Abstract

If you have used the SAS System for any period of time, you are probably familiar with conventional DATA and PROC step processing to extract, merge, analyze, and present data. But you may not be familiar with how the SQL procedure can be used to perform these very same tasks. In fact, using the SQL procedure can often reduce the amount of code needed to perform a task as well as minimize the level of resources required by the system to process a request.

As a method of comparing the SQL procedure with conventional DATA and PROC step methods, this paper presents application examples with their corresponding benchmark statistics to contrast the performance differences. The resulting differences are reported and performance strategies are recommended.

Basic SQL Concepts

SQL boasts the ability to define, manipulate, and control relational databases or tables as well as providing easy user access. The concept behind SQL is that the user does not have to specify physical attributes about the data such as data structure, location, and/or data type. The user will concentrate on what data should be selected, but not how to select them. It also provides methods of making changes to tables without the need to change application programs. Consequently, data independence is one of the design goals of the relational model.

The SQL Procedure in the Base SAS System

PROC SQL has the following features:

- Can run interactively as well as batch (noninteractive).
- Uses the Structured Query Language to create, modify, and retrieve data from tables.
- Can be augmented through the use of Global statements such as TITLE and FOOTNOTE.
Accesses tables via a two-level name where the first level is the Libref and the second level name is the name of the table.

**Terminology**

The following terminology is provided to help relate SQL and SAS System terms and concepts.

- **Column** — the same as a variable in the SAS System.
- **Libref** — acts as the alias or nickname. Points to a SAS data library.
- **Relational Database Management System** — a database system that forms relationships between data items.
- **Row** — the same as an observation in the SAS System.
- **Structured Query Language (SQL)** — a highly standardized high-level language used in relational database management systems to create and alter objects within a database.
- **Table** — the same as a SAS System data set.
- **View** — contains a definition or description of data stored in another location.

**Approach**

The approach we used in conducting comparisons between the SQL processor and the DATA and PROC steps consisted of the following steps.

- Compare SQL to DATA and PROC Methods.
- Analyze Results from Comparisons.
- Explain Outcome.

We tracked CPU time and the amount of memory utilized by each run as our benchmark statistics. This we felt would provide us with a better understanding about performance and efficiency issues.

**Operating Environment/Data Characteristics**

Performance and Efficiency tests were conducted on an IBM 3090 MVS mainframe with TSO as the communications monitor. All tests were run interactively, where at any one time a maximum of 200 TSO users could be logged on.

We decided to conduct our comparisons a maximum of three times. Each test would be run at different hours of the day to reflect those times that were considered peak time (greatest demand on system resources) and other times when demand was not too extensive. We then averaged the results to arrive at an average CPU time and an average memory utilization.

The characteristics of the data used for conducting our test comparisons consisted of 1,285 observations, 61 variables (character and numeric), and a record length of 128 bytes.
Tasks Studied

The tasks that we chose to study under our test comparisons were in areas that we cited as being most requested by the user population or in areas that we thought would be interesting.

- Data Retrieval.
- Rearranging Data.
- Index Creation.
- Merging without an Index.
- Merging with an Index.

Data Retrieval

We examined three methods of retrieving and displaying data specifying a WHERE statement: (1) Custom Report Writing with the DATA _NULL_, (2) PROC PRINT, and (3) PROC SQL.

Method 1:

```
DATA _NULL_;  
SET INFECT;  
FILE PRINT NOTITLE;  
WHERE SPECIMEN = '2';  
PUT @10 SPECIMEN $1.  
@20 CULT_DT MMDDYY8.  
@30 PF $7.;  
RUN;
```

Method 2:

```
PROC PRINT DATA=INFECT;  
WHERE SPECIMEN = '2';  
VAR SPECIMEN CULT_ID PF;  
RUN;
```

Method 3:

```
PROC SQL FEEDBACK STIMER;  
SELECT SPECIMEN, CULT_DT,  
PF FROM INFECT  
WHERE SPECIMEN = '2';  
QUIT;
```

Results:

<table>
<thead>
<tr>
<th>Method</th>
<th>CPU</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA <em>NULL</em></td>
<td>0.13</td>
<td>4553K</td>
</tr>
<tr>
<td>PROC PRINT</td>
<td>0.17</td>
<td>4551K</td>
</tr>
<tr>
<td>PROC SQL</td>
<td>0.16</td>
<td>4551K</td>
</tr>
</tbody>
</table>

Rearranging Data

We examined two methods of rearranging data: (1) PROC SORT and (2) PROC SQL.

Method 1:

```
PROC SORT DATA=INFECT;  
BY HOSP PF CULT_DT;  
RUN;
```
Method 2:

PROC SQL FEEDBACK STIMER NOPRINT;
SELECT * FROM INFECT ORDER BY HOSP, PF, CULT_DT;
QUIT;

Results:

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC SORT</td>
<td>0.38</td>
<td>4492K</td>
</tr>
<tr>
<td>PROC SQL</td>
<td>0.21</td>
<td>4492K</td>
</tr>
</tbody>
</table>

Index Creation

We examined two methods of creating indexes: (1) PROC DATASETS and (2) PROC SQL.

Method 1:

PROC DATASETS DDNAME= WORK;
MODIFY INFECT;
INDEX CREATE SPECIMEN;
QUIT;
RUN;

Method 2:

PROC SQL FEEDBACK STIMER NOPRINT;
CREATE INDEX SPECIMEN ON INFECT(SPECIMEN);
QUIT;

Results:

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC DATASETS</td>
<td>0.12</td>
<td>4551K</td>
</tr>
<tr>
<td>PROC SQL</td>
<td>0.11</td>
<td>4553K</td>
</tr>
</tbody>
</table>

Merge - No Index

We examined two methods of merging without an index: (1) DATA Step and (2) PROC SQL.

Method 1:

DATA MERGE;
  MERGE INFECT (IN=A)
  SPECIMEN (IN=B);
  BY SPECIMEN;
  IF A AND B;
RUN;

Method 2:

PROC SQL FEEDBACK NOPRINT;
CREATE TABLE MERGE AS
  SELECT *
  FROM INFECT AS A,
  SPECIMEN AS B
  WHERE A.SPECIMEN = B.SPECIMEN;
QUIT;

Results:

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA Step</td>
<td>0.11</td>
<td>4551K</td>
</tr>
<tr>
<td>PROC SQL</td>
<td>0.18</td>
<td>4551K</td>
</tr>
</tbody>
</table>
Merge with Index

We examined two methods of merging without an index: (1) DATA Step and (2) PROC SQL.

Method 1:

```lean
DATA MERGE;
  MERGE INFECT (IN=A)
  SPECIMEN (IN=B);
  BY SPECIMEN;
  IF A AND B;
RUN;
```

Method 2:

```lean
PROC SQL FEEDBACK NOPRINT;
CREATE TABLE MERGE AS
  SELECT *
  FROM INFECT AS A,
   SPECIMEN AS B
WHERE A.SPECIMEN =
  B.SPECIMEN;
QUIT;
```

Results:

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA Step</td>
<td>0.18</td>
<td>4551K</td>
</tr>
<tr>
<td>PROC SQL</td>
<td>0.03</td>
<td>4551K</td>
</tr>
</tbody>
</table>

Conclusions

Although our tests were rather limited to the MVS platform, we found many reasons to be excited about the SQL procedure. Coding requirements were often much less than the DATA and/or PROC step counterparts. The SQL procedure was able to combine many operations, such as sorting and printing data, in one single SQL step. Consequently, CPU and I/O requirements often were much less.

Based on the tests that were performed, it becomes apparent that the SQL procedure provides many advantages over traditional DATA and PROC step methods especially in the areas of merging indexed data sets (tables). In this area alone, we give the SQL procedure thumbs up. It performed significantly better than the DATA step method. Consequently, we feel that the SQL procedure is worth looking into, especially since it comes bundled with the base product.

Acknowledgments

Our sincere thanks are extended to everyone involved with the SUGI Conference especially Neil Howard, Section Chair of Information Systems.

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

Author Contacts

The authors will be happy to answer questions and accept suggestions at the following addresses:

Kirk Paul Lafler  
Software Intelligence Corporation  
P.O. Box 1390  
Spring Valley, CA 91979-1390  
Tel: (619) 670-SOFT or (619) 670-7638

Fran Larsen  
County of Los Angeles  
Information Technology Service  
Customer Applications Branch  
9150 East Imperial Highway  
Downey, CA 90242  
Tel: (213) 940-2153