type	Length	Format	Informat	Label
libname	Text	8			Library Name
memname	Text	32			Member Name
memtype	Text	8			Member Type
dbms_memtype	Text	32			DBMS Member Type
memlabel	Text	256			Dataset Label
typemem	Text	8			Dataset Type
crdate	Number	8	DATETIME.	DATETIME.	Date Created
modate	Number	8	DATETIME.	DATETIME.	Date Modified
nobs	Number	8			Number of Physical C
obslen	Number	8			Observation Length
nvar	Number	8			Number of Variables
protect	Text	3			Type of Password Pr
compress	Text	8			Compression Routine
encrypt	Text	8			Encryption
npage	Number	8			Number of Pages
filesize	Number	8			Size of File
pcompress	Number	8			Percent Compression
reuse	Text	3			Reuse Space
bufsize	Number	8			Bufsize
delobs	Number	8			Number of Deleted O

Later we will see yet another way to see the structure of a Dictionary Table.

DICTIONARY TABLES IN MORE DEPTH

In this section we will look at some of the dictionary tables, first looking at the structure of the table and then talk about some of the ways it can be used. The purpose here is not to enumerate each of the columns of each of the tables, but to give a general overview of the tables. We will start with the tables that have the metadata about our data tables and views.

DICTIONARY.MEMBERS

The MEMBERS table contains information about all the library member types – tables, views, and catalogs.

```
create table DICTIONARY.MEMBERS
  (
   libname char(8) label='Library Name',
   memname char(32) label='Member Name',
   memtype char(8) label='Member Type',
   dbms_memtype char(32) label='DBMS Member Type',
   engine char(8) label='Engine Name',
   index char(32) label='Indexes',
   path char(1024) label='Path Name'
  );
```

This table is a general overview all of the libraries allocated in the current SAS session. It can be used to determine the contents of a library, or perhaps to determine the type of a specific member. For example you can use the ***engine*** column to determine which version of SAS was used to create the library member or perhaps verify the location of the file by looking at the ***path*** column. With SAS libraries, the ***dbms_memtype*** is blank since it is described in the ***memtype*** column. However, for external databases (e.g. SQL Server/ODBC database), the ***memtype*** column says "DATA", and the ***dbms_memtype*** column tells whether it is a database table (value of "TABLE") or a database view (value of "VIEW").

DICTIONARY.TABLES

The TABLES table contains more detailed information about the members SAS thinks are tables/datasets; remember, for some external data sources SAS considers DBMS views as tables.

```
create table DICTIONARY.TABLES
  (
   libname char(8) label='Library Name',
   memname char(32) label='Member Name',
   memtype char(8) label='Member Type',
   dbms_memtype char(32) label='DBMS Member Type',
   memlabel char(256) label='Dataset Label',
   typemem char(8) label='Dataset Type',
   crdate num format=DATETIME informat=DATETIME label='Date Created',
   modate num format=DATETIME informat=DATETIME label='Date Modified',
   nobs num label='Number of Physical Observations',
   obslen num label='Observation Length',
   nvar num label='Number of Variables',
   protect char(3) label='Type of Password Protection',
   compress char(8) label='Compression Routine',
   encrypt char(8) label='Encryption',
   npage num label='Number of Pages',
   filesize num label='Size of File',
   pcompress num label='Percent Compression',
   reuse char(3) label='Reuse Space',
   bufsize num label='Bufsize',
   delobs num label='Number of Deleted Observations',
   nlobs num label='Number of Logical Observations',
   maxvar num label='Longest variable name',
   maxlabel num label='Longest label',
   maxgen num label='Maximum number of generations',
   gen num label='Generation number',
   attr char(3) label='Dataset Attributes',
   indxtype char(9) label='Type of Indexes',
   datarep char(32) label='Data Representation',
   sortname char(8) label='Name of Collating Sequence',
   sorttype char(4) label='Sorting Type',
   sortchar char(8) label='Charset Sorted By',
   reqvector char(24) format=$HEX48 informat=$HEX48
     label='Requirements Vector',
   datarepname char(170) label='Data Representation Name',
   encoding char(256) label='Data Encoding',
   audit char(3) label='Audit Trail Active?',
   audit_before char(3) label='Audit Before Image?',
   audit_admin char(3) label='Audit Admin Image?',
   audit_error char(3) label='Audit Error Image?',
   audit_data char(3) label='Audit Data Image?'
  );
```

The TABLES table is commonly used to get some basic information about the table such as number of rows (***nobs***) and/or columns (***nvar***) in the table, or the table creation/modification date. With some external data sources (e.g. ODBC) the ***nobs*** column is set to zero since the SAS/Access driver cannot return the number of rows in a table. Also, as noted above some DBMS views are reported in the TABLES table, although the ***dbms_memtype*** can be used to determine whether we are looking at a DBMS table or view.

DICTIONARY.VIEWS

The VIEWS table contains a more limited set of metadata about the views available. Note that it is reporting on SAS views, not views in external DBMSs.

```
create table DICTIONARY.VIEWS
   (
    libname char(8) label='Library Name',
    memname char(32) label='Member Name',
    memtype char(8) label='Member Type',
    engine char(8) label='Engine Name'
   );
```

Besides letting you determine which views are available, by looking at the ***engine*** column you can determine if the view was created as an SQL view or a DATA step view. This table can be used to list all of the SASHelp views as will be shown later.

DICTIONARY.COLUMNS

The COLUMNS table provides detailed metadata about the columns in all of the tables and views.

```
create table DICTIONARY.COLUMNS
   (
    libname char(8) label='Library Name',
    memname char(32) label='Member Name',
    memtype char(8) label='Member Type',
    name char(32) label='Column Name',
    type char(4) label='Column Type',
    length num label='Column Length',
    npos num label='Column Position',
    varnum num label='Column Number in Table',
    label char(256) label='Column Label',
    format char(49) label='Column Format',
    informat char(49) label='Column Informat',
    idxusage char(9) label='Column Index Type',
    sortedby num label='Order in Key Sequence',
    xtype char(12) label='Extended Type',
    notnull char(3) label='Not NULL?',
    precision num label='Precision',
    scale num label='Scale',
    transcode char(3) label='Transcoded?'
   );
```

This table is commonly used to determine if a column exists (see examples below). It can also be used to verify the column type and format.

When looking at the above table layouts we see some common 'key' columns - particularly ***libname*** and ***memname***. By joining the above three tables on these key columns it is possible to provide a custom report with your data dictionary.

In addition to the dictionary tables that describe your data, there are tables, which describe your SAS session. Let us look at some of these.

DICTIONARY.OPTIONS

The OPTIONS table has an entry for each of the SAS options.

```
create table DICTIONARY.OPTIONS
   (
    optname char(32) label='Option Name',
    opttype char(8) label='Option type',
    setting char(1024) label='Option Setting',
    optdesc char(160) label='Option Description',
    level char(8) label='Option Location',
    group char(32) label='Option Group'
   );
```

DICTIONARY.TITLES

The TITLES table has an entry for each title and footnote line currently in effect. See the example below on how to use this table to save the current titles, and then restore them after running a report.

```
create table DICTIONARY.TITLES
   (
    type char(1) label='Title Location',
    number num label='Title Number',
    text char(256) label='Title Text'
   );
```

DICTIONARY.EXTFILES

The EXTFILES table has an entry for each external file registered `(filerefs)` in the session.

```
create table DICTIONARY.EXTFILES
   (
    fileref char(8) label='Fileref',
    xpath char(1024) label='Path Name',
    xengine char(8) label='Engine Name'
   );
```

This table is useful when you want to document external data sources/output from a run. Be aware that SAS has a number of `filerefs` it uses that you do not see in the explorer window; all of these SAS generated `filerefs` begin with #LN, so you can easily filter them out.

Now that we have looked at a few of the dictionary tables, let us look at some examples of how they could be used.

LOOKING IT UP IN THE DICTIONARY

You have developed an outstanding report that everyone wants included into their SAS runs. The problem is your report sets new title text and some users want their original titles after your report runs. Well, wrap your report in a macro and add two simple data steps, one before and one after your report (Listing 2).

LISTING 2 - RESETTING TITLES

```
%macro myreport;
/* use the dictionary to save all of the old titles and footnotes */
/* force the new title for demonstration purposes */
TITLE1 "The Original SAS Title1";
TITLE2 "The Original SAS Title2";

/* the SAS view sashelp.vtitle has the current titles and footnotes */
DATA __oldtitles;
    set sashelp.vtitle;
RUN;

/* add a new title and run the report */
TITLE1 "This is a GREAT REPORT";
TITLE2 "With TWO Title Lines";
PROC SORT DATA=sashelp.shoes OUT=shoes;
     BY region product;
RUN;

PROC PRINT noobs DATA=shoes;
BY region product;
ID region product;
SUMBY product;
VAR sales returns;
RUN;

/* restore the titles from the dataset */
DATA null;
     SET __oldtitles END=done;
     length title $12.;
/* a title can be 200 chars, allow 2 extra for the quotes */
     length newtext $202.;
/* convert the title number to a char string */
     anum = compress(put(number, 2.));
/* put a single quote at the beginning of the text */
     newtext = "'" || text;
/* find the end of the text and add another quote */
     l = length(newtext);
     substr(newtext, l+1,1) = "'";
/* for titles, type = 'T' */
     if  type = "T"
     then
       do;
        /* first make the string  TITLE1 etc */
        title = "TITLE" || anum;

        /* now add the text so we end up with TITLE1 'This is the TEXT' */
        titletext = title || newtext;
        /* and put it out to the symbol table */
        call symput(title, titletext);
       end;
     /* repeat for footnotes */
     else if  type = "F"
     then
       do;
        /* first make the string  FOOTNOTE1 etc */
        title = "FOOTNOTE" || anum;
        /* now add the text so we end up with FOOTNOTE1 'This is the TEXT' */ titletext =
title || newtext;
        /* and put it out to the symbol table */
        titletext = title || newtext;
        call symput(title, titletext);
     end;

/* create a counter variable */
     if done
     then
       do;
         call symput("titlevars", put(_n_, 2.));
       end;
run;

%* now, pump out the old titles and footnotes ;
%do i = 1 %to &titlevars;
&&title&i;
%symdel title&I;
%end;
%symdel titlevars;
%mend myreport;
%myreport
```

Listing 2 can be viewed as a rough template for saving and restoring most settings in the SAS session. First, open the appropriate view (using a WHERE clause if appropriate) to save the current values. Set and use the new values. Finally, in another data step create the macro variables which are used to reset the original values.

A common use of the Dictionary Tables/Views is to identify and/or enumerate tables and columns available in the session. Let's take a quick look at viewing some of these metadata.

FIND THAT COLUMN

The Dictionary View SASHELP.VCOLUMN has the list of all the columns in all of the tables and views in your current SAS session. We can use this table to create a list of columns that are in multiple tables. First, we could do it with a simple listing (note the WHERE clause that excludes the MAPS and SASHELP libraries):

LISTING 3 - SELECTING COLUMNS IN MULTIPLE TABLES (1)

```
proc sql;
select name, count(*) as occurrences
from sashelp.vcolumn
where libname not in (‘MAPS’, ‘SASHELP’)
group by name
having count(*) > 1
order by name
;
quit;
```

1. Get a count for each column

2. Exclude MAPS and SASHELP libraries

3. Only keep names that occur more than once

Knowing the columns with multiple occurrences is useful, but it would be more useful to know in which tables the columns belong. With SAS this is where it gets interesting since there are usually a number of ways to solve the problem. One way is to create tables with the column names and use this to get more data on the columns:

LISTING 4 - SELECTING COLUMNS IN MULTIPLE TABLES (2)

```
proc sql;
create table MultiColCnt as
select name, count(*) as occurrences
from sashelp.vcolumn
where libname not in (‘MAPS’, ‘SASHELP’)
group by upcase(name)
having count(*) > 1
order by name
;

Create table MultiColCnt1 as
select distinct name from MultiColCnt
;

create table MultiColTables as
select v.name, v.libname, v.memname, v.type, v.length, v.label, v.format
from sashelp.vcolumn as v, MultiColCnt1 as c
where c.name = v.name
and v.libname not in (‘MAPS’, ‘SASHELP’)
order by  v.name, v.libname, v.memname
;
quit;
```

1. Create a new table

2. Get a list of unique names

3. Join back to the full table and keep the selected metadata on the column names that have multiple occurrences

Now, suppose you want to apply a consistent label and format to a specific column that may be found in multiple tables. Well, the Dictionary Tables can help you locate the tables with the column and check whether the label and format need to be changed. For those that need changing, you can apply your favorite proc to change them. The following example uses a data step to select the tables that need changing then uses SQL to change them.

LISTING 5 - CHANGING THE LABEL AND FORMAT OF A SELECTED COLUMN

```
%macro ChangeLabelFormat(colName,   /* column to change */
                          newLabel,  /* label to apply   */
                          newFormat) /* format to apply  */
        ;
%local tblsToChange;
%local i;
%let colName = %upcase(&colName);
%* first locate the tables with the column;
%let tblsToChange = 0;
data _null_;
set sashelp.vcolumn
   (where=(upcase(name) EQ "&colName" AND
           (label NE "&newLabel" OR format NE "&newFormat") )
    keep=libname memname name label format)
     end=done;
chgLib   = compress('ChgLib'   || put(_n_, 7.));
chgTable = compress('ChgTable' || put(_n_, 7.));
call symput(chgLib,   trim(libname));
call symput(chgTable, trim(memname));
if done
    then  call symput('tblsToChange', put(_n_, 7.));
run;

%* if any cols need changing, use SQL to change them;
%if &tblsToChange NE 0
%then
  %do;
    %do i = 1 %to &tblsToChange;
      %let tblToChange = %cmpres(&&chgLib&i...&&chgTable&i);
      proc sql;
           alter table &tblToChange
           modify &colName
           label="&newLabel"
           format=&newFormat
      ;
      quit;
    %end;
  %end;
%mend;
```

1. **Filter on the column name and keep if the label or format is does not match the new format/label**

2. **Create variables indexed by the 'record number'**

3. **Create macro variables with the library and table name**

4. **Tell how many changes needed**

We saw earlier the use of DESCRIBE TABLE to print the table structure to the SAS LOG. Although it gives us the information we need it is not always useful. Similarly we saw how we can also see the structure by looking at the properties of the SASHelp view in the explorer window. Again, it gives us the information, but it is not always useful. What if we want to capture these metadata in a data step, or create a more user-friendly printed output? SAS Dictionary Tables to the rescue

LISTING 8 – CAPTURING THE DICTIONARY METADATA

```
     %macro describeTable(table=tables);
     %let table = %upcase(&table);
     PROC SQL;
     CREATE TABLE _&table as
     SELECT *
```

1. **Get all of the columns in the DICTIONARIES table for the specific Dictionary Table**

```
FROM DICTIONARY.DICTIONARIES
WHERE MEMNAME = "&table"
ORDER BY varnum
;
%let rows = &sqlOBS;
QUIT;
%if &rows > 0
%then
  %do;
ODS RTF BODY="C:\TEMP\&table..RTF";
title1 "Columns in the &table Dictionary Table";
proc print noobs label data=_&table (drop=memname memlabel);
run;
ODS RTF CLOSE;
  %end;
%else
  %do;
     %put Dictionary Table &table does not exist;
  %end;
%mend;
%describeTable(table=tables);
```

2. &sqlOBS will tell us how many rows were returned

3. If the member name we requested was found print the results

4. If the member name we requested was not found put out a message

CONCLUSION

Dictionary tables are an essential part of every SAS developer's toolbox. In the past, Michael Davis at SUGI 26 presented a paper that had a good overview and review of other papers showing different uses of the Dictionary Tables. Also, Pete Lund at SUGI 26 had an excellent paper on the use of the Dictionary Tables to document a project. This paper has provided a brief introduction to Dictionary Tables that I hope has helped you to better understand these concepts and thus they become more easily accessible to you as a useful tool.

REFERENCES

Ravi, Prasad. "Renaming All Variables in a SAS Data Set Using the Information from PROC SQL's Dictionary Tables"
SUGI 28 Seattle SAS Users Group International Proceedings. March 30-April 2, 2003.
http://www2.sas.com/proceedings/sugi28/118-28.pdf

Beakley, Steven & McCoy, Suzanne. "Dynamic SAS Programming Techniques, or How NOT to Create Job Security"
SAS Conference Proceedings SUGI 29. May 2004.
http://www2.sas.com/proceedings/sugi29/078-29.pdf

Lund, Pete. "A Quick and Easy Data Dictionary Macro"
SAS Conference Proceedings: SUGI 27. April 2002.
http://www2.sas.com/proceedings/sugi27/p099-27.pdf

Davis, Michael. "You Could Look it Up: An Introduction to SASHELP Dictionary Views"
SAS Conference Proceeding: SUGI 26. April 2001.
http://www2.sas.com/proceedings/sugi26/p017-26.pdf

Dilorio, Frank & Abolafia, Jeff. "Dictionary Tables and Views: Essential Tools for Serious Applications"
SAS Conference Proceedings: SUGI 29. May 2004.

http://www2.sas.com/proceedings/sugi29/237-29.pdf

Eberhardt, Peter & Brill, Ilene. "How Do I Look It Up If I Cannot Spell It: An Introduction to SAS Dictionary Tables"
SAS Conference Proceedings: SUGI 31. March 2006.
http://www2.sas.com/proceedings/**sugi**31/259-31.pdf

ABOUT THE AUTHOR

Peter Eberhardt is SAS Certified Professional V8, SAS Certified Professional V6, and SAS Certified Professional - Data Management V6. In addition his company, Fernwood Consulting Group Inc. is a SAS Alliance Partner. Peter is a regular speaker at SAS Global Forum, SESUG and NESUG as well as at local user groups in Canada, the US and the Caribbean. If you have any questions or comments you can contact Peter at:

Fernwood Consulting Group Inc.,
288 Laird Dr.,
Toronto ON M4G 3X5
Canada
Voice: (416)429-5705
e-mail: peter@fernwood.ca

SAS, and SAS Alliance Partner are registered trademarks of SAS Institute Inc. in the USA and other countries. Other brand and product names are registered trademarks or trademarks of their respective companies.