

Understanding PROC SQL

*Prepared by: Patricia Ledesma Liébana
e-mail: pledesma@northwestern.edu*

Most of the SAS samples in the Wharton Research Data Services' (WRDS) server use PROC SQL to merge the datasets. This document briefly explains the logic of PROC SQL for SAS users who have experience with the standard MERGE statement.

PROC SQL is SAS' implementation of the ANSI Standard Query Language (SQL) standard. Among the significant differences is that PROC SQL can use SAS data step functions, SAS macros and macro variables. One limitation of PROC SQL is that it can join (SQL's jargon for "merge") up to 16 tables (datasets) at a time. SAS' normal limit is 100.

It is not necessary to learn PROC SQL to use any of the datasets on the WRDS' server. Programs written using the MERGE statement will work just as well.

The following SQL statement appears in the "ina.sas" sample program:

```
1 PROC SQL;  
2     CREATE TABLE temp AS SELECT  
3     inanames.coname, inanames.iname, temp.*  
4     FROM temp, comp.inanames  
5     WHERE temp.cnum = inanames.cnum AND temp.cic = inanames.cic;  
6 QUIT;
```

- In PROC SQL, SAS datasets are called "tables". Line 2 refers to a dataset called "temp" and it can be read as the equivalent of a DATA step.
- Variables within a table can be called by the name of the table, a period, and the variable name. Wild cards are allowed. Thus, in line 3 of the program above, inanames.coname and inanames.iname refer to the variables "coname" and "inames" in the "inanames.ssd01" dataset. Note that this requires a prior LIBNAME statement. "temp.*" refers to all the variables in dataset "temp" (thus, the SQL statement is overwriting the existing "temp").
- Line 4 specifies the source datasets for the table creation and can be read as the MERGE statement.
- Line 5 indicates to PROC SQL that both "cnum" and "cic" in temp have to match "cnum" and "cic" in "inanames". This is the equivalent of the BY statement in a merge.

The same result can be accomplished with DATA steps and merge statements. One advantage of SQL is that it uses less lines of syntax. A regular SAS merge will require sorting the datasets involved.

The SELECT statement has a few sub-clauses that are interesting (other than FROM and WHERE). You can also create summary datasets with summary functions and a GROUP BY statement.

For further reference, see:

- SAS Institute, *Getting Started with the SQL Procedure*
- The SQL chapter in the *SAS Procedures manual* (online, from Research Computing's SAS or manuals pages)

The following example is generates the same dataset twice, using both SQL and regular merge statements. It is based on WRDS' "ina.sas" program.

```
options ls=72 ps=max nocenter nonumber nodate;

libname comp '/wrds/compustat/sasdata';

data temp;
  set comp.ina (keep=cnum cic yeara smbl data6 data12);
  where smbl in ("KO" "IBM") and yeara=1995;

proc sql;
  create table temp1 as select
    inanames.coname, inanames.iname, temp.*
  from temp, comp.inanames
  where temp.cnum = inanames.cnum and temp.cic = inanames.cic;
quit;

proc print data=temp1 noobs label;

/* The same but with merge */

data temp2;          /* Subset data from inanames, in order to sort */
  set comp.inanames (keep=cnum cic smbl coname iname);
  where smbl in ("KO" "IBM");

proc sort data=temp;
  by cnum cic;

proc sort data=temp2;
  by cnum cic;

data temp;
  merge temp2 temp;
  by cnum cic;

proc print data=temp noobs label;
```

First version: February 5, 2001
Last revision: February 16, 2002