

PROC SQL Dictionary Tables - A Great Source of Information about Data Sets

Wendih S. Yao, PAREXEL International Corp, Northbrook, IL

ABSTRACT

PROC SQL provides the capability to retrieve information about all libraries and external files that are allocated in a session. This information will be returned as what appears to be rows in tables and can be accessed from other procedures by creating views of the tables listed in this paper. It can be accessed by PROC SQL with views of the tables or with the table names themselves.

INTRODUCTION

PROC CONTENTS already provides sufficient information about data sets. In this paper, PROC SQL dictionary tables are introduced as an alternative way to get information about data sets, catalogs, libnames, and other SAS objects.

Dictionary tables are a new alternative in a from-list. They are READ-ONLY objects and can not be updated by PROC SQL. These dictionary tables can only be accessed under PROC SQL. However, SAS provides a set of SQL views under the SASHELP library which allows access from normal DATA steps or procedures. They are excellent for finding out information such as the number of observations in a data set, how many data sets are in the library, and what are the data set names in the library.

WHAT'S IN THE DICTIONARY

The information that can be retrieved in PROC SQL dictionary tables depends on your programming needs. These are result sets of information about the current SAS environment (ie, all data sets or other SAS objects). They can be grouped by the following:

DICTIONARY.CATALOGS: contains SAS catalog information.

libname	char(8)	label='Library Name',
memname	char(8)	label='Member Name',
memtype	char(8)	label='Member Type',
objname	char(8)	label='Object Name',
objtype	char(8)	label='Object Type',
objdesc	char(40)	label='Object Description',
modified	char(8)	label='Date Modified',
alias	char(8)	label='Object Alias',

DICTIONARY.COLUMNS: contains SAS data set variable information.

libname	char(8)	label='Library Name',
memname	char(8)	label='Member Name',
memtype	char(8)	label='Member Type',
name	char(8)	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(40)	label='Column Label',
format	char(16)	label='Column Format',
informat	char(16)	label='Column Informat',
idxusage	char(9)	label='Column Index Type',

DICTIONARY.EXTFILES: contains specific external file information.

fileref	char(8)	label='File Reference',
xpath	char(80)	label='Path Name',
xengine	char(8)	label='Engine Name',

DICTIONARY.INDEXES: contains data set index information.

libname	char(8)	label='Library Name',
memname	char(8)	label='Member Name',
memtype	char(8)	label='Member Type',
idxname	char(8)	label='Index Name',
idxusage	char(9)	label='Column Index Type',
name	char(8)	label='Column Name',
indxpos	num	label='Position of Column in Concatenated Key',
nomiss	char(3)	label='Nomiss Option',
unique	char(3)	label='Unique Option',

DICTIONARY.MEMBERS: contains all SAS data sets or other object information.

libname	char(8)	label='Library Name',
memname	char(8)	label='Member Name',
memtype	char(8)	label='Member Type',
engine	char(8)	label='Engine Name',
index	char(8)	label='Indexes',
path	char(80)	label='Path Name',

DICTIONARY.OPTIONS: contains current SAS session options.

optname	char(20)	label='Session Option Name',
setting	char(200)	label='Session Option Setting',

optdesc char(80) label='Option Description'.

DICTIONARY.TABLES: contains detail data sets information.

libname char(8) label='Library Name',
memname char(8) label='Member Name',
memtype char(8) label='Member Type',
memlabel char(40) label='Data Set Label',
typemem char(8) label='Data Set Type',
crdate num label='Date Created',
modate num label='Date Modified',
nobs num label='Number of 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',
reuse char(3) label='Reuse Space',
bufsize num label='Buffer size',
delobs num label='Number of Deleted
Observations',
indxtype char(9) label='Type of Indexes'.

DICTIONARY.VIEWS: contains data views information.

libname char(8) label='Library Name',
memname char(8) label='Member Name',
memtype char(8) label='Member Type',
engine char(8) label='Engine Name'.

DICTIONARY.MACROS (new to SAS 6.11): contains macro variables information.

scope char(9) label='Macro Scope',
namechar(8) label='Macro Variable Name',
offset num label='Offset into Macro Variable',
value char(200) label='Macro Variable Value'.

DICTIONARY.TITLES (new to SAS 6.11): contains titles information.

type char(1) label='Title Location',
number num label='Title Number',
text char(200) label='Title Text'.

SASHELP VIEWS vs. DICTIONARY TABLES

There is an SQL view in SASHELP for each of the above virtual tables, which allows them to be accessed from any SAS procedure. The corresponding views with dictionary tables are as follows:

SASHELP.VCATALG - dictionary.catalogs
SASHELP.VCOLUMN - dictionary.columns
SASHELP.VEXTFL - dictionary.extfiles
SASHELP.VINDEX - dictionary.indexes
SASHELP.VMACRO - dictionary.macros

SASHELP.VMEMBER - dictionary.members
SASHELP.VOPTION - dictionary.options
SASHELP.VTABLE - dictionary.tables
SASHELP.VTITLE - dictionary.titles
SASHELP.VVIEW - dictionary.views

The following additional SASHELP views information can also be retrieved by dictionary MEMBERS table:

SASHELP.VSACCES:

```
select libname, memname
from dictionary.members
where memtype='ACCESS'
order by libname, memname;
```

SASHELP.VSCATLG:

```
select libname, memname
from dictionary.members
where memtype='CATALOG'
order by libname, memname;
```

SASHELP.VSLIB:

```
select distinct(libname), path
from dictionary.members
order by libname;
```

SASHELP.VSTABLE:

```
select libname, memname
from dictionary.members
where memtype='DATA'
order by libname, memname;
```

SASHELP.VSTABVW:

```
select libname, memname, memtype
from dictionary.members
where memtype in ('VIEW', 'DATA')
order by libname, memname;
```

SASHELP.VSVIEW:

```
select libname, memname
from dictionary.members
where memtype = 'VIEW';
```

EXAMPLES

1. Suppose a list of patients with serious adverse events (SAE) is to be generated. Before printing the results, the data set containing the patients with SAEs can be checked to see if it is empty.

First, create a data set, SAE, that contains all serious adverse events:

```
proc sql;
  create table SAE as
  select *
  from ADVERSE
  where SERIOUS='YES';
```

Second, check number of observations in the SAE data set and store the result into a macro variable EXIST:

```
reset noprint;

select NOBS into :EXIST
from dictionary.tables
where memname='SAE';
quit;
```

If EXIST is greater than 0, i.e., the SAE data set is not empty, a list of serious adverse events will be produced. Otherwise, generate the listing with an appropriate note to indicate that there was no serious adverse event in the study.

```
%if &EXIST %then %do;
  data _null_;
    ...
    SAS statements to list serious
    adverse events.
    ...
  run;
%end;

%else %do;
  data _null_;
    #5 @10 put 'There are no records
    of serious adverse events in the
    study.';
  run;
%end;
```

2. This example demonstrates how to get the number of records per patient for each data set in the data library. You don't need to know how many data sets are in the library or the names of these data sets. All you need is the location where the data library resides.

Define a SAS data library.

```
libname MYLIB 'sas-data-library';
```

Create a data set, TABNAME, using LIBNAME and MEMTYPE to subset only the data sets in the MYLIB.

```
proc sql;
  create table TABNAME as
  select MEMNAME
  from dictionary.tables
  where libname='MYLIB' and
  memtype='DATA';
quit;
```

Note: The SQL procedure sets up an automatic macro variable, SQLOBS, after it executes each statement. This macro variable contains the number of rows processed by an SQL procedure statement. In this case, SQLOBS contains the number of data sets in the MYLIB library.

We can now process the frequency count for each data set.

```
%macro FREQCNT;
  %do I = 1 %to &SQLOBS;
    data _null_;
      set TABNAME (firstobs=&I obs=&I);
      call symput ('MODULE', MEMNAME);
    run;

    proc sql;
      select PATNO, count(*) as NUMOBS
      from MYLIB.&MODULE
      group by PATNO
      order by PATNO;
    quit;
  %end;
%mend FREQCNT;
%FREQCNT;
```

CONCLUSION

One of the advantages of using SQL dictionary tables is being able to write concise SAS code. With SQL dictionary tables, we can automatically process all members in a SAS data library, information on all variables in SAS data sets, the current values of SAS options, the current values of SAS macro variables, the titles and footnotes in effect as well as information on external files allocated to the session.

REFERENCE

SAS Institute: *SAS Guide to the SQL Procedure – Usage and Reference*, Version 6, First Edition, SAS Institute Inc., Cary, NC.

Paul Kent: *An SQL Tutorial – Some Random Tips*, SUGI 21 Proceedings, p 237-241.

Phil Mason: *In the Know...SAS Tips and Techniques From Around the Globe*, SAS Institute Inc., Cary, NC (1996).

CONTACT

Wendih Yao
Manager, Statistical Programming
PAREXEL International, Inc.
Two Northbrook Place
60 Revere Dr, Suite 200
Northbrook, IL 60062

Phone: (847) 509-1595 x 6136
Fax: (847) 509-1691
E-mail: wendih.yao@parexel.com