

Data About Data: An Introduction To Dictionary Views And Tables

Frank C. DiIorio, ASG, Inc.

One of the most useful, and least heralded, additions to Version 6.07 of the SAS System is dictionary views and tables. They contain a wealth of meta-data, or data about data, about SAS datasets, catalogs, system options, and external files. This information can be used for a wide variety of data management and reporting applications.

One of their most appealing features is that they are created and maintained automatically - no programmer intervention or special action is necessary for their use. This paper briefly describes the organization of the different tables and views. It also presents a few examples of their use.

Organization

There are two ways to access dictionary information. They are distinguished by whether you access them as tables (with PROC SQL) or views (with other PROCs and DATA steps).

PROC SQL accesses the information as a collection of tables with the reserved LIBNAME of DICTIONARY. To use the COLUMNS table, for example, enter the following:

```
proc sql;
select * from dictionary.columns;
```

The dictionary tables are read-only. You can use them just as you would any other SAS dataset or table in PROC SQL. SAS creates and maintains eight dictionary tables (described below).

Non-SQL access is via a collection of views referenced by the reserved LIBNAME of SASHELP. For example, to use the VCOLUMN view you could code:

```
proc print data=sashelp.vcolumn;
where libname = "PROJECT";
```

The views, like the dictionary tables, are read-only. You can use them as you would any other view. SAS creates and maintains 14 dictionary views (described below). Some are essentially duplicates of the dictionary table, while others are column subsets designed for faster access.

The organization of the tables and views is outlined in Figure 1. Space constraints prevent a complete discussion of the contents of each entity. For more information, refer to "SAS Technical Report P-222: Changes and Enhancements to Base SAS Software."

Examples of Use

The variety of uses for dictionary data is vast. Two macros using the data are presented here. One gives a flavor for the scope of the data, the other is a simple but useful application.

Example 1. The first macro is shown below. It creates a series of macro variables (VIEW1 - VIEWn) containing the names of the Views found in SASHELP. Then it runs the CONTENTS and PRINT procedures on each view. This gives an idea of the contents of the view (variable names, labels, and so on) as well as a listing of a limited number of observations from the view. Studying the output created by SHOWVIEW is a good orientation to the views.

```
%macro showview(obs=10);
data _null_;
set sashelp.vsvview(keep=memname)
end=eof;
call symput('view',
||left(put(_n_,2.)),
memname);
if eof then call symput('nview',
put(_n_,2.));
run;

%do i = 1 %to &nview.;
proc contents
data=sashelp.&&view&i.;
title "View SASHELP.&&view&i.";
run;

proc print
data=sashelp.&&view&i.
(obs=&obs.);
title2 "First &obs. obs.";
run;
%end;
%mend;

%showview(obs=25);
```

Example 2. Projects are often physically arranged into distinct operating system files or directories. All data for a stress management survey, for example, may be stored in a directory called "stress". The next macro uses the VCOLUMN view and the REPORT procedure to produce a listing of all variables in all datasets in a directory.

LIB_REPT identifies all datasets in the directory and lists them sorted by variable name within dataset name. The dataset name, variable name, type, length, and format are displayed in three columns across the page. This is a simple and relatively pain-free way to

document the data in a study. The macro can be readily extended to display the data in other formats (e.g., variable name sorted first, only duplicate variable names).

```
%macro lib_rept(libname=WORK);
%let libname=%upcase(&libname.);
options pageno=1;
proc report panels=3 pspace=2
      spacing=1 headline
      nocenter nowindows
      data=sashelp.vcolumn
      (where=(libname="%&libname."););
column memname name type length
      format ;
define memname / order ;
define name / 'Name' width=8;
define type / 'Type' width=4;
define length / 'Len.' width=4;
define format / 'Format' width=10;
title 'Library &Libname. - Sorted
      by Dataset Name, Var Name';
run;
%mend lib_rept;

%lib_rept(libname=input
```

The output from the invocation of LIB_REPT is contained in Figure 2. The report is informative: you can get a good overview of the library's contents with a minimum amount of coding effort. And since the

report is based on a view, when the data organization changes the view will be automatically updated. The next time the report is run information on the new, updated data sets will be present.

Conclusion

This paper has briefly described the nature of dictionary data and outlined some of its applications. The ease of use and dynamic nature of the data should make it appealing to end users and, especially, to developers.

Contact Information

Your questions and comments are valued and encouraged. Contact the author at:

102 Westbury Drive
Chapel Hill, NC 27516-9154

work: 919-467-0505
message: 919-942-2028

Figure 2: Sample Output from LIB_REPT Macro

Library INPUT, Sorted by Dataset Name, Variable Name									
Member Name	Name	Type	Len.	Format	Member Name	Name	Type	Len.	Format
ANSWER	EMP_NO	num	3		ENT	COMPANY	char	4	\$COMPANY.
	Q	num	3			ENT_CODE	char	4	\$ENT_COD.
	A	num	3			ENT_CODE	char	3	\$ENT_COD.
	A2	num	3			CL_SCOPE	num	3	
	DATE	num	5	DATE7.		CO_SALES	num	3	
	COMPANY	char	4	\$COMPANY.		EMPINCO	num	3	
EMPANS	ENT_CODE	char	4	\$ENT_COD.	BU_SALES	num	3		
	_NOTUSED	num	8			NUMEMPBU	num	3	
	EMP_NO	num	3			PRACTICE	num	3	
	Q	num	3			AREA	num	3	
	A	num	3			COMPANY	char	4	\$COMPANY.
	A2	num	3			ENT_NAME	char	20	
EMPLOY	COMPANY	char	4	\$COMPANY.	MARKET	CON_NAME	char	17	
	ENT_CODE	char	4	\$ENT_COD.		FOUND	num	8	DATE7.
	_NOTUSED	num	8			IND	num	8	
	SEX	num	3			A2	num	3	
	AGE	num	3			EMP_NO	num	3	
	YRCO	num	3			Q	num	3	
EMPLOY	YRJOB	num	3			A	num	3	
	LEV	num	3			COMPANY	char	4	\$COMPANY.
	FUNC	num	3			ENT_CODE	char	4	\$ENT_COD.
	ETH	num	3						
	TYPE	num	3						
	FPART	num	3						
EMPLOY	OTHQU	num	3						
	EMP_NO	num	3						
	ENG_DATE	num	5	DATE7.					
	SEX	num	3						
	AGE	num	3						
	YRCO	num	3						
EMPLOY	YRJOB	num	3						
	LEV	num	3						
	FUNC	num	3						
	ETH	num	3						
	TYPE	num	3						
	FPART	num	3						
EMPLOY	OTHQU	num	3						
	OTHQU	num	3						

Figure 1: Overview of Dictionary Tables and Views