REALIZING IMPROVED EFFICIENCY ON LARGE DATA BASES WITH SQL IN SAS® 6.06, WITH FURTHER IMPROVEMENTS UNDER SAS 6.07

Anne Soeder, Glaxo Inc., RTP, NC

ABSTRACT:

Attempting to JOIN a small SAS dataset with a large DB2 table using Proc SQL can require large amounts of memory and CPU time. This JOIN can be made more efficient with a macro which will soon be available in the SAS.SAMPLE.LIBRARY. Further improvements can be realized with improved SQL processing in SAS version 6.07 with the Pass through feature. This paper will demonstrate how the use of one or both of these techniques will reduce up to seven-fold, without increasing memory requirements.

INTRODUCTION:

SAS version 6.06 and Proc SQL opened the door to the access of DB2 tables allowing JOINs with SAS datasets. This paper will present to you a unique problem that arises quite often in our programming efforts. The joining of a small SAS dataset with a large DB2 table which results in a full tablespace scan. It will present a macro from the SAS.SAMPLE.LIBRARY which will be a solution to this problem under SAS version 6.06. I will then discuss the statistics of both running the job with and without the macro, comparing them for measurements in CPU time and memory requirements. I will then add pass thru logic to illustrate further improvements under SAS version 6.07. I will discuss statistics for running this same query under SAS version 6.07 with and without the macro, and then with the macro adding the pass thru logic.

THE PROBLEM:

Often in the interactive environment we are requesting input from our users through a SAS/AF screen. We then build a SAS dataset with their response to retrieve the requested information from a DB2 table. What has evolved from this process is a small SAS dataset joined with very large DB2 tables. Since SAS cannot take advantage of the DB2 indexes, it does a Fulltable space scan (for every record in the smaller table, a comparison is made against every record in the larger table). This can be very costly both in time and memory resources.

THE SOLUTION:

I have found, through testing, that by using a WHERE statement in our joins to link the key fields in the tables, the choice to run a Fulltable scan may be avoided. It seemed logical to create the WHERE clause from the entries in the smaller table. But since some of our WHERE statements could still be hundreds of lines long we chose to implement the statement through a macro. (This macro now resides in the SAS.SAMPLE LIBRARY and was written by Paul Kent of SAS Institute, Inc.).

IMPLEMENTATION OF THE SOLUTION:

Understanding the data is very important in determining whether this solution will actually help. Let's review the requirements for using this macro. This macro will be beneficial, when joining a small SAS dataset with a large DB2 table, by its key fields and the expected retrieval will be less than half the records in the DB2 table. Below I have included information about the datasets that I will be using.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Observations</th>
<th>Columns</th>
<th>Record Size</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODSLCT</td>
<td>4</td>
<td>3</td>
<td>SAS dataset</td>
<td></td>
</tr>
<tr>
<td>PSLAO12</td>
<td>4.6 mil</td>
<td>23</td>
<td>DB2 table</td>
<td></td>
</tr>
<tr>
<td>LISTPRC</td>
<td>4</td>
<td>6</td>
<td>SAS dataset</td>
<td></td>
</tr>
<tr>
<td>GOVSSALE</td>
<td>115,276</td>
<td>19</td>
<td>output table</td>
<td></td>
</tr>
</tbody>
</table>

I have selected a query which is used in an online system to help illustrate this problem. In this query the user has been asked to select the product number(s) of the sales information they wish to retrieve based on a certain time frame. Their selection has been stored in a temporary SAS dataset (PRODSLCT). The sales information will be found in a large DB2 table (PSLA012) which is populated with approximately 4.6 million records. The join is performed by the product number (PRODNO) which is a key field in the sales table.

```
PROC SQL;
CREATE TABLE GOVSSALE AS
  SELECT A.DESC
  , B.PRODNO, B.INVCHQ
  , B.CONTNO, B.COUNTVR, B.CONTLIN, B.SLSDATE
  , B.SLSTYPE, B.PRODPRC, B.SLSQTY, B.SALESAMT
  , B.DATEAOD, B.SHIPTONO, B.SALESNO
  , B.PROONO, B.INVCNO
  , B.CONTRNO, B.CONTRVER, B.CONTRLINE
  , C.DESCCHR, C.PROOPRC AS LPRODPRC, C.EFFOT, C.EXPIRDT
FROM PRODSLCT A
  , SCAN18P.PSLAO12 B
  , LISTPRC C
WHERE (A.PRODNO = B.PRODNO)
  AND (B.SLSTYPE = 'I')
  AND (B.SLSDATE >= #START#)
  AND (B.SLSDATE <= #END#)
  AND (B.PRODNO = C.PRODNO)
  AND (C.EFFOT = B.SLSDATE)
  AND B.SLSDATE <= C.EXPIRDT;
QUIT;
```

Figure 1

In order to use the macro only three simple steps are required.

1. Include the macro into the program.
2. Call the PREJOIN macro to set up the small table.
3. Call the KEYJOIN macro to link with the large table.

Illustrated below is the same query from Figure 1 with the macro code included. The WHERE statement would resemble (WHERE Prodno = '9999999' or prodno = '88888' ...) and
would be inserted where the %KEYJOIN line is. Modifications to the query have been highlighted.

```sql
%INCLUDE 'STAABA.BC.PROGRAM(MACPMK6)';RUN;
%PREJOIN(DS=PRODSLCT,KEY=PRODNO,KEYTYPE=C);
PRO SQL:
CREATE TABLE GOVSSALE AS
SELECT A.DESC,
    B.PRODNO, B.INVCHN,
    B.CONTNO, B.CONTVER, B.CONTINV, B.SLSDATE,
    B.SLSTYPE, B.PRODPRC, B.SLSQTY, B.SALESAMT,
    B.DATEADD, B.SLSLINNO, B.SHIPFROM, B.SALESNO,
    C.DESCSON, C.PRODPRC AS LPRODPRC, C.EFFDT, C.EXPIRDT
FROM PRODSLCT A
    BCNSLHP.PSLA012 B
    LSTIFRC C
WHERE (A.PRODNO = B.PRODNO)
AND %KEYJOIN(ALIAS=A,KEY=PRODNO)
AND B.SLSTYPE = 'I'
AND (B.SLSDATE <='STAR'D'
AND B.SLSDATE <= C.EXPIRDT)
AND (B.PRODNO = C.PRODNO)
AND (C.EFFDT <= B.SLSDATE)
AND B.SLSDATE <= C.EXPIRDT);
QUIT;
```

**Figure 2**

**DISCUSSION OF MEASUREMENT AND RESULTS:**

To write a SAS program that optimizes performance of the SAS system, we are concerned about minimizing three areas: CPU time, I/O count, and memory requirements. These performance areas were measured using SAS system options STIMER, MEMRPT, and FULLSTATS. Each query was tested in three runs, and the results were averaged to arrive at a single measurement.

The following table shows a comparison of the techniques in terms of the memory requirements and CPU time. The results show a decrease in CPU time, holding the memory requirement nearly constant. The query with the macro was 4.5 times faster than the plain query.

<table>
<thead>
<tr>
<th>ACTION</th>
<th>CPU TIME</th>
<th>TASK MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain Query 6.06</td>
<td>00:22:20.85</td>
<td>1247K</td>
</tr>
<tr>
<td>Query With Macro 6.06</td>
<td>00:04:50.10</td>
<td>1370K</td>
</tr>
</tbody>
</table>

**FURTHER IMPROVEMENTS USING PASS THRU WITH SAS 6.07:**

The SQL passthru feature in SAS version 6.07 allows the programmer to pass logic in the Proc SQL to DB2, allowing the evaluation of this code to be done by DB2. Only the results of the passed thru logic are then returned to Proc SQL as an intermediary table. With this in mind, I recognized the possibility of improvements in the Join using Pass thru logic:

1. Allows DB2 to use its indexing.
2. Passed logic reduces the size of the returned table from DB2.
3. Retrieves only those columns that the query needs.

Because the Pass thru feature is only capable at this time to pass logic pertaining to DB2 tables, I felt the combination of the macro with this new feature was my strongest assault on this query for efficiency. The following query is the same as in Figure 1 and 2 only modified to add the Pass thru feature.

```sql
%INCLUDE 'STAABA.BC.PROGRAM(MACPMK6)';RUN;
%PREJOIN(DS=PRODSLCT,KEY=PRODNO,KEYTYPE=C);
PRO SQL:
CREATE TABLE GOVSSALE AS
SELECT A.DESC,
    B.PRODNO, B.INVCHN,
    B.CONTNO, B.CONTVER, B.CONTINV, B.SLSDATE,
    B.SLSTYPE, B.PRODPRC, B.SLSQTY, B.SALESAMT,
    B.DATEADD, B.SLSLINNO, B.SHIPFROM, B.SALESNO,
    C.DESCSON, C.PRODPRC AS LPRODPRC, C.EFFDT, C.EXPIRDT
FROM PRODSLCT A
    CONNECTION TO DB2(SELECT PRODNO, INVCNO, CONTNO, CONTVER,
    CONTINV, SLSDATE, SLSTYPE, PRODPRC, SLSQTY, SALESAMT, SLSDATE, SLSLINNO, SHIPFROM, SLESAMT, SLESAMT, SLESAMT, PRODNO, INVCNO
    FROM PROD.SALES_0012
    WHERE %KEYJOIN(ALIAS=A,KEY=PRODNO)
    AND SLSTYPE = 'I')
    B
    LSTIFRC C
WHERE (A.PRODNO = B.PRODNO)
AND %KEYJOIN(ALIAS=A,KEY=PRODNO)
AND (B.SLSDATE <= 'START'D'
AND B.SLSDATE <= C.EXPIRDT)
AND (B.PRODNO = C.PRODNO)
AND (C.EFFDT <= B.SLSDATE)
AND B.SLSDATE <= C.EXPIRDT);
QUIT;
```

**DISCUSSION OF MEASUREMENT AND RESULTS:**

The same testing techniques were used with SAS 6.07 as the above techniques with SAS 6.06. To test the efficiency of SAS 6.07 the code in Figure 1 and 2 were rerun under SAS 6.07 to establish a base. After the results from these runs were favorable I then ran the query with Pass thru under SAS 6.07.

The following table shows a comparison of the techniques in terms of the memory requirements and CPU time. The results show a decrease in CPU time while holding the memory requirement constant. The query with the macro was 5.4 times faster than the plain query. and version 6.07 was 6.7 times faster than version 6.06's plain query.

<table>
<thead>
<tr>
<th>ACTION</th>
<th>CPU TIME</th>
<th>TASK MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain Query 6.07</td>
<td>00:17:42.75</td>
<td>4078K</td>
</tr>
<tr>
<td>Query with Macro 6.07</td>
<td>00:03:17.56</td>
<td>4078K</td>
</tr>
<tr>
<td>Query with Macro and Pass Thru Feature 6.07</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CONCLUSION:**

Attempting to JOIN a small SAS dataset with a large DB2 table using PROC SQL can require large amounts of memory and CPU time. This paper has shown that using the SAS.SAMPLE.LIBRARY macro MACPMK, the CPU time can be greatly reduced under both SAS version 6.06 and 6.07, without affecting the memory requirements. An additional savings can be realized by adding the SAS version 6.07 SQL passthru feature. The greatest savings is realized with the combination of SAS's new version 6.07, the SAS.SAMPLE.LIBRARY macro and passthru, improving CPU time by 6.7 times over version 6.06.

328
REFERENCES

TRADEMARK INFORMATION
SAS is a registered trademark of the SAS Institute Inc., Cary, NC, USA.