Paper 3455 - 2015

Nifty Uses of SQL Reflexive Join and Sub-query in SAS®

Cynthia Trinidad, Theorem Clinical Research, Orange County, CA

ABSTRACT

SAS SQL is so powerful that you hardly miss using Oracle PL/SQL. One SAS SQL forte can be found in using the SQL Reflexive Join. Another area of SAS SQL strength is the SQL sub-query concept.

The focus of this paper is to show alternative approaches to data reporting and to show how to surface data quality problems using Reflexive Join and Sub-query SQL concepts.

The target audience for this paper is the intermediate SAS programmer or the experienced ANSI SQL programmer new to SAS programming.

INTRODUCTION

According to Katherine Prairie[1], reflexive join or a self-join is the ability to join a table to itself thereby allowing you to search the table second time using information selected from the first pass through the table. I like how succinctly Jessica Hampton[2] defines it – "'Sometimes, you may want to use a table twice in the same query'.

Additionally, Prairie defines sub-query as the ability to retrieve information from one or more tables to control the rows retrieved from another table.

In Data Management, there are instances where one can utilize both reflexive join and sub-query concepts to generate/satisfy varying report requirements. I will go over a couple of these situations using SQL code that I have used and tested. Before I show you the SQL code, here are notes and hints.

The Reflexive Join:

a) Usually is on a table that contain recursive data (where one column's value has many associations within another column). An example of recursive relationship is that of the employees table - there are many employees to one supervisor.
b) Require usage of aliases.

The Sub-query:

a) *Easily* compares each variable in a SELECT statement to a set of values retrieved from the same or another table.
b) Each statement relies on the outcome of the nested statement/query.

CONCEPTS APPLICATION:

Example #1: Generate report that tracks subject's visits

A reflexive join is when you join a table to itself. In Clinical Data Management, one instance where this is used is in tracking subject visits using one long table, such as Vital Signs (VS) measurement data. VS data has many visit rows to one subject (a recursive relationship).

To track the subject visits, vertical subject + visit data needs to be translated (/transposed) into horizontal, one row per subject with multiple visit columns. The desired result should quickly show which visit a subject has completed, missed or have yet to complete. Let's see below how the use of reflexive join helps accomplish this goal.

SUBJECT	SBP	DBP	VISITDT	VISIT
A001	97	47	8-Jun-2012	SCR
A001	114	66	7-Jul-2012	W4
A001	113	69	4-Aug-2012	W8_EOS
A002	98	67	25-Oct-2012	SCR
A003	87	48	20-Sep-2012	SCR
A003	113	75	17-Oct-2012	W4
A003	105	81	14-Nov-2012	W8_EOS
A004	137	73	24-Oct-2012	SCR
A002	100	71	19-Dec-2012	W8_EOS

← sample Vital Signs (VS) Measurement data

In the reflexive SQL statement below, I repeatedly read visit fields but each time assigned different aliases and qualifying criteria in the where clause from the same Vitals table to produce horizontal transposition.

```
proc sql;
 create table pt_track as
 select a.subject, a.visit as V1, a.visitdt as VDT1,
                   b.visit as V2, b.visitdt as VDT2,
                    c.visit as V3, c.visitdt as VDT3
 from vitals a,
      vitals b,
      vitals c
 where a.subject = b.subject and b.subject = c.subject and
       a.visit = 'SCR' and b.visit = 'W4' and c.visit = 'W8_EOS';
quit;
```

← These are 3 distinct Visit values

← reflexive nature = reading same table more than once but varying aliases

RESULT:

SUBJECT	V1	VDT1	V2	VDT2	V3	VDT3
A001	SCR	8-Jun-12	W4	7-Jul-12	W8_EOS	4-Aug-12
A003	SCR	20-Sep-12	W4	17-Oct-12	W8_EOS	14-Nov-12

← A002 & A004 are missing from list

Although the statement above has the basic elements of a SQL reflexive join, this only resulted in listing equi-joined table values - only those subjects (A001 and A003) that have values for all three visits were listed. The ones (A002 and A004) with some missing data were excluded.

CORRECTED SOLUTION:

A true tracking report will list missing Visit information along with data that is present. The modified query shown below uses Outer Join to the same table with nested conditionals (*/where* clause) to satisfy this report requirement.

```
proc sql;
   create table pt_track as
   select coalesce(a.subject, b.subject) label = 'Subject' as subject,
          coalesce(a.visit, b.visit) as V1 ,
          coalesce(a.visitdt, b.visitdt) as VDT1,
          b.folder as V2, b.visitdt as VDT2,
          c.folder as V3, c.visitdt as VDT3
     from vitals (where=(folder = 'SCR')) as a Full JOIN
          vitals (where=(folder = 'W4')) as B
       on a.subject = b.subject
          Full JOIN vitals (where=(folder = 'W8_EOS'))  as c
       on a.subject = c.subject;
quit;
```

←Coalesce function applied here to ensure fields returned are not missing

RESULT:

SUBJECT	V1	VDT1	V2	VDT2	V3	VDT3
A001	SCR	8-Jun-12	W4	7-Jul-12	W8_EOS	4-Aug-12
A002	SCR	25-Oct-12			W8_EOS	19-Dec-12
A003	SCR	20-Sep-12	W4	17-Oct-12	W8_EOS	14-Nov-12
A004	SCR	24-Oct-12				

Example #2: Identify ECG data quality issues.

The reflexive sub-query offers easy comparison of variables in a SELECT statement to a set of values retrieved from the same table. In Clinical Data Management, one instance where this is used is in identifying discrepancy with lab data. In this example we need to identify where data content (chronological ordering) conflicts with the date content (actual collected date).

ECG DATA

SUBJECT	Visit	Scan Seq	EGDT	EGTM
1001	SCR	Pre-scan 1	26-Sep-12	7:04
1001	SCR	Pre-scan 2	24-Sep-12	8:02
1001	SCR	Pre-scan 3	24-Sep-12	8:43
1002	SCR	Pre-scan 1	24-Sep-12	6:56
1002	SCR	Pre-scan 2	24-Sep-12	7:56
1002	SCR	Pre-scan 3	24-Sep-12	8:39

← Many Scan records are associated with the same Subject+Visit+Scan Seq

To surface the problematic records, in the SQL below the primary statement selects ECG variables and the primary statement relies on the outcome of the nested statement/query.

Each row of ECG Table A is held constant while the sub-query traverses through the same table's records searching for values that match the where clause. Thus, promoting a side-by-side comparison of each subject's rows.

```
proc sql;
   create table DifEcgDat as
   select Subject, Visit, Scan_seq, Egdt, Egtm
     from ECG a
     where subject in
        (select distinct subject
           from ECG
           where egdt NE a.egdt and
                 visit EQ a.visit)
   order by subject, visit, egdt;
quit;
```

← The power in the sub-query lies on using the alias, ***a***; without it there can be no relationship established between the nested variable to the primary variable.

RESULT

SUBJECT	Visit	Scan Seq	EGDT	EGTM
1001	SCR	Pre-scan 2	24-Sep-12	8:02
1001	SCR	Pre-scan 3	24-Sep-12	8:43
1001	SCR	Pre-scan 1	26-Sep-12	7:04

←The query identified the inconsistency as Subject 1001's observation: Pre-scan 1 date, 26-Sep-12, is different from Pre-scan 2 and 3 date, 24-Sep-12.

▲Subject 1002 is not in the output because after comparison the query identified all date values were consistent.

The Sub-query is also a very powerful feature when relating more than just one table, in multiple tables and when used in conjunction with SQL operations UNION, EXCEPT and INTERSECT. Lei Zhang and Danbo Yi[3] said it best "SAS SQL feature provides great flexibility in manipulating and querying data in multiple tables simultaneously."

CONCLUSION

The SAS SQL concepts discussed above provide the Data Management SAS Programmer alternatives to the usual SAS Data Step logic. In certain cases, writing code using SQL can: entail less code, facilitate less errors and be all around more efficient. It all hinges on applications available to and the comfort level of the Programmer. Even with the Reflexive SQL examples above, there is still more than one option for the Programmer. He/she is in control. These are the primary reasons why I don't miss using Oracle SQL, the first SQL language I programmed with.

REFERENCES

1. Katherine Prairie, "The Essential PROC SQL Handbook"
2. "Reflexive Joins in SAS with Proc SQL" Jessica Hampton SAS Tips, October 18, 2013
3. "Working with Subquery in the SQL Procedure" by Lei Zhang and Danbo Yi, NESUG98 Proceedings
4. "Joined Table" article, SAS Base 9.2 Procedures Guide, website URL = http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a002473691.htm

CONTACT INFORMATION

Cynthia Trinidad
Theorem Clinical Research
Email Cynthia.Trinidad@TheoremClinical.com
Mobile/Cell 714 423 5095