Paper 132-26

Digging for Gold: SAS® Tools for Extracting Clinical Study Information from Oracle Clinical

Nancy Brucken, Pfizer Global R&D – Ann Arbor Laboratories, Ann Arbor, MI

ABSTRACT

Back in the “old” days, information such as study titles, locations of commonly collected fields and treatment group labels was hard-coded into SAS programs written to summarize data for a single clinical study. During the process of developing standard cross-therapy area reports to summarize data for any clinical study, we found that retrieving this information directly from the database allowed for much more flexibility in coding the programs and reusability across multiple studies. This paper demonstrates tools we have developed using SQL and the SAS SQL procedure to extract study titles, view creation dates, the names of the views containing specific questions, treatment group labels and treatment group print order, and other items.

CLINICAL STUDY TITLES

The title of a clinical study or protocol is displayed on every summary and listing table generated for the study. Corporate standards here require that it appear on every table included in the appendix of a Research Report or publication. Prior to the implementation of Oracle Clinical, SAS programmers would hard-code the study title into a macro variable, and then reference that macro variable in code used to display the title. That system worked fine for programs that ran on a single study, but often began to break down when programmers needed to run the same set of programs multiple times for multiple studies.

The CLINICAL_STUDIES table in Oracle Clinical contains both short (80 characters) and long (2000 characters) study titles. If the study title is entered into the database at study set-up, it can be extracted using the same piece of SQL code for any study, and stored in a global SAS macro variable, to be displayed on any table produced for a study, without the need for custom coding.

The following code extracts the short study title from Oracle Clinical:

```
proc sql noprint;
  connect to oracle (path="&instance");
  select title into :stitle
    from connection to oracle
      (select c.short_title TITLE
         from clinical_studies C
         where c.study = &_study);
  disconnect from oracle;
quit;
```

&INSTANCE is a SAS macro variable containing the name of the Oracle instance. **&_STUDY** is a SAS macro variable that contains the name of the study, surrounded by single quotes. The code passes the short title back into the SAS macro variable **&STITLE**.

Due to a 200-character maximum imposed on character variables by SAS 6.12, the long study title has to be broken up into segments before being passed to SAS and reassembled into a single macro variable. Our study titles tend to be far less than 600 characters, so the code that we use is as follows:

```
proc sql noprint;
  connect to oracle (path="&instance");
  select t1, t2, t3 into :t1, :t2, :t3
  from connection to oracle
   (select substr(c.title,1,200) T1,
           substr(c.title,201,400) T2,
           substr(c.title,401,600) T3
      from clinical_studies C
      where c.study = &_study);
  disconnect from oracle;
quit;
```

The SAS macro variables **&T1**, **&T2** and **&T3** then contain the title segments.

VIEW CREATION DATES

Given the large numbers of views that can potentially be created for a single study, it is often useful to add the view name and creation date to output tables, so that the results can more easily be reproduced. These view creation dates could be entered into the programs every time tables are run, but that quickly becomes a maintenance nightmare for programs run on multiple studies, and has a high potential for error. Creation dates for different types of views are stored in different locations within Oracle Clinical. We retrieve the creation date for the desired view using SQL, and pass it into a global SAS macro variable.

By definition, the creation date for the current view is the current date. While this value could be derived within Oracle, it is easier in this case to retrieve the current system date using SAS.

The creation date for the stable view has been defined here as the date that the last successful batch validation was completed. The BATCH_DM_RUNS table contains information on batch validations. This date can then be retrieved from the BATCH_DM_RUNS table like so:

```
select
  to_char(b.batch_end_ts,'DDMon YYYY')
    SNAPDATE
from study_access_accounts S,
     batch_dm_runs B
where
   s.clinical_study_id =
     b.clinical_study_id and
   s.study_access_acct_name = &_study and
   b.current_flag='Y' and
   b.success_flag='Y');
```

Again, **&_STUDY** is a SAS macro variable containing the name of the study and view, surrounded by single quotes.

Creation dates for snapshot and rollsnap views are stored in the STUDY_ACCESS_ACCOUNTS table, and are retrieved as follows:

```
select
  to_char(CREATION_TS,'DDMon YYYY')
    SNAPDATE
from study_access_accounts
where study_access_acct_name = &_study;
```

The surrounding SAS PROC SQL code to pass these results into macro variables is similar to the preceding examples; therefore it is not shown here.

LOCATIONS OF QUESTIONS

While we have tried to place a question in the same DCM for every study in which it is used, that does not always happen. For example, patient randomization numbers are sometimes collected with demographic data, and sometimes with patient status or dosing information. In order to write programs that could extract and display randomization number for any study, regardless of its location in the database, we needed a utility that could find it.

Being primarily SAS programmers, we first tried using the SAS dictionary tables to extract field locations, once the Oracle Clinical SAS views for the study had been defined. However, that method proved to be prohibitively slow. For example, a study containing 43 views took approximately 18 seconds to locate a given field.

Oracle Clinical contains a table called ALL_TAB_COLUMNS, which contains the names of all fields collected for all studies in the database. The following code creates a list of all views for a given study that contain the specified field name:

```
proc sql noprint;
  connect to oracle (path="&instance");
  select tabname into :tname
    separated by ' '
  from connection to oracle
    (select table_name TABNAME
      from all_tab_columns
      where owner=&_study and
            column_name=&_colname
     order by table_name);
  disconnect from oracle;
quit;
```

&_STUDY is a SAS macro variable containing the name of the study, surrounded by quotes, and **&COLNAME** is a SAS macro variable containing the name of the field being hunted. The resulting list is then passed into the SAS macro variable **&TNAME**, with multiple view names separated by spaces; **&TNAME** is then scanned for the preferred location of the field in question. This piece of code ran in a little over one second on the same 43-view study.

TREATMENT GROUP LABELS

For programs that are run on a single study, treatment group labels can be placed in a single format, and that format referenced where needed later on in the programs. However, this system becomes more complex when programs need to run on multiple studies. We could add a new format to the program every time it is run on a new study, but that requires additional maintenance and validation.

Instead, we decided to store treatment group labels inside Oracle Clinical. When a study is unblinded and treatment group codes are added to the database, we create a Discrete Value Group (DVG) for the question containing treatment group codes. The DVG is given the same name as the study, and the treatment group labels are entered into the long value field. These labels contain the names of the treatment groups as they are to appear on output tables. Our solution is a function, which can be called from within a SQL block, to look up each patient's treatment group value in the DISCRETE_VALUES table, and return its treatment group label (LONG_LABEL). Data is then summarized by the treatment group label values, and the labels are picked up and placed on the output tables by SAS. The code calling the treatment group label function appears as follows:

```
select distinct d.study,
    d.invsite,
    d.pt,
    pd_rxc.get_treat(d.rxgrp,d.study) CMED
 from s&cipr.$&view.demo;
```

This code then places the treatment group label in the CMED field. The function PD_RXC.GET_TREAT is stored inside Oracle Clinical, with the rest of the user-written procedures and functions. **&CIPR** is a SAS macro variable containing the name of the compound and protocol, while **&VIEW** contains the name of the study view (current, stable, etc.).

TREATMENT GROUP PRINT ORDER

Our standard practice is to report results for placebo patients first, followed by results for patients taking the study drug, by increasing dosage level, and finally by results for patients who received a comparator drug. Sometimes, the treatment group labels, when sorted alphabetically, end up following those requirements. Unfortunately, those occasions are rare. In most cases, we need to assign a print sequence number to each treatment group so that summarized results will be displayed in the desired order.

Again, we could hard-code the treatment group print sequence numbers into the output table programs. However, the same problems already encountered with treatment group labels appear again. Our solution is to store the print sequence numbers in Oracle Clinical, using the same DVG created to hold treatment group labels. The print sequence number is entered into the DVG display sequence number field, and can then be retrieved from the DISPLAY_SN field in the DISCRETE_VALUES table with another user-defined function.

The code calling the treatment group print sequence number function appears as follows:

```
select distinct d.study,
    d.invsite,
    d.pt),
    pd_rxc.get_seq_no(d.rxgrp,d.study) MED
 from s&cipr.$&view.demo;
```

This code places the treatment group print sequence number in the MED field. The function PD_RXC.GET_SEQ_NO is again stored inside of Oracle Clinical, with the rest of the user-written procedures and functions. **&CIPR** is a SAS macro variable containing the name of the compound and protocol, while **&VIEW** contains the name of the study view (current, stable, etc.).

CONCLUSION

Oracle Clinical contains useful information regarding properties of a study, such as the study title, view creation dates and locations of commonly-collected fields, and can be configured to hold treatment group labels and print sequence numbers. During the process of developing standard cross-therapy area reports to summarize data for any clinical study, we found that retrieving this information from the database allowed for much more flexibility in coding the programs. It also significantly reduced the amount of maintenance work required to adapt a program to run on a new study. This paper serves to illustrate some of the techniques used for retrieving this information. It is only a partial list of the study "gold" buried inside of Oracle Clinical. Happy treasure hunting!

ACKNOWLEDGEMENTS

Thanks to Michael Turomsha for allowing me to discuss the treatment group label and print sequence number functions which he wrote, and to Maggie Smith and Bryan Judge for helping me to locate much of this information inside of the Oracle Clinical database.

CONTACT INFORMATION

Nancy Brucken
Pfizer Global Research & Development-
   Ann Arbor Laboratories
2800 Plymouth Rd.
Ann Arbor, MI 48105

(734) 622-5767

Nancy.Brucken@pfizer.com

TRADEMARKS