

PROC SQL: Why and How

Nate Derby

Stakana Analytics
Seattle, WA

SUCCESS, 4/27/11

Outline

- 1 Why?
 - What?
 - Why?
- 2 How?
 - Introduction, Examples
 - Working with SAS Data Sets
 - Other Aspects
- 3 Conclusions

What is SQL?

SQL = **S**tructured **Q**uery **L**anguage

- Pronounced “Sequel” or “S-Q-L”.
- Intended for managing data in relational database systems.
- Developed by IBM in early 1970s.
- Now very robust:
 - Lots of functionality (not just for queries!).
 - Implemented in most computing packages (including SAS).

Why SQL?

Because of what we just said:

- Designed for databases:
 - Works very well/quickly with large data sets.
 - Sometimes faster than SAS code.
 - Sometimes works when SAS code doesn't.
 - Easily implements **data constraints**.
 - Easily implements **views**.
 - Easily implements **indexes**.

Why SQL?

Because of what we just said:

- Lots of functionality:
 - SQL can be used to do many things.
 - Some things easier in SQL than in base SAS code.
 - Examples to follow.
- Implemented in most computing packages:
 - Great for *program portability*.
 - Can cut and paste between SAS and another language.
 - Some things aren't portable.

Intro to SQL in SAS

Implemented in SAS via PROC SQL:

PROC SQL Setup

```
PROC SQL <options>;
```

```
[SQL Statements]
```

```
QUIT;
```

Easy Examples

PROC SQL Code

```
PROC SQL;
  SELECT name, sex, age, height, weight
  FROM sashelp.class;
QUIT;
```

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112

Easy Examples

PROC SQL Code

```
PROC SQL;
  SELECT a.name, a.sex, a.age, a.height, a.weight
  FROM sashelp.class a;
QUIT;
```

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112

Easy Examples

PROC SQL Code

```
PROC SQL;
  SELECT boo.name, boo.sex, boo.age, boo.height, boo.weight
  FROM sashelp.class boo;
QUIT;
```

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112

Easy Examples

PROC SQL Code

```
PROC SQL;
  SELECT *
  FROM sashelp.class;
QUIT;
```

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112

Easy Examples

PROC SQL Code

```
PROC SQL;  
  SELECT a.*  
    FROM sashelp.class a;  
QUIT;
```

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112

Easy Examples

PROC SQL Code

```
PROC SQL;  
  SELECT name, sex, age, height format=6.2, weight  
         format=6.2  
  FROM sashelp.class;  
QUIT;
```

Name	Sex	Age	Height	Weight
Alfred	M	14	69.00	112.50
Alice	F	13	56.50	84.00
Barbara	F	13	65.30	98.00
Carol	F	14	62.80	102.50
Henry	M	14	63.50	102.50
James	M	12	57.30	83.00
Jane	F	12	59.80	84.50
Janet	F	15	62.50	112.50
Jeffrey	M	13	62.50	84.00
John	M	12	59.00	99.50
Joyce	F	11	51.30	50.50
Judy	F	14	64.30	90.00
Louise	F	12	56.30	77.00
Mary	F	15	66.50	112.00
Philip	M	16	72.00	150.00
Robert	M	12	64.80	128.00
Ronald	M	15	67.00	133.00
Thomas	M	11	57.50	85.00
William	M	15	66.50	112.00

Easy Examples

PROC SQL Code

```
PROC SQL;  
  SELECT name, sex, age, height format=6.2, weight  
         format=6.2  
  FROM sashelp.class  
  WHERE sex = 'M';  
QUIT;
```

Name	Sex	Age	Height	Weight
Alfred	M	14	69.00	112.50
Henry	M	14	63.50	102.50
James	M	12	57.30	83.00
Jeffrey	M	13	62.50	84.00
John	M	12	59.00	99.50
Philip	M	16	72.00	150.00
Robert	M	12	64.80	128.00
Ronald	M	15	67.00	133.00
Thomas	M	11	57.50	85.00
William	M	15	66.50	112.00

Easy Examples

PROC SQL Code

```
PROC SQL;  
  SELECT name, sex, age, height format=6.2, weight  
         format=6.2  
  FROM sashelp.class  
  WHERE sex = 'M'  
  ORDER BY age;  
QUIT;
```

Name	Sex	Age	Height	Weight
Thomas	M	11	57.50	85.00
James	M	12	57.30	83.00
Robert	M	12	64.80	128.00
John	M	12	59.00	99.50
Jeffrey	M	13	62.50	84.00
Alfred	M	14	69.00	112.50
Henry	M	14	63.50	102.50
William	M	15	66.50	112.00
Ronald	M	15	67.00	133.00
Philip	M	16	72.00	150.00

More Interesting Example

PROC SQL Code

```

PROC SQL;
  SELECT sex,
         mean( age ) as mage label='Mean Age' format=6.2,
         min( height ) as minh label='Min Height' format=6.2,
         max( height ) as maxh label='Max Height' format=6.2
  FROM sashelp.class
  GROUP BY sex;
QUIT;

```

Sex	Mean Age	Min Height	Max Height
F	13.22	51.30	66.50
M	13.40	57.30	72.00

Much More Interesting Example

Name	Sex	Age	Mean Age by Sex
Joyce	F	11	13.22
Thomas	M	11	13.40
James	M	12	13.40
Jane	F	12	13.22
John	M	12	13.40
Louise	F	12	13.22
Robert	M	12	13.40
Alice	F	13	13.22
Barbara	F	13	13.22
Jeffrey	M	13	13.40
Alfred	M	14	13.40
Carol	F	14	13.22
Henry	M	14	13.40
Judy	F	14	13.22
Janet	F	15	13.22
Mary	F	15	13.22
Ronald	M	15	13.40
William	M	15	13.40
Philip	M	16	13.40

Much More Interesting Example

PROC SQL Code

```
PROC SQL;
  SELECT a.name, a.sex, a.age, b.mage
  FROM sashelp.class a JOIN (
    SELECT sex, mean( age ) as mage
      label='Mean Age by Sex' format=6.2
    FROM sashelp.class
    GROUP BY sex
  ) b
  ON a.sex = b.sex
  ORDER BY age, name;
QUIT;
```

Creating Tables

PROC SQL Code

```
PROC SQL; NOPRINT;  
  CREATE TABLE work.blah AS  
    SELECT name, sex, age, height, weight  
      FROM sashelp.class;  
QUIT;
```

Creating Tables

PROC SQL Code

```
PROC SQL NOPRINT;  
  CREATE TABLE blah AS  
    SELECT sex, mean( age ) as mage  
      label='Mean Age by Sex' format=6.2  
  FROM sashelp.class  
  GROUP BY sex;  
CREATE TABLE final AS  
  SELECT a.name, a.sex, a.age, b.mage  
    FROM sashelp.class a join blah b  
    ON a.sex = b.sex  
  ORDER BY age, name;  
QUIT;
```

Log Output

Since all within one PROC, we get timing for entire block of code:

```
133 proc sql;
134   create table blah as
135     select sex, mean( age ) as mage label='Mean Age by Sex' format=6.2
136     from sashelp.class
137     group by sex;
```

NOTE: Table WORK.BLAH created, with 2 rows and 2 columns.

```
138   create table final as
139     select a.name, a.sex, a.age, b.mage
140     from sashelp.class a join blah b
141     on a.sex = b.sex
142     order by age, name;
```

NOTE: Table WORK.FINAL created, with 19 rows and 4 columns.

```
143 quit;
```

NOTE: PROCEDURE SQL used (Total process time):

real time	0.03 seconds
cpu time	0.03 seconds

The STIMER Option

The `STIMER` gives times for each step within `PROC SQL`:

PROC SQL Code

```
PROC SQL NOPRINT STIMER;  
  CREATE TABLE blah AS  
    SELECT sex, mean( age ) as mage  
      label='Mean Age by Sex' format=6.2  
    FROM sashelp.class  
    GROUP BY sex;  
  CREATE TABLE final AS  
    SELECT a.name, a.sex, a.age, b.mage  
    FROM sashelp.class a join blah b  
    ON a.sex = b.sex  
    ORDER BY age, name;  
QUIT;
```

Log Output

Now we get listings for each step:

```
223 proc sql noprint stimer;
```

```
NOTE: SQL Statement used (Total process time):
```

```
real time          0.00 seconds
cpu time           0.00 seconds
```

```
224 create table blah as
```

```
225     select sex, mean( age ) as mage label='Mean Age by Sex' format=6.2
```

```
226     from sashelp.class
```

```
227     group by sex;
```

```
NOTE: Table WORK.BLAH created, with 2 rows and 2 columns.
```

```
NOTE: SQL Statement used (Total process time):
```

```
real time          0.01 seconds
cpu time           0.00 seconds
```

```
228 create table final as
```

```
229     select a.name, a.sex, a.age, b.mage
```

```
230     from sashelp.class a join blah b
```

```
231     on a.sex = b.sex
```

```
232     order by age, name;
```

```
NOTE: Table WORK.FINAL created, with 19 rows and 4 columns.
```

```
NOTE: SQL Statement used (Total process time):
```

```
real time          0.01 seconds
cpu time           0.01 seconds
```

```
233 quit;
```

```
NOTE: PROCEDURE SQL used (Total process time):
```

```
real time          0.00 seconds
```

Use Different Table Names

Valid in SAS code, but PROC SQL complains:

PROC SQL Code

```
PROC SQL NOPRINT STIMER;
  CREATE TABLE blah AS
    ...;
  CREATE TABLE blah AS
    SELECT a.name, a.sex, a.age, b.mage
      FROM sashelp.class a join blah b
      ON a.sex = b.sex
    ORDER BY age, name;
QUIT;
```

WARNING: This CREATE TABLE statement recursively references the target table. A consequence of this is a possible data integrity problem.

ALTER, UPDATE, DELETE Statements

ALTER TABLE **statement** changes columns:

PROC SQL Code

```
PROC SQL NOPRINT STIMER;  
  CREATE TABLE blah AS  
    SELECT *  
      FROM sashelp.class;  
ALTER TABLE blah  
  DROP age  
  MODIFY height FORMAT=6.2, weight FORMAT=6.2;  
QUIT;
```

The ALTER TABLE **statement** cannot be used to change the data!

ALTER, UPDATE, DELETE Statements

UPDATE statement changes data, DELETE FROM deletes rows:

PROC SQL Code

```
PROC SQL NOPRINT STIMER;  
  CREATE TABLE blah AS  
    SELECT *  
      FROM sashelp.class;  
UPDATE blah  
  SET weight  
  WHERE name = 'Robert';  
DELETE FROM blah  
  WHERE name = 'Judy';  
QUIT;
```

UPDATE, DELETE FROM cannot be used to modify the columns!

Views and Indexes

SQL was developed for databases, so PROC SQL works really well with **views** and **indexes**:

- A **view** a “virtual table” = executable instructions for a table, but holds no data.
 - Great for getting updated info, restricting access.
- An **index** is one or more columns used to identify each row within a table
 - Great for cutting down on time for processing data in (very) large tables.

More info: See Kirk Lafler's book!

Accessing a Database

Surprise! `PROC SQL` can access a database really well.

Weird: No one writes about this ... except Katherine Prairie.

Two ways:

- Pass-Through facility:
 - Uses ODBC (external to SAS).
 - Nests non-SAS SQL code within SAS code.
 - Thus, much SAS functionality can't be used. Sometimes have to be creative with SAS code to make non-SAS code.
 - Also: Messy code!
- SAS/ACCESS and the `LIBNAME` statement: *Much* cleaner code, allows using SAS functionality with the database.

Conclusions

PROC SQL can be a powerful tool!

- Very flexibly queries (often more flexible than the standard SAS DATA steps).
- Sometimes easier than in standard SAS code.
- Facilitates program portability.
- Allows for views, indexes and integrity constraints.
- Quicker than SAS code for large data sets.
- For *very* large data sets, sometimes SAS code crashes, SQL code doesn't.

Further Resources



Pete Lund.

An Introduction to SQL in SAS.

<http://pugsug.org> (Presentations), 2010.



Howard Schreier.

PROC SQL by Example: Using SQL within SAS.

SAS Press, 2008.



Kirk Lafler.

PROC SQL: Beyond the Basics Using SAS.

SAS Press, 2004.



Katherine Prairie.

The Essential PROC SQL Handbook.

SAS Press, 2005.

Further Resources

Nate Derby: `http://nderby.org`
`nderby@stakana.com`