

The SQL Procedure: When and How to Use It?

Ying Feng, CTB/McGraw-Hill, Hightstown, NJ

ABSTRACT

As a data analyst who did Oracle and SQL Server Database programming before diving into the SAS® world, I'm thrilled to find SAS also provides its own version of Structured Query Language (SQL). Sure, the DATA step can do a lot of work, but sometimes PROC SQL can accomplish the same work more efficiently and with fewer steps. At the very least, it provides a nice alternate to the DATA step.

In this paper, some scenarios will be identified and PROC SQL will be used to demonstrate them.

INTRODUCTION

Developed by IBM in the mid-1970s, SQL is a programming language for getting information into or out of relational databases or non-relational-based tables. A fundamental difference between SQL and other standard programming languages is that SQL is declarative. You specify what kind of data you want; the database system is responsible for figuring out how to retrieve it.

Although SQL is both an ANSI (American National Standards Institute) and an ISO (International Standards Organization) standard, many database products provide proprietary extensions to the standard SQL language. PROC SQL is the SAS implementation of SQL. Among the significant differences from other extended versions of SQL is that PROC SQL can use SAS DATA step functions, SAS macros and macro variables.

In this paper, I review several scenarios demonstrating how to tackle common issues using PROC SQL. This will illustrate that PROC SQL can often achieve the same goals with fewer steps as compared to non-SQL Base SAS techniques, thus consuming fewer computer resources.

Scenario 1: Merging Tables -- No Presorting and No Common Variable Needed

With non-SQL Base SAS, a lot of procedures require a SORT in advance. With PROC SQL, no beforehand SORT procedure is needed. PROC SQL does sorting behind the scene and it's arguable that whether it's more efficient than PROC SORT, but it's clear that PROC SQL requires less coding. Note that in the following example, the SQL procedure takes only one step and the non-SQL takes three.

Non-SQL Base SAS:

```
PROC SORT DATA=TABLE1 (RENAME = (VAR1=COMMONVAR)) OUT=TABLE1_2;  
  BY COMMONVAR;  
RUN;  
  
PROC SORT DATA=TABLE2 (RENAME = (VAR2=COMMONVAR)) OUT=TABLE2_2;  
  BY COMMONVAR;  
RUN;  
  
DATA MERGE;  
  MERGE TABLE1_2 TABLE2_2;  
  BY COMMONVAR;  
  
RUN;
```

PROC SQL:

```
PROC SQL;  
  CREATE TABLE MERGE AS  
    SELECT *  
      FROM TABLE1, TABLE2  
     WHERE TABLE1.VAR1=TABLE2.VAR2;  
QUIT;
```

Scenario 2: Transforming Data – Creating, Renaming New Variables and Ordering Output

Sometimes we need to output a file with the variables sorted in a certain order (not necessarily in an alphabetic order) and give those variables explicit names.

Assuming we have a data set called Table1, with 10 variables ordered from var1 to var10. What we want to accomplish is:

1. Add two new variables:
 - newVar1 by concatenating var1 and var2
 - newVar2 as the sum of var3;
2. Rename var1 and var2 as out1 and out2;
3. Only output var1, var2, and var3 and the two new variables;
4. Order the output variables as out2, out1, NewVar1, var3, NewVar2.

This task can be accomplished in PROC SQL easily, using the following code.

```
PROC SQL;  
  CREATE TABLE TABLE2 AS  
    SELECT VAR2 AS OUT2, VAR1 AS OUT1, VAR1||VAR2 AS NEWVAR1,  
          VAR3, SUM (VAR3) AS NEWVAR2  
      FROM TABLE1;  
QUIT;
```

Scenario 3: Access Relational Databases

To access relational database systems, you need to have a SAS License for SAS/ACCESS. SAS/ACCESS products are data access engines. They are a family of interfaces (each of which is licensed separately) that enable you to interact with data in other vendors' databases from within SAS. You can either use the LIBNAME Statement or the Pass-Through Facility to access the database.

If you use the LIBNAME Statement to connect to the database, you can choose either the DATA step or PROC SQL to process data. However, if you choose the Pass-Through Facility to talk with the database, PROC SQL is your only choice. The Pass-Through Facility enables you to interact with a data source using its native SQL syntax without leaving your SAS session, a definite advantage. Below is the syntax for the Pass-Through Facility.

```

PROC SQL;
  CONNECT TO DBMS-NAME;
  SELECT *
    FROM CONNECTION
    TO DBMS-NAME
    (DBMS-SPECIFIC-SELECT-STATEMENT);
EXECUTE
  (DBMS-SPECIFIC-SQL-STATEMENT)
  BY DBMS-NAME;
DISCONNECT FROM DBMS-NAME;
QUIT;

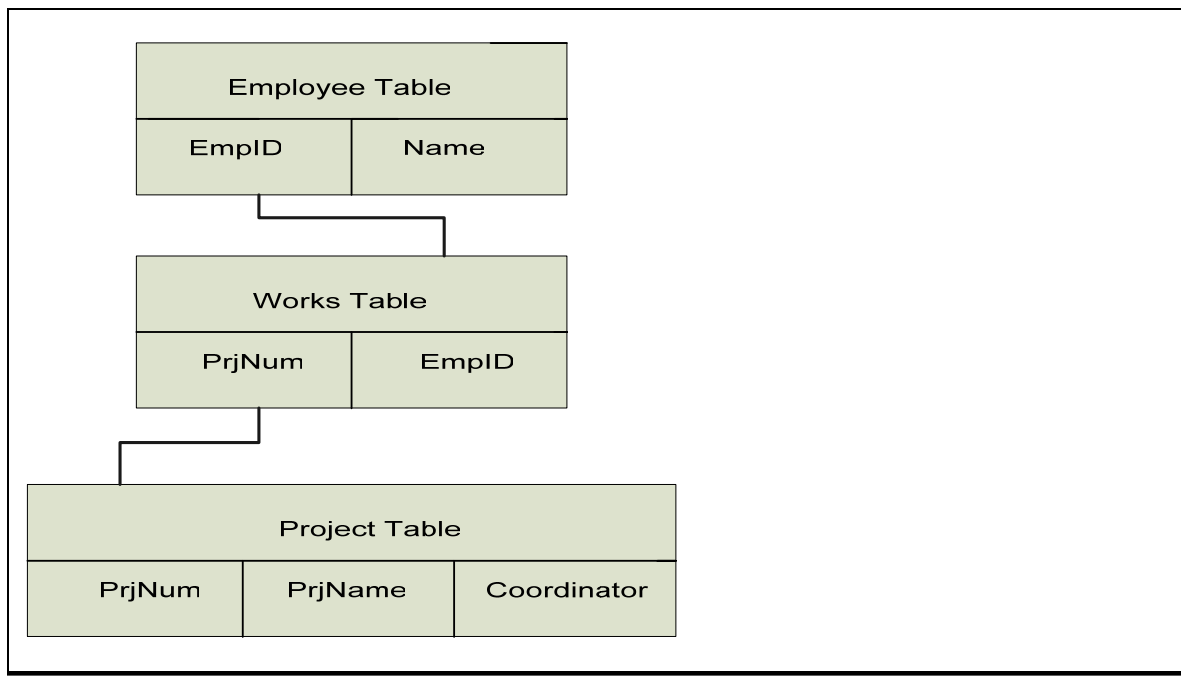
```

Scenario 4: Complex Joining

Combining data sets or tables together is quite a common operation. For DATA Steps, you have to sort the data set, rename common variables if they do not have common names, and perform multiple merge steps if the join is complex. But for PROC SQL, all this can be done in one statement. With many techniques provided by the SQL language, the complex join can be accomplished quite easily.

Suppose we have three data sets as shown below: employee information in the Employee table, project numbers used by employees in the Works table, and project information in the Project table. We want to create a new dataset with the name and ID of employees who worked on all the projects.

The following is the data sets relationship chart.



Here is the SQL code.

```
PROC SQL;
  CREATE TABLE GRP1 AS
  SELECT *
  FROM EMPLOYEE AS E
  WHERE (
    (SELECT PROJNUM
     FROM WORKS AS W
     WHERE E.EMPID=W.EMPID)
    CONTAINS
    (SELECT PROJNUM
     FROM PROJECT)
  );
QUIT;
```

Isn't that simple?

Scenario 5: Create a Macro List Using the Into Clause

The ability of PROC SQL to create a string of macro variables is quite powerful. Combined with a do-loop, it can solve a lot of headaches.

Consider this scenario: Every month, we get a list of organization IDs. We need to evaluate the performance of each organization. The list of organization IDs will differ each month, so we need to generate macro variables containing the codes and call them separately to perform the evaluation.

In PROC SQL, we can use the Into Clause to put the list of organization IDs into a string of macro variables. Then, we call them individually in a do-loop to perform the statistical evaluations.

```
*LISTORG IS A DATASET CONTAINING THE LIST OF ORGANIZATION IDS (IDORG)AND
ORGANIZATION NAMES;

PROC SQL NOPRINT;
  SELECT COUNT(*) INTO: NUMROWS
  FROM LISTORG;

  SELECT IDORG INTO :CDE1-:CDE%TRIM(%LEFT(&NUMROWS))
  FROM LISTORG;
QUIT;

%MACRO ITER;
  %DO I=1 %TO &NUMROWS;
    .....
    .....
    %RPTG(IDORG=&&CDE&I)
  %END;
RUN;
%MEND ITER;

%ITER;
```

CAVEATS

Every approach has its own caveats. For PROC SQL, one of the most important issues is to avoid Cartesian products if you want to merge several large data sets. If you forget to specify merging variables, SQL will join every row of one table to every row of the other tables. Creating the Cartesian product is quite resource consuming. Investing enough time to learn the SQL algorithms and how to code them efficiently is a nice way to avoid such caveats.

CONCLUSION

The purpose of the paper is to discuss a nice alternate to non-SQL Base SAS programming, PROC SQL. Based on the above discussion, we can see PROC SQL is a powerful tool. However, there are many different ways to get any job done. What can get the job done fast and right is what is appropriate for you. The methodology you choose isn't that important.

REFERENCES

Dickstein, Craig and Pass, Ray. 2001. "DATA Step vs. PROC SQL: What's a Neophyte to Do?" *Proceedings of the Twenty-Sixth Annual SAS Users Group international Conference*, paper 61.

Hu, Weiming. 2004. "Top Ten Reasons to Use PROC SQL". *Proceedings of the Twenty-Ninth Annual SAS Users Group international Conference*, paper 42.

Whitlock, Ian. 2001. "PROC SQL - Is it a Required Tool for Good SAS Programming?" *Proceedings of the Twenty-Sixth Annual SAS Users Group international Conference*, paper 60.

AUTHOR CONTACT INFORMATION

Ying Feng
Statistical Analyst

CTB/McGraw-Hill
148 Princeton-Hightstown Road
East Windsor, NJ 08520

Email: ying_feng@ctb.com

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.