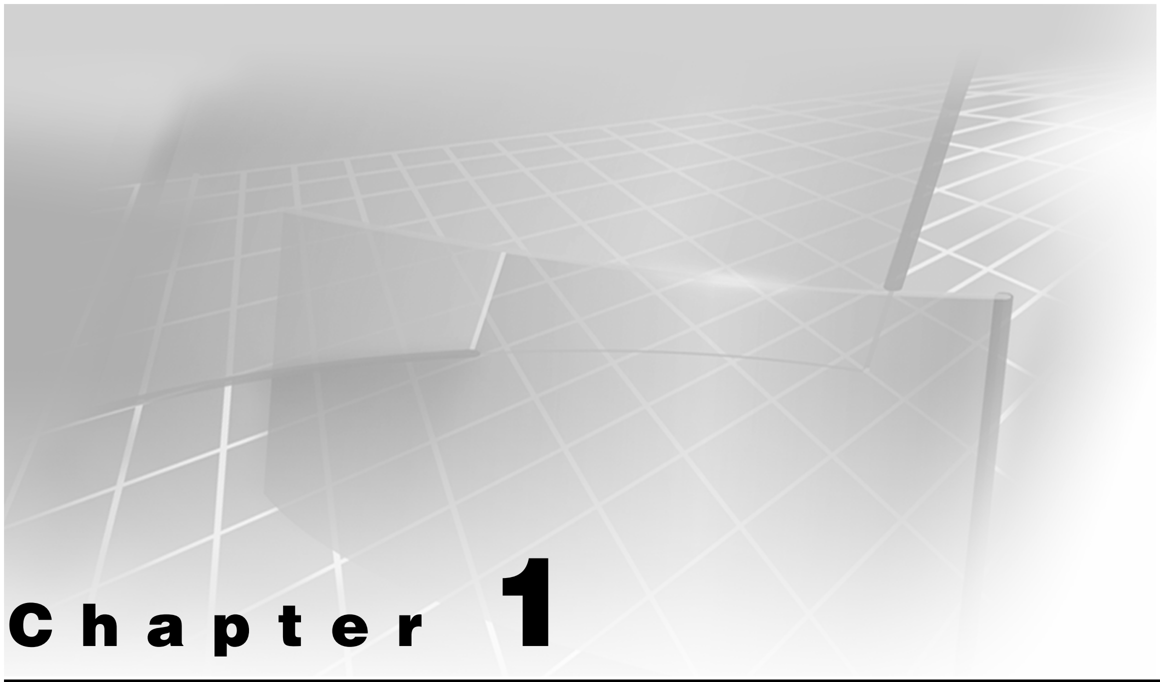

Chapter 1

Introduction to PROC SQL

A little history

SQL is a powerful, flexible, fourth-generation sublanguage that enables complex processing through a few simple statements. You need only to indicate the desired outcome rather than outline each of the steps necessary to reach that outcome because SQL is a nonprocedural language. SQL statements allow for the complete creation, maintenance, and reporting of relational database systems using English-like statements.

In the mid-1970s the Structured Query Language (SQL) was developed by IBM researchers in San Jose, California, to support a new relational database model. In June 1970, Dr. E. F. Codd, a researcher with IBM, published his mathematical theory of data management in a paper entitled "A Relational Model of Data for Large Shared Data Banks." His ideas resulted in the definition of a new form of data storage structure, a table consisting of rows and columns. The relational database model was thus born from tables and the relationships between tables.

SQL was designed to enable access to data stored in a relational database. It allows you to create, alter, and delete tables as well as modify or delete existing records or add new records to tables.

By the late 1980s and early 1990s, each database vendor had its own version of SQL. In an effort to minimize the inconsistencies and provide portability of SQL statements, the American National Standards Institute (ANSI) developed a set of international standards to be applied to the language. Several standards have been published by ANSI since 1986, including SQL-89, SQL-92, SQL-99 and SQL-2003.

Each successive SQL language release extends functionality. However, the foundations of the SQL language have remained mostly unchanged. Vendors that are compliant with the ANSI SQL-92 standard, for example, are also compliant with the SQL-99 core function standards.

The power and ease-of-use of SQL has resulted in its use in hundreds of database products today. Companies such as Oracle, Microsoft, Sybase, and IBM depend heavily on SQL in their database products regardless of operating system. As a result, anyone working with databases today must be proficient in SQL. The ANSI standards have resulted in a set of more or less common statements with agreed upon functionality from each vendor. However, many different ideas and syntactical differences are found in each flavor of SQL.

PROC SQL underwent major change in SAS Version 8, resulting in a more versatile procedure that is also more closely in line with the ANSI SQL-92 standard. The new version extends the functionality of the SQL language with elements from Base SAS.

PROC SQL statements

There are approximately 40 statements in SQL which can be grouped into several categories. Data Definition Language (DDL) statements such as CREATE, ALTER, and DROP are used to create and maintain tables and other objects such as indexes and views in a relational database. Data Manipulation Language (DML) statements such as SELECT, INSERT, UPDATE, and DELETE are used to retrieve and maintain rows of table data. The final group of statements, COMMIT, ROLLBACK, GRANT, and REVOKE are all Data Control Language (DCL) statements, which are focused on database security.

This book concentrates on the DDL and DML statements that support reporting or selection of data and the creation and maintence of tables and views. With knowledge of just seven statements, you have the powerful functionality of SQL at your fingertips.

Why learn PROC SQL?

Why learn PROC SQL instead of continuing to use the SAS programming language and the standard procedures? A single SELECT statement may encompass selection, manipulation, merging, sorting, summary, filtration, and reporting processes, eliminating several SAS procedures and DATA steps in your programs.

You will find that you can apply many of the concepts you already know to PROC SQL. Whether it is retrieving data from a data set or database, the SQL procedure treats everything as though it were a two-dimensional table composed of rows and columns. This is similar to the SAS data set where the observations are rows and variables are columns.

The SQL procedure completes many tasks with a single SQL statement while the more traditional SAS solution may involve several SAS procedures and DATA steps.

Tip: A table generated in PROC SQL can be referenced in a DATA step in exactly the same way as a data set. The reverse is also true; data sets created in a DATA step can be used in PROC SQL.

PROC SQL allows you to modify and maintain tables within a database from the SAS session. Statements to modify existing records, add new records, and delete records can all be incorporated into a PROC SQL statement and applied via the SQL Pass-Through Facility or LIBNAME engine.

The compact nature of a SQL statement allows for quick updates of programming code if the data changes. SQL statements can also include macro variables allowing for a generic program that can be dynamically updated.

Reporting that suits your needs

A PROC SQL query generates a summarized, sorted report without a call to any other procedures. The SELECT statement automatically produces printed output, eliminating the need for a PRINT procedure. The SELECT statement also sorts the result set without a call to PROC SORT.

SAS standard formats and user-defined formats created in PROC FORMAT may be applied within a SELECT statement.

Output Delivery System (ODS) destinations may be specified for the reports generated by a SELECT statement. A single SELECT statement may generate several formatted reports such as HTML and PDF reports using ODS statements. A document destination may also be specified, providing for flexible output options without the need to re-execute the query.

The TEMPLATE procedure may be used to create custom styles for reports. Other SAS procedures such as PROC TABULATE and PROC REPORT may operate on tables created by the SQL procedure.

Tip: A simple SELECT statement produces a printed report unless you specify the NOPRINT option. SELECT statements that are part of a subquery in a CREATE, ALTER, or UPDATE statement do not generate output. In each of these cases, the output returned by the SELECT statement is used as input to the CREATE, ALTER, or UPDATE statement.

The power to create

The PROC SQL CREATE statement provides for the creation of tables, views, and SAS data sets in either a database or a native SAS library from within a SAS session. Users working within a database require appropriate privileges in order to execute a CREATE statement in a database. The new tables can be created within a database or stored in a native SAS data library.

There are several options available for the creation of tables in PROC SQL. Each column and column modifier can be specified much in the same way as empty fields in a data set can be established in a DATA step. However, tables can also be created using the structure of an existing table or data set as a template, but leaving an empty table. Alternatively, the CREATE statement can mimic the DATA step data set creation process, taking both structure and data from an existing table or data set.

Views provide the ability to selectively allow users to see columns and rows of information in one or more tables and data sets. Views can be generated from a complex SELECT statement that retrieves information from one or more tables or data sets. However, you issue a simple SELECT statement against the view, retrieving up-to-date information because the view is re-created each time it is queried. The dynamic nature of views makes them invaluable for reporting applications.

Tip: Most table creation in a database can be accomplished using PROC SQL statements. If additional database-specific statements are required, they can be passed directly to the Relational Database Management Systems (RDBMS) for processing using the SQL Pass-Through facility. Only those users with appropriate database privileges can successfully submit CREATE statements.

Ease of maintenance

PROC SQL statements such as ALTER, INSERT, UPDATE, DROP, and DELETE provide for the addition of new data and the modification or update of existing data in a table. From within a SAS session, tables in a database as well as tables and data sets stored in native SAS libraries can be maintained by users with appropriate database privileges.

New rows can be added to tables directly using the INSERT statement, or they may be taken from one or more tables or data sets in any active library, including a database. One or more criteria can be set, limiting the rows taken from the other sources.

In PROC SQL, a single UPDATE statement applies the changes to existing rows in a table without creating a new table or data set. The rows modified by the UPDATE statement can be limited through WHERE clause criteria based on values within the same table or other tables. Moreover, updates can be easily applied from records in one or more other data sets or tables.

Existing tables and data sets can also be altered using the ALTER statement to include additional columns and add column modifiers such as labels or formats. In each case, the work is done on the existing table without the need to create a new table or data set.

Security

USER, a special keyword, when added to a PROC SQL INSERT or UPDATE statement can be used to store the user ID associated with the action in a table. Such statements can be triggered by specific events to execute in the background of applications, allowing for the creation of an effective audit table. Dates and other information collected from the SAS session can also be added to the entry.

Database security is also maintained. A user must have the appropriate security to create tables, views, and indexes within a database. Only those users with appropriate database privileges can successfully submit INSERT, UPDATE, ALTER, and DELETE statements. In addition, only those users with read-access to tables may report from them or views built from those tables using the SELECT statement.

Data integrity checks

Integrity constraints such as primary key, foreign key, check, unique, and not null can all be added either at the time a table is created or later with an ALTER statement. Once set, they automatically check all incoming data values.

Primary key integrity constraints check for duplicate values as data is entered into one or more columns of a table. Foreign key integrity constraints prevent data from being entered into the column of one table unless the value is already in another. This form of constraint is useful for ensuring that each row of your incoming data has a valid code such as a state abbreviation.

The check integrity constraint can be used to check all incoming data against one or more criteria. For example, it can be used to limit your incoming data to a specific date range. The NOT NULL integrity constraint can be applied to prevent the entry of null values.

Foreign key constraints make the task of synchronizing column values common to two tables or data sets effortless through the specification of referential actions. When a value in the parent or reference table is modified, the value in the linked or child table is automatically modified in the same way. In addition, the constraint can be set up so that the value in the child table is changed to a missing value if the value in the linked parent table is deleted.

Optimized performance

The PROC SQL optimizer automatically works out a plan for executing SQL statements in the most efficient manner possible. Indexes can be built to provide better performance for your queries. In addition, directions for the optimizer may be added as statement options.

Performance optimization is important when tables are stored in one or more databases as well as native SAS libraries. The SQL optimizer determines whether processing should be transferred to the database or be handled by SAS. Options such as DBMASTER, new in SAS 9, assist in the optimizer in efficient handling of queries involving tables residing in different database locations.

Adaptability

Flexibility in a changing environment

We all know how unstable the computing environment in companies is today. Your company may decide to implement a new data warehouse using Sybase instead of DB2. A new package may be introduced to generate reports from your Oracle database.

If you are using a SAS/ACCESS LIBNAME statement to connect to a database, the only change needed regardless of the database is to the LIBNAME connection string required to establish a connection to the database.

Using the familiar SAS interface, you can easily create new tables and update and retrieve your data. There is no need to learn another product or interface such as Oracle SQL*Plus or ISQL. Moreover, the information extracted using SQL is directly available for further processing in a SAS DATA step or other SAS procedures.

SAS PROC SQL allows you to apply your standard SAS output formats and labeling options and almost all of the SAS functions. In fact, if you are a SAS programmer, you already know more about SQL statements than most other programmers!

Table merges and Cartesian products

One of the strengths of SQL is its ability to join or merge two or more tables, generating result sets or new tables that include data from one or more of the tables. Criteria can be built from any column in the joined tables as well. Unlike the SAS data set merge, the tables do not have to be presorted or indexed on a key variable in order to accomplish the join. With PROC SQL, the procedure sorts and merges the tables in the same step.

There are several ways to accomplish a join or merge in SQL. It is possible to match common values in a column in two or more tables. However, it is also possible to produce a Cartesian product of all combinations of all rows from all of the tables.

Two or more tables can also be joined using outer joins which retrieve nonmatching rows from one or both tables along with the unique matching rows. Set operations provide additional merge or join functionality based on intersect, union, and exception operations.

Fuzzy logic

A wide range of criteria can be applied to SQL queries thereby limiting the rows retrieved or manipulated by the query. However, often the criteria we wish to apply cannot be written in a simple fashion using mathematical operators such as =, <, or >.

PROC SQL WHERE clauses may include conditions that require fuzzy logic or inexact matching.

Fuzzy logic can be applied to pattern-matching criteria in an SQL query. SAS functions such as SCAN and CONTAINS allow us to parse a string for the inclusion of various characters. The LIKE operator can be used in conjunction with wildcard symbols to restrict character string matches to a particular position within a column value. For criteria based on a range, the BETWEEN operator can be used to set the bounds.

PROC SQL may also include the SAS SOUNDEX function in the WHERE clause of a query. This function will match column values that sound similar to the given value.

Criteria built from stored data

Subqueries provide the ability to calculate or retrieve one or more values which are then substituted into a WHERE clause. Essentially they allow report criteria to be built at the time the query is run. Subqueries are executed first, returning one or more values to the main query.

Work within a database from a SAS session

An advantage of the SQL procedure is its ability to establish a connection to a database through SAS/ACCESS and the SQL Pass-Through facility. PROC SQL statements may be submitted interactively within a SAS session directly to a database regardless of its location. For example, database servers can be accessed from client machines, or the database may be distributed between several machines.

In addition, a single SAS session can support multiple connections to one or more databases enabling multisource data delivery for data warehouse extraction, transformation and loading processes and other applications. SAS procedures and DATA steps can seamlessly combine database tables and tables and data sets stored in native SAS libraries.

SQL Pass-Through facility

SAS took the new tool one step further by incorporating the ability to execute SQL statements generated from within a SAS session directly against a variety of database systems. The SQL Pass-Through facility enables a user to retrieve information from a database and incorporate it into SAS data sets, all from within the familiar SAS environment.

A database specific CONNECT statement first opens communication with the database. SAS then passes the SQL statements to the database for execution and returns the results to the SAS session. Finally, a DISCONNECT statement closes the database connection.

SAS/ACCESS LIBNAME statement

Since SAS 8, connecting to a database has become even easier with the enhancement of the LIBNAME statement. With PROC SQL and SAS/ACCESS, the export and import steps are eliminated. A LIBNAME statement provides the means to connect to the database, and these tables can be accessed directly by either a DATA step or PROC SQL. Moreover, a data set stored in a native SAS library may be matched to records in tables stored in two different database systems using fields in common between the tables.

The LIBNAME statement specifies the parameters needed to establish a database connection and associate that connection with a libref. Once assigned, the libref can be used as part of two-level names in any SAS procedure or DATA step. In addition, views can store embedded LIBNAME statements making the connection to a database invisible to the user.

Interactive and Web-based applications

Internet and intranet-driven applications can incorporate SQL statements using htmSQL input files that provide access to tables and data sets, both in native SAS libraries and databases. In addition, entire SQL statements can easily be built from parameters passed through interactive Web pages.

In SAS/AF FRAME entry and webAF, objects can be populated through data retrieved from either native SAS libraries or databases using the SQL procedure. Parameters passed through FRAME objects can also be used to set criteria used in SQL statements.

Values retrieved from tables or data sets using SQL queries can also be easily bound to macro variables using the INTO clause. These variables can be passed to macro programs in FRAME source code or other SAS programs.

When not to use PROC SQL

The SQL procedure does not read or write text files directly. Nor is it well suited to the creation of a data set using instream record images.

The SAS DATA step offers a wide range of delimiters and other formatting options commonly encountered in data files. It provides more flexibility when reading files such as spreadsheets that may include missing values when they are imported into SAS. The SAS INPUT statement also allows for named input.

Although the PROC SQL INSERT statement may be used to add rows to a table or data set, a VALUES keyword is required for each row and the complete record must be enclosed in parentheses. In addition, all character and missing variables must be enclosed in quotation marks. Because many files that are imported contain, at best, a delimiter between fields, the added syntax required by the INSERT statement can significantly add to the workload of data imports.

The SAS DATA step is your only solution if you are attempting to import fixed-width columnar data into SAS. PROC SQL does not allow for positional column references in the INSERT statement.

The PROC SQL advantage

SQL is an important component of every database today regardless of its function. It is used for transactional databases as well as data warehouses and data marts supporting data mining activities. The effort you put into learning PROC SQL in SAS provides you with a skill that can be used in environments other than SAS. Once you've completed this book, you will be ready to tackle reporting, table creation, and maintenance in any database system your company decides to implement.