Paper 255

A Windowed Application for Efficient Access to Data Warehouse DB2® Data with PROC SQL Pass-Through Facility and Macros

James Yang, Highmark Blue Cross Blue Shield, Pittsburgh, PA

ABSTRACT

This paper is a follow up of a NESUG'98 paper (Yang 1998). Because of the attraction of its efficiency, macro %JOINDB2 has been revised to add the capability of joining flat files with DB2 tables for non-SAS® users. Also, two macros %FLATOUT and %QUERYDB2 have been developed to enhance %JOINDB2. Based on these macros, a windowed application has been developed on MVS®(OS/390) using SAS macro windows to improve the efficiency when joining/querying data warehouse data (DB2). It can be used by SAS programmers, non-SAS programmers, and users without any programming experience. It checks the validation of user input (including existence of files, tables, and columns), joins flat file/SAS data set with DB2 tables, queries DB2 tables, writes out flat file/SAS data set, creates file structure description and FOCUS master file description for output flat files, generates SAS codes (including JCL® for batch job), etc..

INTRODUCTION

As discussed on Yang's NESUG'98 paper, PROC SQL Pass-Through facility has superior performance over SAS/ACCESS® views when accessing DB2 data on MVS even they both use the same interface view engine. The main reason is the KEEP and VAR statements are not passed to DB2 through view descriptor for processing. By utilizing the Pass-Through facility and the technique of listing key values into a macro variable and embedding it into DB2 where clause, macro %JOINDB2 transforms the joining into a simple query within the DB2 environment and therefore greatly improves the performance. If the key is indexed, it may save over 99% of CPU time and bypass the day-time DB2 CPU governor.

Because of the attraction of its efficiency, Macro %JOINDB2 has been revised to add the capability of joining flat files with DB2 tables for non-SAS users. Also, two macros %FLATOUT and %QUERYDB2 have been developed to output flat file and query DB2 tables, respectively. To ease the use of these macros for all users (even without programming experience), a windowed application on MVS has been developed using SAS macro windows.

Because of the page limit, this paper will only give you an overview of the application, explain the approach and some techniques in the window interface, and go through macro %FLATOUT. Macro %QUERYDB2 is a simple form of macro %JOINDB2, which has been explained thoroughly on Yang's NESUG'98 paper.

OVERVIEW OF THE APPLICATION

This application consists of four macros %WINDOWS, %JOINDB2, %QUERYDB2, %FLATOUT, and a program WINCNTL. Macro %WINDOWS defines 15 macro windows for user input and information about this application. Program WINCNTL contains 16 cooperative macros to control the running of the windows, to check the validation of user input (including existence of files, tables, and columns), and to prepare, write out and submit the SAS codes for batch (JCL included) or interactive job. The backbone of this application is macros %JOINDB2, %QUERYDB2, and %FLATOUT. They are called to run the actual jobs. All 20 of the macros have been compiled and stored in a SAS library using the stored compiled macro facility. This saves CPU time because these macros are compiled only once, not every run of the application.

The functionality of this application includes: (1) Join flat file or SAS data set with DB2 tables on the data warehouse; (2) Query DB2 tables on the data warehouse; (3) Output flat file or SAS data set using data warehouse standard formats (making data more sharable); (4) Create file structure description and FOCUS Master File Description for output flat file; (5) Generate SAS codes for the actual job (including JCL for batch job) for documentation or resubmitting the job under ISPF/TSO.

The main advantage of this application is its efficiency, especially when joining external data with DB2 tables. If the joined key is indexed, it may save you over 99% of CPU time compared to SAS/ACCESS views, FOCUS views, and even the Pass-Through facility itself. Most of the time, you can bypass the day-time DB2 CPU governor with a 5-minute-CPU class job or even run the job interactively (may be used as an on-line inquiry tool).

Another advantage of this application is its flexibility. You don't need to know any programming language, JCL, or data set allocation on MVS. You can join or query several tables at the same time as long as they all have the columns you want. You can include WHERE/GROUP BY/HAVING clauses, create new columns with arithmetic, summary functions (SUM, COUNT, AVG), and DB2 SQL functions such as SUBSTR. You can also use SAS data sets or flat files as input/output, specify the storage media (Disk/Tape), the job name and job class. Furthermore, you can use both upper case and lower case letters and run the job in batch or interactively. You can also submit more than one job without leaving the application, or go to interactive SAS and relaunch the application. Finally, you can have the flat file description, the FOCUS MFD, the SAS codes and JCL, etc..

The last advantage of this application is its window interface. It not only avoids the programming, but also checks your input and prompts you with warning/error messages and sounds if necessary. It checks the validation of your selection/input, the existence of files, DB2 tables, and DB2 columns. It lets you page backward and forward to make changes, and allows you to quit the application at any time.

THE WINDOW INTERFACE

This application utilizes macro windows to implement the window interface. The window interface on MVS with SAS/SCL® is not so user-friendly and effective as on PC. Also, there are too many tables and columns in the data warehouse for this application to utilize the useful list box in SAS/SCL. If SAS/SCL on PC is used with SAS/CONNECT®, the performance will be affected by the connection (especially the data transfer) between MVS and PC. Also, users need to log on MVS anyway to check the batch jobs (recommended for most of the jobs submitted by this application). The interface for this application is mainly to provide help information, check and capture user input to submit actual job. It doesn't require too many windows. Macro window is a traditional, handy and quick, but still effective solution for this application.

This application uses TSO CLIST command to invoke interactive SAS with an autoexec file to launch the application. To avoid the confusion of interactive SAS for non-SAS users, all windows take up the full screen to hide the interactive SAS on the background. But SAS users have more conveniences: go to interactive SAS and ISPF to check information or do whatever allowed, or relaunch the application with all previous user input (may save typing). Windows are called by each other and they are under users' control: page backward or forward, go to interactive SAS or quit the application. All windows have instructions and examples (where available) for user input and actions. Each window checks the validation of user selections and input (including the existence of files, tables, and columns), and prompts user with warning/error messages and sounds if any information is invalid.

The first window provides the following options for information about this application (including update information) to help users understand and use the application: (1) What can this application do? (2) What are the advantages of this application? (3) What can you do with this application? (4) What should you be aware of? (5) What's NEW? (6) Questions/Problems/Bugs? These serve as a readme/help file.

Appendix I is a window control macro %DB2INFO, one of the 16 cooperative macros in the program WINCNTL. Like the other 15 macros, it extensively uses macro functions (especially the quoting functions), macro label statement, and macro control flow such as the %DO %WHILE loop and %IF %THEN %ELSE statement. Macro label statement %STARTOVER: is used to restart the window input and recheck the input from the beginning, when invalid input information has been captured anywhere on the window. This allows users to even make changes to previously checked input information. It mainly checks the required input fields and the existence of or access to DB2 tables and columns using %DO %WHILE loop and macro function %QSCAN. PROC SQL option OUTOBS=1 along with its Pass-Through facility is used to do the quick check on tables and columns. Macro functions %QSUBSTR and %INDEX are used to extract DB2 columns from SQL functions (e.g. SUBSTR and SUM), DISTINCT keyword, and operators (e.g. arithmetic +-*/ and concatenation ||).

After capturing all necessary user input information into various macro variables, this application uses the PUT statement with macro function %NRSTR to write out SAS codes (mainly %LET statements and macro calls to the three backbone macros) to MVS internal reader to submit the actual job. If an external file is specified to store the SAS codes, SAS codes are written out to the external file first. Then the job is submitted to the internal reader with the following codes (EXT_FILE is the fileref for the external file):

```
  FILENAME SUB SYSOUT=A PGM=INTRDR LRECL=80 RECFM=F;
  DATA _NULL_; INFILE EXT_FILE; INPUT; FILE SUB;
  PUT _INFILE_; RUN;
```

TSO command SUBMIT can't be used directly with SAS X statement because the macro variable for the external file can't be resolved within the SUBMIT command. One way is to write the X statement to a temporary external file and then use the %INCLUDE statement to execute it.

MACRO %FLATOUT

Macro %FLATOUT on Appendix II is called when users need to output the data to a flat file. It gathers the output formats from the data warehouse standard formats stored in the SAS/ACCESS views for columns/aliases found in the views. Format DATE7. is changed to MMDDYY4.||YEAR4. to fit the FOCUS date format MDYY in the data warehouse. For columns/aliases not in the views (usually new columns created from arithmetic and DB2 SQL functions), format 12.2 is used for numeric columns and '$'||TRIM(LENGTH)||'.' for character columns (LENGTH comes from DICTIONARY.COLUMNS table in PROC SQL). Then it determines the variable lengths from the formats by capturing the number between the dot and the 1st non-numeric byte on the left of the dot. Based on the lengths, it calculates the logical record length to allocate the output flat file on disk or tape (overwrite if exist). Finally, it writes out the data to the flat file and prints out the file structure description for the flat file. The nested macro %MASTERFD is called when users want a FOCUS master file description (MFD) for the output flat file. %MASTERFD reads in the FOCUS usage formats and the aliases from existing MFDs for the FOCUS views in the data warehouse (P12.2 for numeric and 'A'||LENGTH for character columns/aliases not found in data warehouse). Finally, %MASTERFD determines the storage formats and writes out the MFD (including starting and ending positions for each field and detailed header information—number of fields, record length, etc.). By using data warehouse standard formats and including starting and ending positions for each field (not necessary for MFD), the data are more easily shared and understood for different users.

CONCLUSION

Based on an efficient macro %JOINDB2, two additional macros %QUERYDB2 and %FLATOUT, and a window interface have been developed. They work together as an application to improve the efficiency (both computer and human) when accessing data warehouse DB2 data. Because of its efficiency, flexibility and easy-to-use window interface, this application has been extensively used for various projects by all levels of users (including non-programmers). The development of this application demonstrates that (1) a small but efficient program (e.g., macro %JOINDB2) may evolve into a useful application with further investigations/explorations; and (2) traditional and basic concepts/approaches (macro windows along with their control macros in this application) are still very powerful and may give you a handy, quick but still very effective solution.

REFERENCES

Yang, J. (1998), "Efficiently Access Data Warehouse with PROC SQL Pass-Through Facility and Macro," *Proceedings of the Eleventh Annual NorthEast SAS Users Group Conference*, 11, 600-606.

SAS Institute Inc. (1994), *SAS/ACCESS Software for Relational Databases: Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1987), *SAS Guide to Macro Processing, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

I would like to thank John Newell for his review and suggestions on this paper. Also, I want to thank John Newell and Highmark Blue Cross Blue Shield for their support on developing this application.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

James Yang
Highmark Blue Cross Blue Shield
120 Fifth Avenue, Suite P7205
Pittsburgh, PA 15222-3099
Phone: (412) 544-2433
Fax: (412) 544-0700
James.Yang@Highmark.com

APPENDIX I

```
%MACRO DB2INFO;
```

```
 %LET BELL=;
 %STRTOVER: %DISPLAY DB2INFO BLANK &BELL;
 %IF %UPCASE(&SYSCMD)=QT %THEN %STR(ENDSAS;);                /*quit application*/
 %ELSE %IF %UPCASE(&SYSCMD)=BK AND &ACTION=F %THEN %FLATIN;  /*back a window*/
 %ELSE %IF %UPCASE(&SYSCMD)=BK AND &ACTION=S %THEN %SASIN;   /*back a window*/
 %ELSE %DO;
  *-----Fill in all fields except WHERE/GROUP BY clause-----*;
  %LET TABLES=%QUPCASE(&TABLES1&TABLES2&TABLES3);
  %LET DB2COLM=%QUPCASE(&DB2KEY,&COLS1&COLS2&COLS3&COLS4&COLS5);
  %DO %WHILE(&TABLES EQ OR &DB2COLM EQ OR
             %BQUOTE(&VIEWNAME) EQ OR %BQUOTE(&DB2KEY) EQ );
    %LET SYSMSG=All fields are required except WHERE/GROUP BY clause.;
    %LET BELL=BELL;  %GOTO STRTOVER;
  %END;
  %LET I=1;      /*check existence of tables*/
  %LET TABLE=%QSCAN(&TABLES,&I,',');
  %DO %WHILE (&TABLE NE );
    PROC SQL NOPRINT OUTOBS=1; CONNECT TO DB2(SSID=WDB2);
      SELECT * FROM CONNECTION TO DB2 (SELECT * FROM &TABLE);
      %LET OBS=&SQLOBS;   DISCONNECT FROM DB2;
    QUIT;
    %IF &OBS=0 %THEN %DO;
     %LET SYSMSG=Table &TABLE does not exist or you do not have access.;
     %LET BELL=BELL;  %GOTO STRTOVER;
    %END;
    %ELSE %DO;   /*check existence of columns on an existing table*/
     %LET J=1;
     %LET COLM=%QSCAN(&DB2COLM,&J,',+-*/|');
     %DO %WHILE (&COLM NE );
      %LET P1=%QSUBSTR(&COLM,1,1);
      %IF &P1=1 OR &P1=2 OR &P1=3 OR &P1=4 OR &P1=5 OR
          &P1=6 OR &P1=7 OR &P1=8 OR &P1=9 OR &P1=0 %THEN %GOTO BYPASS;
      %LET LP=%INDEX(&COLM,%STR(%()); %LET RP=%INDEX(&COLM,%STR(%)));
      %IF &LP NE 0 AND &RP NE 0 %THEN %LET COLM=%QSUBSTR(&COLM,&LP+1,&RP-&LP-1);
      %IF &LP NE 0 AND &RP EQ 0 %THEN %LET COLM=%QSUBSTR(&COLM,&LP+1);
      %IF &LP EQ 0 AND &RP NE 0 %THEN %LET COLM=%QSUBSTR(&COLM,1,&RP-1);
      %LET DT=%INDEX(&COLM,DISTINCT);
      %IF &DT NE 0 %THEN %LET COLM=%QSUBSTR(&COLM,&DT+9);
      %IF &COLM EQ %THEN %GOTO BYPASS;
      PROC SQL NOPRINT OUTOBS=1; CONNECT TO DB2(SSID=WDB2);
       SELECT * FROM CONNECTION TO DB2 (SELECT &COLM FROM &TABLE);
       %LET OBS=&SQLOBS;  DISCONNECT FROM DB2;
      QUIT;
      %IF &OBS=0 %THEN %DO;
       %IF %QSCAN(&COLM,1)=&COLM
       %THEN %LET SYSMSG=Column &COLM is not in table &TABLE.;
       %ELSE %LET SYSMSG=Not all columns in &COLM are in table &TABLE.;
       %LET BELL=BELL;  %GOTO STRTOVER;
      %END;
      %BYPASS: %LET J=%EVAL(&J+1);
               %LET COLM=%QSCAN(&DB2COLM,&J,',+-*/|');
     %END;
    %END;        /*end checking existence of columns*/
    %LET I=%EVAL(&I+1);
    %LET TABLE=%QSCAN(&TABLES,&I,',');
  %END;          /*end checking existence of tables*/
  %IF %UPCASE(&MORELINE)=Y %THEN %MORELINE;
  %ELSE %OUTPUT;       /*next window*/
 %END;
%MEND DB2INFO;
```

APPENDIX II

```
%MACRO FLATOUT
```

```
( VIEWNAME,    /*SAS/ACCESS view name for any one of the DB2 tables where data &SASOUT
                 comes from.*/
  FLATFILE,    /*full name of the output flat file. Overwritten if exist.*/
  SASOUT=TEMP, /*any SAS dataset from data warehouse which needs to be output to flat file.
                 Default is WORK.TEMP, the default output from %JOINDB2 and %QUERYDB2.*/
  UNIT=DISK,   /*storage media for output flat file: DISK (default) or TAPE.*/
  TAPRETPD=30, /*retention period (default 30 days) for tape if UNIT=TAPE.*/
  MASTERFD=    /*full name of FOCUS MFD for output flat file. Default is no MFD.*/
);

/*-------get libname and member name in &SASOUT------*/

  %LET SASOUT=%UPCASE(&SASOUT);
  %IF %INDEX(&SASOUT,.)=0 %THEN %DO;
    %LET LIBNAME=WORK; %LET MEMNAME=&SASOUT;  %END;
  %ELSE %DO; %LET LIBNAME=%SCAN(&SASOUT,1,.);
             %LET MEMNAME=%SCAN(&SASOUT,2,.); %END;

/*--------get variable names & their formats--------*/

  LIBNAME HIW 'PDATAMGT.SAS.VIEWS' DISP=SHR;     /*data warehouse SAS/ACCESS views*/
  PROC SQL;
    CREATE TABLE VARNAME AS
    SELECT NAME,VARNUM,TYPE,LEFT(PUT(LENGTH,3.)) AS LENGTH
    FROM DICTIONARY.COLUMNS
    WHERE LIBNAME="&LIBNAME" AND MEMNAME="&MEMNAME"
    ORDER BY VARNUM;

    CREATE TABLE VARFMT AS
    SELECT V.*, D.FORMAT
    FROM   VARNAME V LEFT JOIN DICTIONARY.COLUMNS D
      ON   LIBNAME='HIW' AND MEMNAME="&VIEWNAME" AND V.NAME=D.NAME
    ORDER BY V.VARNUM;
  QUIT;

/*change format DATE7. to MMDDCCYY and assign formats to new columns*/

  DATA VARFMT(DROP=LENGTH); SET VARFMT;
    IF FORMAT=' ' AND TYPE='char' THEN FORMAT='$'||TRIM(LENGTH)||'.';
    IF FORMAT=' ' AND TYPE='num'  THEN FORMAT='12.2';     /*columns not in data warehouse*/
    IF FORMAT NE 'DATE7.' THEN OUTPUT;
    ELSE DO; FORMAT='MMDDYY4.'; OUTPUT;
             FORMAT='YEAR4.';   OUTPUT;   END;
  RUN;

/*--------determine the lengths for variables-------*/

  DATA VARFMT; SET VARFMT;
    LENGTH LEN_C $3 NOTNUM $1;
    DOTPOSIT=INDEX(FORMAT,'.');
    DO I=(DOTPOSIT-1) TO 1 BY -1 UNTIL(INDEXC(NOTNUM,'0123456789')=0);
      NOTNUM=SUBSTR(FORMAT,I,1);
    END;              /*1st non-numeric byte on the left of the dot in the format*/
    LEN_C=SUBSTR(FORMAT,I+1,DOTPOSIT-I-1);
    LEN_N=INPUT(LEN_C,3.);
    DROP NOTNUM DOTPOSIT I;
  RUN;




/*--put variables & their formats into &VAR_FMT for PUT statement--*/
/*----calculate logical record length (LRECL) for the flat file----*/
```

```
  PROC SQL NOPRINT;
   SELECT NAME||' '||FORMAT INTO :VAR_FMT SEPARATED BY ' '
   FROM VARFMT ORDER BY VARNUM,FORMAT;
   SELECT LEFT(PUT(SUM(LEN_N),4.)) INTO :LRECL FROM VARFMT;
  QUIT;

/*-----allocate flat file (if exist then overwrite)-----*/

  DSNEXST &FLATFILE;
  %IF &SYSDEXST %THEN %LET DISP=OLD;                    /*overwrite*/
  %ELSE %LET DISP=NEW;                                   /*create*/

  %IF %UPCASE(&UNIT)=TAPE %THEN %DO;
   %LET BLKSIZE=%EVAL(&LRECL*%SYSFUNC(FLOOR(32760/&LRECL)));
   FILENAME FLATFILE "&FLATFILE"
            DISP=(&DISP,CATLG,DELETE) UNIT=CTAPE LABEL=RETPD=&TAPRETPD
            RECFM=FB LRECL=&LRECL BLKSIZE=&BLKSIZE;
   %END;
  %ELSE %DO;
   %LET BLKSIZE=%EVAL(&LRECL*%SYSFUNC(FLOOR(6400/&LRECL)));
   FILENAME FLATFILE "&FLATFILE"
            DISP=(&DISP,CATLG,DELETE) UNIT=SYSTSO
            SPACE=(CYL,(150,50),RLSE)
            RECFM=FB LRECL=&LRECL BLKSIZE=&BLKSIZE;
  %END;

/*-----------write out flat file----------*/

  OPTIONS MISSING=' ';     /*blank instead of default '.'*/
  DATA _NULL_; SET &SASOUT; FILE FLATFILE; PUT &VAR_FMT; RUN;

/*--------------print flat file structure description------------*/

  PROC SORT DATA=VARFMT NODUPKEY; BY VARNUM; RUN;
  DATA VARFMT; SET VARFMT;
    LENGTH SASFMT $16 DATATYPE $14;
      SASFMT=FORMAT;  DATATYPE='NUMERIC';
    IF FORMAT IN ('MMDDYY4.','YEAR4.') THEN DO;
     SASFMT='MMDDYY4.||YEAR4.';  DATATYPE='DATE(MMDDCCYY)';
     LEN_N=8;  LEN_C='8';
    END;
    IF FORMAT=:'$' THEN DATATYPE='CHARACTER';
    RETAIN STARTPOS 1;
    STARTPOS=SUM(STARTPOS,LAG(LEN_N));  /*LAG() in 1st obs is missing*/
    ENDPOS=STARTPOS + LEN_N - 1;
  RUN;
  PROC PRINT NOOBS SPLIT='*';
    TITLE5 "FLAT FILE DESCRIPTION FOR &FLATFILE";
    VAR VARNUM NAME SASFMT DATATYPE STARTPOS LEN_N;
    LABEL VARNUM='FIELD ORDER' NAME='FIELD NAME' SASFMT='SAS FORMAT'
          DATATYPE='DATA TYPE' STARTPOS='START POSITION'
          LEN_N='FIELD LENGTH';
  RUN;

/*-------nested macro to create FOCUS master file description------*/

 %MACRO MASTERFD;
  DATA MFD;  INFILE "PFOCUS.BCWP.MASTER.DATA(&VIEWNAME)"     /*MFD in data warehouse*/
             DELIMITER = ',' MISSOVER DSD;
    FORMAT  FIELD $22. ALIAS $22. USAGE $15.;
    INPUT FIELD $ ALIAS $ USAGE $;
    IF FIELD =: 'FIELD=';
    LENGTH FLD_NAM $8; FLD_NAM=SCAN(FIELD,2,'=');
  RUN;
  PROC SQL NOPRINT;
    CREATE TABLE MFD AS
    SELECT VARFMT.*, MFD.*, 'A'||LEN_C AS ACTUAL
    FROM   VARFMT LEFT JOIN MFD
    ON     FLD_NAM=NAME
    ORDER BY VARNUM;
    SELECT LEFT(PUT(MAX(VARNUM),3.)) INTO :VARCNT FROM VARFMT;
  QUIT;

  DATA _NULL_; SET MFD;
    IF FIELD=' ' THEN DO;     /*assign usage formats to new columns*/
      FIELD='FIELD='||NAME; ALIAS=NAME;
      IF TYPE='char' THEN USAGE=ACTUAL; ELSE USAGE='P'||FORMAT;
    END;                                          /*numeric*/
    FILE "&MASTERFD";
    IF _N_=1 THEN
    PUT '$$*' 77*'*' /
        '$$*' 32*' ' "&MEMBER" /
        '$$*' 77*' ' /
        "$$* FOCUS FILE DEFINITION FOR FLAT FILE &FLATFILE" /
        '$$*' 77*' ' /
        '$$*' 20*' ' "(&VARCNT FIELDS,   &LRECL BYTES)" /
        '$$*' 77*' ' /
        "$$* LOCATION:  &MFDPDS" /
        "$$*  CREATED:  &TODAY" /
        '$$*' 77*' ' /
        '$$*' 77*'*' /
        " FILENAME=&MEMBER,SUFFIX=FIX,$" /
        ' SEGNAME=SEG0,$' /
        '$$';
    PUT @2 FIELD $22.   @24 ','  @25 ALIAS $22.  @47 ','
        @48 USAGE $15.  @63 ','  @64 ACTUAL $5.  @69 ',$'
        @71 STARTPOS Z4. @75 '-'  @76 ENDPOS Z4.;
  RUN;
 %MEND MASTERFD;

/*--------create FOCUS master file description--------*/

 %IF %BQUOTE(&MASTERFD) NE  %THEN %DO;
  %LET MFDPDS=%SCAN(&MASTERFD,1,'(');          /*get PDS name*/
  %LET MEMBER=%SCAN(&MASTERFD,2,'(');
  %LET MEMBER=%SCAN(%BQUOTE(&MEMBER),1,')');   /*get PDS Member name*/
  %LET TODAY=%SYSFUNC(DATE(),YYMMDD10.);
  %MASTERFD;
 %END;

%MEND FLATOUT;
```