USING PROC SQL TO CREATE AD HOC REPORTS

Anne Marie S. Smith, Information Systems Engineer, Philadelphia, Pa.

ABSTRACT

The purpose of this paper is to demonstrate the powerful features of PROC SQL in the creation of ad hoc reports for clinical trial research. The reports created are for the display, summation and statistical analysis of clinical trial research data. The SAS programs discussed are flexible and summarize Phase I to Phase III clinical trial data; the adverse events, study medications and potentially clinically significant lab test values. In the SAS programs, the use of PROC SQL affords for shorter, easier programs with short execution times, while making SAS program code efficient, flexible and maintainable.

INTRODUCTION

PROC SQL is used for ad hoc report programming, to perform the following tasks: 1) data manipulation of multiple views in Oracle and 2) creation of the report of clinical trial research data. You preplan and do ad hoc joining of multiple Oracle views: demographic data with study medications, laboratory tests and adverse events views; by the key fields; patient and investigator identification, test ids, start date, stop date, relative or sequence day and treatment period or visit code for your reports.

These associations of different alphanumeric data types, which are potentially relateable by the six key fields, are needed to obtain the desired result; a correct investigator and patient based ad hoc report. You need to achieve the correct results on the patient basis, while dealing with the issues of overlap among treatment periods and start and stop dates. On a investigator and patient basis, you need to associate correctly by the relative day or sequence day of the clinical trial study. "The correct approach to get the intended result from the query, of course, would be one which, on a patient basis, somehow accounted for the overlapping time frames between the Startdates and EndDates. But users cannot express such a relationship using the SQL query tools generally available in CANDA's. So the key to easy and correct use of a CANDA in this situation (concomitant meds and adverse events) is to pre-define and pre-code the rule for combining this data during querying; in essence, to leave nothing up to the user (Cohen 1995, pg.8)."

PROBLEM

While participating on a Clinical Project Team, I was faced with the following problem. Much more information on our IV study drug was needed for safety monitoring purposes. This was a Phase I drug study and it was still ongoing at this point in time. I received detailed derived data requests from the CPT statistician for the following new information.

1) Total Dose
2) Days in Study
3) Last Cycle number

Additionally, the last dose date was the first day of a 28 day review period, and not really considered post study information.

SOLUTION

My solution was this; to write 3 short SAS programs, 50 lines or less, and create an informative study medication report summary. This report summary included all of the essential keys, investigator and patient identifications with the start of study day , the dose group and the newly derived data fields of days in study, total dose and last cycle number. Now all of the requested information for the continual safety monitoring of our study drug was contained within one nifty study medication summary report.

This programming effort took 1.5 days to design, write the SAS programs, test and validate the output. The choice of a 3 part design was due to my years of software development training and my knowledge of IEEE software engineering standards. My programs and output are included in the appendix following my paper.

DESIGN

The design of our Ad Hoc Reports needs to be a well thought out and planned and can be easily and readily displayed in table form or with spreadsheets. You need to predefine and precode how data of various alphanumeric types need to be associated at the view level to combine and accurately satisfy the ad hoc report request. You'll dynamically implement the design with PROC SQL which is powerful, flexible, easy and accurate.

Using PROC SQL you will create associations of different data types in the clinical Oracle database

and you will query the data views on an investigator and patient basis. The views that will be needed for the examples are the Patderiv view with the demography data, Patmeds view with the study medication data, Patstev view with the adverse events data and Patlabs view with the lab test results.

Design Overview - Pseudocode

1) Data access - Set up the input/output environment, all library definitions for Slice, Supplemental, Macro and Format libraries and set SAS programming options such as page size and date.

2) Ad Hoc queries - Extract clinical data records by vital keys; investid, patid, testid, startdt, stopdt, tstdt and seq_day, from views in slice and supplemental clinical drug research data libraries.

3) Correct repetitious data dynamically by summarizing on vital keys and collapsing records of data. Join data of different types during the PROC SQL queries and create new views to satisfy the clinical ad hoc report request. This is considered background processing and is well stated in the literature. "That is, it concerns the data access, the ad hoc querying, the removal of repeated data after joining data of different types during querying, the data subsets that are saved and retrieved, and the count and statistics reports that are produced by the system (Cohen 1995,pg.10)".

4) Save the new query which will create the clinical trial data subset; retrieve and output the result ad hoc report which includes the appropriate titles and headings in a clear readable format for the investigator and patient ids, test ids, dates including relative or sequence days, counts and statistical analysis with graphs.

DATA ASSOCIATIONS , MULTIPLE DATA VIEWS

PATMEDS ***P_MEDS*** ***P_WONDER*** ***PATDER01***

PATMEDS	<------->	P_MEDS	<------->	P_WONDER	JOIN	PATDER01		P_FNL01
Investid		investid		investid	<----->	investid	---->	investid
Patid		patid		patid	<----->	patid	---->	patid
Startdt	MIN	startdt		startdt		invtxtlg		startdt
Stopdt	MAX	stopdt		stopdt		tretxtlg		stopdt
Startdt	MAX	lst_dsdt		lst_dsdt		tretxtsh		lst_dsdt
Stopdtp		stopdtp		stopdtp		thecode		stopdtp
Startdt	CALC	duration		duration				duration
Duratio		duratio	Collapse	duratio				duratio
Treatco		treatco	<------->	treatco				treatco
Trpstrng	INPUT, SUM	tot_dose		tot_dose				tot_dose
Trpstrng	INPUT, MIN	dosegrp		dosegrp				dosegrp
Visitco	INPUT, MAX	lst_cycl		lst_cycl				lst_cycl
								invtxtlg
								tretxtlg
								tretxtsh
								thecode

An explanation of the data is given to you at this point to help you with some of the derivations in the above table. The lst_dsdt variable, the last dose date, is derived by using the maximum function on the startdt, the start date variable. The trpstrng variable, the treatment strength value, stored in character form is converted to numeric and then summed up with the sum function to derive the total dose variable. At this point let me just say that any numbers which are stored as characters can be converted to the true numeric representation and only then will correct arithmetic calculations be possible. Incorrect results will be obtained if you try to perform arithmetic calculations on numeric character strings. Again, the trpstrng variable is converted from character to numeric via the input function and its minimum value is obtained by using the minimum function. Now you derive the dose group and call it the dosegrp variable. Finally, the visitco variable, the visit code, is converted to its numeric representation by the input function and then its maximum value is derived via the maximum function; and now you've got the lst_cycl variable, the last cycle number. This study medication is a liquid dose of variable strength measured in milligrams and dosed intravenously in cycles. SEE APPENDIX FOR PROGRAMS AND LOGS

SOME SHORT EXAMPLES

To get started, lets begin with a couple of sample programs to give you patient outlier information for the blood glucose test taken durind the clinical trial study. The query is simply this, create a view of all the patients whose blood glucose test value is greater than 126 and give the date and time it occurs. Assume lab records exist in the Patlabs view with investigator, patient, test id's, dates including sequence day and that the format $el_name exists in our format library.

```
PROC SQL;

   Create Supp.Outliers as
      Select investid||patid as patient label=PATIENT format=$char9.,
         tstdt label=DATE:TIME format=datetime16.,
         seq_day label=SEQUENCE_DAY,
         /* default numeric format taken here */
         testid format=$el_name. label=TEST_NAME ,
         put(testid, $el_name.) as test,
         tstvalql label=VALUE format=8.
      From Slice.Patlabs
      Where calculated test='GLUCOSE' and tstvalql > 126
      Order by 1, 2, 3;

Quit;
```

Our next example is another query but this time you want to ask if there were any adverse events associated with a patient's elevated glucose level. Again please assume that the adverse events are in the Patstev view with the appropriate vital keys and sequence day. At this point you may ask why use PROC SQL rather than PROC SORT and the DATA STEP to solve my programming problems. In my opinion, PROC SQL is more straight foward in it's approach to the problem. You pick over the data fields, choose exactly which data fields are needed from each view. You have the capability of deriving new information from existing data fields while setting up the necessary format and label. I like PROC SQL's stlyle, it's a classy procedure.

When you use the DATA STEP, you need to presort and preindex your data before reading it into the DATA STEP so that you do not lose any records. PROC SQL does the sorting and indexing automatically for you. Finally I believe that one can safely assume that both methods require just sbout the same workspace. My best guess is that the DATA STEP in certain cases requires much more I/O work because every record is read. There are some wonderful papers in the SUGI 22 conference proceedings, while you may want more information from the SAS experts on PROC SQL versus PROC SORT and the DATA STEP.

```
PROC SQL;

    Create view Supp.GluAE as
    Select a.*,
           b.event label=ADVERSE_EVENT
format=$char60.
     From Supp.Outliers a,
          Slice.Patstev b
     Where a.patient=b.investid||b.patid And
a.seq_day=b.seq_day
     Order by 1, 2;

      Select *
         from Supp.GluAE;
Quit;
```

Your output was automatically sorted by patient and the test date and it looks like the following table.

PATIENT	DATE:TIME	TEST_NAME	VALUE	SEQUENCE _DAY	EVENT
101010001	07JUL97:09:00:00	GLUCOSE	136	15	Nausea
101010004	08JUN97:08:00:00	GLUCOSE	140	24	Headache
101020003	09APR97:08:30:00	GLUCOSE	129	12	Headache
101040002	01JUN97:08:00:00	GLUCOSE	140	15	Heart Palp

SUMMARY

I've attempted to demonstrate with my study medication summary example that PROC SQL is easy to use and allows the Information Systems Engineer and Programming community much flexibility to select various views with many alphanumeric data types of their choice for the creation of their reports. Once I selected my clinical trial database views and the key variables, its was easy to subset, join on keys and manipulate the research data by mathematical formulas. It's easy to use functions such as the MIN and MAX, or to convert using the INPUT function, and rename the new data to other data types for the display and creation of the ad hoc clinical trial report.

I've also demonstrated one approach to making SAS programs more maintainable by a 3 part program design: the setup of global drug data parameters in the first SAS program; the data manipulation step by subsets of records, joins of data on keys and mathematical formulas or functions in the second program; and finally the display and creation of the ad hoc clinical trial data in the third SAS program. This 3 part setup is easy to maintain across projects and runs quite efficiently and the SAS programs and logs are provided for your review.

REFERENCES

- 1) SAS Language: Reference Version 6 First Edition, SAS INSTITUTE, Inc.
- 2) SAS Guide to the SQL Procedure, Usage and Reference Version 6 First Edition and SQL Processing with the SAS System, SAS INSTITUTE, Inc.
- 3) "The choice of SAS as your Application Development Tool for your CANDA", NESUG 1995 Conference

proceedings, written by Barry Cohen, Planning Data Systems and President Of PhilaSUG.

- 4) SAS Guide to Macro Processing, Version 6 Second Edition and SAS Macro Language Course Notes, SAS Institute Inc.

ACKNOWLEDGMENTS

Many Thanks to The PhilaSUG board members, especially to our Webmaster, Robert Schechter, our President , Barry Cohen and Perry Watts for their help, encouraging words, comments and support. Finally I want to thank Dr. John O'Neill at LaSalle University, for sharing his technical expertise on Structured Query Language with me.

Contact Information:

Anne Marie S. Smith
Information Systems Engineer
145 King of Prussia Road
Radnor, Pa. 19087
Tel: 1-610-341-2707(W), 215-654-1394(H)
E-mail: smitha@war.wyeth.com (Work)
annie1126@aol.com (Home)

APPENDIX

```
*****  WONDER 101 GLOBAL PARAMETER FILE   GLBPOST1.SAS ;
*****  Written by by A. M. Smith on 2/06/97;
/*%ALLOCLIB(WONDER,P101); for now */
%let subset = 1;        /* Patient Subset Numberfor report title */
%let rev ='1';        /* readyrev = '1' for patient subset, N=1 */
%let drug = WONDER;             /* DRUGNAME */
%let dno = '664';          /* Text number */
%let drugname = WONDER ;   /* DRUGNAME for Report titling  */
%let SUBPATT = PATIENT;     /* Selection text for title in capitals */
%let protocol = 007ACE-101-US ;  /* Protocol text for report titling */
%let prot = 101;               /* Protocol number */
%let ps = 60;                  /* Set page size = 60 lines */
*****%let reportno = &reportno (SUBSET N );
%macro database;
  %if &outdest ^=  %then
    %let outdest = &outdest.&subset;
  %if &subset ^=  %then
    %let reportno = SUBSET( &subset);
%mend database;
***** WONDER DRUG Formulation, FORMAT Details ************************;
Proc Format;
  Value Drugform
   00.001-00.500 = 'WONDER 0.50 mg.'
   00.501-01.000 = 'WONDER 1.00 mg.'
   01.001-01.500 = 'WONDER 1.50 mg.'
   01.501-02.000 = 'WONDER 2.00 mg.'
   02.001-03.000 = 'WONDER 3.00 mg.'
   03.001-04.000 = 'WONDER 4.00 mg.'
   04.001-05.000 = 'WONDER 5.00 mg.'
   05.001-06.000 = 'WONDER 6.00 mg.'
   06.001-07.000 = 'WONDER 7.00 mg.'
   07.001-08.000 = 'WONDER 8.00 mg.'
   08.001-09.500 = 'WONDER 9.00 mg.'
   other         = 'UNKNOWN        ';
Proc Format;
   Value $DRID
   '10132' = 'SHERLOCK HOLMES, M.D.'
```

```
   '10138' = 'JOHN WATSON, M.D.';
%database
*============ end of file ==========================;

*========= SASPOST1.SAS ==================;
**** The Data Manipulation Program, PhilaSUG Poster 2/26/97******;
**** Author - Anne Marie S. Smith on  2/6/97 *******;
*= Make P_FINAL0N views for Patient Subsets,N=1-->PROJSUP LIBRARY**;
**** Projsup Library will contain the views we create after a *****;
**** sucessful run of this program which is verifiable with the ***;
**** SAS log output. ***********************************************;
**** View Names: P_Meds-Medication Data, P_Wonder-Wonder Rx Subset*;
**** P_Fnl0N-Wonder drug Subset joined with existing Demographics;
**** Data from the Patderiv view **********************************;
options center ps=60 ls=132 mautosource missing=' ' symbolgen;
*****Global parameters needed to set up your wonder drug environment**;
%let outdest = ;
%include GBLPOST1;
**** Create Unblinded Multiple Treatment View, P_Meds *****;
Proc Sql;
  create view Projsup.P_Meds as
    select a.investid, a.patid, min(a.startdt) as startdt
        label='STUDY_MED_START_DATE' format=datetime16.,
        (max(a.startdt)+(60*60*24*28)) as stopdt
        label='STUDY_MED_STOP_DATE' format=datetime16.,
        max(a.startdt) as lst_dsdt
        label='LAST_DOSE_DATE' format=datetime16., "N" as stopdtp format=$1.,
        (round(((((max(a.startdt)+(60*60*24*28))-min(a.startdt))/60/60)/24)))
        as duration label='STUDY_DURATION', "D" as DURATIOU format=$1.,
        a.treatco, sum(input(a.trpstrng,7.3)) as tot_dose
        label= 'TOTAL_DOSE',
        min(input(a.trpstrng,7.3)) as dosegrp
        label='DOSE_GROUP_CODE' format=drugform.,
        max(input(a.visitco,7.)) as lst_cycl label='LAST_CYCLE_NO'
    from projdb.patmeds a
    Group by investid, patid, treatco;
***** Create View P_WONDER  with DRUG treatments ***********;
Proc SQL;
  Create view Projsup.P_Wonder as
    Select *
  from projsup.P_Meds
  where treatco=&dno
  Group by investid, patid, treatco;
***** UPDATE PATDER0N where n=subset no. ***********************;
data projsup.patder0&subset;
   set projsup.patder0&subset;
   invtxtlg=put(investid,$DRID.);
   tretxtlg="&DRUG OPEN-LABEL";
   tretxtsh="&drug";
   thecode="007";
** Create Patient Subset in view P_FNL0N, by subset number N *****;
Proc SQL;
 Create view projsup.P_Fnl0&Subset as
   Select a.*,
   b.invtxtlg, b.tretxtlg, b.tretxtsh, b.thecode
   From Projsup.P_Wonder as a, Projsup.Patder0&Subset as b
   where a.investid||a.patid = b.investid||b.patid
   group by a.investid, a.patid;
```

```
run;
quit;
*====== End of SASPOST1.SAS Program==========================;

********SASPOST2.SAS   Report Generator using PROC SQL *****************;
** written by Anne Marie Smith, 2/18/97 for PhilaSUG Meeting   ********;
options nocenter mautosource symbolgen sasautos=(cpmacs,cpmacsi,sasautos) missing=' ';
**** Set up Global variables and i/o drug environment   ***************;
%let outdest = POSTER.lis ;
%let reportid = POSTER;
%let reportno = ;
%include gblpost1;
%outroute(&OUTDEST);
*** Query the P_FNL0N View where n=subset number and write the report;
options date pageno=1;
Proc SQL double;
   TITLE1 "                         CLINICAL INVESTIGATION OF A &&DRUGNAME DRUG";
   TITLE2 " ";
   TITLE3 "REPORT &&reportid &&Reportno SUMMARY LISTING OF STUDY MEDICATION BY &&Subpatt";
   TITLE5 "PROTOCOL  : &&protocol";
   TITLE6 "THERAPY   : &&DRUGNAME-OPEN-LABEL";
   select invtxtlg label='INVESTIGATOR' format=$char40., patid label='PATIENT',
   startdt label='START     DATE:TIME',
   duration label='DAYS IN STUDY' format=5.,
   lst_cycl label='LAST CYCLE', tot_dose label='TOTAL DOSE GIVEN',
   dosegrp label='DOSE GROUP'
   from projsup.P_FNL0&Subset
   order by invtxtlg, patid;
quit;
run;
%outclose(&outdest);

*====== End of SASPOST2.SAS Program==========================;
          CLINICAL INVESTIGATION OF A WONDER DRUG     16:28 Tuesday, February 25, 1997 1

REPORT POSTER SUBSET( 1) SUMMARY LISTING OF STUDY MEDICATION BY PATIENT

PROTOCOL  : 007ACE-101-US
THERAPY   : WONDER-OPEN-LABEL

                                                 DAYS              TOTAL
                                               START         IN     LAST     DOSE
INVESTIGATOR               PATIENT       DATE:TIME  STUDY   CYCLE   GIVEN  DOSE GROUP
-------------------------------------------------------------------------------------------------------------------

JOHN WATSON, M.D.             0001     07AUG95:11:40:00    57        3     3.12   WONDER 1.50 mg.

JOHN WATSON, M.D.             0002     14AUG95:11:30:00    44        2     2.04   WONDER 1.50 mg.

SHERLOCK HOLMES, M.D.     0001     03APR95:12:25:00     28        1    0.425   WONDER 0.50 mg.

SHERLOCK HOLMES, M.D.     0002     24APR95:09:45:00     56        3    1.395   WONDER 0.50 mg.

SHERLOCK HOLMES, M.D.     0003     15MAY95:10:25:00    28        1      0.5     WONDER 0.50 mg.

SHERLOCK HOLMES, M.D.     0004     17MAY95:09:45:00    28        1     0.42    WONDER 0.50 mg.

SHERLOCK HOLMES, M.D.     0005     17JUL95:11:10:00      42        2      1.5    WONDER 1.00 mg.

END OF OUTPUT
```