How Are All of These Tables Related? - Relational Database Map - RDB_MAP.SAS

Eric Losby, HealthCare COMPARE Corp., Downers Grove, IL

Abstract

This simple, yet highly useful SAS® program generates a "relational map" of permanent SAS datasets or SAS/ACCESS® views within a common libname reference. Basically, RDB_MAP.SAS produces a tabular or matrix report showing the common variables shared by all the datasets or views in the libname. The column headings of the report are the SAS dataset or view names and the row headings are all the SAS variable names. An 'X' (or your favorite alpha-numeric character or word) is printed under the dataset or view the variable is found. One can then easily see which tables share a common variable.

RDB_MAP.SAS utilizes the SASHELP.VCOLUMN dataset , a PROC SORT, a PROC TRANSPOSE, and a simple PROC PRINT. This is a fast and easy way to display all relationships between multiple tables of information.

Introduction

This utility code is designed to lessen the initial shock one goes through when starting up a new project involving the combination of data from multiple datasets. Its basic design is to give the programmer an overview of all the datasets or database views just thrown at him/her. One of the more popular approaches maybe the proc contents. The "proc contents data=libref._all_;" may be the simplest approach, but will generate large amounts of paper, not to mention the time to sort through it all. RDB_MAP.SAS is a more elegant approach.

Figure 1 shows the code used to produce the report. Only input required is to change the libname to fit your platform.

Figure 1 SAS Code

```
*********************************************
* RDB_MAP.SAS
*
*  Author:  Eric Losby  Data Interpretation
*
*  Description:
*  Generates a 'Map' which shows
*  relationship between tables by common
*  variables.  Must supply a libname and
*  optional memtype. (DATA,View,...)
*
*********************************************;

options ps=80 nodate ;

%let libref = ACCESS;

libname &libref 'drive:[directory]';

TITLE "Relational Database Map of &libref.";

* SECTION 1;
data temp1(keep=memname name x);
set SASHELP.VCOLUMN;

if libname="&libref." AND memtype='VIEW' then do;
    x = 'X';
    output;
end;
else delete;
run;

* SECTION 2;
proc sort data=temp1;
by name memname;
run;

* SECTION 3;
proc transpose data=temp1 out=list(drop=_name_);
id memname;
var X;
by name;
run;

* SECTION 4;
proc print data=list;
run;
```

Section 1 - The SASHELP.VCOLUMN View

The sashelp views or dictionary tables hold great deal of data about your data, or metadata, that should not go unnoticed. The RDB_MAP.SAS code looks at one specific sashelp view, the SASHELP.VCOLUMN. This holds all the "column" or variable level data in the default libnames and specified libnames. The information available is the LIBNAME, MEMNAME (dataset), MEMTYPE, NAME (variable name), TYPE, LENGTH, NPOS, VARNUM, LABEL (rdb column name if view), FORMAT, INFORMAT and IDXUSAGE.

Figure 2 shows a sample print of the information available in the SASHELP.VCOLUMN.

Also, a PROC CONTENTS for the SASHELP.VCOLUMN is in Figure 3.

For a simple example, just consider the variables MEMNAME and NAME (or LABEL). For more advance uses, one could look at TYPE, LENGTH, FORMAT and/or INFORMAT.

At this point, we define our common variable identifier, 'X' to add to the SASHELP.VCOLUMN information. This will be the symbol displayed on the table identifying the existence of a variable in a dataset.

Figure 2 Sample print of information found in SASHELP.VCOLUMN

OBS	LIBNAME	MEMNAME	MEMTYPE	NAME	TYPE	LENGTH	NPOS	VARNUM	LABEL	FORMAT	INFORMAT	IDXUSAGE
1	ACCESS	ADDRESS	VIEW	ADDR_ID	num	8	0	1	addr_id	11.	11.	
2	ACCESS	ADDRESS	VIEW	ZIP_CD	char	5	8	2	zip_cd	$5.	$5.	
3	ACCESS	ADDRESS	VIEW	CNTRY_CD	char	5	13	3	cntry_cd	$5.	$5.	
.												
.												
23	ACCESS	ADD_PRSN	VIEW	ADDR_TYP	char	4	40	6	addr_typ_cd	$4.	$4.	
24	ACCESS	ADD_PRSN	VIEW	GEN_STAT	char	1	44	7	gen_status_cd	$1.	$1.	
25	ACCESS	ADD_PRSN	VIEW	CREAT_DT	num	8	45	8	gen_status_cd_dt	DATETIME21.	DATETIME21.	
.												
.												
101	ACCESS	PERSON	VIEW	PRSN_ID	num	8	0	1	prsn_id	11.	11.	
102	ACCESS	PERSON	VIEW	LAST_NM	char	25	8	2	last_nm	$25.	$25.	
103	ACCESS	PERSON	VIEW	FIRST_NM	char	25	33	3	first_nm	$25.	$25.	
.												
.												

Figure 3 PROC CONTENTS output of SASHELP.VCOLUMN

```
 Data Set Name: SASHELP.VCOLUMN                          Observations:         .
 Member Type:   VIEW                                     Variables:            12
 Engine:        SASESQL                                  Indexes:              0
 Created:       22:06 Thursday, February 4, 1993         Observation Length:   145
 Last Modified: 22:06 Thursday, February 4, 1993         Deleted Observations: 0
 Protection:                                             Compressed:           NO
 Data Set Type:                                          Sorted:               NO
 Label:

      -----Alphabetic List of Variables and Attributes-----

  #    Variable    Type    Len    Pos    Label
 ------------------------------------------------------------
 10    FORMAT      Char     16    104    Column Format
 12    IDXUSAGE    Char      9    136    Column Index Type
 11    INFORMAT    Char     16    120    Column Informat
  9    LABEL       Char     40     64    Column Label
  6    LENGTH      Num       8     40    Column Length
  1    LIBNAME     Char      8      0    Library Name
  2    MEMNAME     Char      8      8    Member Name
  3    MEMTYPE     Char      8     16    Member Type
  4    NAME        Char      8     24    Column Name
  7    NPOS        Num       8     48    Column Position
  5    TYPE        Char      4     32    Column Type
  8    VARNUM      Num       8     56    Column Number in Table
```

Section 2 - The PROC SORT

To use the PROC TRANSPOSE we need to sort the data by "name memname". This puts the information in the correct order. Remember, name is the variable names in all the datasets or views and memname are the names of the datasets or views.

Section 3 - The PROC TRANSPOSE

This whole utility relies on the ability to transpose information from datasets and create new datasets. The PROC TRANSPOSE basically turns a variable values into variables (or columns, if you prefer). One then needs to decide how the output will look. If one wants the dataset names as the column headings, then the ID in the PROC TRANSPOSE should be MEMNAME. These are transposed by NAME, using X as the VAR or values to use for the ID columns. It is not necessary to keep the variable _name_ in the transposed dataset.

For additional help with PROC TRANSPOSE, please refer to your SAS documentation.

Section 4 - The PROC PRINT

What would be complete without a PROC PRINT? One can easily change the position of the columns by using a VAR statement. Some options to consider might be to specify “HEADING = HORIZONTAL” and “UNIFORM” in the PROC PRINT.

Figure 4 shows the general form that the output will have.

Figure 4 General form the final report will have

OBS	NAME	dsn1	dsn2	dsn3	dsn4 ...
1	var1	X		X	
2	var2		X		
3	var3	X	X	X	X
4	var4		X	X	
5	var5		X	X	
6	var6	X			
.	.				
.	.				

Case Study 1

Gathering Address Information

If given a task to construct an address from a relational database, one may have to draw on several tables for the information. Let's say you are given the tables that you need to get the information from (not always the case though!) There are tables: person, addr_prsn_addl_addr_h (add_prsn), address, addl_addr_lin (add_line), and city_r table. One could do a PROC CONTENTS on all of these, but here's how RDB_MAP.SAS would display the information. This uses views created by SAS/ACCESS.

From Figure 5, one can easily see relationships among the datasets/views. The person and add_prsn table share the prsn_id, while the address and add_prsn share the addr_id. This connects the name information with the address information. Also, the address information is related to the add_line by the ad_ad_id and to the city_r table by city_id.

Figure 5 Sample print for Case Study #1

OBS	NAME	ADDRESS	ADD_PRSN	ADD_LINE	CITY_R	PERSON
1	ADDR_ID	X	X			
2	ADDR_TYP		X			
3	AD_AD_ID		X	X		
4	AD_AD_OR		X			
5	AS_OF_DT	X			X	
6	CITY_ID	X			X	
7	CITY_NM				X	
8	CNTRY_CD	X				
9	CNTY_ID	X				
10	CREAT_DT		X			
11	DATA_VER	X			X	
12	FIRST_NM					X
13	GEN_STAT		X			
14	HNDCP_AC	X				
15	LAST_NM					X
16	LIN_TXT			X		
17	MID_NM					X
18	PRSN_ID		X			X
19	PRSN_SAL					X
20	PRSN_TIT					X
21	SOUNDEX_					X
22	SRC_CD					X
23	STATE_CD	X				
24	STE_NM	X				
25	STRET_DI	X				
26	STRET_NM	X				
27	STRET_NU	X				
28	STRET_TY	X				
29	SYST_TBL		X			
30	ZIP_CD	X				
31	ZIP_PLUS	X				

Case Study 2

Using the variable type and formats

As mentioned before, one can use the TYPE, LENGTH, FORMAT variables from the SASHELP.VCOLUMN view to get specific variable information. This adds another dimension to the report. All that is required is to change Section 1 of RDB_MAP.SAS. See Figure 6.

Note that the type variable will only have the value of "char" or "num". Also, if there exists a format for a variable, a date format for example, it would make sense to report that format. Informat could also be used.

Using the enhancement on the previous case study yields the output in Figure 7.

Not only does the output show relations, but now one has verification if two (or more) common variable share the same length and/or format.

Figure 6 Modification to Section 1

```
* SECTION 1;
data temp1(keep=memname name x);
 set SASHELP.VCOLUMN;
format x $12.;

if libname="&libref." AND memtype='VIEW' then do;

  if type = "char" then x = compress("$"||length||".");
  else x = compress(length||".");

  if format ne " then x = format;

  output;

end;
else delete;
run;
```

Figure 7

OBS	NAME	ADDRESS	ADD_PRSN	ADD_LINE	CITY_R	PERSON
1	ADDR_ID	11.	11.			
2	ADDR_TYP		$4.			
3	AD_AD_ID		11.	11.		
4	AD_AD_OR		11.			
5	AS_OF_DT	DATETIME21.			DATETIME21.	
6	CITY_ID	11.			11.	
7	CITY_NM				$35.	
8	CNTRY_CD	$5.				
9	CNTY_ID	11.				
10	CREAT_DT		DATETIME21.			
11	DATA_VER	$5.			$5.	
12	FIRST_NM					$25.
13	GEN_STAT		$1.			
14	HNDCP_AC	$1.				
15	LAST_NM					$25.
16	LIN_TXT			$35.		
17	MID_NM					$25.
18	PRSN_ID		11.			11.
19	PRSN_SAL					$4.
20	PRSN_TIT					$5.
21	SOUNDEX_					$4.
22	SRC_CD					$5.
23	STATE_CD	$2.				
24	STE_NM	$6.				
25	STRET_DI	$2.				
26	STRET_NM	$30.				
27	STRET_NU	$6.				
28	STRET_TY	$7.				
29	SYST_TBL		11.			
30	ZIP_CD	$5.				
31	ZIP_PLUS	$4.				

Summary

RDB_MAP.SAS has many advantages.

- It's simple
- Saves paper
- Highly portable from one project to the next
- Great first step while undertaking any new project
- Creates output that will impress your co-workers

Contact: Eric Losby
Programmer/Analyst
HealthCare COMPARE Corp.
3200 Highland Avenue
Downers Grove, IL 60515

e-mail: Eric_Losby@hccompare.com