Using the SQL Procedure

Kirk Paul Laffler
Software Intelligence Corporation
Spring Valley, CA 91979-1390 (619) 670-S0FT

Abstract

The SQL procedure follows most of the guidelines established by the American National Standards Institute. In its current form, PROC SQL provides many features available in the DATA step and the MEANS, PRINT, and SORT procedures. An advantage of using PROC SQL is that it can often result in fewer and shorter statements than using existing DATA step methods and procedures. This hands-on workshop illustrates numerous examples of how PROC SQL and its many statements can be used.

Prerequisites

This workshop introduces the syntax and uses of the SQL Procedure. No prior knowledge of SQL is required since this material is designed to acquaint the user and/or programmer to the many features found within the SQL Procedure. You should have a working knowledge of the following concepts before enrolling in this workshop:

- how to use the SAS System on your host computer system
- how to interact with the SAS System in your environment
- how to create and process SAS System data sets under Version 6.06
- how to use the SAS System Display Manager System windows and text editor
- how to assign and reference Librefs and Filerefs in your environment
- what the rows and columns represent in a SAS System data set.

Workshop Objectives

After completion of this workshop, you should have a working knowledge of the following concepts:

- basic SQL concepts
- specific SQL syntax requirements
- how to construct queries to retrieve, sum, group, and sort data
- how to create and modify simple tables and views
- why the ORDER BY clause is not very efficient.

Brief History of Structured Query Language (SQL)

Structured Query Language (SQL) has an interesting history. It is a universal language that was developed to easily access data that is stored in relational databases or tables. A relation is represented as a two-dimensional table consisting of rows and columns. A series of guidelines were developed by Dr. E. F. Codd, an IBM mathematician, through the use of relational mathematics. SQL evolved over time to access data regardless of the application software (e.g., COBOL, PL/1, etc.) being used.

Basic SQL Concepts

SQL boasts the ability to define, manipulate, and control relational databases or tables as well as providing easy user access. The concept behind SQL is that the user does not have to specify physical attributes about the data such as data structure, location, and/or data type. The user will concentrate on what data should be selected, but not how to select them. It also provides methods of making changes to tables without the need to change application programs. Consequently, data independence is one of the design goals of the relational model.

The SQL Procedure in the Base SAS System

PROC SQL has the following features:

- can run interactively as well as batch (noninteractive)
- uses the Structured Query Language to create, modify, and retrieve data from tables
- can be augmented through the use of Global statements such as TITLE and FOOTNOTE
- accesses tables via a two-level name where the first level is the Libref and the second level name is the name of the table.

Terminology

The following terminology is provided to help relate SQL and SAS System terms and concepts.

Column -- the same as a variable in the SAS System.

Libref -- acts as the alias or nickname. Points to a SAS data library.

Relational Database Management System -- a database system that forms relationships between data items.

Row -- the same as an observation in the SAS System.
Structured Query Language (SQL) — a highly standardized high-level language used in relational database management systems to create and alter objects within a database.

Table — the same as a SAS System data set.

View — contains a definition or description of data stored in another location.

Retrieve and Display Data

To extract and retrieve data, we will use the SQL Procedure's SELECT statement. You will see in the following examples that the desired column(s) or variable(s) are displayed in the order indicated in the SELECT statement.

Example 1:

PROC SQL;
SELECT SSN, SEX
FROM libref.PATIENTS;
In Example 1, the columns SSN (Social Security Number) and SEX are selected and displayed from the PATIENTS SAS System data set.

Example 2:

PROC SQL;
SELECT SSN, LASTNAME
FROM libref.PATIENTS;
In Example 2, the columns SSN and Patient’s Last Name from the PATIENTS data set.

Sum and Display Data in Groups

To sum and display data in groups, we will use the SQL statement GROUP BY. In the next example the GROUP BY clause is used when a summary function is used in a query. The column being summed is WEIGHT (Patient’s Weight) with the results of the query being grouped by SEX (Patient’s Gender).

Example:

PROC SQL;
SELECT SEX, SUM(WEIGHT) AS TOTWEIGHT
FROM libref.PATIENTS
/* Shortness of Breath */
WHERE SYMPTOM='10'
GROUP BY SEX;

Output:

SEX TOTWEIGHT
F 1800
M 3155

Arrange Results in Ascending Order

To arrange results in ascending order, we will direct SQL by using the ORDER BY clause. One or more columns can be selected for sorting. One or more columns can be ordered in either ascending and/or descending order. The default sort order is ascending (lowest to highest). If you want to override the default order, arrange in descending order, you need to specify DESC following the column-name that is specified.

Example:

PROC SQL;
SELECT LASTNAME, EDUC
FROM libref.PATIENTS
ORDER BY EDUC;

Output:

LASTNAME EDUC
Candle 10
Robertson 12
Cranberry 13

Syntax Requirements

The Structured Query Language (SQL) is directed by the statements, options, and components within the SQL procedure to create, retrieve, and modify data from tables and views. The syntax requirements for using the SQL procedure within the SAS System follow.

PROC SQL < option(s) > ;
CREATE create-statement;
DELETE delete-statement;
DROP drop-statement;
SELECT select-statement;
UPDATE update-statement;

SQL Procedure Statements

SQL procedure statements are presented on the following pages.
CREATE Statement  
The CREATE statement allows for the creation of tables, views, and indexes on columns in tables. There are multiple ways of creating tables, views, and indexes. This workshop will illustrate the statement for creating a new table with columns that are not existing within current tables.

General Format - CREATE TABLE:

PROC SQL;
CREATE TABLE table-name
(column-definition(s));

Example:

PROC SQL;
CREATE TABLE CUNICS(CUN NO char(6),
DIRECTOR char(20));
PROC CONTENTS DATA=CUNICS;
RUN;

Data Types and Widths for Columns:

Data Types: CHARACTER, INTEGER, DECIMAL  
Widths: Character columns default to 8 characters, Numeric columns are created using the maximum precision possible by the SAS System. A LENGTH statement can be specified in the DATA step.

Example:

PROC SQL;
CREATE TABLE CLINICS(CLIN_NO char(6),
DIRECTOR char(20));
PROC CONTENTS DATA=CLINICS;
RUN;

DELETE Statement  
The DELETE statement removes rows from a table as indicated in the WHERE clause.

General Format:

PROC SQL;
DELETE FROM table-name
WHERE sql-expression;

Example:

PROC SQL;
DELETE FROM CLINICS
WHERE CLIN_NO='011234';
/* Displays all fields in CLINICS */
SELECT * FROM CLINICS;

DROP Statement  
The DROP statement deletes a table, view, or index. You must specify the libref. when tables, views, or indexes are stored permanently.

General Format:

PROC SQL;
DROP TABLE table-name;
DROP VIEW view-name;
DROP INDEX index-name FROM table-name;

Example:

PROC SQL;
DROP TABLE libref.CLINICS;
DROP VIEW libref.CONTACTS;

SELECT Statement  
The SELECT statement fetches data for the query, formats desired information, and displays it as output.

General Format:

PROC SQL;
SELECT query-expression;

Examples:

PROC SQL;
SELECT LASTNAME, DOB FROM libref.PATIENTS;

817
**UPDATE Statement**

The UPDATE statement allows (or columns within existing observations (rows) of a table or view to be changed. Caution should be exercised while updating tables. Make sure you backup your tables prior to changing data since accidents can happen.

**General Format:**

```sql
PROC SQL;
    UPDATE table-name | libref.view-name
    SET set-clause
    <WHERE where-expression>
;
```

**Example:**

```sql
PROC SQL;
    UPDATE libref.PATCOPY /* Back-up Copy */
    SET WEIGHT=WEIGHT + 1;
    SELECT LASTNAME, WEIGHT
    FROM libref.PATCOPY;
```

**SQL Procedure Statement Components**

The SQL procedure statement components provide a way to further specify your selection and/or update criteria. The following components are briefly presented.

- **BETWEEN** -- searches for data lying within certain parameters.
- **Column-Definition** -- defines data types and widths.
- **Column-Modifier** -- establishes column attributes.
- **From-List** -- indicates what table or view to use in a FROM clause.
- **Group-by-Item** -- indicates the groups of variables values processed in a GROUP BY clause.
- **Order-by-Item** -- indicates the order in which observations are displayed in an ORDER BY clause.
- **SQL-Expression** -- identifies functions, expressions, and operators that are used to connect them.

**Conclusion**

Structured Query Language (SQL) is a universal language that was developed to easily access data that is stored in relational databases or tables. Relations can be thought of as two-dimensional tables consisting of rows and columns. Through the use of relational mathematics a series of guidelines for defining, manipulating, and controlling tables were developed by Dr. E. F. Codd.

The concept behind SQL is to free the user from having to specify physical attributes about the data such as data structure, location, and/or data type. The user concentrates on what data should be selected, but not how to select it.

The SQL procedure provides a standardized way to retrieve and display data, sum and display data in groups, and arrange results in ascending or descending order by columns.

PROC SQL can run in both interactive and batch environments. It uses the Structured Query Language to create, modify, and retrieve data from tables. Global statements such as TITLE and OPTIONS can be used with PROC SQL. Tables are accessed via two-level names where the first is the Libref and the second level is the name of the table.

**Acknowledgments**

My sincere thanks are extended to everyone involved with the SUGI Conference especially Richard La Valley and Andrew J. L. Cary, Section Chairs. Their encouragement and support in the preparation of this paper and workshop was invaluable.

**References**

SAS and SAS/SQL-DS are registered trademarks of the SAS Institute Inc.

**Author Contact**

The author will be glad to answer questions and accept suggestions at the following address:

Kirk Paul Lafler
Software Intelligence Corporation
P.O. Box 1390
Spring Valley, CA 91979-1390
Tel: (619) 670-SOFT or (619) 670-7638