Paper QT-09

Why choose between SAS Data Step and PROC SQL when you can have both?

Charu Shankar, SAS® Canada

ABSTRACT

As a SAS coder, you've often wondered what the SQL buzz is about. Or vice versa you breathe SQL & don't have time for SAS. Learn where the SAS data step has a distinct advantage over SQL. Learn where you just can't beat SQL.

1. READING RAW DATA

The data step is able to read raw data. Since PROC SQL does not read raw data, turn to the data step to read your flat files.

```
data dsrawdata;
   infile datalines dlm=',';
   input name $ gender $  age height;
   datalines;
   Alfred,M,14,69
   Alice,F,13,56.5
   Barbara,F,13,65.3
   ;
run;
```

Display 1. Read raw data with the data step

WINNER: SAS DATA STEP

2. JOINING DATA

The SAS data step uses Merging techniques to join tables while PROC SQL uses join algorithms. PROC SQL offers more flexibility in joins: you don't necessarily have to join on same named columns, nor are you limited to joining only on equality, nor do you have to explicitly pre-sort data.

Data Step Merge

```
* prepare data for merging;
proc sort data=sashelp.prdsal2
out=prdsal2;
by state;
run;

proc sort data=sashelp.us_data
out=us_data(drop=state);
by statename;
run;
```

Proc SQL Join

```
proc sql;
create table sqljoin as
select
COUNTRY,COUNTY,PRODUCT,STATENAME
,POPULATION_2010
from sashelp.prdsal2 as p,
sashelp.us_data as us
where p.state = us.statename;
quit;
```

```
*rename variables to allow merging;
data dsmerge;
merge prdsal2(in=inprd) us_data(in=inus
rename=(statename=state));
by state;
if inprd and inus;
keep COUNTRY COUNTY PRODUCT STATE
POPULATION_2010;
run;
```

Display 2. Data Step Merge vs. PROC SQL Join

	Match-Merge	SQL Inner Join
Number of datasets/ size	No limit to number or size other than disk space.	Max number of tables in join 256.
Data processing	sequential so that observations with duplicate BY values are joined one-to-one.	Cartesian product for duplicate BY values.
Output datasets	Multiple data sets can be created.	Only one data set can be created
Complex business logic	using IF-THEN or SELECT/WHEN logic.	CASE logic; however, not as flexible as DATA step syntax.
Sorted/indexed datasets	Prerequisite for merging	Not necessary
Join condition	Equality only	Inequal joins can be performed.
Same named variables	Same named BY variables must be available in all data sets.	Same named variables do not have to be in all data sets.

Display 3. Data Step Merge vs. PROC SQL Join comparison

WINNER – PROC SQL

3. ACCUMULATING DATA

The data step can get an accumulating or running total with greater ease by using a SUM statement. The PROC SQL step is a little bit more involved.

A running total is the summation of a sequence of numbers which is updated each time a new number is added to the sequence, by adding the value of the new number to the previous running total.

Proc sql accumulating data

```
data shoes;
set sashelp.shoes;
obs=_n_;
run;

proc sql;
create table sqlrunning as
 select region, product, sales,
 (select sum(a.sales) from shoes as  a
where a.obs <= b.obs) as Running_total
 from shoes as b;
quit;
```

Data step accumulating data

```
data dsrunning;
set shoes;
keep region product sales
running_total;
running_total + sales;
run;
```

Display 4. Accumulating data

WINNER: DATA STEP

4. AGGREGATING DATA

The Boolean expression is a logical expression that evaluates to either true or false. With PROC SQL you can get really creative in what this expression looks like.

Just like adults, newborns come in a range of healthy sizes. Most babies born between 37 and 40 weeks weigh somewhere between 5 pounds, 8 ounces (2,500 grams) and 8 pounds, 13 ounces (4,000 grams). If you want to see the number of over average weight babies who were born to married women and whose mom were smokers, use the slick Boolean operation in Proc Sql.

Proc Sql aggregating data

```
proc sql;
create table sqlboolean as
select visit,
sum(weight > 4000 and married=1
and momsmoke=1) as wgt4000 'over
average weight',
sum(weight <=2500 and married=1
and momsmoke=1) as wle2500 'under
average weight'
from sashelp.bweight
group by visit;
quit;
```

Data step aggregating data

```
/*1. prep data for summarizing*/
data dsboolean;
set bweight;
by visit;
if first.visit then  do;
   wgt4000=0;
   wle2500=0;
end;
if weight > 4000 and married=1 and
momsmoke=1 then wgt4000 + 1;
else if weight <=2500 and married=1
and momsmoke=1 then wle2500 + 1;
if last.visit;
label wgt4000 ='over average weight'
      wle2500 ='under average weight';
keep visit wgt4000 wle2500;
run;
```

Display 5. Aggregating data

WINNER: PROC SQL

5. MANAGING DATA

Dictionary tables are a fascinating way to get metadata quickly. If you are trying to locate ID named columns that are also numeric, PROC SQL will do the trick speedily.

```
proc sql;
select libname, memname, name, type, length
from dictionary.columns
where upcase(name) contains 'ID'
and libname='SASHELP' and type='num';
quit;

NOTE: Table WORK.SQLDICT created, with 34 rows and 5 columns.
quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.77 seconds
      user cpu time       0.37 seconds
      system cpu time     0.34 seconds
      memory              5623.92k
      OS Memory           29176.00k
      Timestamp           03/24/2018 12:38:22 AM
```

Display 6. PROC SQL to locate all numeric ID columns in the SASHELP library

```
data dsdict;
set sashelp.vcolumn;
keep libname memname name type length;
where upcase(name) contains 'ID' and libname='SASHELP' and type='num';
run;

NOTE: There were 34 observations read from the data set SASHELP.VCOLUMN.
      WHERE UPCASE(name) contains 'ID' and (libname='SASHELP') and
(type='num');
NOTE: The data set WORK.DSDICT has 34 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           2.86 seconds
      user cpu time       1.26 seconds
      system cpu time     1.42 seconds
      memory              6505.20k
      OS Memory           29432.00k
```

Display 6. Data step to locate all numeric ID columns in the SASHELP library

Here's why PROC SQL was faster. While querying a DICTIONARY table, SAS launches a discovery process. Depending on the DICTIONARY table being queried, this discovery process can search libraries, open tables, and execute views. The PROC SQL step runs much faster than other SAS procedures and the DATA step. This is because PROC SQL can optimize the query before the discovery process is launched. The WHERE clause is processed before the tables referenced by the SASHELP.VCOLUMN view are opened.

WINNER: PROC SQL

CONCLUSION

Both the SAS data step and PROC SQL are powerful tools to read, analyze, manage and report on data. The goal of this paper was to shine light on 5 common data scenarios and showcase how sometimes the data step is crisper and more efficient and sometimes PROC SQL rises above the data step. Knowing the strengths of each of these tools in your SAS toolkit will help draw upon the right tool at the right time.

REFERENCES

Huang, Chao “Top 10 SQL trips in SAS”, SAS Global Forum 2014
https://support.sas.com/resources/papers/proceedings14/1561-2014.pdf

Shankar, Charu, “#1 SAS programming tip for 2012”, SAS Training Post, May 10, 2012
https://blogs.sas.com/content/sastraining/2012/05/10/1-sas-programming-tip-for-2012/

ACKNOWLEDGMENTS

Charu is grateful to Pharmasug for accepting her paper. This paper originally was presented at the Wisconsin Illinois SAS Users group as an invited paper. Charu has modified this paper for Pharmasug and included up to date references. She appreciates her manager Stephen Keelan and SAS Canada for the support and encouragement to share her SAS® and SQL knowledge. She is grateful to her many wonderful customers and students whose ongoing questions provided the impetus to research & share SAS data step and PROC SQL techniques.

CONTACT INFORMATION

The author welcomes correspondence about this work. You can contact her at:

Charu Shankar
Senior Technical Training Specialist
SAS® Institute Inc.
280 King Street East,
Toronto, ON M5A 1K7
Charu.Shankar@sas.com

Why