

Finding Answers in the PROC SQL Documentation

Howard Schreier, Howles Informatics, Arlington VA

ABSTRACT

SQL is more than another SAS[®] procedure; it is a language, and one which was not designed by SAS developers. As a consequence, the documentation for PROC SQL is different from what is familiar to most SAS users. This paper will present a series of tips designed to help SAS users find the answers they need in the PROC SQL documentation.

WHERE TO START?

If you look at the SAS Version 9.1.3 online documentation (OnlineDoc[®]), you will see that the “Base SAS” heading can be expanded as shown here:

- Base SAS
 - + Base SAS Procedures Guide
 - + Base SAS Procedures Guide: Statistical Procedures
 - + SAS Language Reference: Concepts
 - + SAS Language Reference: Dictionary
 - ...
 - + Data Security Technologies in SAS
 - + SAS SQL Query Window: User's Guide
 - + SAS SQL Procedure User's Guide
 - + SAS Macro Language: Reference
 - ...

Notice the subheadings for the *SAS SQL Query Window: User's Guide* and the *SAS SQL Procedure User's Guide*. There is more. If you expand the “Base SAS Procedures Guide” subheading, and the “Procedures” subheading which then appears at the next level, you see

Base SAS

Base SAS Procedures Guide

What's New in Base SAS 9.0, 9.1, and 9.1.3 Procedures

Concepts

Procedures

The APPEND Procedure

...

The SORT Procedure

The SQL Procedure

The STANDARD Procedure

...

The UNIVARIATE Procedure

Appendices

Notice the section on The SQL Procedure. It corresponds to a chapter within the *SAS Procedures Guide* (whereas the two SQL subheadings we found at a higher level in this tree correspond to two separate, stand-alone books).

So what's the difference, and where do you turn? It turns out that there are four options:

- *SAS SQL Query Window: User's Guide*
- *SAS SQL Procedure User's Guide*
- "The SQL Procedure" chapter of the *SAS Procedures Guide*
- None of the above

Note: At this point we will shift our attention from the OnlineDoc to the printed version. That's just for convenience; the content is the same. Excerpts from the printed documentation which appear in boxes below are direct quotes, although typography, indentation, and the like have been altered for the sake of emphasis.

SAS SQL QUERY WINDOW: USER'S GUIDE

This is a specialized document pertaining to a graphical interface which lets you generate SQL code by doing a lot of pointing and clicking. By today's standards, and compared to tools like Enterprise Guide[®] and SAS[®] Data Integration Studio, the SQL Query Window is rather primitive. So, unless you have a very specific reason for working with the SQL Query Window, you probably want to look elsewhere for information about PROC SQL.

SAS SQL PROCEDURE USER'S GUIDE

To get a sense of what the *SAS SQL Procedure User's Guide* is about, let's take a look at the table of contents. The first chapter is an introduction:

Chapter 1 _ Introduction to the SQL Procedure 1

What Is SQL? 1

What Is the SQL Procedure? 1

Terminology 2

Comparing PROC SQL with the SAS DATA Step 3
Notes about the Example Tables 4

The next chapter turns to some basics:

Chapter 2 _ Retrieving Data from a Single Table 11

Overview of the SELECT Statement 12
Selecting Columns in a Table 14
Creating New Columns 18
Sorting Data 25
Retrieving Rows That Satisfy a Condition 30
Summarizing Data 39
Grouping Data 45
Filtering Grouped Data 50
Validating a Query 52

The following chapter introduces the added complexity of combining data from different tables:

Chapter 3 _ Retrieving Data from Multiple Tables 55

Introduction 56
Selecting Data from More Than One Table by Using Joins 56
Using Subqueries to Select Data 74
When to Use Joins and Subqueries 80
Combining Queries with Set Operators 81

The book then deals with tables and views:

Chapter 4 _ Creating and Updating Tables and Views 89

Introduction 90
Creating Tables 90
Inserting Rows into Tables 93
Updating Data Values in a Table 96
Deleting Rows 98
Altering Columns 99
Creating an Index 102
Deleting a Table 103
Using SQL Procedure Tables in SAS Software 103
Creating and Using Integrity Constraints in a Table 103
Creating and Using PROC SQL Views 105

Some relatively advanced topics appear in the fifth chapter:

Chapter 5 _ Programming with the SQL Procedure 111

Introduction 111
Using PROC SQL Options to Create and Debug Queries 112
Improving Query Performance 115
Accessing SAS System Information Using DICTIONARY Tables 117
Using PROC SQL with the SAS Macro Facility 120
Formatting PROC SQL Output Using the REPORT Procedure 127
Accessing a DBMS with SAS/ACCESS Software 128
Using the Output Delivery System (ODS) with PROC SQL 132

Examples conclude the main body of the book:

Chapter 6 _ Practical Problem-Solving with PROC SQL 133
Overview 134
Computing a Weighted Average 134
Comparing Tables 136
Overlaying Missing Data Values 138
Computing Percentages within Subtotals 140
Counting Duplicate Rows in a Table 141
Expanding Hierarchical Data in a Table 143
Summarizing Data in Multiple Columns 144
Creating a Summary Report 146
Creating a Customized Sort Order 148
Conditionally Updating a Table 150
Updating a Table with Values from Another Table 153
Creating and Using Macro Variables 154
Using PROC SQL Tables in Other SAS Procedures 157

There is pretty clearly a progression from the simple and elementary to the complex and advanced. This is an instructional work, one which does a pretty good job of explaining SQL as it is implemented within SAS. However, it is not a complete reference and syntax guide.

SAS PROCEDURES GUIDE (SQL CHAPTER)

This is the location of the syntax reference for PROC SQL, and will be the main focus of the remainder of this paper. However, it does not resemble the documentation for most SAS procedures; it is much more modular and much more dependent on cross references.

Let's take a look at the relevant part of the table of contents from the front of the *SAS Procedures Guide*:

Chapter 49 _ The SQL Procedure 1065
Overview: SQL Procedure 1067
Syntax: SQL Procedure 1069
SQL Procedure Component Dictionary 1108
Concepts: SQL Procedure 1152
PROC SQL and the ANSI Standard 1160
Examples: SQL Procedure 1163

We can see that most of the pages are in the "Syntax" section and the "Component Dictionary". When we look at the beginning of the chapter, a more detailed table of contents is available. There we can see that the "Syntax" section presents the **statements** which constitute SQL as implemented by SAS. The PROC statement appears first, followed by the others, in alphabetical order. To this extent, the structure does conform to the usual template for documenting SAS procedure syntax.

<i>Syntax: SQL Procedure 1069</i>
<i>PROC SQL Statement 1072</i>
<i>ALTER TABLE Statement 1077</i>
<i>CONNECT Statement 1081</i>
<i>CREATE INDEX Statement 1081</i>
<i>CREATE TABLE Statement 1083</i>
<i>CREATE VIEW Statement 1087</i>
<i>DELETE Statement 1089</i>
<i>DESCRIBE Statement 1090</i>
<i>DISCONNECT Statement 1091</i>
<i>DROP Statement 1092</i>

EXECUTE Statement 1093
INSERT Statement 1093
RESET Statement 1095
SELECT Statement 1096
UPDATE Statement 1107
VALIDATE Statement 1108

Among these statements, SELECT is very much the “800-pound gorilla”. It is complex and versatile, and much used (either as a standalone statement or, more often, as a clause within another statement) in most SQL applications. Most of the other statements are relatively specialized (such as those pertaining to the pass-thru facility, and those only used in applications which change existing tables), relatively simple (eg, the DROP statement), or even non-essential (eg, the VALIDATE statement). That is not to say that they are not useful or important. The point is that this part of the table of contents does not provide much “differential guidance”; many users find themselves delving into the SELECT documentation constantly and the rest of the statement explanations seldom or never.

Now let’s turn to the sections of the Component Dictionary.

SQL Procedure Component Dictionary 1108

BETWEEN condition 1109
BTRIM function 1109
CALCULATED 1110
CASE expression 1111
COALESCE Function 1112
column-definition 1113
column-modifier 1114
column-name 1116
CONNECTION TO 1117
CONTAINS condition 1117
EXISTS condition 1118
IN condition 1118
IS condition 1119
joined-table 1120
LIKE condition 1129
LOWER function 1131
query-expression 1131
sql-expression 1137
SUBSTRING function 1144
summary-function 1145
table-expression 1151
UPPER function 1152

Words and phrases in upper case are used literally in SQL code. Components which are entirely in lower case are placeholders which serve only in the documentation; they are replaced by other elements in SQL code, according to menus and rules which appear in the documentation.

The Component Dictionary is something of an odd mixture in other respects as well. For the most part, it includes things **not** specific to a single context (statement, clause, or component) or which don’t fit in anywhere else, but there are exceptions. On the one hand, the ORDER BY clause (which is multi-context, being usable in the SELECT, CREATE TABLE, and CREATE VIEW statements) is **not** in the table of contents. On the other hand, the joined-table item is used **only** in the FROM clause of the SELECT statement, and so might have been

subsumed there. Many of the Component Dictionary items are specialized or are presented for completeness (eg, the UPPER function, which is just an SQL-standard alias for the SAS UPCASE function).

The important point here is that this dictionary is definitely **not** a complete presentation of important SQL components, many of which are submerged elsewhere, either in the documentation of statements or in the documentation of other components. For example, subqueries are covered in the sql-expression section.

In looking at the Component Dictionary, we see a fair amount of jargon. That is, there are terms like “sql-expression” which are not part of the language per se (and which are therefore in lower case) but which are used in the documentation to categorize and classify language constructs. Some of these terms are rather similar, which can make things more confusing. It is likely that many highly competent SQL programmers would find it difficult to distinguish among

- sql-expression
- table-expression
- query-expression

Such experts would know very well how to form and use these three elements, but the **terminology** is not in everyday use. We will revisit these three terms a couple of times before we are done.

It is the jargon and the extensive cross references which can make the SQL reference documentation presented in the *SAS Procedures Guide* somewhat perplexing. The remainder of this paper will be devoted to tips and explanations intended to reduce the confusion.

NONE OF THE ABOVE

Before we continue examining the SQL reference documentation, it is important to understand that a lot of code elements allowed within a PROC SQL step are not, strictly speaking, part of SQL, and thus are not documented in the SQL chapter of the *SAS Procedures Guide*. PROC SQL is part of SAS, and therefore can “borrow” a lot of SAS features. We can see this at two distinct levels:

- Nearly all SAS functions (but not CALL routines), formats, informats, and data set options can be used in appropriate contexts **within** SQL statements. However, the SQL Component Dictionary only covers functions which are exclusive to SQL and not usable in the DATA step.
- Most if not all SAS global statements can be interspersed with the SQL statements within a PROC SQL step (that is, after the PROC SQL statement and before the QUIT statement).

These non-SQL language elements used within PROC SQL steps are documented primarily in the *SAS Language Reference: Dictionary*, so that manual should be considered part of the SQL user’s reference set.

FOLLOWING CROSS REFERENCES

The PROC SQL reference documentation is characterized by an unusual amount of cross referencing. We can get a feel for this by starting with the centerpiece of SQL, the SELECT statement.

SELECT STATEMENT

The syntax skeleton for the SELECT statement begins with:

```
SELECT <DISTINCT> object-item <, ...object-item>
```

Here we’ve encountered another bit of jargon, “object item”. It’s just a placeholder, and the documentation following the skeleton lists the possible substitutions:

object-item is one of the following:

- *
represents all columns in the tables or views that are listed in the FROM clause.
- **case-expression** <AS alias>
derives a column from a CASE expression. See “CASE expression” on page 1111.
- **column-name** <<AS> alias> <column-modifier <... column-modifier>>
names a single column. See “column -name” on page 1116 and “column -modifier” on page 1114.
- **sql-expression** <AS alias> <column-modifier <... column-modifier>>
derives a column from an sql-expression. See “sql-expression” on page 1137 and “column -modifier” on page 1114.

Of these possibilities, “sql-expression” is the most vague, so its definition may be the one we are most likely to need. We can follow the cross reference to “sql-expression”.

SQL-EXPRESSION

The definition tells us that an

sql-expression produces a value from a sequence of operands and operators.

It then presents the syntax skeleton

operand operator operand

and states that

operand is one of the following:

- a **constant**, which is a number or a quoted character string (or other special notation) that indicates a fixed value. Constants are also called literals. Constants are described in *SAS Language Reference: Dictionary*.
- a **column-name**, which is described in “column -name” on page 1116.
- a **CASE expression**, which is described in “CASE expression” on page 1111.
- a **SAS function**, which is any SAS function except LAG, DIF, and SOUND. Functions are described in *SAS Language Reference: Dictionary*.
- the **ANSI SQL functions** COALESCE, BTRIM, LOWER, UPPER, and SUBSTRING.
- a **summary-function**, which is described in “summary -function” on page 1145.
- a **query-expression**, which is described in “query -expression” on page 1131.
- the **USER** literal, which references the userid of the person who submitted the program. The userid that is returned is operating environment-dependent, but PROC SQL uses the same value that the &SYSJOBID macro variable has on the operating environment.

Basically, this is telling us that an sql-expression is much like an expression that is coded in a DATA step, involving constants, variables (here called columns), operators and functions. CASE expressions and summary functions are notable SQL extensions to this vocabulary. Putting aside the extremely specialized ANSI SQL functions and USER literal, we are left with “query-expression”, the least concrete of the permitted ingredients. So we follow its cross reference.

QUERY-EXPRESSION

The explanation of query-expression is that it

retrieves data from tables.

The documentation then offers several cross references for the whole concept:

See also:

- “table-expression” on page 1151,
- “Query Expressions (Subqueries)” on page 1140, and
- “In-Line Views” on page 1102

Then we see the syntax skeleton:

```
table-expression <set-operator table-expression> <...set-operator table-expression>
```

This is followed by a list of the components

Arguments

- **table-expression**
is described in “table-expression” on page 1151.
- **set-operator**
is one of the following:
 - INTERSECT <CORRESPONDING> <ALL>
 - OUTER UNION <CORRESPONDING>
 - UNION <CORRESPONDING> <ALL>
 - EXCEPT <CORRESPONDING> <ALL>

It then goes on to explain in considerable detail these set operators. However, the basic ingredient here is “table-expression”; we follow its cross reference.

TABLE-EXPRESSION

The documentation tells us that a table-expression

```
defines part or all of a query-expression.
```

It then offers a cross reference

```
See also: “query-expression” on page 1131
```

But it’s a cross reference **from** query-expression that brought us here, so we don’t want to follow this one right back there. Reading on, we get the syntax skeleton for a table-expression:

```
SELECT <DISTINCT> object-item<, ... object-item>  
<INTO :macro-variable-specification <, ... :macro-variable-specification>>  
FROM from-list  
<WHERE sql-expression>  
<GROUP BY group-by-item <, ... group-by-item>>  
<HAVING sql-expression>
```

This is what you may have learned as the nucleus of SQL. (though it deliberately and appropriately excludes the ORDER BY clause, which is often taught as being part of the nucleus) Yet the clauses are not explained below the skeleton. Moreover, it turns out that very few cross references from elsewhere in the documentation point here. That’s a bit strange, for the nucleus of the language.

There is one more cross reference:

See “SELECT Statement” on page 1096 for complete information on the SELECT statement.

This tells us where the details are. However, we **started** this tour with the SELECT statement documentation. Now we’ve come around in a circle. The problem is not as serious as it seems, and does not indicate any flaw in the manual. It is, in essence, due to the nestability of SQL, and, in particular, to the use of inline views and subqueries. When we saw the admissibility of a query-expression as an operand within an sql-expression, we were in fact reading about subqueries. The lesson learned is that, in the PROC SQL reference documentation, you cannot mechanically follow cross references from entities to their constituent entities and expect the process to end by leading you to nothing but primitives.

THE THREE EXPRESSIONS REVISITED

Our circular tour touched on the three types of expressions (sql-, table-, and query-). You may have found the explanations to be a bit nuanced. Let’s consider the three again, using an example. Once again, our starting point will be the SELECT statement.

SELECT STATEMENT

Here is a valid (though not very useful) SELECT statement

```
select name, age      from sashelp.class where sex='F'
union
select name, age + 1 from sashelp.class
order by age
;
```

QUERY-EXPRESSION

A query-expression is either a table-expression or two or more table-expressions connected by set operators. To illustrate, we’ll underline the query-expression in our SELECT statement:

```
select name, age      from sashelp.class where sex='F'
union
select name, age + 1 from sashelp.class
order by age
;
```

It’s a little odd to see that the statement does not “sandwich” its major ingredient (the query-expression). Instead; the statement **begins** with the query-expression, followed by the ORDER BY clause and the terminating semicolon. The documentation does not exactly recognize this structure, and it tends to blur the distinction between the SELECT statement and its major ingredient, the query-expression.

TABLE-EXPRESSION

A table-expression is essentially a SELECT clause (not statement), with its subordinate clauses. We can underline the two table-expressions in our example:

```
select name, age      from sashelp.class where sex='F'
union
select name, age + 1 from sashelp.class
order by age
;
```

SQL-EXPRESSION

An sql-expression is a scalar expression (one which evaluates to a single value, and **not** to multiple rows or columns). In that way it's a lot like a formula or expression you might code in a DATA step. An sql-expression can incorporate subqueries, but each such subquery must ultimately evaluate to a scalar. Of course an sql-expression which applies to a data source comprising multiple rows will be evaluated repeatedly, once for each row. In that sense it can give rise to a vector, even though each evaluation generates a scalar result.

Let's continue our example by underlining just the sql-expressions:

```
select name, age      from sashelp.class where sex='F'
union
select name, age + 1 from sashelp.class
order by age
;
```

The sql-expression is a ubiquitous construct. In this example we see one as an object-item (that is, in a SELECT list) and one in a WHERE clause. Sql-expressions are also permitted in the ON clause (part of a join specification), the GROUP BY clause, the HAVING clause, and the ORDER BY clause.

COULD IT BE MORE LOGICAL?

We noted earlier a bit of oddity in the documentation of the table-expression. Even though the table-expression (that is, the SELECT/FROM/WHERE/GROUP BY/HAVING sequence) is the nucleus of SQL, its documentation is only sketchy, the details being located instead under the SELECT statement. This is a reasonable arrangement in that it conforms to what most people probably expect. It is not, however, the most logical arrangement.

Before continuing this discussion of the documentation structure, let's consider another question: What are the devices available to make SQL results available **outside** PROC SQL? There are basically four such vehicles.

- Tables. SQL results can be placed in tables which in turn can be used by other parts of the SAS System. This can be done with either the CREATE TABLE statement or the INSERT statement.
- Views. The CREATE VIEW statement can be used to make SQL results available to other parts of the SAS System when such results are needed.
- Output Delivery System. With or without the use of explicit ODS code, a standalone or "naked" SELECT statement (that is, a statement which begins with the keyword "SELECT") will ordinarily send its results to the Output Delivery System.
- Macro variables. A standalone SELECT statement which includes an INTO clause and which does not involve set operators (that is, which incorporates a simple table-expression and not a general query-expression) will populate one or more macro variables with its results.

The example we used above,

```
select name, age      from sashelp.class where sex='F'
union
select name, age + 1 from sashelp.class
order by age
;
```

sends its results to the Output Delivery System. In the absence of any explicit ODS coding, and assuming that the code is run via the SAS Display Manager, that simply means that the results appear in the Output window.

Now suppose that instead of seeing the results, we want them in a SAS data set to be used as input to some SAS procedure. We can accomplish this by composing a CREATE TABLE statement:

```
create table mytable as
select name, age      from sashelp.class where sex='F'
union
select name, age + 1 from sashelp.class
order by age
;
```

Each of these statements evaluates the same query-expression, then sorts the rows in the result set, if necessary, to conform to the ORDER BY specification. Instead of sending the ordered result set to ODS, the CREATE TABLE statement sends it to a SAS data set. Even though the CREATE TABLE statement appears to be a derivative of the SELECT statement, the two are, functionally, more like peers with a common core (the query-expression).

The syntax skeleton for the form of CREATE TABLE statement we are using here is:

```
CREATE TABLE table-name AS query-expression
<ORDER BY order-by-item<, ... order-by-item>>;
```

This suggests that the syntax skeleton for the standalone SELECT statement (which can be thought of as the “Feed ODS and/or Populate Macro Variables” statement) **could** be simply:

```
query-expression
<ORDER BY order-by-item<, ... order-by-item>>;
```

Note that this is **not** a quote from the manual.

The details of the SELECT/FROM/WHERE/GROUP BY/HAVING sequence could then be relocated to the section of the documentation which explains table-expressions.

CONCLUSIONS

SQL is very different from other SAS procedures, and as a consequence its documentation is organized differently. Here are some things to keep in mind when looking for answers about PROC SQL. We’ll start with tips about **where** to look.

- Don’t look for information in the *SAS SQL Query Window: User’s Guide* unless you have a specific interest in that product (SQL Query Window).
- Use the *SAS SQL Procedure User’s Guide* to learn about SQL from explanations and examples.
- Consult the SQL chapter in the *SAS Procedures Guide* to find the syntax rules for PROC SQL and to determine just what is allowed within each construct.
- Remember that the table of contents at the beginning of that chapter does not identify all of the major elements of SQL, but does point to a number of very minor elements.
- PROC SQL permits you to use many non-SQL SAS language elements (functions, data set options, global statements, etc.), so consider the *SAS Language Reference: Dictionary* to be part of the documentation of PROC SQL.

The tips which follow all pertain to the syntax information in the *SAS Procedures Guide*.

Try to learn the terminology and concepts used in the SQL syntax explanations, especially the three generic-sounding “expressions” (sql-expression, table-expression, and query-expression).

- An sql-expression is basically a scalar formula. Subqueries are components of sql-expressions, so the syntax documentation for subqueries is found in the sql-expression section.
- A table-expression is a SELECT clause (not statement) together with its required FROM clause and any of the optional subordinate clauses (INTO, WHERE, GROUP BY, or HAVING).
- A query-expression is either a single table-expression or multiple table-expressions connected using set operators. Because this is the only context in which set operators are used, they are documented in the query-expression section.

The modularity of SQL leads to a lot of cross referencing in the syntax documentation. The nestability provided by subqueries and inline views creates some circular paths through the cross referencing.

The syntax documentation for the nucleus of SQL (the SELECT clause with its subordinate FROM, INTO, WHERE, GROUP BY, and HAVING clauses) is found in the section for the SELECT statement, even though it logically belongs under the table-expression concept, and is often used in CREATE TABLE and other statements.

REFERENCES

SAS Institute Inc., 2006. *SAS OnLine Doc 9.1.3*. Cary, NC: SAS Institute Inc.
<http://support.sas.com/onlinedoc/913/>

SAS Institute Inc., 2004. *SAS 9.1 SQL Procedure User's Guide*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2006. *Base SAS 9.1.3 Procedures Guide, Second Edition*, Volumes 1, 2, 3, and 4. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2006. *SAS 9.1.3 Language Reference: Dictionary, Fifth Edition*. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

SAS is a Registered Trademark of the SAS Institute, Inc. of Cary, North Carolina.

CONTACT INFORMATION

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

Howard Schreier
Howles Informatics
Arlington VA

703-979-2720

hs AT howles DOT com
<http://howles.com/saspapers/>