Paper 97

PROC FORMAT Provides Efficient Current Descriptions of Codes in Large Tables

Bob Romero, U S WEST Communications, Littleton, CO

Abstract

Relational Databases often store codes in large tables in order to save space and improve processing speed. Library look-up tables provide descriptions with the codes in order to decipher what the codes mean. A table join is needed so that the description is available when the data is printed in a report. PROC FORMAT allows the descriptions to be printed when needed, without the extra processing of joining tables. A SAS® view of a relational library table is then used to provide up-to-date descriptions of the code without using extra space.

Introduction

Codes are more efficient to store in large relational tables than are long descriptions. However, if the description is needed in a report so that the audience can make sense of the report, a table join is needed which requires more processing time and space, especially if the data table is extremely large. PROC FORMAT allows the use of a SAS data set to provide a format of a variable without extra processing done on the large table. Also, if a recent description of each code is required, a SAS view of the library table will provide the most recent descriptions. The following example illustrates the use of PROC FORMAT and the UNIX “cron” facility to provide meaningful information on a report.

Data Preparation

A library look-up table contains numeric status codes and descriptions. For Example:

TICKET_TYPE_ID	TICKET_TYPE_NAME
29	ISDN_BRI
2	POTS
3	FEATURE_GROUP
4	DESIGN_LOW
5	SA_INQUIRY
22	56KB
23	64KB
37	LMU_NOTIFIER
42	MESSAGE
43	DSL
1	DS1
44	DS3
45	DS1_LIS
46	ISDN_PRI
48	VDSL

The view of the ORACLE® table was established using SAS/ACCESS® to ORACLE and the SQL Procedure Pass-Through Facility. The code for this table would be:

```
proc sql;
connect to oracle as currcon(path=”@DATABASE”);
create view libref.view_1 as
select * from connection to currcon
( SELECT TICKET_TYPE_ID, TICKET_TYPE_NAME
FROM TABLE );
disconnect from currcon;
quit;
```

The code to create the view is run only once and because it is a view, the view is automatically updated as the table is updated.

The data step reads the contents of the view into a temporary SAS data set and renames the code and description variables into the valid variable names for PROC FORMAT. Also, a value is assigned to the variable, fmtname, which contains the name of the format to be used for the code variable in the large data set.

```
libname library 'path of format library';
data tcktype;
set libref.view;
length label $20;
start = ticket_t;
label = ticket_0;
fmtname = 'tckfmt';
keep start label fmtname;
run;

proc format library=library cntlin = tcktype;
run;
```

The above program is part of the UNIX script that is cronned each day. The format is stored in a permanent library and then is assigned in the SAS program that processes the large data set with a format statement. For example:

```
format ticket_t tckfmt.;
```

If the numeric value is requested instead of the description, then a numeric format to the variable is assigned.

```
format ticket_t 4.;
```

Conclusion

This facility of PROC FORMAT expedites the processing of the large data set and still produces a column of meaningful information that is up to date.

AUTHOR CONTACTS

Bob Romero
U S West Communications
700 W. Mineral, Room OR H11.15
Littleton, CO
(303) 707-7706
rromero@uswest.com