The SQL Procedure and DB2\textsuperscript{2} : Coding Without Confusion

Nancy Fictum, Wisconsin Public Service Corporation, Green Bay, WI

Abstract

This paper illustrates how to create an application that allows you to "cut" and "paste" PROC SQL code which uses the Pass Through Facility. The benefits of this application are:

* It can be used on a corporate wide basis for all DB2 tables.
* It addresses the problem of the 8 character SAS names vs. 18 character DB2 names.
* It runs in a batch environment.
* It eliminates LOTS of key strokes.
* It's easy to implement and maintain so you can get a life again.

The audience for this paper should understand the basic concepts of the Pass Through Facility and an SQL SELECT statement.

Introduction

Wisconsin Public Service Corporation implemented a DB2 based Customer Information System (CIS) in mid 1989. Because of possible contention problems against the live data for corporate wide reporting, most tables were extracted to flat files and The SAS System\textsuperscript{3} was used to create reports. Even as Version 6 introduced SAS Access and View descriptors, performance was no better than the sequential access of the flat file extracts. It was not until the introduction of PROC SQL and the Pass Through Facility that allowed DB2 to optimize the data retrieval and increase performance.

Once the decision was made to access data directly from the DB2 tables, a large conversion effort was required to change each existing application from DATA/INFILE code to PROC SQL.

Prior to 1989, the Customer Information System was stored in IBM's IMS\textsuperscript{4} database. This is important because all element names were defined with a maximum of 8 characters. Those names were retained and documented with the 18 character DB2 names in our corporate dictionary (Brownstone Solutions' Brownstone dictionary). This provides a significant benefit to all existing applications because all data analysis and reports did not have to be recoded since those same 8 character names are still used in this new application.

The remainder of this paper will describe how to set up the application (one time), and execute it multiple times for any defined DB2 table.

Setting up the Application

This application runs in an IBM MVS/TSO environment, and has two parts:

1) defining the DB2 tables to create the PROC SQL code, and
2) using a "paste" facility to create the code for the Pass Through Facility.

Defining the DB2 Tables

A TSO/REXX named CVIEW is used to prompt for the following:

* DB2 Subsystem : RP  (RP = DB2 Production)
  (RO = DB2 Development)
* DB2 Creator Name: CISEXTR
* DB2 Table Name: CIS006V

Note: Actually, any software that can create an interactive screen to prompt for the above values and pass them to a batch job, can be used.

The words above in bold, are examples of what you would fill in on the screen.

505
FIGURE 1 is a partial listing of the REXX after the above values have been filled in on the screen.

***************FIGURE 1**************
SET PARTTABL = &SUBSTR(1:6,&TABLE)
SET TBLORVU = &SUBSTR(7:7,&TABLE)
SET ALETTR = A
SET VLETTR = V
SET ACCESS = &PARTTABL&LETTR
"CISOO5A"
SET VIEW = &PARTTABL&LETTR
"CISOO5V"
IF &HIQUAL = CISPROD +
THEN DO
  SET &CREATOR = &HIQUAL
  SET &HIQUAL = CISEXTR
  SET &TBNAME = &VIEW
ENDO
ELSE DO
  SET &TBNAME = &TABLE
  SET &CREATOR = &HIQUAL
ENDO
IF &TBLORVU = T +
THEN DO
  SET DD = 1V
ENDO
ELSE DO
  SET DD = 2V
ENDO
IF &SS = RP +
THEN SET SSID = DB2P
ELSE SET SSID = DB2D

These values are then passed to a batch job. This is really the core of the PROC SQL definition. FIGURE 2 is the actual code that is executed.

***************FIGURE 2**************
LOG=X,PRINT=X,COPY=1,VIEW=AN.PROD,
DB2="&SS"
//SASFILE DD DSN=AZ.&SS..SAS.SOLCODE
//DB2FILE DD DSN=AZ.&SS..DB2.SOLCODE
//SYSIN DD *
LIBNAME VIEWS 'AZ.SOLPFT.&SS..SASVIEWS'
SERVER=SERVER6;

* THIS JOB CREATES ACCESS AND VIEW DESCRIPTORS IN A BATCH ENVIRONMENT*
* THERE ARE 2 DIFFERENT SAS LIBRARIES: ONE FOR DEVELOPMENT, ONE FOR PRODUCTION;

PROC ACCESS DBMS = DB2;
  CREATE VIEWS.&ACCESS .. ACCESS;
  SSID=&SSID;
  TABLE=&HIQUAL .. &TBNAME;
  ASSIGN=YES;
  UNIQUE=YES;
  CREATE VIEWS.&VIEW .. VIEW;
SELECT ALL;
/*PROC CONTENTS DATA=VIEWS.&VIEW OUT=VIEWOUT NOPRINT;*/
PROC SOL;
  CREATE TABLE SASNAMES AS
    SELECT DB2_NAME / 'FROM DICTIONARY'/
       ,SAS_NAME / 'FROM DICTIONARY'/
       ,NAME / 'FROM SAS VIEW'/
FROM ANVIEW,DD000&D A
WHERE ACREATOR = &CREATOR'
AND ATBL_NAME = &TABLE'
AND A.DB2_NAME = B.LABEL
ORDER BY A_SEQNO;
RUN;
PROC SORT DATA=SASNAMES NODUPKEY;
BY DB2_NAME;
RUN;
DATA_NULL_;  
SET SASNAMES END = LAST; 
FILE SASFILE(&TBNAME);  /SAS SELECT*/ IF N = 1 
THEN DO;  
PUT @2 'PROC SOL;'; 
PUT @2 'CONNECT TO DB2(SSID=&SSID);'; 
PUT @2 'CREATE TABLE FIRST AS SELECT; 
PUT @5 NAME @15 'AS' @20 SAS_NAME ; 
END; 
ELSE DO; 
PUT @3 ',' @5 NAME @15 'AS' @20 SAS_NAME ; 
END; 
IF LAST THEN DO; 
PUT @5 '/ * .... +++ ............. ++/..; 
PUT @2 'FROM CONNECTION TO DB2; 
END; 
DATA_NULL_;  
SET SASNAMES END = EOF; 
FILE DB2FILE(&TBNAME);  /Db2 SELECT*/ IF N = 1 
THEN DO; 
PUT @3 'Z: @5 DB2_NAME @30 /* @33 
SAS_NAME @42 */'; 
END; 
ELSE DO; 
PUT @2 ',' @3 'Z: @5 DB2_NAME @30 /* @33 
SAS_NAME @42 */'; 
END; 
IF EOF 
THEN DO; 
PUT @5 /*FROM &HIOUAL. &TBNAME */; 
PUT @5 ');'; 
END;

The above code is executed once for each DB2 table/view or whenever the table structure changes. The members created from this code are used company wide by "pasting" them into your own batch program. Next, FIGURE 3 is an example of the SAS SELECT statement.

**********FIGURE 3**********

PROC SQL; 
CONNECT TO DB2(SSID=Db2p); 
CREATE TABLE FIRST AS SELECT 
U_REV_TA AS EURVTAXD 
C_STATE AS ECTSTATE 
T_TAX_DI AS ETTDSHRT 
U_TAX_DI AS EUTAXDST 
U_OP_DIS AS EUOPRDST 
T_TAX_D AS ETIAXDST 
C_MNCPLT AS ECTDMCPY 
N_COUNTY AS ENTAXCY 
F_DLC_SG AS FDLCSGNL 
F_SESNL_ AS FSESNLRO 
/***********/ FROM CONNECTION TO DB2 
***********/

The code above creates the SAS select statement listing each column in the table on a different line (so columns not required can be easily deleted), and the 8 character SAS data dictionary name stored on the corporate data dictionary. The "AS" keyword creates the alias name to be stored in the SAS dataset.

FIGURE 4 is the DB2 select statement which is passed directly to DB2 for processing. Again each column is selected with the 8 character SAS dictionary name as a comment. This helps to match and delete columns that were deleted in the SAS select statement.

**********FIGURE 4**********

(SELECT 
 Z.U_REV_TAX_DIS /* EURVTAXD */ 
Z.C_STATE /* ECTSTATE */ 
Z.T_TAX_DIST_SHORT /* ETTDSHRT */ 
Z.U_TAX_D /* EUTAXDST */ 
Z.U_OP_DIS /* EUOPRDST */ 
Z.T_TAX_D /* ETIAXDST */ 
Z.C_MNCPLT /* ECTDMCPY */ 
Z.N_COUNT /* ENTAXCY */ 
Z.F_DLC_SGNL_AVAIL /* FDLCSGNL */ 
Z.F_SESNL_ENTRY_RORD /* FSESNLRO */ 
FROM CISEXTR.CISW24V );

507
The code puts a "Z." in front of each column to create a table alias name. You can perform a CHANGE ALL on "Z." to another letter in case more than one table is joined in the code. The last parenthesis and semicolon (;) are on a line themselves so any WHERE, ORDER BY, GROUP BY, etc. can be inserted within the DB2 select statement.

**Pasting the PROC SQL code**

Once the ISPF members have been created, you can now "paste" them into another member after the JCL. In EDIT mode of ISPF, you type the following on the COMMAND line:

```
DP CISO05V
```

The CISO05V is the name of the table or view.

**Figure 5** shows a partial listing of the EDIT MACRO (DP) code used to paste the DB2 SQL SELECT statement.

```
008 Note: SP CISO05V on the ISPF COMMAND line will paste in the SAS SELECT statement. Be sure you place the A (after) or B (before) on the line numbers.
```

---

**The Final Product**

Figure 6 shows an example of two tables joined using this application. The only coding required is the WHERE clause and some details such as commas, and alias names.

```
//STEP1 EXEC SASV6,LOG=X,PRINT=X,SSID=DB2P;
//WORK DD UNIT=DISK,SPACE=(CYL,(50,5))
//SYSIN DD *
PROC SQL;
CONNECT TO DB2(SSID=DB2P);
CREATE TABLE FIRST AS
  SELECT C_MNCPLT AS ECTDMCPY,
       N_COUNTY AS ENNTAXCTY
  FROM CISEXTR.CISW24V A,
   CISEXTR.CISW25V B
IM-IERE A.U_OP_DIST = B.U_OP_DIST AND
   B.U_OP_DIST = 81;
%PUT &SOLXRC;
%PUT &SOLXMSG;
```

---

508
Conclusion

We have found that applications converted from the flat file extracts to this application using the Pass Through Facility run about twice as fast. They run about 4 times faster than using SAS ACCESS and VIEW descriptors.

The most significant benefits of using this application include:

* you do not need to know the truncated 8 character SAS name.

* you do not need to know the DB2 name (which can be up to 18 characters in length).

* you can make use of the efficiencies PROC SQL has to offer (i.e. WHERE, ORDER BY, faster processing times).

* if your SAS Consultant or DB2 DBA maintains the TABLE/VIEW definitions, you only need to know how to use the "paste" facility.

Future Directions

As with most applications, improvements could always be made based on changing business conditions and/or user feedback. Some of the enhancements to this one include:

A. Creating a batch job that sets up the application for two or more tables vs. one table at a time.

B. Writing an edit macro so any columns deleted in the SAS SELECT are matched and deleted in the DB2 SELECT (and vice versa).

C. Creating an enhanced "cut" and "paste" routine or "undo" routine so that if column(s) are inadvertently deleted, they can be recaptured.

D. Making sure ALL DB2 Tables have the 8 character SAS dictionary name.

Acknowledgements

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

DB2, IMS are registered trademarks or trademarks of IBM Corporation in the USA and other countries. ® indicates USA registration.

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

Author Contact

Nancy Fictum
Wisconsin Public Service Corporation
600 N. Adams St.
Green Bay, WI  54307-9002
(414) 433-1880