Discovering the Power of SAS Metadata: An Introduction to Dictionary Tables and Views

Frank DiIorio
CodeCrafters, Inc.
Philadelphia PA

Agenda

- What are Dictionary Tables?
- Resources
- Accessing them efficiently
- Commonly used tables
- Examples of use

What are Dictionary Tables?

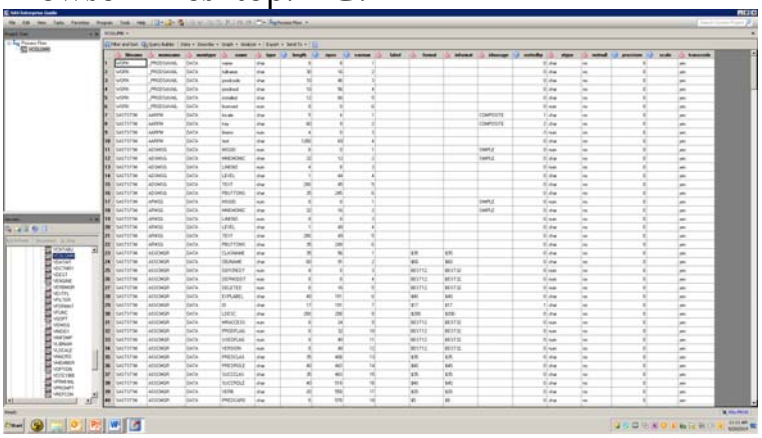
- Metadata describing activity in a SAS program
  - Available since SAS V6.07
  - Columns and tables added with new releases
- Metadata?
  - Generally: "data about data"
  - SAS-centric: data describing the environment in which a program is executing
  - Can be either ignored or exploited
  - But helpful to know about their presence and contents ("One thing you can never say is that your haven't been told." Dr. Krakower, *The Sopranos)*

What are Dictionary Tables? (cont.)

- Dictionary Tables:
  - Describe current settings and resources used by a program
  - Are *always* created at the start of a session (not optional!)
  - Are constantly and automatically updated during the session
  - Are read-only (they're *modified* by activity in your program, but you cannot *directly* change their content)
  - Cannot have structure changed (add/delete variable, change format, etc.)
  - Have 1 or more views defined in SASHELP

Resources

- Conference papers (LexJansen.com)
- Forums: SAS Community, SAS-L (better for usage q's)
- SAS online documentation? *Not very helpful.*
- Browse in desktop/EG:



Macro to Describe Tables

```
%macro DictInfo;
proc sql noprint;
     select distinct memname,         count(distinct memname)
     into   :tbl separated by ' ',    :ntbl
     from dictionary.dictionaries ;

     select memname,                  count(distinct memname)
       into :view separated by ' ',   :nview
     from dictionary.views
     where memname like "V%" & libname = 'SASHELP'
           & memtype = 'VIEW' ;

     %do i = 1 %to &ntbl.;
         %let item = %scan(&tbl., &i.);
         describe table dictionary.&item.;
     %end;

     %do i = 1 %to &nview.;
         %let item = %scan(&view., &i.);
         describe view sashelp.&item.;
     %end;

quit;
%mend;
```

Partial Macro Output

For each table and view …

```
NOTE: SQL table DICTIONARY.COLUMNS was created like:

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(16) label='Column Format',
   informat char(16) label='Column Informat',
   idxusage char(9) label='Column Index Type'
  );
```

Somewhat helpful, but doesn't give much of an idea of content, what values are stored, case-sensitivity, etc.

Using the Tables

- SQL
  - Access **Tables** with LIBNAME DICTIONARY (yes, that's a 10 character LIBNAME)
  - Access **Views** with LIBNAME SASHELP
  - Generally, Tables are accessed faster than views
- Elsewhere
  - No access to DICTIONARY.*table*, just SASHELP.*view*
- Anywhere
  - Faster access if you don't modify columns that are part of a table/view's index

Commonly Used Tables

- 32 tables in Version 9.4
- Nice, but not realistic, to know everything about all of them. High-altitude view is good for starters.
- Let's look at some of the more frequently used tables
- We'll present examples of use later

Commonly Used Tables

Table/View usage in Rho macro library

Table/View	N
COLUMNS	63
TABLES	37
FORMATS	15
MACROS	10
EXTFILES	9
LIBNAMES	8
OPTIONS	5
MEMBERS	3
CATALOGS	3
TITLES	2
STYLES	2
VIEWS	1

Commonly Used Tables

Table	Comments
dictionaries	"data about data about data"
tables	Non-SAS member names can be problematic
columns	Case-sensitivity! Compare w/ CONTENTS output
options	Beware of OFFSET!
titles	Titles *and* footnotes
formats	System and user-defined
xattrs	"Extended attributes?!" *Next slide. Be patient…*
macros	As with OPTIONS, beware of OFFSET.
extfiles	Allocated automatically by SAS; FILENAME; XPT; other
functions	An example of "Why would I ever need this?"

Table Reference

Briefly: Extended Attributes

- New in V9.4
- Create user-defined metadata at dataset and/or variable level
- Add/Remove/Update using PROC DATASETS:

```
modify dataset;
   xattr add ds attrib1=value1;
   xattr add var var1(attrib1=value1)
                 var2(attrib1=value1 attrib2=value2);
run;
```

- DICTIONARY.XATTRS

```
libname memname name xattr  xtype  xoffset xvalue
X       TESTX        sortBy char   0       study subject
X       TESTX        scope  char   0       internal
X       TESTX   v1   role   char   0       char demo
X       TESTX   v2   long   char   0       really really really
X       TESTX   v2   long   char   200     ly long
X       TESTX   v2   int'l  num    0       1
```

Examples of Use

As we go through examples keep in mind what's needed to use the Tables and Views effectively:

- Knowledge of content (values, case-sensitivity, possible quirks) and granularity
- Use is most effective via PROC SQL

These are simple, "proof of usefulness" examples. Think how these code snippets can be "macrotized" and made into powerful, general-use applications.

1: List/Count of Datasets

Any number of ways to do this!

```
proc sql noprint;
  select memname into :datasets separated by ' '
  from dictionary.members
  where memtype = 'DATA' & libname = 'PROD'
  ;
  %let datasetsN = &SQLobs.;
quit;


%do idx = 1 %to &datasetsN.;
    %let dsn = %scan(&datasets., &idx.);
    proc print data=prod.&dsn.(obs=5);
    title "First 5 obs from PROD.&dsn.";
    run;
%end;
```

Table Reference

1: List/Count of Datasets (cont.)

"Consider the source" →


```
proc sql noprint;
  select memname into :datasets separated by ' '
  from dictionary.members
  where memtype = 'DATA' & libname = 'PROD'
  ;
  %let datasetsN = &SQLobs.;
quit;

&datasets = 'First sheet$' Second$
&datasetsN = 2
```

Count is correct but `%scan` would create incorrect DSN's (`'First` and `sheet$'`).This is when knowledge of quirks/rules/etc. becomes important.

Table Reference

1: List/Count of Datasets (cont.)

```
proc sql noprint;
  select memname into :datasets separated by '~'
  from dictionary.members
  where memtype = 'DATA' & libname = 'PROD'
  ;
  %let datasetsN = &SQLobs.;
quit;

&datasets = 'First sheet$'~Second$
&datasetsN = 2
```

We can parse &datasets reliably once we add ~ as the 3rd argument to %scan

Table Reference

2: Variables Not Compatible with XPT

```
data PROD.FINAL;
length 'has gap'n LongVarName clean $1 longlbl 3
       TooLong $1000;
label longlbl = 'This label exceeds the 40 char limit!!!!!!';
run;

proc sql noprint;
     create table notValid as
     select name, label, length
     from dictionary.columns
     where libname = 'PROD' and memname = 'FINAL' &
                    (index(trim(name), ' ') > 0 |
                     length(name)     > 8 |
                     length(label)    > 40 |
                     length           > 200
                    ) ;
```

Table Reference

3: Identify User-Written Formats

[1] Simply display them:

```
proc print data=sashelp.vformat(where=(source='C'));
```

[2] Save for later use (possible in PROC FMT CNTLIN dataset):

```
%let userFmtN = 0;
proc sql noprint;
     create table userFmt as
     select *
     from dictionary.formats
     where fmtType = 'F'
     ;
     %let userFmtN = &SQLobs.;
quit;
```

Table Reference

4: Option Capture, Reset, Restore

Macro best practice: don't overwrite user's settings

```
* Capture ;
proc sql noprint;
select setting into :OPTcent
   from dictionary.options
   where optname="CENTER";
select catT('ls=', setting) into :OPTlinesize
   from dictionary.options
   where optname='LINESIZE';  /* Full name, not alias (LS)! */
quit;

* Reset ;
options linesize=200 center=0;
... procs go here ...

* Restore ;
options &OPTcent. &OPTlinesize.;
```

Table Reference

5: Count Observations in a Dataset

Just one of *many* ways to do this:

```
%macro countobs(data=, count=_count_);
       %global &count.;

       %let data = %upcase(&data.);
       %if %index(&data., .) > 0 %then %do;
           %let libname = %scan(&data., 1, .);
           %let memname = %scan(&data., 2, .);
           %end;
           %else %do;
                 %let libname = WORK;
                 %let memname = &data.;
                 %end;
       proc sql noprint;
            select nobs into :&count
            from dictionary.tables
            where libname="&libname." & memname="&memname."
                  & memtype="&data." ;
       quit;
       %if &sqlobs. = 0 %then %let &count. = -1;
       %put COUNTOBS: Count variable &count. [&&&count];
%mend;
```

Table Reference

6: Display Global Macro Variables

More work, but more readable, than `%put _global_;`

```
proc sql noprint;
     create table _macvars_ as
     select name, offset, scope, value
     from dictionary.macros
     where offset=0 and scope='GLOBAL'
     order by name
     ;
quit;

data _null_;
     set _macvars_;
     put name @20 value $80.;
run;
```

Table Reference

7: Identify Conflicting Attributes

Like-named variables in a library should have identical type and length.

```
proc sql noprint;
     create table tl_discrepancies as
     select distinct catT(type, length) as tl, name
     from dictionary.columns
     where libname="PROD"
     group by name
     having count(distinct tl) > 1
     ;
quit;
```

Table Reference

Wrapping Up …

- Dictionary tables are: powerful; predictable; available by default
- Using them requires knowledge of: SQL, table structure
- Most effective use is via SQL, saving the information as a table or macro variable(s)
- The Tables are an integral part of general-purpose macros (consider the potential for expansion of Example 7, previous slide)
- Uses are "only limited by your imagination"

Thanks for Coming!

Your comments and questions are valued and encouraged:

FrankDiIorio@gmail.com

Note: CodeCraftersInc.com web site and email addresses have been terminated (Phase I of my Ease into Retirement plan).