

Comparing SAS steps and PROC SQL: Coding and Performance

Robert Rosofsky

Health Information Systems Consulting



Boston Area SAS Users Group

December 4, 2003

Quick Intro to SQL

- Structured Query Language, developed in 1980s
- Developed to standardize the query of (*duh!*) and manipulation of relational databases
- Is now an ANSI-standard language

What is ANSI?

- American National Standards Institute
 - Standards for materials, processes, computer languages, etc.
 - COBOL, FORTRAN have standards
- SQL now in ANSI standard 1999
- SQL-2003 is in draft form

SAS Proc SQL

- Implements SQL-1992 within SAS programs
- PROC SQL as a superset of ANSI:
Permits some SAS specific keywords not found in the standard, and
- PROC SQL as a subset of ANSI:
 - Some database features are not available in SAS

Datasets and SQL Tables

SAS Nomenclature

- SAS Dataset
- Observation
- Variable

SQL Nomenclature

- Table
- Row
- Column

SQL Queries

- “Queries” in SQL are not only *requests* for information
- “Query” is also the word that you use to describe almost any SQL-coding:
 - e.g. a make-table query, an update query, an alter-table query, etc.
- Commands are available for creating tables, changing table structures, changing values in tables, and more

Proc SQL Syntax

Proc SQL *<options>* ;

- Multiple SQL statements (or queries) can be put into a single Proc
- You can have a series of SQL queries that are your entire program
- Think of the PROC SQL/QUIT; combination as boundaries for a different coding environment

Quit;

The *SELECT* Statement

- The “workhorse” of Proc SQL
- It selects rows and columns from:
 - tables/datasets and views
- Can subset by read, created, or summarized variables

The *Select* Statement (Cont'd)

- Creates:
 - new columns/variables
- Can combine datasets through concatenation, interleaving, match-merging, and updating

The *Select* Statement (Cont'd)

- Can generate reports and output to the *Output* window
- Through use of a “*Create table tablename as*” clause, can create:
 - datasets/tables
- Through use of a “*Create view viewname as*” clause, can create:
 - views

Print All Observations and Variables

```
Proc Print
```

```
  Data= First;
```

```
Run;
```

```
Proc SQL;
```

```
  Select *
```

```
  From First;
```

```
Quit;
```

Print All Observations and Variables

SAS Proc Print

Obs	patient	wealth	AGE	losses
1	48787024	20384	99	10
2	46872304	20094	98	20
3	84128158	19896	97	30
4	41338574	19702	97	40
5	95500229	19541	96	50

SQL Select...

patient	wealth	AGE	losses
48787024	20384	99	10
46872304	20094	98	20
84128158	19896	97	30
41338574	19702	97	40
95500229	19541	96	50

Subset by Var for Output

Proc Print

 Data=First;

 Var Patient Age

 Losses;

Run;

Proc SQL;

 Select Patient,
 Age, Losses

 From First;

Quit;

Subset by Var for Output

SAS Proc Print

Obs	patient	AGE	losses
1	48787024	99	10
2	46872304	98	20
3	84128158	97	30
4	41338574	97	40
5	95500229	96	50

SQL Select...

patient	AGE	losses
48787024	99	10
46872304	98	20
84128158	97	30
41338574	97	40
95500229	96	50

Subset by Record

Proc Print

 Data=First;

 Where Age >=64;

Run;

Proc SQL;

 Select *

 From First

 Where Age >=64;

Quit;

Subset by Record

SAS Proc Print

Obs	patient	wealth	AGE	losses
1	48787024	20384	99	10
2	46872304	20094	98	20
3	84128158	19896	97	30
4	41338574	19702	97	40
5	95500229	19541	96	50

SQL Select...

patient	wealth	AGE	losses
48787024	20384	99	10
46872304	20094	98	20
84128158	19896	97	30
41338574	19702	97	40
95500229	19541	96	50

Subset by Variable and Observation

Proc Print

 Data=First;

 Var Patient Age

 Losses;

 Where Age >=64;

Run;

Proc SQL;

 Select Patient,
 Age, Losses

 From First

 Where Age >=64;

Quit;

Subset by Variable and Observation

SAS

SQL

Obs	patient	AGE	losses	patient	AGE	losses
1	48787024	99	10	48787024	99	10
2	46872304	98	20	46872304	98	20
3	84128158	97	30	84128158	97	30
4	41338574	97	40	41338574	97	40
5	95500229	96	50	95500229	96	50
6	07635890	96	60	07635890	96	60
7	70238377	96	70	70238377	96	70
8	09689050	96	80	09689050	96	80
9	54968905	95	90	54968905	95	90
10	86506538	95	100	86506538	95	100

Create One Dataset From Another

```
Data Second;  
  Set First;  
Run;
```

```
Proc SQL;  
  Create table Second  
  as  
  Select *  
  From First;  
Quit;
```

Create One Dataset From Another

```
217 Data Second;  
218     Set First;  
219 Run;
```

NOTE: There were 1729 observations read from the data set WORK.FIRST.

NOTE: The data set WORK.SECOND has 1729 observations and 4 variables.

NOTE: DATA statement used:

real time 0.05 seconds

```
220 Proc SQL;  
221     Create table Second as  
222     Select *  
223     From First;
```

NOTE: Table WORK.SECOND created, with 1729 rows and 4 columns.

```
224 Quit;
```

NOTE: PROCEDURE SQL used:

real time 0.00 seconds

Creating New Variables

```
Data Second;  
  Set First;  
  Emotion = Losses  
  / Wealth * 100;  
Run;
```

```
Proc SQL;  
  Create table Second  
  as  
  Select *, Losses /  
  Wealth * 100 as  
  Emotion  
  From First;  
Quit;
```

Creating New Variables

```
236 Data Second;  
237     Set First;  
238     Emotion = Losses / Wealth * 100;  
239 Run;
```

NOTE: There were 1729 observations read from the data set
WORK.FIRST.

NOTE: The data set WORK.SECOND has 1729 observations and 5
variables.

NOTE: DATA statement used:
real time 0.05 seconds

```
240 Proc SQL;  
241     Create table Second as  
242     Select *, Losses / Wealth * 100 as Emotion  
243     From First;
```

NOTE: Table WORK.SECOND created, with 1729 rows and 5 columns.

```
244 Quit;
```

NOTE: PROCEDURE SQL used:
real time 0.05 seconds

Summary Statistics – Entire Dataset

```
Proc Means Data=  
    First Mean;  
    Var Wealth;  
Run;
```

```
Proc SQL;  
    Select mean(Wealth)  
    From First;  
Quit;
```

Summary Statistics – Entire Dataset

SAS Proc Means

Summary Statistics - Entire Dataset

The MEANS Procedure

Analysis Variable : wealth
Mean

7012.72

SQL Select...

Summary Statistics - Entire Dataset

7012.718

Summary Statistics – By Levels

```
Proc Means Data=
    First Mean;
Class Sex;
    Var Wealth;
Run;
```

```
Proc SQL;
    Select Sex,
           mean(Wealth) as
           Mean
    From First
    Group By Sex;
Quit;
```

Summary Statistics – By Levels

SAS Proc Means

Analysis Variable : wealth		
	N	
sex	Obs	Mean
F	864	7015.39
M	865	7010.05

SQL Select...

sex	Mean
F	7015.387
M	7010.052

Multiple Operations 1

What are we doing?

- Create a variable from a dataset,
- Sort the data by created variable,
- Print the data

Multiple Operations 1

```
Data Second;  
  Set First;  
  Emotion = Losses /  
    Wealth * 100;  
Run;  
Proc Sort Data= Second;  
  By Descending  
    Emotion;  
Run;  
Proc Print data=Second;  
  Var Patient Name Phone  
    Emotion;  
Run;
```

```
Proc SQL;  
  Select Patient, Name,  
    Phone, Losses /  
    Wealth * 100 as  
    Emotion  
  From First  
  Order By Emotion Desc;  
Quit;
```

Multiple Operations 1

SAS

Obs	patient	name	phone	Emotion
1	09965380	ETTA DIMARTINO	698-2302	99.4979
2	30132903	DOUGLAS DIMARTINIS	698-2324	98.9439
3	22303322	SARAH DIMARTINIS	698-5777	98.4303
4	95331699	CAROLYN DILLON	698-0646	97.8661
5	70993677	PAUL DILLON	698-5777	97.3597

SQL

patient	Name	phone	Emotion
09965380	ETTA DIMARTINO	698-2302	99.49786
30132903	DOUGLAS DIMARTINIS	698-2324	98.94386
22303322	SARAH DIMARTINIS	698-5777	98.43029
95331699	CAROLYN DILLON	698-0646	97.86608
70993677	PAUL DILLON	698-5777	97.35974

Summarizing By Created Variable

```
Data Second;  
  Set First;  
  Emotion = Losses /  
    Wealth * 100;  
Run;
```

```
Proc Means Data= Second  
  NWAY Mean;  
  Class Sex;  
  Var Emotion;  
Run;
```

```
Proc SQL;  
  Select Sex, Mean(Losses /  
    Wealth * 100)  
    as M_Emotion  
  From First  
  Group By Sex;  
Quit;
```

Multiple Operations 2...

What are we doing?

- Create a variable from a dataset,
- Summarize the created variable by a classification variable, and
- Print by order of the summarized variable

Multiple Operations 2...(SAS)

```
Data Second;  
  Set First;  
  Emotion = Losses /  
    Wealth * 100;  
Run;  
  
Proc Summary Data=  
  Second NWAY;  
  Class State;  
  Var Emotion;  
  Output Out= Third  
    Mean=M_Emotion;  
Run;
```

Code continues...

```
Proc Sort Data=Third;  
  By Descending  
    M_Emotion;  
Run;  
  
Proc Print Data= Third;  
  Var State M_Emotion;  
Run;
```

Multiple Operations 2...(SQL)

```
Proc SQL;  
  Select State, Mean(Losses / Wealth * 100)  
    as M_Emotion  
  From First  
  Group By State  
  Order by M_Emotion DESC;  
Quit;
```

Multiple Operations 2

Obs	state	M_Emotion	state	M_Emotion
1	MA	58.6025	MA	58.60249
2	RI	58.5784	RI	58.57843
3	VT	58.5181	VT	58.51809
4	CT	58.4691	CT	58.46911

Multiple Summary Statistics

```
Data Second;  
  Set First;  
  Emotion = Losses /  
    Wealth * 100;  
Run;
```

```
Proc Means Data=  
  Second NWAY  
  Mean N;  
  Class State;  
  Var Emotion Age;  
Run;
```

```
Proc SQL;  
  Select State, Mean(Losses  
    / Wealth * 100)  
  as M_Emotion,  
  Count(*) as  
  Num_Patients,  
  Mean(Age) as M_Age  
  From First  
  Group By State;  
Quit;
```

Multiple Summary Statistics

SAS Data step & Proc Means

state	Obs	Variable	Mean	N
CT	346	Emotion	58.4691141	346
		AGE	46.1618497	346
MA	691	Emotion	58.6024883	691
		AGE	46.1722142	691
RI	346	Emotion	58.5784260	346
		AGE	46.0289017	346
VT	346	Emotion	58.5180931	346
		AGE	46.0809249	346

SQL Select...

state	M_Emotion	Num_Patients	M_Age
CT	58.46911	346	46.16185
MA	58.60249	691	46.17221
RI	58.57843	346	46.0289
VT	58.51809	346	46.08092

Multiple Summary Statistics: Multiple Classification Vars

```
Proc Means Data= Second  
    NWAY;  
Class State City;  
Var Emotion Age;  
Output Out= Third  
    (Rename=(_FREQ_=  
        Num_Patients))  
    Mean=M_Emotion  
    M_Age;  
Run;
```

```
Proc SQL;  
    Select State, City,  
        Mean(Losses / Wealth  
            * 100) as  
            M_Emotion,  
        Count(*) as  
            Num_Patients,  
        Mean(Age) as M_Age  
    From First  
Group By State, City;  
Quit;
```

Multiple Summary Statistics:

Multiple Classification Variables & Subsetting

```
Proc Means Data= Second
  NWAY;
  Class State City;
  Var Emotion Age;
Where Wealth > 500000;
  Output Out= Third
(Where=(M_Emotion > 50))
  Rename=(_FREQ_=
    Num_Patients))
  Mean=M_Emotion M_Age;
Run;
```

```
Proc SQL;
  Select State, City
    Mean(Losses / Wealth
      * 100) as
      M_Emotion, Count(*) as
      Num_Patients,
    Mean(Age) as M_Age
  From First
Where Wealth > 500000
  Group By State, City
Having M_Emotion > 50;
Quit;
```

Combining Datasets: Concatenation

```
Data Third;  
    Set First Second;  
Run;
```

```
Proc SQL;  
    Create table third as  
        Select *  
        From First  
        Union  
        Select *  
        From Second;  
Quit;
```

Combining Datasets: Concatenation

SAS

```
364 Data Third;
```

```
365     Set First Second;
```

```
366 Run;
```

NOTE: There were 1729 observations read from the data set WORK.FIRST.

NOTE: There were 1729 observations read from the data set
WORK.SECOND.

NOTE: The data set WORK.THIRD has 3458 observations and 9 variables.

SQL

```
384 Proc SQL;
```

```
385     Create table fourth as
```

```
386         Select * From First
```

```
387         Union
```

```
388         Select * From Second;
```

WARNING: A table has been extended with null columns to perform the
UNION set operation.

NOTE: Table WORK.FOURTH created, with 3458 rows and 9 columns.

```
389 Quit;
```

Combining Datasets: Interleaving

```
Proc Sort data = first;  
    By Patient;  
Run;  
  
Proc Sort data = second;  
    By Patient;  
Run;  
  
Data Third;  
    Set First Second;  
    By Patient;  
Run;
```

```
Proc SQL;  
    Create table fourth as  
    Select * From First  
    Union  
    Select * From Second  
    Order by Patient;  
Quit;
```

Combining Datasets: Merging

```
Proc Sort data = claims;  
  By Patient;  
Run;
```

```
Proc Sort data = members;  
  By Patient;  
Run;
```

```
Data Third;  
  Merge Claims(in = in_claims)  
         members(in= in_mems);  
  By Patient;  
  If in_claims and in_mems;  
Run;
```

```
Proc SQL;  
  Create table third as  
  Select *  
  From claims, members  
  Where claims.Patient =  
         members.Patient  
  Order by Patient;  
Quit;
```

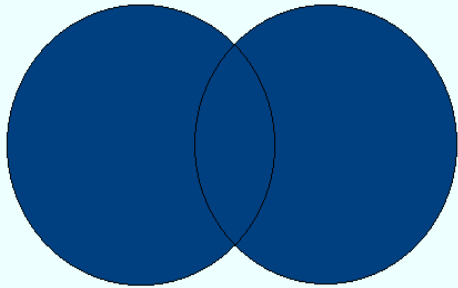
Combining Datasets: Merging (Differently Named Key Variables)

```
Proc Sort data = claims;  
  By Patient;  
Proc Sort data = members;  
  By Patient_ID;
```

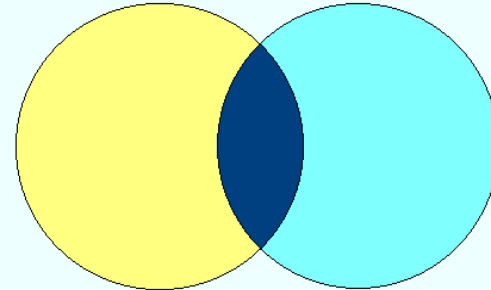
```
Data Third;  
  Merge  
    claims (in = in_claims)  
    members (in= in_mems  
              rename=  
              (Patient_ID=Patient));  
  By Patient;  
  If in_claims and in_mems;  
Run;
```

```
Proc SQL;  
  Create table third as  
  Select *  
  From claims, members  
  Where claims.Patient =  
        members.Patient_ID  
  Order by Patient;  
Quit;
```

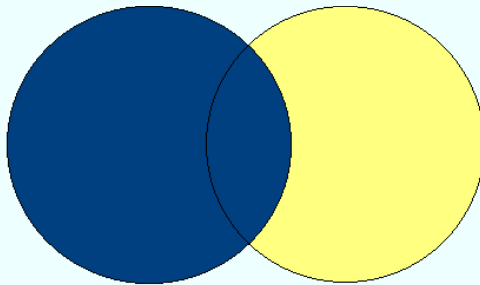
Combining Datasets: Joins



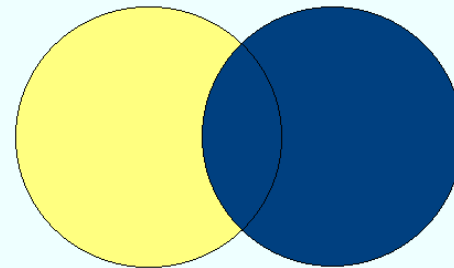
Full Join



InnerJoin



Left Join



Right Join

Joins: Inner

Only Those Records From Both Files That Match

```
Proc Sort data = claims;
  By Patient;
Run;
Proc Sort data = members;
  By Patient;
Run;

Data Third;
  Merge Claims(in = in_claims)
        members(in= in_mems);
  By Patient;
  If in_claims and in_mems;
Run;
```

```
Proc SQL;
  Create table third as
  Select *
  From Claims
Inner join Members
On Claims.Patient=
      Members.Patient;
Quit;
```

Joins: Full

Records From Both Files That Match + All Others

2 sorts are assumed...

```
Data Third;  
  Merge Claims(in = in_claims)  
         members(in= in_mems);  
  By Patient;  
  If in_claims OR in_mems;  
Run;
```

```
Proc SQL;  
  Create table third as  
  Select *  
  From Claims  
  Full join Members  
  On Claims.Patient=  
      Members.Patient;  
Quit;
```

Joins: Left Join

Records From Both Files That Match +
All Remaining Records In *First Named* Dataset

2 sorts are assumed...

```
Data Third;  
  Merge Claims(in=in_claims)  
         members(in= in_mems);  
  By Patient;  
  If in_claims;  
Run;
```

```
Proc SQL;  
  Create table third as  
  Select *  
  From Claims  
  Left join Members  
  On Claims.Patient=  
      Members.Patient;  
Quit;
```

Joins: Right Join

Records From Both Files That Match +
All Remaining Records In *Second Named Dataset*

2 sorts are assumed...

```
Data Third;  
  Merge  
  Claims(in=in_claims)  
    members(in= in_mems);  
  By Patient;  
  If in_mems;  
Run;
```

```
Proc SQL;  
  Create table third as  
  Select *  
  From Claims  
  Right join Members  
  On Claims.Patient=  
    Members.Patient;  
Quit;
```

Multiple Operations 3...

What are we doing?

- Merge 2 datasets/tables,
- Create a variable dependent on a variable from each merged dataset, and
- Subset by row, based on the value of the calculated variable

Multiple Operations 3...

2 sorts are assumed...

```
Data Third;  
  Merge Claims(in = in_cl)  
    Members(in= in_mem);  
  By Patient;  
  If In_cl and In_mem;  
  AgeEventDays= Date-DOB;  
  If AgeEventDays > 10000 ;  
  Keep Patient Sex Age  
    AgeEventDays;  
Run;
```

```
Proc SQL;  
  Create table third as  
  Select Patient, Sex, Age,  
    Date - DOB as  
      AgeEventDays  
  From Claims as C  
  Inner join Members as M  
  On C.Patient =M.Patient  
  Having AgeEventDays >  
    10000;  
Quit;
```

Using Subqueries

What are we doing?

- **Subset** one of 2 source tables, used in a merge
- Merge the 2 datasets/tables,
- Create a new variable, that is dependent on a variable from each merged dataset

Using Subqueries

```
Proc Sort data = Members  
  (Where= (Age > 64))  
  Out = Elders;  
  By Patient;  
Proc Sort data = Claims;  
  By Patient;  
Data Third;  
  Merge Claims (in = in_cl)  
             Elders(in= in_elders);  
  By Patient;  
  If In_cl and In_elders;  
  keep patient doctor date  
  claimid;  
Run;
```

```
Proc SQL;  
  Create table fourth as  
  Select patient, doctor,  
         date, claimid  
  From Claims  
  Where patient in  
    (Select Patient  
     From Members  
     Where Age > 64);  
Quit;
```

Performance:

Summarization (SAS)

```
proc summary data=claims nway;  
  class doctor patient;  
  var paid days;  
  output out=Large1Class1Var  
    (rename=(_freq_=ClaimN) drop=_type_)  
    sum=SumPaid SumDays  
    mean=MeanPaid MeanDays;  
run;
```

Performance:

Summarization (SQL)

```
proc sql;  
  create table Large1Class1VarSQL as  
  select doctor, patient, sum(paid) as  
    SumPaid, count(*) as ClaimN, sum(Days)  
    as SumDays, Mean(paid) as MeanPaid,  
    Mean(Days) as MeanDays  
  from claims  
  group by doctor, patient;  
quit;
```

Performance: Summarization *Results*

# of obs	Total# of vars	# of class vars	# of analysis vars	# of stats	SAS	SQL
1,133,538	53	1	1	1	1.8	7.8
1,133,538	53	1	1	2	1.8	8.0
1,133,538	53	1	2	3	1.8	8.4
1,133,538	53	2	2	5	3.8	12.7

Performance:

Merging (SAS)

```
Proc Sort data = SQLPre.Claims  
  (keep = Patient Doctor Paid  
  Days)  
  Out = Claims;  
  By Patient;  
Run;
```

```
Proc Sort data =  
  SQLPre.Members  
  (keep = Pat_ID DOB Age)  
  Out = Members;  
  By Pat_ID;  
Run;
```

```
Data PatClSAS;  
  Merge Claims(In = In_Cl)  
  Members (In= In_Mem  
  rename=(Pat_ID=Patient));  
  By Patient;  
  If In_Cl and In_Members;  
Run;
```

```
Proc sort data= PatClSAS;  
  By Descending Age DOB;  
Run;
```

Performance: Merging (SQL)

Proc SQL;

Create table PatientClaimsSQL as

Select Patient, Doctor, Paid, Days, Age

From

SQLPre.Claims as C

inner join

SQLPre.Members as M

on C.Patient = M.Pat_ID

Order by Age Desc , DOB;

Quit;

Performance: Join/Merge

Results

Dataset 1		Dataset 2						
				# of key vars	Order by # of Vars	Resulting # of Rows		
# of obs	# of vars	# of obs	# of vars				SAS	SQL
1,133,538	53	174,868	3	1	0	1,133,538	10.7	6.3
1,133,538	53	174,868	3	1	1	1,133,538	28.2	30.2
1,133,538	53	174,868	3	1	2	1,133,538	26.9	30.7
1,133,538	53	1,000	3	1	0	5,403	39.6	13.1
1,133,538	53	1,000	3	1	1	5,403	32.9	13.0
1,133,538	53	1,000	3	1	2	5,403	37.2	15.4

Finale

- PROC SQL is an additional tool with its own strengths and challenges
- Imperatives of data accuracy require examining results as you learn and use PROC SQL

Questions?...

Contact Information

Robert Rosofsky

Health Information Systems Consulting

www.HealthInfoSys.net