Paper 220

An SQL List Macro to Retrieve Data from Large SAS/DB2 Databases

Thiru Satchi
Blue Cross and Blue Shield of Massachusetts

Abstract

The SQL Pass-Through facility in the SAS/Access interface software is a tool to retrieve data from DB2 Relational Database Management System (RDMS) tables and to convert them into a SAS dataset. This paper demonstrates the benefits of using this feature, along with a macro that converts a SAS dataset into a list of key values. If these key variables are indexed in a DB2 table, the macro will significantly decrease the required programming and computer processing time.

Introduction

Creating, accessing, and manipulating data located in the relational objects of a database management system (DBMS) can require extensive programming and computer processing time. Depending on the programming method used, time can be wasted when working with very large DB2 or SAS (1) tables. The application of macros that use indexes has been proven a very time-efficient strategy (2). However, these macros require extensive coding and are complex.

The following presentation will introduce an SQL macro, which creates a list of key variables from a SAS dataset that is then passed to a DB2 or another SAS table to extract data.

Merging Two or More SAS Datasets

A merge involving multiple variables from two or more large SAS datasets requires a large amount of computer processing time. To circumvent this problem, this merge can be accomplished by focusing on a single key variable from one SAS dataset to extract data from the other SAS dataset(s). This will produce a table that programmers can then refine even further with less chance of exhausting DBMS resources. However, there are limitations to listing out every single key variable: it is a tedious process and it increases the risk that typographical errors will be made.

An SQL list macro prevents the aforementioned limitations. Here is an example to demonstrate how this macro works. Suppose that information (date of birth: DOB; diagnosis: DIAG) available in a comprehensive SAS file (Appendix A) was needed for 8 individuals, who have a unique character identification (EMPID) (Listing 1).

Listing 1. SAS dataset with selected values for a character key variable.

```
1   DATA CHAR_KEY;
2    INPUT EMPID $5.;
3   CARDS;
4    T7564
5    P6504
6    T7707
7    M5596
8    V7500
9    T2117
10   A5987
11   S7654;
12  PROC SORT NODUPKEY ; BY EMPID;
```

These two SAS datasets can be merged using the SQL list macro presented in Listing 2. The SQL list macro creates a unique list of values, which are flanked by apostrophes and separated by a comma. Thus, the list of EMPID values from our example would be formatted as the following: T7564','P6504','T7707','M5596','V7500',T2117','A5987', 'S7654'.

Listing 2. SQL list macro for a character key variable.

```
1   PROC SQL NOPRINT;
2    SELECT " ' "  ||  TRIM(EMPID)  ||  " ' ";
3     INTO : CHARLIST SEPARATED BY  " , "
4     FROM CHAR_KEY;
5   QUIT;
```

This list ("CHARLIST") is then referenced in the WHERE statement and matched with the same key variable in the comprehensive SAS dataset (Listing 3). The results of the merge can be saved as a subset of the comprehensive SAS dataset or it can be printed (output is presented in Appendix B). However, the list of key variables is limited to 32K characters (Listing 4).

Listing 3. Reference to the key variable list in the WHERE statement.

```
1   PROC PRINT DATA = MASTER;
2    WHERE ID IN (&CHARLIST);
3   RUN;
```

Listing 4. WHERE statement limitation.

```
1   PROC PRINT DATA=MASTER;
2    WHERE ID IN ('T7564','P6504','T7707','M5596','V7500',
3                 'Y2117','A5987','S7654');
4   RUN;
```

Key Variable List ≤ 32K bytes

A variation of this SQL list macro is also available for numeric key variables (Listing 6). This macro requires that the length of the key variable be defined in the SELECT statement. If the length is not verified at this step, the numeric variable will have a maximum width of 8 bytes by default. Therefore, this step is important for numeric key variables with an actual display length greater than 8 bytes.

This macro also creates a list of values separated by comma. Thus for the numeric variable "MEMID" in Listing 5, the resulting list would be the following: 346987,3469871, 258766,566511,846420987,73771,464467896,921396,9213 9678,87932. A printout of the resulting merge is presented in Appendix C.

Listing 5. SAS dataset with selected values for a key variable.

```
1   DATA NUM_KEY;
2     INPUT MEMID;
3   CARDS;
4     346987
5     3469871
6     258766
7.    566511
8     846420987
9     73711
10.   464467896
11.   92139678
12.   87932
13.  921396;
14.  PROC SORT NODUPKEY ; BY MEMID;
```

Listing 6. SQL list macro for a numeric key variable.

```
1   PROC SQL NOPRINT;
2     SELECT   MEMID   FORMAT 9.
3       INTO : NUMLIST SEPARATED BY ","
4       FROM NUM_KEY ;
5   QUIT;
```

These two SQL list macros overcome the tediousness of writing out a potentially long list of variables, and the risking making a typographical error. The result is a key variable list that is automatically created from an existing dataset, along with appropriate apostrophes and commas.

Merging a SAS Table with a DB2 Table(s) in a Single Iteration

The SQL list macro is able to access DB2 tables with the Pass-Through facility (Listing 7), which is a feature of SAS PROC SQL (introduced in version 6.07) (a similar macro was developed by Craig Gavin, Reference 3). The Pass-Through facility allows SQL statements to be passed directly to a DBMS for processing without the use of Access and View descriptors and thus dramatically reduces the computer processing time (2). However, these statements are limited to 32K bytes.

Like the case of merging two or more SAS tables, this macro begins with the creation of a list that is then passed to the WHERE statement. It also takes advantage of indexes on the DB2 table(s), which reduces the computer processing time significantly, (2).

In Listing 7, data is extracted from the "DB2_IP_CLAIM" DB2 table for the individuals in the key variable list ("KEY_LIST") and is then passed to a temporary SAS dataset ("TEMPDATA").

Listing 7. SQL list macro access to a DB2 table(s).

```
1   %MACRO DB2PULL(TABLE);
2   PROC SQL;
3     CONNECT TO DB2(SSID=D2P6);
4   CREATE TABLE TEMPDATA AS
5     SELECT *
6   FROM CONNECTION TO DB2
7.  (SELECT *
8   FROM DB2_&TABLE
9         WHERE ID IN (&KEY_LIST));
10. DISCONNECT FROM DB2;
11. QUIT;
12. %PUT &SQLMSG;
13. %MEND DB2PULL;
14.
15. %MACRO EXTRACT;
16.       %DB2PULL(IP_CLAIM);
17. %MEND EXTRACT;
18. %EXTRACT;
```

Merging a Large SAS File with a DB2 Table(s) in Multiple Iterations

The limitation that only 32K bytes can be passed to the DB2 environment using the Pass-Through facility is resolved by passing the list in segments and then appending the extracted data into one file. This is accomplished by additional macros (Appendix D).

The first step in using the macros is to establish the global parameters needed for the macros (Appendix D, Lines 1-2). The parameters in Line 1 (defined in Table 1) are assigned values in a later step (Line 79, explained later). The parameters in Line 2 are temporary macro variables created and executed when various macros are called.

Table 1. Global parameters needed to execute macros needed to overcome the Pass-Through facility/DB2 data transfer restrictions.

Variable	Definition
KEY_DSN	The SAS dataset containing the key variable, which will be used to join to the DB2 table(s).
KEY_VAR	SAS variable name of the key variable used to create the list.
VAR_TYP	The key variable data type – coded as "C" for character variables or "N" for numeric ones.
VAR_LEN	The actual length (not the storage length) of the key variable.
OUT_DSN	The output SAS dataset name.
LIST_LEN	The total length of the key variable list segments.

The "%DB2PULL" macro (Appendix D, Lines 5-18) is similar to the macro in Listing 7. However, it also includes a PROC APPEND statement to join multiple DB2 data extracts into one SAS dataset. The %DB2PULL macro is also able to extract data from multiple DB2 tables with the assistance of the "%EXTRACT" macro (Appendix D, Lines 20-24), which would be the IP_CLAIM, OP_CLAIM1, and OP_CLAIM2 tables in Appendix D (Lines 21-23). There is no limit to the number of DB2 tables that can be listed in the %EXTRACT macro.

The "%RUNCOUNT" macro (Appendix D, Lines 28-40) then does several things. First, it determines the total number of observations of the key variable. Depending on the key variable type, the %RUNCOUNT macro then determines the number of observations per segment. This is accomplished by multiplying the number of key variables by the sum of the length of the key variable plus 3 for character key variables (two flanking apostrophes and one comma) or plus 1 for numeric key variables (one comma).

The %RUNCOUNT macro also determines the number of list segments that will be passed through the Pass-Through facility to DB2. This is calculated by dividing the total

length of the list by the LIST_LEN value (the CEIL function is used to round the number up to the next integer).

Next, the "%LISTSQL" macro (Appendix D, Lines 44-54) creates segmented lists of values ("KEY_LIST"), which are separated by appropriate apostrophes and commas. The creation of these lists is based on the first (BEG_OBS) and last (END_OBS) observation numbers of each segment. For each instance an SQL list macro is run, DB2-extracted data is appended to any existing data in a SAS output dataset (OUT_DSN).

If the macro is rerun, additional extracts will be appended to the original extract. To resolve this potential problem, the "%CLEARDSN" macro (Appendix D, Lines 58-62) deletes any existing data in the SAS output dataset (OUT_DSN).

The final macro, "%DATAPULL" (Appendix D, Lines 66-75), calls the %EXTRACT, %RUNCOUNT, %LISTSQL, and %CLEARDSN macros. It also determines the beginning and ending key variable observation number for each list segment, which are needed for the %LISTSQL macro.

The %DATAPULL macro is called in Line 78 of Appendix D. It is here where the values assigned to the parameters are needed for the macros. Suppose for example, we have a list of patients ("PATLIST") with an identifying character key variable ("EMPID"), which has a length of 9 bytes. We wish to extract data for these individuals from a DB2 table(s) and store this information in a permanent SAS dataset ("HISTORY"). The macro call statement would appear as the following:

```
%DATAPULL(PATLIST,EMPID,C,9,HISTORY,31000);
```

The last parameter in the above call statement refers to the total length of the key variable list segments (LIST_LEN). Note that this setting will vary from program to program, because the size of the entire WHERE statement is limited to 32K bytes, not just the list.

To help illustrate how these macros work, suppose we have 10,000 patients. Each of these patients has a patient identification number (MEMID), which is a numeric variable with length of 9 digits.

Aside from restricting our data extract to the MEMID in the WHERE statement, assume that we will also include 1,000 drug codes, each of which is an 11-digit, character variable. As has been discussed earlier, character variables in a list should be flanked by apostrophes (2 digits) and separated by a comma (1 digit). Therefore, this list of drug codes will account for 14,000 bytes [= 1,000 * (11 + 2 + 1)].

In order not exceed the 32K WHERE clause limit, the MEMID list can be no be larger than 18K (= 32K - 14K); this value should be entered as the LIST_LEN value. The %RUNCOUNT macro will then calculate the size of each list segment, as well as the number of iterations that the macro is run:

$$\text{PASS\_OBS} = \text{INT}\left(\frac{18000}{9+1}\right) = 1800$$

$$\text{NUM\_PASS} = \text{CEIL}\left(\frac{10000 * (9+1)}{18000}\right) = \text{CEIL}(5.55) = 6$$

The %DATAPULL macro creates the following macro call statements.

```
%LISTSQL (    1, 1800);
%LISTSQL (1801, 3600);
%LISTSQL (3601, 5400);
%LISTSQL (5401, 7200);
%LISTSQL (7201, 9000);
%LISTSQL (9001,10800);
```

After each %LISTSQL call macro, %EXTRACT is executed and the extracted data is appended to a permanent dataset. The %LISTSQL macro will end after the 10,000th and final observation.

Multiple Key Variables

During instances the key dataset contains more than one key variable, use one key variable, preferably that is indexed, to extract data. The resulting SAS dataset (OUT_DSN) can then be merged with the key dataset using the original (used in the DB2 extract) and other key variables in SAS.

Discussion

An SQL list macro has been introduced. Variations of it can accommodate either a character or numeric key variable. This macro is able to automatically pass an entire list of key variables in segments, without encountering impedance due to volume restrictions discussed in earlier presentations (2). In doing so, it decreases the required programming and computer processing time. The macro is especially time efficient if the key variable listed is indexed on the SAS or DB2 table(s) (2). Indeed, this SQL list macro is a potentially powerful tool when working with large SAS and/or DB2 table(s).

Acknowledgements

I am indebted to my colleagues within the Health Services Evaluation department at Blue Cross and Blue Shield of Massachusetts, particularly Edgar Mounib, Robert Swanton, Sara Metrick, and Fran Brindel.

References

1. SAS® is a registered trademark of SAS Institute, Inc., Cary, NC.
2. Loren J. SAS connections to DB2: tools and techniques. *SUGI 1996 Proceedings* pp 498-507.
3. Gavin, C. SAS/ACCESS Techniques for large DB2 databases. *SESUG 1993 Proceedings* pp 99-105.

Contact

Address suggestions and questions to Thiru Satchi at Health Services Evaluation, Blue Cross and Blue Shield of Massachusetts, 100 Summer Street (MS 01/07), Boston, MA 02110. The author can also be reached at (617) 832-3432 or at Thiru_Satchi@bcbsma.com.

Appendix A

SAS dataset example.

```
DATA MASTER;
INPUT EMPID $5. MEMID  9. DOB DATE7.DIAG  $5. ;
 CARDS;
```

1	P6504	51675	29-Nov-57	53071
2	P6504	51675	29-Nov-57	99215
3	A7028	258766	6-Feb-96	99201
4	M5596	55969587	23-Feb-62	45355
5	T7707	45634578	11-Jan-60	45383
6	T7427	92139678	3-May-93	43260
7	M5596	55969587	23-Feb-62	99213
8	M5596	55969587	23-Feb-62	45380
9	A1260	921396	17-Nov-97	88714
10	Y5530	566511	13-Jun-58	55431
11	S5648	464467896	11-Jan-69	99382
12	G2312	87932	12-Feb-57	77210
13	A5987	666886887	10-Feb-71	99211
14	Y2117	7457778	16-Jan-69	99847
15	R3421	73771	8-May-64	53290
16	V7504	846420987	25-Sep-97	99214
17	V7500	567534567	22-Jul-64	64378
18	V2594	87540987	11-Jan-60	53010
19	A7356	346987	14-Jun-62	76576
20	A7236	3469871	16-Nov-95	53101
21	S7654	19970104	14-Dec-26	99211
22	T7564	19970104	16-Mar-42	74747
23	Z0578	986547897	16-Oct-95	53101
24	;			

Appendix B

Sample output for the character key variable.

The SAS System

OBS	EMPID	MEMID	DOB	DIAG
1	P6504	51675	29NOV57	53071
2	P6504	51675	29NOV57	99215
4	M5596	55969587	23FEB62	45355
5	T7707	45634578	11JAN60	45383
7	M5596	55969587	23FEB62	99213
8	M5596	55969587	23FEB62	45380
13	A5987	666886887	10FEB71	99211
14	Y2117	7457778	16JAN69	99847
17	V7500	567534567	22JUL64	64378
21	S7654	19970104	14DEC26	99211
22	T7564	19970104	16MAR42	74747

Appendix C

Sample output for the numeric key variable.

The SAS System

OBS	EMPID	MEMID	DOB	DIAG
3	A7028	258766	06FEB96	99201
6	T7427	92139678	03MAY93	43260
9	A1260	921396	17NOV97	88714
10	Y5530	566511	13JUN58	55431
11	S5648	464467896	11JAN69	99382
12	G2312	87932	12FEB57	77210
15	R3421	73771	08MAY64	53290
16	V7504	846420987	25SEP97	99214
19	A7356	346987	14JUN62	76576
20	A7236	3469871	16NOV95	53101

Appendix D

Macros to resolve the Pass Through facility/DB2 32K byte transfer limitation

```
1.  %GLOBAL        KEY_DSN KEY_VAR VAR_TYP VAR_LEN KEY_LIST OUT_DSN LIST_LEN;
2.  %GLOBAL        BEG_OBS END_OBS PASS_OBS NUM_PASS  KEY_LIST;
3.
4.  ************************************************************************************;
5.  %MACRO DB2PULL(TABLE);
6.        PROC SQL;
7.           CONNECT TO DB2(SSID=D2P6);
8.              CREATE TABLE TEMPDATA AS
9.                SELECT *
10.             FROM CONNECTION TO DB2
11.               (SELECT *
12.             FROM DB2_&TABLE
13.              WHERE ID IN (&KEY_LIST) AND DX1_CD IN ('53000','23999');
14.             DISCONNECT FROM DB2;
15.       QUIT;
16.       %PUT &SQLMSG;
17.       PROC APPEND BASE=&OUT_DSN DATA=TEMPDATA;
18. %MEND DB2PULL;
19.
20. %MACRO EXTRACT;
21.     %DB2PULL(IP_CLAIM);
22.     %DB2PULL(OP_CLAIM1);
23.     %DB2PULL(OP_CLAIM2);
24. %MEND EXTRACT;
25.
26. ************************************************************************************;
27.
28. %MACRO RUNCOUNT;
29.    DATA _NULL_; SET &KEY_DSN NOBS=NUMOBS;
30.      IF _N_ = 1;
31.         %IF %STR(&VAR_TYPE) = C %THEN %DO;
32.              CALL SYMPUT('PASS_OBS',INT(&LIST_LEN/(&VAR_LEN+3)));
33.              CALL SYMPUT('NUM_PASS',CEIL((NUMOBS*(&VAR_LEN+3))/&LIST_LEN));
34.           %END;
35.         %ELSE  %DO;
36.              CALL SYMPUT('PASS_OBS',INT(&LIST_LEN/(&VAR_LEN+1)));
37.              CALL SYMPUT('NUM_PASS',CEIL((NUMOBS*(&VAR_LEN+1))/&LIST_LEN));
38.          %END;
39.    RUN;
40. %MEND RUNCOUNT;
41.
42. ************************************************************************************;
43.
44. %MACRO LISTSQL(BEG_OBS,END_OBS);
45.      PROC SQL NOPRINT ;
46.         %IF %STR(&VAR_TYPE) = C %THEN %DO;
47.              SELECT " ' " || TRIM(&KEY_VAR) || " ' "
48.           %END;
49.        %ELSE %DO;
50.              SELECT &KEY_VAR FORMAT &VAR_LEN..
51.        %END;
52.              INTO : KEY_LIST SEPERATED BY ","
53.           FROM &KEYDSN(FIRSTOBS=&BEG_OBS OBS=&END_OBS);
54.  %MEND LISTSQL ;
55.
56. ************************************************************************************;
57.
```

```
58.  %MACRO CLEARDSN;
59.  %IF  %SYSFUNC(EXIST(&OUT_DSN)) % THEN %DO;
60.              PROC DELETE DATA=&OUT_DSN) ; RUN;
61.        %END;
62.  %MEND CLEARDSN ;
63.
64.  ************************************************************************************************;
65.
66.   %MACRO DATAPULL (KEY_DSN,KEY_VAR,VAR_TYP,VAR_LEN);
67.              %CLEARDSN;
68.              %RUNCOUNT;
69.            %DO I= 1 %TO &NUM_PASS;
70.                    %LET BEG = %EVAL(1+(&I-1)*&PASS_OBS);
71.                    %LET END = %EVAL(&I*&PASS_OBS);
72.                 %LISTSQL(&BEG,&END);
73.                 %EXTRACT;
74.  %END;
75.  %MEND DATAPULL;
76.
77.
78.  %DATAPULL(KEY_DSN=PATLIST,KEY_VAR=EMPID,VAR_TYPE=C,VAR_LEN=9,OUT_DSN=HISTORY,LIST_LEN=31000);
```