SAS seminar – Proc SQL

2004-10-26

Gustaf Edgren
MEB, Karolinska Institutet

Outline

- Background
  - SQL history and terminology
- Introduction
  - Proc SQL
  - Intro to SQL
- Syntax
  - Retrieving data
  - Modifying data
  - Summary functions
  - Combining tables
- Conclusion

Non-outline

- I will NOT cover:
  - How to create or modify tables
  - How to append tables
  - How to perform sub-queries
  - How to work with views
  - Database specific issues like relational integrity, primary keys etc.

…Maybe next time?

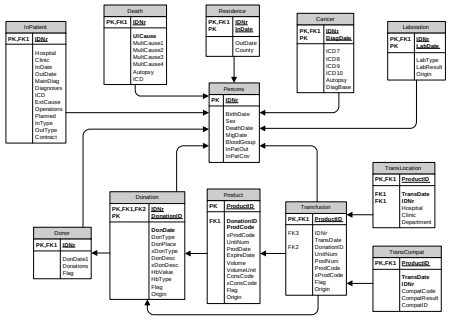
Background – SQL history

- Structured Query Language.
- Developed by IBM in the 1970s for querying, modifying and updating relational databases
- Adopted and standardized by ANSI and ISO during the 1980s
- Used with almost all RDBMS (Relational Data Base Management System) e.g. Oracle, DB2, Access etc.
- Most RDBMS also offer enhancements to ANSI SQL

Background - terminology

SAS Data step	Proc SQL
Dataset	Table
Variable	Column
Observation	Row
Merge	Join
Append	Union

Background – example database



Introduction – what is Proc SQL?

- Proc SQL is the SAS implementation of SQL
- Proc SQL is a powerful SAS procedure that combines the functionality of the SAS data step with the SQL language
- Proc SQL can sort, subset, merge and summarize data – all at once
- Proc SQL can combine standard SQL functions with virtually all SAS functions
- Proc SQL can work remotely with RDBMS:s such as Oracle

Introduction – syntax

- Very straightforward and extremely structured syntax
- Highly scalable, from simple one row queries, to complicated queries with several sub-queries
- Possible to combine with <u>all</u> SAS functions (except the lag function)

Retrieving data

- The simplest SQL queries only retrieve and subset data from a specified source
- This is can usually be achieved just as easily with SAS code
- Some advantages with SQL does however exist…

Retrieving data – syntax

proc sql;
create table *tablename* as
select [distinct]
column1,
column2,
[*], …
from *library.table*
where *expression*
order by *column1 etc.*;
quit;
* = all columns

Example 1

- With our example database in mind, how would one create a table with all the men in the database?

Example 1

SQL approach

proc sql;
create table *men* as
select *
from *cblood.persons*
where sex = 1;
quit;

SAS approach

data men;
set cblood.persons;
where sex=1;
run;

SAS – SQL 1 – 0

Example 2

- What if we wanted a table with all men and their birthdates; sorted ascending by birthdate?

Example 2

SQL approach

```
proc sql;
   create table men as
   select
      idnr,
      birthdate
   from cblood.persons
   where sex = 1
   order by birthdate; *
   quit;
```

SAS approach

```
data men;
   set cblood.persons;
   where sex=1;
   keep idnr birthdate;
run;

proc sort data=men;
   by birthdate;
run;
```

* Sort performed by data source

SAS – SQL 1 – 1

Example 3

- Well then, that was simple (?), what if we wanted a list of all the people that ever received blood?

Example 3

SQL approach

```
proc sql;
   create table patient as
   select
      distinct idnr
   from cblood.transfusion;
   quit;
```

SAS approach

```
data patient;
   set cblood.transfusion;
   keep idnr;
run;
proc sort data=patient
   nodup;
   by idnr;
run;
```


SAS – SQL 1 – 2

Modifying columns

- SQL, just like SAS, offers the possibility to create new columns (variables) with:
  - New values
  - Other columns (variables)
  - Combinations of new values and other columns
  - SAS and/or SQL functions applied to any value or column

Modifying columns – syntax

proc sql;
create table *tablename* as
select
function(*column1) as newcolumn1*,
column2 [+|-|*|/] *column3 as newcolumn2*,
…
from *library.table;*
quit;

Example 4

- Lets say we want a table with the age (in years) at death of all the people in our cohort that have died this far
  - Simple?

Example 4

SQL approach

```
proc sql;
   create table dead as
   select
      idnr,
      (deathdate-birthdate)
      /365.25 as deathage
   from cblood.transfusion
   where deathdate ^= .;
   quit;
```

SAS approach

```
data dead(keep=idnr
   deathage);
   set cblood.transfusion
   (keep=idnr birthdate
   deathdate);
   where deathdate ^=.;
   deathage=(deathdate-
   birthdate) /365.25;
run;
```


SAS – SQL 2 – 3

Example 5

- So, what if we want to use a SAS function? How do we do that?
- Lets extract the blood central ID from the donation ID to see what blood centrals have been involved

Example 5

SQL approach

```
proc sql;
   create table blc as
   select distinct
      substr(donationid,2,3)
         as blc
   from cblood.donation;
   quit;
```

SAS approach

```
data blc(keep=blc);
   set cblood.donation
   (keep=donationid);
   blc=substr(donationid,3,3);
run;
proc sort data=blc nodup;
   by blc;
run;
```


SAS – SQL
3 – 4

Summary functions

- SQL also has the ability to summarize data
- Counts, means, etc are easily calculated and presented or stored in new or existing tables

Summary functions – syntax

```
proc sql;
  create table tablename as
      select function(*) as alias
      from libname.table
      group by byvariable1
      having conditions;
  quit;
```


Example 6

- Lets say we want to calculate the total number of donations per person.
- How does one do that?

Example 6

SQL approach

```
proc sql;
   create table donations as
   select
      idnr,
      count(*) as count
   from cblood.donation
   group by idnr;
   quit;
```

SAS approach

```
data temp;
    set cblood.donation;
    keep idnr;
proc freq;
    table idnr /
    out=donations(keep=idnr count);
run;
```


SAS – SQL 3 – 5

Example 7

- In order to protect donors, the maximum number of whole blood donations one is allowed to make each year is limited to four (men) or three (women)
- Is there a simple way to identify people who have given too many times?

Example 7

SQL approach

```
proc sql;
   create table toomany as
   select
      idnr,
      year(dondate) as year
   from cblood.donation
   where sex=1
   group by idnr, year(dondate)
   having count(*) > 4;
   quit;
```

SAS approach

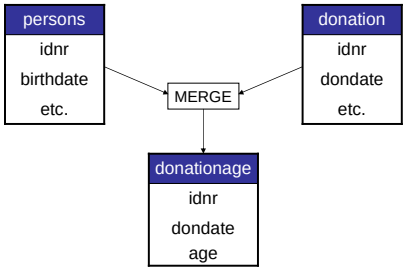
```
data temp(keep=idnr year);
   set cblood.donation(keep=idnr
   dondate);
   where sex=1;
   year=year(dondate);
proc freq data=temp noprint;
   table idnr*year / out=temp
   (keep=idnr year count);
data toomany;
   set temp;
   where count > 4;
run;
```


SAS – SQL 3 – 6

Combining tables

- So, what if we want to select data from multiple tables and store it in one table?
- In essence what we want to do is perform a merge, or as it is called in SQL a join
- Lets look at an example; we want to join the person and donation table and calculate age a each donation

Example 8



Example 8

- The SAS code is simple, but how do we do it with SQL?
- Several possible ways exist, lets start with the simplest variant

Syntax – combining tables

proc sql;
create table *tablename* as
select
[*alias1.column, alias2.column*, *, etc.]
from
libname.table1 as *alias1, libname.table2* as *alias2*
where *alias1.column=alias2.column*;
quit;

Example 8

SQL approach

```
proc sql;
    create table donationage as
    select
        a.idnr,
        a.dondate,
        %age(a.dondate,
            b.birthdate) as age
    from cblood.donation as a,
        cblood.persons as b
    where a.idnr=b.idnr;
    quit;
```

SAS approach

```
proc sort data=cblood.donation
    out=donation;
    keep idnr dondate;
    by idnr;
proc sort data=cblood.persons
    out=persons;
    keep idnr birthdate;
    by idnr;
data donationage
    merge donation(in=a)
        persons(in=b);
    by idnr;
    if a and b;
    age=%age(dondate, birthdate);
run;
```

SAS – SQL 3 – 7

Syntax – combining tables 2

```
proc sql;
   create table tablename as
   select
        [alias1.column, alias2.column, *, etc.]
   from
        libname.table1 as alias1
        [inner | outer | left | right] join
        libname.table2 as alias2
        on alias1.column=alias2.column;
   quit;
```


Syntax – combining tables 2

- So what's a full / inner / left / right join?
  - Full join:
    - if a or b;
  - Inner join:
    - if a and b;
  - Left join:
    - if a;
  - Right join:
    - if b;

```
data newtable;
   merge
        table1(in=a)
        table2(in=b);
   by keyvariable;

   if ?????;
run;
```


Example 9

- So, lets create a table with all donors and their calculated time as blood donor and all the cancer events they ever had

Example 9

SQL approach

```
proc sql;
    create table cancerdonor as
    select
        a.idnr,
        (max(dondate)-min(dondate))
            / 365.25 as dontime,
        b.icd7,
        b.diadate
    from cblood.donation as a
        left join cblood.cancer as b
    on a.idnr=b.idnr
    group by a.idnr, b.icd7, b.diadate;
    quit;
```

SAS approach

Anyone wants to have a go?

SAS – SQL
3 – 8

Combining tables – SQL pros

- Regarding performance and typing, probably a draw, but:
  - SQL allows merging (joining) where key variables have different names
  - SQL does not require sorting
  - SQL allows remote processing of query – hence your computer will remain available

Conclusions

- Proc SQL wont replace the SAS dataset, but is a useful tool when:
  - Working with multiple large datasets
  - Working remotely against a database server
  - Performing complicated merges of multiple datasets
- And don't forget, SQL beat SAS 8 – 3!