Accessing Data Faster Than a SAS® Index
Daniel Dower, AMP Incorporated, Harrisburg, PA

ABSTRACT

Subsets of a large SAS® data set can quickly be accessed through the use of a SAS index. However, other programming techniques can be applied at certain times to obtain greater efficiency than SAS indexes when accessing subsets of large data sets. An alternative to SAS indexes is zoned-index pointer data sets. Depending on the type and amount of data being accessed, the zoned-index pointer data sets allow subsets of large data sets to be extracted faster than it would take with a SAS index. Usually substantial savings in computer resources are also realized.

This paper focuses on the concept of a zoned-index pointer data set. Code will be given showing how to create and use this method of accessing large data sets. The advantages and disadvantages of the zoned-index pointer data set method will also be presented.

INTRODUCTION

Release 6.06 of the SAS System introduced the concept of an indexed SAS data set. One of the benefits of an indexed data set is fast access to a subset of the observations when selecting by WHERE-expression processing. Release 6.07 improved the index processing through the addition of the INDEX= option, the KEY= option, and the optimization of the WHERE clause to accept additional clauses and SAS functions.

A zoned-index pointer data set is an alternative to SAS indexed data sets that allows even faster access to subsets of observations depending on the type and amount of data being accessed. This paper presents an overview of zoned-index pointer data sets and assumes a basic knowledge of SAS indexes and the SAS macro language.

CONCEPT

In order to use a zoned-index pointer data set, the master data must be in an uncompressed SAS data set which is sorted by the value of the index variable. A separate “pointer” SAS data set exists which contains one observation for each value of the index variable and is comprised of three variables:

1. The index variable
2. START -- equal to the beginning observation number of the subset
3. STOP -- equal to the last observation number of the subset.

Refer to Figure 1 for a visual description of the pointer data set.

FIGURE 1:

<table>
<thead>
<tr>
<th>Observation Number</th>
<th>Value of Index</th>
<th>Index Value</th>
<th>START</th>
<th>STOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x-2</td>
<td>n</td>
<td></td>
<td>x-2</td>
<td>x</td>
</tr>
<tr>
<td>x-1</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To access a subset of the master data set, it is necessary to find the observation within the pointer data set which corresponds to the value of the index variable. The starting and ending observation numbers of the subset then can be concatenated with the FIRSTOBS= and OBS= options and placed in a macro variable. This macro variable is used as a data set option as part of a SET or MERGE statement against the master data set.
The concept of the zoned-index pointer data set method simply is that it is more efficient if the SAS System knows the actual observation numbers of the subset than it is to use a SAS index.

CREATING THE POINTER DATA SET

The pointer data set is created with a relatively simple DATA step. This DATA step will traverse the master data set to find the starting and ending observation numbers of each possible subset of data. Figure 2 may be used as a template to create a pointer data set.

FIGURE 2:

```sas
DATA LIBNM.POINTER (KEEP=LAST START STOP);
   SET LIBNM.MASTERDT (KEEP=<index-var>);
   END=EOF;
   LENGTH START STOP LAST $8;
   RETAIN START STOP LAST;
   IF N=1 THEN DO;
      START=1; STOP=0; LAST=<index-var>;
   END;
   IF <index-var> NE LAST THEN DO;
      STOP=N-1; OUTPUT; START=N ;
      LAST=<index-var>;
   END;
   IF EOF THEN DO;
      STOP=N ; OUTPUT;
   END;
   RUN;

*** NOTE: For this example, the variable LAST was defined as an alphanumeric variable of length 8. However, the variable LAST must be defined as the same type and length as the index variable.
```

If the index variable contains a significant number of values, place a SAS index on the index variable to obtain even greater efficiency. This can be done by the INDEX= data set option when creating the pointer data set, or by using the PROC DATASETS procedure immediately after the pointer data set is created.

ACCESSING THE SUBSET

Because the zoned-index pointer data set method utilizes a macro variable to store the observation numbers, the code used to access a subset must be executed from within a SAS macro. Accessing the subset is a simple two-step process:

1. Find the observation in the pointer data set which corresponds to the value of the index variable for the desired subset. Build a macro variable (named &OP_STR) which is comprised of a string setting the FIRSTOBS= option to the beginning observation number of the subset and the OBS= option to the ending observation number of the subset. This can be accomplished with the sample code in Figure 3.

   FIGURE 3:

   ```sas
   DATA _NULL_;
   IF N=1 THEN DO;
      SET LIBNM.POINTER
         (WHERE=(<index-var>=<index-val>));
      CALL SYMPUT('OP_STR', '('II
         'FIRSTOBS='IILEFT(PUT(START,8.))II
         'OBS='IILEFT(PUT(STOP,8.))II')
      ;
   END;
   STOP; RUN;
   ```

   The FIRSTOBS=x option specifies that processing is to begin on the xth observation, and the OBS=n option states that processing will end with the nth observation. A WHERE statement cannot be placed within this DATA step when the FIRSTOBS= or OBS= options are used. However, that does not matter since it is much more efficient to use the actual observation numbers than to combine the WHERE statement with a SAS index.

2. Access the large data set through any valid SAS statement (SET, MERGE, etc.). Place the &OP_STR macro variable that contains the FIRSTOBS= and OBS= values directly after the data set name, as shown in Figure 4.

   FIGURE 4:

   ```sas
   DATA SUBSET;
   SET LIBNM.MASTERDT &OP_STR;
   RUN;
   ```

   The FIRSTOBS=x option specifies that processing is to begin on the xth observation, and the OBS=n option states that processing will end with the nth observation. A WHERE statement cannot be placed within this DATA step when the FIRSTOBS= or OBS= options are used. However, that does not matter since it is much more efficient to use the actual observation numbers than to combine the WHERE statement with a SAS index.

AMOUNT OF SAVINGS

The amount of savings realized by the zoned-index pointer data set method vary greatly depending upon the size of the master data set and the size of the subset of observations. No significant savings will be found if the subset is to contain only a handful of
In those instances the SAS index will outperform the zoned-index pointer data set method. Also consider using the SAS index when the master data set contains a relatively small number of observations because there will be little savings. The zoned-index pointer data sets do produce a notable savings in computer resources when the master data set is extremely large and when the subset has a significant number of observations.

The amount of savings may also differ by computing platform. The zoned-index pointer data set method has been tested using SAS Release 6.08 under the VM/CMS environment and under Release 6.08 of the SAS System For Windows™. It is safe to assume that other operating platforms that run the SAS System will also realize some savings in computer resources when using the zoned-index pointer data set method.

**AN EXAMPLE OF SAVINGS**

To test the zoned-index pointer data set method, a SAS data set of approximately 300,000 observations was created. This data set represents the sales statistics for “ABC Company” with each observation of the data set corresponding to one month’s sales for one account. Each account is assigned to a salesperson who has a unique identifying sales code. For comparison purposes between the SAS index and the zoned-index pointer data set method, a SAS index on the sales code was created.

Sales code “MAA” represents 3,812 observations located near the center of the data set. To extract the observations for this sales code using each method, the sets of code in Figure 5 and Figure 6 are used.

**FIGURE 5:**  
Utilizing the SAS Index:

```sas
data subset;
  set accts.data;
  where code='MAA';
run;
```

**FIGURE 6:**  
Utilizing the Zoned-Index Pointer Data Sets:

```sas
data _null_;  
if _n_=1 then do;
  set accts.ptr (where=(code="MAA"));
  call symput('op_str','firstobs='||
    left(put(start,8.))||'obs='||
    left(put(stop,8.))||')';
  end;
stop; run;

data subset;
  set accts.data &op_str;
rung
```

Running this set of code under the VM/CMS operating system, the average results shown in Figure 7 were obtained.

**FIGURE 7:**

<table>
<thead>
<tr>
<th></th>
<th>SAS Index</th>
<th>Zoned-Index Pointer Data Sets</th>
<th>Percent Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elapsed Time</td>
<td>2.50 sec</td>
<td>2.30 sec</td>
<td>8.00%</td>
</tr>
<tr>
<td>CPU Time</td>
<td>0.54 sec</td>
<td>0.40 sec</td>
<td>25.93%</td>
</tr>
</tbody>
</table>

This same data set was downloaded to the PC and the same code was executed using Release 6.08 of the SAS System For Windows. The average results are displayed in Figure 8.

**FIGURE 8:**

<table>
<thead>
<tr>
<th></th>
<th>SAS Index</th>
<th>Zoned-Index Pointer Data Sets</th>
<th>Percent Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elapsed Time</td>
<td>8.1 sec</td>
<td>6.0 sec</td>
<td>25.93%</td>
</tr>
</tbody>
</table>

In general, the amount of savings will increase as the size of the master data set and the size of the subset of observations increases.

**ADVANTAGES OF THIS METHOD**

The zoned-index pointer data set method of accessing a subset of a large data set has a number of advantages that include the following:
• A significant savings in the amount of computer resources needed to access the subset of observations when large amounts of data are involved.

• Faster response time for on-line applications that need to extract data from a large data set.

• It will take significantly less disk space (greater than 95% in most cases) to store a pointer data set than it does to store the SAS index for a data set.

DISADVANTAGES OF THIS METHOD

While there are advantages to the zoned-index pointer data set method, there are also a number of drawbacks that must be taken into consideration before using this method. These disadvantages include:

• Compression of the master SAS data set cannot be used since this method needs to utilize the observation numbers. This may require an enormous amount of disk space that would not be needed with a SAS indexed compressed data set.

• The zoned-index pointer data set method allows only one index to be utilized. The pointer data sets can only be created based on the value of one index, and the master data set must be sorted by that index variable. By contrast, SAS indexes allow more than one variable to be indexed which allows a data set to be accessed by an index even if the data set is not sorted by that index variable.

• The pointer data sets must be recreated each time an update is made to the master data set. For that reason this method should not be used if the master data set is to be updated frequently.

• The programmer must have a basic knowledge of the SAS macro language to use this method. If you are unfamiliar with the SAS macro language, take time to learn this powerful facility.

• It takes a few more lines of code to create and access a pointer data set. But the extra typing is worth the trouble when the savings are realized!!!

CONCLUSION

The zoned-index pointer data set method offers the potential to save extraordinary amounts of processing when accessing subsets of a large data set. Carefully weigh the advantages and disadvantages listed above to see if zoned-index pointer data sets will be beneficial to your particular data processing needs.

ACKNOWLEDGMENTS

The author would like to thank the employees of the Workgroup Information Systems and Business Analysis Information Systems organizations within AMP Incorporated for their help in introducing me to this technique and refining it to meet my needs.

REFERENCES


The author will be glad to answer questions and accept suggestions at the following address:

Daniel Dower
AMP Incorporated
PO Box 3608, Mailstop 24-20
Harrisburg, PA 17105-3608
E-Mail: DJDOWER@AMP.COM

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

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