

**USING SAS, SAS/ACCESS®, AND SQL PASSTHROUGH TO QUERY AND JOIN ORACLE TABLES:
An Example Using the Health Care Finance Administration's SDPS (Medicare) Database
Barbara B. Okerson, Ph.D.
Mid-South Foundation for Medical Care, Inc.**

ABSTRACT

Medicare Encounter data is received by each state's Peer Review Organization as an Oracle database known as the SDPS (Standard Data Processing System) data set. This data set is used for quality improvement projects, integrity investigations, and research. This data set is then queried with SAS software and the SAS SQL pass-through facility. Often the criteria for selecting data from the SDPS data set are contained in a pre-extracted SAS data set. Using the criterion information from the SAS data set, additional data is extracted from the SDPS database and merged or combined with the existing SAS data set or its information. In this example used to document comorbidities and disease progression for HIV, a SAS data set is run against the SDPS Oracle database to obtain subsequent admissions and diagnoses for each patient (Medicare beneficiary). This paper illustrates four methods for performing this query using SAS/ACCESS, SAS/CONNECT®, and Proc SQL.

INTRODUCTION

Mid-South Foundation for Medical Care, Inc. functions as the Peer Review Organization (PRO) for the state of Tennessee. The PRO program exists to ensure that Medicare beneficiaries receive appropriate, high quality care as mandated by HCFA (Health Care Finance Administration). Among the tasks required for the PROs are:

- Research and distribution of statistical analysis and data on the quality of care among Medicare beneficiaries within the state.
- Monitoring of trends and variations in medical care by geographic area within the state and in state-to-nation comparisons.
- Identification of sentinel events that may indicate less-than-optimal care.
- Identification, prioritization and action upon opportunities for improvement.
- Evaluation of the impact of each quality improvement intervention.

In accordance with this tasking, Mid-South Foundation monitors medical care, diagnosis groups, and treatment patterns throughout the state of Tennessee. Part of this monitoring involves following the normal progression of a diagnosis to identify commonalities in additional diagnoses, comorbidities, length of stays, and outcomes.

This paper follows the progression of Medicare beneficiaries after the first in-patient encounter for any beneficiary with a primary or other diagnosis code equaling HIV (042 ICD-9). The methodology involves identification of those patients, isolating first occurrence, then, for each patient, identifying all subsequent

admissions and their diagnosis and procedure codes. Progression, comorbidities, treatment, and outcome are compared and contrasted by beneficiary, hospital, and region. All of this information is contained in the Oracle SDPS database.

SAS software provides an interface with Oracle that provides a number of options for interfacing and isolating the data needed for this analysis. This paper contrasts four methods for solving this Oracle query in terms of ease of use, ease of programming, time and space considerations, and situational advantages and disadvantages.

These examples use SAS version 6.12 running on a Dell Pentium running Windows 95, also Oracle version 8 and SAS version 6.12 on an RS/6000 model 390 running AIX version 4.1. Communication between the two platforms is achieved with SAS/ACCESS, SAS/CONNECT, and SQL Passthrough.

STEP ONE: QUERYING THE DATABASE TO ISOLATE THE AFFECTED POPULATION

The first step is to extract the first admission in the database for code 042. Because the total HIV admissions by year are used as a part of the report, all of these are written to a SAS data set and a new data set created that includes the beneficiary number (bene_clm) and the discharge date (hse_clm0) of the first admission. The following code creates these two data sets.

```
rsubmit SASSVR1;
proc sql;
  connect to oracle(user='bokerson' pass=xxxxxx
    path='tn_dbsvr');
  create table hivtemp as
  select bene_clm, hsp_id, hse_clm_, hse_clm0,
    hse_inpa, hse_bene, bene_sta, bene_sex,
    bene_rac, bene_mdc, hse_dgns, dgns_cd, hse_dgn0
  from connection to oracle(select A1."BENE_CLM_NUM",
    A1."HSP_ID", A1."HSE_CLM_FROM_DT",
    A1."HSE_CLM_THRU_DT", A1."HSE_INPAT_OUTPAT_IND",
    A1."HSE_BENE_AGE", A1."BENE_STATE_CODE",
    A1."BENE_SEX_IDENT_CD", A1."BENE_RACE_CD",
    A1."BENE_MDCR_STUS_CD", A2."HSE_DGNS_SEQ",
    A2."DGNS_CD", A2."HSE_DGNS_COMORBID"
  from TN_SDPS.HSE A1, TN_SDPS.HSE_DIAG A2
  where A1."BENE_CLM_NUM" = A2."BENE_CLM_NUM" AND
    A1."HSE_CLM_FROM_DT" = A2."HSE_CLM_FROM_DT" AND
    A1."HSE_CLM_THRU_DT" = A2."HSE_CLM_THRU_DT" AND
    A2."DGNS_CD" = '042'
  order by A1."BENE_CLM_NUM", A1."HSE_CLM_THRU_DT") as
  t1( bene_clm, hsp_id, hse_clm_, hse_clm0,
    hse_inpa, hse_bene, bene_sta, bene_sex,
    bene_rac, bene_mdc, hse_dgns, dgns_cd,
    hse_dgn0);
quit;
endrsubmit;
```

This data set is then used for admission, length of stay, and other trends for HIV admissions.

The following code is then used to isolate the first admission for each beneficiary for this diagnosis.

```
rsubmit SASSVR1;
data tempbbo;
    set hivtemp; by bene_clm hse_clm0;
    if first.bene_clm;
run;
endrsubmit;
```

The beneficiary numbers and dates contained in this data set are then used as the criteria for querying and selecting all encounter data for each of these beneficiaries after their initial admission from the Oracle database. Each of the methods given have advantages and disadvantages.

STEP TWO: USING THE SAS DATA SET TO FURTHER QUERY ORACLE

The programming task required is to use the SAS data set as a means for querying the Oracle database. The resultant database should have all the beneficiary information contained in the original SAS data set as well as any additional diagnoses for each beneficiary.

In terms of ease of programming, this task could be accomplished by extracting the encounter table and diagnosis table (both millions of records) and then merging with SAS. If one has nothing but unlimited space and time and is not worried about computing costs this could be your method. Since that is not usually the case in the real world, a more efficient and expedient solution is necessary. Four such methods are illustrated here:

- Joining the SAS data set directly with the Oracle tables
- Joining the SAS data set with a view of the Oracle table
- Create a temporary Oracle table to join with the SAS data set
- Pass the matching information to Oracle using the SAS macro language

Method One: Merging the SAS Data Set Directly with the Oracle Tables

When merging the SAS data set with the Oracle table in a single step, advantages include keeping the processing on the UNIX server, accomplishing the join in a single program, and ease of programming. Disadvantages include processing time and the work space required for the intermediate tables.

In this solution, all the required variables are extracted from Oracle into a SAS work file called ora. This file is joined with the original SAS data set using the defined criteria.

The code for this solution follows. Note that it is necessary to list the data set name prefix to identify those variables selected from

the SAS data set. In this method, the SAS data set variable names cannot be the same variable names as those from the Oracle data base.

```
rsubmit SASSVR1;
proc sql;
connect to oracle(user='bokerson' pass=xxxxxx
    path='tn_dbsvr');
create table work.ora as
select bene_cl, hsp_id2, hse_clf, hse_clt, hse_dgn,
    hse_seq, hse_cmbd, tempbbo.bene_clm, tempbbo.hsp_id,
    tempbbo.hse_clm, tempbbo.hse_clm0, tempbbo.hse_bene,
    tempbbo.bene_sta, tempbbo.bene_sex, tempbbo.bene_rac,
    tempbbo.bene_mdc, tempbbo.hse_dgns, tempbbo.dgns_cd,
    tempbbo.hse_dgn0
from connection to oracle(select B1."BENE_CLM_NUM",
    B1."HSP_ID", B1."HSE_CLM_FROM_DT",
    B1."HSE_CLM_THRU_DT", B1."HSE_DGNS_SEQ",
    B1."DGNS_CD", B1."HSE_DGNS_SEQ"
from TN_SDPS.HSE_DIAG B1) as or1(bene_cl, hsp_id2,
    hse_clf, hse_clt, hse_dgn, hse_seq, hse_cmbd)
right join work.tempbbo as sas1
    on or1.bene_cl=sas1.bene_clm and or1.hse_clt ge
    sas1.hse_clm0;
quit;
endrsubmit;
```

Method Two: Creating a SAS Data View of the Joined SAS Data Set and Oracle Tables

The code to create a data set view differs very little from the code in method one. What differs is when and where in the processing the use of space and time occurs. However, interestingly, even if the view is used only one time to create a permanent SAS data set, the total processing time is less than creating a SAS data set as illustrated in method one (by about two seconds.)

The code follows:

```
rsubmit SASSVR1;
proc sql;
connect to oracle(user='bokerson' pass=xxxxxx
    path='tn_dbsvr');
create view work.ora as
select bene_cl, hsp_id2, hse_clf, hse_clt, hse_dgn,
    hse_seq, hse_cmbd, tempbbo.bene_clm, tempbbo.hsp_id,
    tempbbo.hse_clm, tempbbo.hse_clm0, tempbbo.hse_bene,
    tempbbo.bene_sta, tempbbo.bene_sex, tempbbo.bene_rac,
    tempbbo.bene_mdc, tempbbo.hse_dgns, tempbbo.dgns_cd,
    tempbbo.hse_dgn0
from connection to oracle(select B1."BENE_CLM_NUM",
    B1."HSP_ID", B1."HSE_CLM_FROM_DT",
    B1."HSE_CLM_THRU_DT", B1."HSE_DGNS_SEQ",
    B1."DGNS_CD", B1."HSE_DGNS_SEQ"
from TN_SDPS.HSE_DIAG B1) as or1(bene_cl, hsp_id2,
    hse_clf, hse_clt, hse_dgn, hse_seq, hse_cmbd)
right join work.tempbbo as sas1
    on or1.bene_cl=sas1.bene_clm and or1.hse_clt ge
    sas1.hse_clm0;
quit;
endrsubmit;
```

Any time subsequently that the view would be used instead of a data set would result in considerable processing time. However, if a number of such joins are used regularly, the storage space saved may be worth the increased processing time.

Method Three: Create a Temporary Oracle Table to Join with the SAS Data Set

In this method, the SAS data set containing the join criteria is written to the Oracle data base as a temporary Oracle table. The advantages are that all processing again remains on the UNIX side, all processing is in Oracle, the amount of data processed by Oracle and by SAS is reduced, the actual join is in a single step, and the programming is relatively easy.

In order for this method to work, it is necessary for the programmer to have CREATE TABLE and DROP TABLE privileges for the Oracle data base. Where this is true, this method probably provides the best solution. The following code writes the SAS data set as an Oracle table.

```
rsubmit SASSVR1;
proc dbload dbms=oracle data=tempbbo;
    user=bokerson; orapw=xxxxxx; path='@f:SDPS';
    table tritemp; load; run;
endrsubmit;
```

It is important to make sure that the criterion variables have the same characteristics in the temporary Oracle table as they do in the permanent Oracle data base. SAS software automatically converts variable length formats to fixed formats when creating SAS data sets from Oracle. Although that code is not given, in this example it was necessary to reformat the beneficiary number variable (bene_clm) to a varying format. This was not necessary for the previous two methods. The join code follows:

```
rsubmit SASSVR1;
proc sql;
connect to oracle(user='bokerson' pass=XXXXXX
    path='tn_dbsvr');
create table work.ora as
select bene_clm, hsp_id, hse_clm_, hse_clm0,
    hse_inpa, hse_bene, bene_sta, bene_sex, bene_rac,
    bene_mdc, hse_dgns, dgns_cd, hse_dgn0, hse_clmf,
    hse_clmt, dgns_cd1, hsedgns, hsedgn0
from connection to oracle(select A1."BENE_CLM",
    A1."HSP_ID", A1."HSE_CLM_", A1."HSE_CLM0",
    A1."HSE_INPA", A1."HSE_BENE", A1."BENE_STA",
    A1."BENE_SEX", A1."BENE_RAC", A1."BENE_MDC",
    A1."HSE_DGNS", A1."DGNS_CD", A1."HSE_DGN0",
    A2."HSE_CLM_FROM_DT", A2."HSE_CLM_THRU_DT",
    A2."DGNS_CD", A2."HSE_DGNS_SEQ",
    A2."HSE_DGNS_COMORBD"
from BOKERSON.TRITEMP A1, TN_SDPS.HSE_DIAG A2
where A1."BENE_CLM" = A2."BENE_CLM_NUM"
    AND A1."HSE_CLM0" le A2."HSE_CLM_THRU_DT"
) as t1( bene_clm, hsp_id, hse_clm_, hse_clm0,
    hse_inpa, hse_bene, bene_sta, bene_sex, bene_rac,
    bene_mdc, hse_dgns, dgns_cd, hse_dgn0, hse_clmf,
    hse_clmt, dgns_cd1, hsedgns, hsedgn0)
quit;
endrsubmit;
```

Method Four: Pass the Criterion Information to Oracle using the SAS Macro Language

In this method, the beneficiary number and date information are passed to Oracle through macro processing. Again, like the previous solution, the main advantage is that the processing remains in Oracle. In this instance, Oracle privileges are not required.

Although difficult to code, with difficulty increasing by the number of variables that comprise the query criteria, the result is very efficient code that takes advantage of client-server technology, keeping all the selection processing in Oracle, while eliminating the necessity of creating a temporary Oracle table.

In the following code, two macros are created. In the first (bene), all the beneficiary numbers to be selected are written out as a string or array of macro variables that can be retrieved by calling the macro. The SAS Autocall macro, %cmpres is used to eliminate any multiple blanks from the string. The %cmpres macro calls both the %left and the %trim macros to remove any leading and trailing blanks.

In the second macro (benedate), a series of macro variables are created that pair the beneficiary numbers with the initial discharge date. These can then be called by the SQL to Oracle code. It is necessary to use the %global statement when creating these variables; it is not implicit in this case.

The code to create the macros follows:

```
rsubmit;
data _null_;
    set tempbbo nobs=nobs;
call symput("count",put(nobs,4.));
call symput("bene"||left(nobs)," "||trim(bene_clm)||");
run;
%macro bene;
    %do i = 1 %to &count;
        "%cmpres(&&bene&i)", %end;
    %mend;
%macro benedate;
%do i = 1 %to &count;
    data _null_; set one;
        if nobs=&i;
        %global name&i date&i;
            call symput("NAME&i",trim(bene_clm));
            call symput("DATE&i",(hse_clm0));
    %end; %mend;
```

Although it would be possible to use only the paired macro variables to query Oracle, it is actually more efficient to isolate within Oracle those beneficiary numbers that match before performing the date processing. If processing space considerations are the primary concern, rather than a balance of time and space, this problem can be addressed using only the second macro.

The following code uses the macro variables, rather than joins, to create the data set containing the identical information resulting from the previous three methods.

```

Proc sql;
connect to oracle(user='bokerson' pass=XXXXXX
    path='tn_dbsvr');
create table orahiv (bene_clm char, hsp_id char,
    hse_clm_date, hse_clm0 date, hse_inpa char,
    hse_bene num, bene_sta char, bene_sex char,
    bene_rac char, bene_mdc char, hse_dgns char,
    dgns_cd char, hse_dgn0 char);
create table temp.ora as select bene_clm, hsp_id,
    hse_clm_, hse_clm0, hse_inpa, hse_bene,
    bene_sta, bene_sex, bene_rac, bene_mdc,
    hse_dgns, dgns_cd, hse_dgn0
from connection to oracle(select A1."BENE_CLM_NUM",
    A1."HSP_ID", A1."HSE_CLM_FROM_DT",
    A1."HSE_CLM_THRU_DT",
    A1."HSE_INPAT_OUTPAT_IND", A1."HSE_BENE_AGE",
    A1."BENE_STATE_CODE", A1."BENE_SEX_IDENT_CD",
    A1."BENE_RACE_CD", A1."BENE_MDCR_STUS_CD",
    B1."HSE_DGNS_SEQ", B1."DGNS_CD",
    B1."HSE_DGNS_COMORBID"
from TN_SDPS.HSE_DIAG A1, TN_SDPS.HSE_DIAG B1 where
    A1.=BENE_CLM_NUM= in(%bene) and
    A1.=BENE_CLM_NUM= B1.=BENE_CLM_NUM=) as
or1(bene_clm, hsp_id, hse_clm_, hse_clm0,
    hse_inpa, hse_bene, bene_sta, bene_sex,
    bene_rac, bene_mdc, hse_dgns, dgns_cd,
    hse_dgn0);
%macro mdata;
%do i = 1 %to &count;
    %let name="&&name&i";
    %let date= &&date&i;
    insert into temp.orahiv
    select * from ora where ora.bene_clm=&name and
        ora.hse_clm0 ge &date;
%end;
%mend;
%mdata;
quit;
endrsubmit;

```

STEP THREE: PROCESS THE DATA AND WRITE THE REPORTS

This is where the fun begins: the actual analysis, search for commonalties, trends, problems, and reportable phenomena.

CONCLUSION

The best solution generally becomes a combination of programming ability, database size, data needs, hardware and software limitations, and application. In this situation, ease of programming was sacrificed for efficiency because of the size of the database.

In all cases, good knowledge of SAS/ACCESS, SAS/CONNECT, SQL language including Pass-Through syntax, macro language, and Oracle formats is imperative. The code for any good solution should ultimately be generic enough that it can be adapted to future applications. This code can be used to track any group of patients through subsequent admissions.

BIBLIOGRAPHY

Buffum, Henry W., "Strategic Uses of SAS Data Step Programming and SQL Passthrough to Query Oracle Databases," *Proceedings of the Twenty-first Annual SAS Users Group International Conference*, Cary, NC: SAS Institute, Inc., 1996.

Carpenter, Art, *Carpenter's Complete Guide to the SAS Macro Language*, Cary, NC: SAS Institute, 1998.

Health Care Finance Organization. "Peer Review Organization Overview." http://www.sdps.org/pro_over.html.

Health Care Finance Administration, "Introducing the Medicare Quality of Care Surveillance System," *Quality Resume*, 1(1997): 1-2.

Klenz, Bradley W. , "Using the SAS/ACCESS Interface with Relational DBMSs: Beyond the Basics," *Proceedings of the Sixteenth Annual SAS Users Group International Conference*, Cary, NC: SAS Institute, Inc., 1991.

Okerson, Barbara B., "Using the SAS SQL Procedure," The University of Memphis, 1993, 1996.

SAS Institute. *SAS/ACCESS Software for Relational Databases: Reference, Version 6*, Cary, NC.

SAS Institute. *SAS Guide to the SQL Procedure. Usage and Reference, Version 6.*, Cary, NC.

SAS Institute. *SAS Language. Reference, Version 6*, Cary, NC.

ACKNOWLEDGMENTS

SAS, SAS/ACCESS, and SAS/CONNECT are registered trademarks of SAS Institute, Inc. of Cary, North Carolina.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The analyses upon which this publication is based were performed under Contract Number 500- - , entitled "Utilization and Quality Control Peer review organization for the State of Tennessee," sponsored by the Health Care Financing Administration, Department of Health and Human Services. The content of this publication does not necessarily reflect the views or policies of the Department of Health and Human Services, nor does mention of trade names, commercial products or organizations imply endorsement by the U.S. Government.

The author assumes full responsibility for the accuracy and completeness of the ideas presented. This article is a direct result of the Health Care Quality Improvement Program initiated by the Health Care Financing Administration, which has encouraged identification of quality improvement projects derived from analysis of patterns of care, and therefore required no special funding on the part of this contractor. Ideas and contributions to the author concerning experience in engaging with issues presented are welcome.

For more information contact:

Barbara B. Okerson, Ph.D.
Biostatistician/Health Care Data Analyst
Mid-South Foundation for Medical Care, Inc.
6401 Poplar Ave
Memphis, TN 38119
(901) 682-0381, ext. 304
e-mail: tnpro.bokerson@sdps.org