A Universal SAS/Access Macro to Retrieve Data from DB2 Tables

Chen Xi
PharmClinet Co.

1. Introduction

The SQL Pass-Through facility of SAS/Access interface software has provided users a convenient tool to retrieve data from DB2 RDMIS tables and convert them into SAS format. Generally, people write access programs in the form of SAS macros which can be shared by a group of users. With more and more applications developed and DB2 tables created, the size of the access macro library can grow rapidly and is hard to maintain. For example, if a new field is added into a DB2 table, a number of access macros may need to be removed from the production environment, revised, validated and restored. Also, the fields which an access macro retrieves are generally hard coded into the programs, and it is not easy for a user to retrieve an arbitrary subset of these fields. Macro parameters can be widely used to make a macro flexible, but no one is in favor of writing a macro with many required parameters or with parameters which need long values. There is always a trade off between the flexibility of a macro and the convenience of making a call. In this paper, we present a macro which can be used for many DB2 tables with similar structures without using awkward parameters. Note that the implementation of this strategy relies on the specific database design, and it again emphasizes the coordination between SAS programming team and database management team.

2. Problem

We consider only a simple case to demonstrate our strategy. In this case, there is only one core table to be retrieved and one reference table to be outer joined a number of times. For example, the core table named Status looks like:

<table>
<thead>
<tr>
<th>Bgt Code</th>
<th>Res Code</th>
<th>ID No</th>
<th>Visit</th>
<th>Stat Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1A</td>
<td>ABC</td>
<td>1011</td>
<td>3</td>
<td>P</td>
</tr>
<tr>
<td>F2B</td>
<td>EDF</td>
<td>1201</td>
<td>5</td>
<td>R</td>
</tr>
</tbody>
</table>

Another core table named Outcome looks like:

<table>
<thead>
<tr>
<th>Res Code</th>
<th>Col Code</th>
<th>Col Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>Athlin</td>
<td></td>
</tr>
</tbody>
</table>

The reference table named Reference looks like:

<table>
<thead>
<tr>
<th>Col Name</th>
<th>Col Code</th>
<th>Col Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res Code</td>
<td>ABC</td>
<td></td>
</tr>
<tr>
<td>Out Code</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Stat Code</td>
<td>P</td>
<td></td>
</tr>
</tbody>
</table>

Our goal is to retrieve data from a core table, then outer join the reference text based on the various codes. For example, when using Status and Reference, we can develop a macro so that the output SAS data set may look like this:

<table>
<thead>
<tr>
<th>B</th>
<th>R</th>
<th>R</th>
<th>I</th>
<th>V</th>
<th>S</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>E</td>
<td>E</td>
<td>D</td>
<td>I</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>S</td>
<td>S</td>
<td>N</td>
<td>S</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>T</td>
<td>O</td>
<td>C</td>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bgt Code</th>
<th>Res Code</th>
<th>ID No</th>
<th>Visit</th>
<th>Stat Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1A</td>
<td>ABC</td>
<td>1011</td>
<td>3</td>
<td>P</td>
</tr>
<tr>
<td>F2B</td>
<td>EDF</td>
<td>1201</td>
<td>5</td>
<td>R</td>
</tr>
</tbody>
</table>

The problem is that if we refer to Outcome and Reference tables, can we still use the same access macro to reach our goal? Generally, since the Status and Outcome tables are different, we have to write two separate macros. There might be hundreds of core tables, so that hundreds of macros need to be maintained. We will show that since the two tasks are so similar, it is possible to develop a universal access macro to accomplish the task.

3. Strategy

Before going into the details of our macro, it will be helpful to take a look of the conventional SAS program to do the same work.

PROC SQL;
CONNECT TO DB2 (SSID=DB2F);
CREATE TABLE TEMPAS AS
SELECT * FROM CONNECTION TO DB2
(SELECT S.BGT_CODE, S.RES_CODE, S.ID_NO,
S.RESP_CODE, S.OUT_CODE)

Chen Xi was a consultant in Eli Lilly and Company when this paper was written, and he is currently working in Pfizer, Inc. as a consultant.

Nesug '96 Proceedings
Our strategy is as following:

1. User will provide core table name and table creator’s name;
2. The macro will retrieve DB2 field names and the corresponding SAS field names from DB2 system catalog;
3. The macro will decide which codes can be resolved, and the SAS names for text fields;
4. The core part of the Where clause is hard coded with macro parameters, and user can use an optional macro parameter to provide the rest.

In the following, we will present the basic structure of our universal access macro.

4. DB2 field names and SAS field names

DB2 maintains a system catalog which consists of a series tables. One of the tables is SYSCOLUMNS. Given a DB2 table name and table creator’s name, three columns in SYSCOLUMNS are of our special interests. They are NAME, LABEL and KEYSEQ respectively. For a specific table, NAME contains the DB2 field names of all the columns; LABEL contains the DB2 labels of all the columns, and we make them identical to the corresponding SAS names; KEYSEQ contains the sequence number of the primary keys in that table, and for any non-key field, the sequence number is 0.

The above module simply extracts the three fields and save them into a SAS data set named AAA.

Finally, three sets of macro variables are created and they take the values of DB2 field names, SAS field names and Key field names respectively. Each of the macro variables has a sequence number associated with it, and this keeps the information about the order of the fields. The macro variable NF represents the number of the fields, and it is the maximum number of the fields the macro will handle.

5. Retrieve Data From Core Table

The following SQL module is used to retrieve data from the core table and create a SAS data set named TEMP:

Advanced Tutorials

Nesug '96 Proceedings
In the call, the table creator's name and the table name are passed in by the macro parameters CLIN and TBNAME; the macro parameters BGTCODE and RESCODE are used to generate the core part of the Where clause, and the users can take advantage of the macro parameter MORECODE to fulfill their specific needs. For example, one can say:

%LET MORECODE = AND T.ID_NO = 1011;

and pass it in when he makes the macro call. The generation of the DB2 field names and the SAS field names is accomplished by the following macro program:

%MACRO QQQ (NF=,&SYS=,&JOIN=,&KEYNM=,&CODELIST=,&OUT=,&PRE=,&NF=,&NV=);%MENDQQQ;

In the call, NF represents the number of the fields to be retrieved. When &SYS=DB2NM, it generates all the DB2 field names, and when &SYS=SASNM, it generates all the SAS field names. See the SAS program in section 3, each of the field names is preceded with a coma except the first one, it is controlled by a %IF .. %ELSE. By default, each of the fields has a prefix T. When generating the SAS field names, it is not needed so that the macro parameter PRE is overwritten by a blank value. The parts related to the macro parameters JOIN, KEYNM and CODE will be used in outer join which will be discussed in the later sections.

In implementation, the macro QQQ must be defined before it is called.

6. Which Codes Can be Resolved

After the core data set is available, there might be some codes needing to be resolved. The macro tries to find all the fields whose DB2 name has the word Code in it.

%LET CODECOM = ;
%DO I = 1 %TO &NF;
      %IF %INDEX(ANOSOLVE,&DB2NM&I,CODE) = 0 AND %INDEX(ANOSOLVE,&DB2NM&I,CODE)= 0 THEN %DO;
      %LET CODECOM = &CODECOM,%STR(&%DB2NM&I,%STR(1));
      %END;
%LET CODELIST = (%CMPRES(%QUOTF(,(&CODECOM,2,%LENGTH(%QUOTF(,(&CODECOM)))-1));
%LET CODELIST = (%CMPRES(%QUOTF(,(&CODECOM));

The macro function SUBSTR is used to eliminate the coma in the front of the first field. The macro variable NOSOLVE is used to eliminate the code fields which do not need to be resolved.

PROC SQL; CREATE TABLE REF AS
  SELECT * FROM CONNECTION TO DB2 (SSID=DB2P)
  (SELECT G.COL_NAME, G.COL_CODE, G.COL_TEXT
   FROM &CLIN . .REFERENCE G
   WHERE G.COL_NAME IN &CODELIST)
AS REF (COLNAME,COODE,COOLTEXT);
DISCONNECT FROM DB2;

The above SQL module will retrieve all the reference text information from the REFERENCE table.

PROC SQL;
CREATE TABLE ADDCODE AS
SELECT DISTINCT A.NAME, A.LABEL, B.COLNAME, A.KEYSEQ
FROM AAA A
LEFT JOIN REF B
ON A.NAME = B.COLNAME;
PROC SORT DATA=ADDCCODE;
BY KEYSEQ NAME;

After the table REF is created, there might be some code fields which do not exist. For example, there is no records in REF with Col_Name Med_Code. In that case, there should not be a text field associated with it. In this module, the first step tries to determine which codes can be found in REF. If the code field has text associated with it, COLNAME will not be null.

DATA _NULL_;
SET ADDCODE;
IF COLNAME = ' ' THEN CALL SYMPUT('CODE','LEFT(_N_,)');
ELSE CALL SYMPUT(CODE,'LEFT(_N_).
SUBSTR(LABEL,1,LENGTH(LABEL)-1))T;

Finally, in this data step, a series of the macro variables, CODEi, are created. For a record if COLNAME is null, CODE will be also null; if COLNAME is not null, the CODE will take the corresponding SAS name and change the last character from C to T, for text.

7. Outer Join to Resolve the Codes

With all the information available, it is easy to do the outer join as follows:

PROC SQL;
CREATE TABLE &OUT AS
SELECT %QQQ(NF=&NF,&SYS=SASNM,&JOIN=Y)
FROM TEMP T
%IF(NF=&NF)
ORDER BY %QQQ(NF=&NF,&SYS=KEYNM);

Where macro parameter OUT is used to represent the name of the output data set, and the data set is sorted based on the key fields. Again, the macro program QQQ is used here to generate the SAS field names.

The creation of the outer join criteria is done by the following macro program:

%MACRO QQQ(NF=&NF);
%DO I = 1 %TO &NF;
%IF &CODEI = ' ' THEN %DO;
LEFT JOIN REF G&I
ON T.&:&:SASNM&I = G&I.COLCODE
AND G&I.COLNAME = %UNQUOTE(%STR(&%&DB2NM&I,%STR(1));
%END;
%MACRO P(NFO&NF);
%MACRO V(NFO&NF,8YS=KEYSEQ).
%MACRO L(NFO&NF,8YS=KEYSEQ).
As discussed previously, when the macro variable CODE is not null for some CODEi, an out join is needed. Note that the UNQUOTE function is needed. Without it, the macro will generate the code, but will not execute it. Note that we have assumed that the first field, Bgt_Code, does not need to be resolved. If it does, the macro needs a small change.

8. Discussion

As we have claimed in the beginning, we have presented only a simple form of this macro. For the people who are interested in putting the macro in their application library, it will be helpful to discuss some of the implications and potential enhancement.

8.1 Coordination with DBA

This macro heavily relies on the DB2 database design, in particular, the naming convention. For example, it is always assumed that the label defined for a DB2 field is the corresponding SAS field name; any code needing to be resolved must have the word CODE in its DB2 field name. Also, the SAS names of the reference text field are created by the macro, and it should not conflict with the database original design. All of these emphasize the coordination between SAS programming team with application DBA. In many cases, a small change in database design may create a user friendly working environment, and provide a big convenience for users.

8.2 More Than One Core Table

In the previous sample, we discussed only the case where there is one core table. A natural question is how to expand the macro to cover the case of multiple core tables. For example, if we need to join Status and Outcome tables together at first, then out join the Reference table. Since the information about the key fields of the both table can be found in SYSCOLUMNS table, it will not be hard to construct the join criteria. The macro variable MORECODE will provide plenty of the flexibility for the join. But, some fields, such as the key fields, will be common to both tables, and the programmer needs to define from which table a field will be extracted. Again, the creation of the join criteria rely on the naming convention. We assume that the common key fields in different tables have the same field names. If this is not the case, it is possible to set up a table controlling the mapping; or, set up a table to store the join criteria.

8.3 More Than One Reference Table

In many cases, the reference information does not fit into a single reference table. More importantly, the previous reference table is based on vertical design, a number of codes can be put in the same table. A more conventional reference table may look like

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F1A</td>
<td>A</td>
<td>A Pain Reliever</td>
</tr>
<tr>
<td>F2B</td>
<td>C</td>
<td>C Pain Reliever</td>
</tr>
<tr>
<td>F3C</td>
<td>A</td>
<td>ABC Drug</td>
</tr>
</tbody>
</table>

we call this kind of the reference table a horizontal reference table. In this case, only one type of code is resolved in the table, and the code name appears as the column name. To join this kind of the table, the join criteria might be different. A convenient way to deal with codes that are not resolved from REF table is to go through the SYSCOLUMNS table again to find where in the horizontal reference table the code can be found. To prevent retrieving the field from the core table again, one may need to name the core table and reference table differently.

9 Conclusion

In this paper, a simple universal access macro is introduced. It can be used to retrieve data from many DB2 tables with similar structure, and users can choose arbitrary subsets of fields within a table. To implement this strategy, the coordination between SAS programming team and application DBA is emphasized. Combined with some other utility tables and programs, the flexibility of this macro can be further enhanced.

10 Acknowledgment

The authors would like to express their appreciation to Mike Million, DBA of the Medical System, Eli Lilly and Company, for his consultation on the DB2 database structure.

The authors welcome any kind of comments. Please contact Chen Xi at chenxi@easyway.net.