Paper 318-2013

Don't Let the Number of Columns Hold You Back!

Wayne Crews, SAS Institute Inc., Cary, NC

ABSTRACT

Many databases have a column limit of approximately 2,000 columns. Several SAS® PROCs, such as PROC NEURAL and PROC TRANSPOSE, produce output that easily exceeds 2,000 columns. This paper presents a technique to code around this business problem when using databases on the back end. Maximize your columns using SAS to talk to the databases via multiple tables, pull them together, and then split them back out.

INTRODUCTION

As data itself and manipulating data get more complex, data sets are growing in the number of rows as well as the number of columns. Working with various procedures could put you in a situation where suddenly you go from the number of rows becoming the number of columns. When working with SAS data sets, this is no problem! For all practical purposes, SAS does not have a limit on the number of columns in a SAS data set. What do you do when your ETL process or back-end data store does not support the number of columns that you are working with?

THE LIMITS

When running up against the column limit, the database commonly returns the high-water value with the error message. The SAS/ACCESS engine passes the error along with the SAS error. Figure 1 represents an example of trying to load too many columns into a database back end. Figure 1 is followed by Table 1, which shows a summary of various databases and their column limit in a standard table.

Figure 1. Surpassing the Column Limit

```
118  data gp.wide;
119     set sgflocal.wide;
120  run;
ERROR: Error attempting to CREATE a DBMS table. ERROR: CLI execute error: [SAS
ACCESS to Greenplum][ODBC Greenplum Wire Protocol driver][Greenplum]ERROR: tables
can have at most 1600 columns(File tablecmds.c;Line 2207;Routine MergeAttributes;).

NOTE: The DATA step has been abnormally terminated.
NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set GP.WIDE may be incomplete.  When this step was stopped there
were 0 observations and 3003 variables.
ERROR: ROLLBACK issued due to errors for data set GP.WIDE.DATA.
```

Table 1. Maximum Number of Columns by Database Vendor

Database Vendor	Maximum Number of Columns
Oracle	1000
Greenplum	1600
DB2	1012
Teradata	2048
SQL Server	1024

GETTING AROUND THE LIMIT

You can store your wide data set using SAS. When it comes time to store your wide data set to your back-end relational database, it is easy to do with SAS. The solution is to add a way to uniquely identify each row, and then slice the original SAS data set into multiple tables that have fewer columns than the maximum allowable columns for that database. In the following code example, I used a Greenplum database, which has a column limitation of 1600. The SAS data set named wide has 3000 data columns named x1..x3000, and one column named ID that stores the row_id. Each execution of the DATA step creates a Greenplum table with 1000 columns and one column for the ID.

This allows three Greenplum tables to store all 3000 data columns in the RDBMS. This data can be reported on in the RDBMS or pulled back into SAS for computations.

```
libname saslib "\\wayne\SGF2013";
libname gp greenplm server='greenarrow.unx.sas.com' user=wayne password=xx
schema=wayne database=hps;
%macro portion(fvar,lvar);
data gp.col&fvar.to&lvar(dbcommit=100);
   set saslib.wide (keep=x&fvar-x&lvar id);
run;
%mend;
%portion(1,1000);
%portion(1001,2000);
%portion(2001,3000);
```

PULLING IT BACK INTO A WIDE SAS DATA SET

If the data is in the RDBMS, you can read it and merge it back into one wide SAS data set. In this coding example, three Greenplum tables are read. Each has 1000 data columns. The row_id in each table is stored in a column named ID. Using PROC SQL, all 1001 columns of each table are selected and written to one SAS view table. These three SAS view tables are merged into the new wide SAS data set. Now that the data is in one SAS data set with 3001 columns, you can use the power and speed of SAS to do your analysis and reporting.

```
libname saslib "\\wayne\SGF2013";
libname gp greenplm server='greenarrow.unx.sas.com' user=wayne password=xx
schema=wayne database=hps;
%macro portion(fvar,lvar);
proc sql;
   connect using gp as mygp;
   create view strip&fvar as select * from connection to mygp
    ( select * from col&fvar.to&lvar order by id );  quit;
%mend;
%portion(1,1000);
%portion(1001,2000);
%portion(2001,3000);
data saslib.widenew;
   merge strip1 strip1001 strip2001;
   by id;  run;
```

CONCLUSION

In summary, data is becoming bigger and more complex. Business has driven the need to store many columns in one data set or table. However, relational databases have the limitation of only being able to save between 1000 to 2000 columns per table. SAS has no column limitations. You can still use your back-end database to store your wide table, and then pull it back into SAS as one wide data set for fast analysis.

RECOMMENDED READING

- *Base SAS® Procedures Guide*
- *SAS® For Dummies®*

CONTACT INFORMATION

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

Wayne Crews
100 SAS Campus Dr.
Cary, NC 27513
Wayne.Crews@sas.com
http://www.sas.com