# SAS/DB2 - DO's and DON'Ts

Michael L. Sperling, City of Phoenix, AZ.

## INTRODUCTION

This presentation follows my evolutionary use of SAS with DB2. The use of views with SAS/ACCESS, at first, seemed the most logical approach to the SAS/DB2 world. The second step was PROC SQL. Both methods of DB2 access have gone through variations which are described below.

Regardless of the access method chosen, more often than not the interoperability of SAS and DB2 caused problems which transcended the physical transfer of data. The problems were sometimes perceptual and sometimes real. In either case, SAS became the culprit and the SAS user became the victim. I will attempt to chronicle an approach to alleviating these situations.

## VIEWS

The saga begins with SAS 6.06 - the interactive creation of views. The only way to create views in SAS 6.06 was interactively. I created the views, placing them in a single RACF controlled SAS Data Library. The end-users had read-only access to the views and to the DB2 tables. The primary problem with the interactive creation of views is the possible overlay of another view of the same name without warning. (Example 1)

SAS 6.07 offered the ability to create views in a batch job using PROC ACCESS. This speeds up the process of maintenance and keeps a record of the process and activity. Any future changes in the DB2 database can be easily updated in the batch descriptions with comments to indicate the nature and date of the change. While I have only worked with DB2, this process works with the SAS interface to any database. (Example 2). Views are probably the easiest method of database access for end-users; especially, those who do not wish to be programmers. (Example 3).

## PROC SQL

The use of PROC SQL provides a simple and effective way to access a DB2 database without having to worry about the creation and maintenance of views. However, it adds the requirement to learn another language, even though SQL is relatively simple. While the interoperability of SAS and DB2 is not very well defined, the pass-thru function of PROC SQL provides an effective solution to performance issues. The simplest form of SQL to access a DB2 table is the 'Select *', where all rows (observations) and columns (variables) are selected (Example 3). You can select the columns you want by specifying them instead of the '*' in the select statement. You can also limit the observations which you chose by the use of a Where clause (Example 4).

While SAS provides the means for

accessing the database, the execution of SQL in the SAS/DB2 environment is less than perfect. Using simple SQL statements I found that I could interfere with the use of the database by other on-line applications. Even though I was operating in a low priority batch initiator, when the SQL code was passed to DB2, the batch job took on the attributes of the DB2 priority. If the SQL code was not accessing the table using the primary key, a sequential search of the entire table was done. If a Where clause was included, then multiple sequential searches of the table were initiated - one search for each of the values being requested. This can be a tremendous workload on a large database and can negatively impact other users of that database. With the higher DB2 priority, this impact is even worse.

Our next phase of SAS/DB2 interaction then became a search to eliminate multiple sequential searches of a DB2 table. The solution was to generate the SQL code with Where clause values and store them in an external dataset. Follow this with a PROC SQL step to execute the code using a %INCLUDE statement (Example 5). This will cause a single sequential search of the table regardless of the number of values being requested. The primary limitation is that DB2 can only handle a 32K select statement. Therefore, if you have more than 400 lines in your WHERE clause, you will get a DB2 error. The only viable solution is to create multiple executions of these steps extracting 400 values each time. You can possibly get 1200 values extracted at a time by stringing 3 values per line.

## PERFORMANCE

If the DB2 table has only 10,000 rows, the search is quick and system performance does not appear to be degraded. However, if the table contains 5,000,000 rows, system performance is noticeably poor across the board. Variables that affect performance include:

1. table size;
2. conditions of the search;
3. machine capacity in terms of memory;
4. disk availability;
5. placement of datasets on disk packs, strings and channels; and,
6. volume of traffic against the same tables by other applications.

## SOLUTIONS

The effectiveness/performance of SAS with DB2 is often perception. If you cause (or appear to cause) slow response time once -- you (SAS) are forever to blame. The best way to avoid this is to keep communications open --

1. you and DBA;
2. you and management;
3. you and end-user;
4. you and the vendor;
5. you and auditors;
6. you and everyone.

The common thread is YOU!

Resolve problems quickly, making sure everyone knows the cause, whatever it is, including SAS and DB2.

Very often you can reduce or eliminate the perception of poor

system performance linked to SAS by instituting procedures specifically aimed at SAS. These are procedures which should be implemented anyway, such as standards. This will make almost everyone feel good:

1. management (you have established constraints);
2. DBA (you have established parameters and limits); and,
3. auditors (you have established controls).

Technical solutions which come to mind are:

1. create a separate subset of DB2 data which is only accessed by the SAS programmers but maintained by the usual DB2 applications;
2. create a duplicate set of tables as SAS datasets on the mainframe; and,
3. create a duplicate set of tables as SAS datasets on another platform, such as UNIX, RISC, PC, WINDOWS.

**SUMMARY**

There is "good news" and "bad news" in the SAS/DB2 world. The good news for the use of views is that it provides: security of the DB2 tables, ease of use for the end-users, and quick creation of views in the interactive mode. The batch procedures to create views provides: ease of maintenance, documentation, and an audit trail.

The "bad news" is: views of the same name will overlay each other without warning; maintenance of a changing database is time consuming, at best; and, large DB2 tables impact total system response.

The good news for the use of PROC SQL is that the pass-thru function allows DB2 to work in its own environment using indexes or keys and faster response time due to less interaction between SAS and DB2.

The bad news is the 32K select statement limitation of DB2 and the extended searches required when the WHERE clause does not use the key of the DB2 table.

## Example 1

```
Command ===>

Libref: RMS
Dataset: ADVFILE

     Variable Length Format         Informat       Key Label

   _ CENTURY   8     4.              4.              N  FILE_CENTURY
   _ YEAR      8     4.              4.              N  FILE_YEAR
   _ FILE_NBR  8     6.              6.              N  FILE_NBR
   _ SEQ_NBR   8     5.              5.              N  ADV_SEQ_NBR
   _ ADV_LNAM $20    $20.            $20.            N  ADV_LNAME
   _ ADV_FNAM $15    $15.            $15.            N  ADV_FNAME
   _ ADV_MI   $1     $1.             $1.             N  ADV_MI
   _ ADV_TYPE $1     $1.             $1.             N  ADV_TYPE
   _ ADV_STAT $3     $3.             $3.             N  ADV_STATUS
   _ STAT_DAT  8     DATE7.          DATE7.          N  ADV_STATUS_DATE
   _ ST_NBR   $10    $10.            $10.            N  ADV_STREET_NBR
   _ ST_DIR   $2     $2.             $2.             N  ADV_STREET_DIR
   _ ST_NAME  $24    $24.            $24.            N  ADV_STREET_NAME
   _ ST_TYPE  $3     $3.             $3.             N  ADV_STREET_TYPE
   _ ADV_CITY $26    $26.            $26.            N  ADV_CITY
```

## Example 2

```
CREATE AA.ACCTBYP.ACCESS;
  TABLE=WHS.ACCT_BYPASS_TABLE;
  SSID=DSNP;
  RENAME   ACCT_NBR          =  ACCTNBR
           STORE_DATE        =  STORDATE
           ACCT_BYP_ORIGIN   =  BYPORIGN
           ACCT_BYP_STATUS   =  BYPSTATS
           ACCT_BYP_LID      =  BYPLID
           ACCT_BYP_REASON   =  BYPREASN
           ACCT_BYP_MGMT     =  BYPMGMT
           UPDATE_YMD        =  UPDTEYMD
                                ;

CREATE AA.ACCTBYP.VIEW;
  SELECT   ACCT_NBR
           STORE_DATE
           ACCT_BYP_ORIGIN
           ACCT_BYP_STATUS
           ACCT_BYP_LID
           ACCT_BYP_REASON
           ACCT_BYP_MGMT
           UPDATE_YMD
                                ;
```

## Example 3

```
libname WII 'MMSPROD.WII.sasacces.data'  disp=shr;

DATA  WIIBIND;
 MERGE  WII.WII
        WII.WIICOMB (KEEP= bsafname bsalname
         deptcode divprio inifname inilname
         iniphone
         justify usernum wiinum)   ;
   BY WIINUM;
RUN;
```

## Example 4

```
proc sql ;

  create TABLE  mmm.match     AS

  select *

    from mmm.type1, whs.metracct

    where type1.acctnbr=metracct.acctnbr   ;

quit;
```

## Example 5

```
proc sql ;

  create TABLE  mmm.match     AS

  select type1.acctnbr, name,  dir,
         street, city, zipcode, metERMAK,
         housenbr,
         metracct.acctnbr, mtrINymd,
         amtrloc

    from mmm.type1, whs.metracct

    where type1.acctnbr=metracct.acctnbr
                                          ;
quit;
```

## Example 6

```
data _null_;
  file 'tdm470.private.sascode(custcode)';
  set awwa2.sample    end=eof;
  if _n_ = 1 then do;
     put 'connect to db2 (ssid=dsnp);' ;
     put 'create table awwa2.custdata as' ;
     put 'select *'  ;
     put 'from connection to db2'  ;

     put '(select a.acct_nbr, a.cust_nbr,'  ;
     put ' start_ymd, current_cust_code'  ;

     put '  from whs.customer_table a' ;
     put '       where a.acct_nbr =' acctnbr;
  end;
  else do;
     put '         or a.acct_nbr =' acctnbr;
  end;
  if eof then do;
     put 'for fetch only)'  ;
     put 'as cust (acctnbr, custnbr, startymd,
currcucd);'  ;
     put '%put  macro This is SQLXMSG  &SQLXMSG;'
;
     put '%put  macro This is SQLXRC   &SQLXRC ;'
;
     put 'disconnect from db2;' ;

  end;

run;


proc sql;

%include  'tdm470.private.sascode(custcode)'  ;

quit;
```