Indexes are an extension of the data sets they are associated with even though they are stored in separate files in the SAS® data library. The main purpose of using indexing is to assist in the exact location and selection of observations. When a data set is indexed any observation in it can be accessed directly whereas in a nonindexed data set, the SAS system has to start from the first observation. The WHERE and BY statements are used with indexed data to subset or select observations and can be used both in a DATA step or with a PROC statement.

There are two types of indexes depending on the number of key variable used to index the data. When one key variable is used, the index is a simple one and if more than one key variable is used, the index is called composite. This paper used a simple index.

**CREATING INDEXES**

An index is created using the INDEX CREATE statement. But before any data set can be indexed, it has to be sorted by the variables on which it is to be indexed. For example, we want to index the data set ABC on the variable X. The code to do this is:

```sas
libname old 'sr1 [scratch]' ;
proc sort data = old.ABC out = SORT ;
   by X ;
run ;
proc datasets library = old ;
   modify SORT ;
   index create X ;
run ;
```

This program creates a simple index. If more than 1 variable is to be indexed on, the index is called a composite index and the INDEX CREATE statement has the following form:

```
INDEX CREATE index = (X1, X2, X3, ..., Xn)
```

Indexes use a lot of storage space and so to make the most efficient use of indexing, the variables to be indexed on should be those used frequently either in BY statements or for subsetsenting the data.

An index can be deleted using an INDEX DELETE statement after a MODIFY statement. Once a data set has been indexed, a BY statement can be used without first sorting the data set by the variables in the BY statement. The only condition is that the variables in the BY statement must be in the same order as the indexing variables. When composite indexing has been done, it is not necessary to use all the indexing variables in the BY statement. For example, if a data set has been indexed on the variables X1, X2, and X3; it is possible to use just X1 and X2 in the BY statement, but not X2 and X1 or X2, X1 and X3 in that order.

The use of WHERE statement enables an efficient use of indexes. A WHERE statement allows the user to specify a condition which the observation must satisfy in order to be included in the data set. This is useful as SAS does not have to read all the observations from the input data set. This statement can be used in the DATA step or the PROC step, but it cannot be used as part of an IF-THEN statement. Some operators that can be used with the WHERE statement are arithmetic, comparison, logical and < or > etc. There are some special operators which can be used only with the WHERE statement like the BETWEEN-AND operator which has the form:

```
WHERE variable BETWEEN value AND value ;
```

The WHERE statement can be used in the input data step with procedures like the APPEND procedure, the COMPARE procedure and the DATASETS procedure with an APPEND statement. In the COMPARE procedure, the WHERE statement is used to limit observations from the data sets to be compared. The SAS system uses the index to select observations in a WHERE statement only if the one or more variable in the WHERE
expression has a simple index, if the WHERE statement is selecting only one-thirds or less of the total number of observations in the data set and if the data set has more than one index that meets the conditions of the WHERE statement. If the SAS system uses an index, then it retrieves the observations in the indexed order otherwise the observations are retrieved in the physical order in which they occur in the data set.

An alternative to using the OR operator to get a list of variables is the IN operator. An IN operator can be used to determine whether a value is among a specified range of numbers or a list of character values. It compares the value produced by an expression on the left of the operator to a list of values given on the right. The general form is

\[ \text{expression IN (value-1, \ldots, value-n)} \]

The expression can be any valid SAS expression but is usually a variable name, but the value must be a SAS constant.

METHODS

The American College of Radiology (ACR) frequently needs to extract data from the Part B Medicare Annual Data (BMAD) to do extensive analyses. BMAD data is collected annually by HCFA and gives the number of procedures performed for Medicare as well as the payments made. We used this data to compare the different methods of subsetting data sets using both indexed and non-indexed datasets for the comparison. Only the data on radiologic services which is about a million records was used. The data is by individual procedure codes and since at ACR there is frequently a need to subset the data based on procedure code (proc), this variable was used as the indexing variable. We used SAS 6.06 on the VMS™ system. The data set used 152,000 blocks of storage space and the index on just this one variable used an additional 35,700 blocks of storage space. The nonindexed data set is called 89DATA while the indexed data is called INDEX89.

To compare the different methods, 9 procedure codes were subsetted from this large data set. We compared the CPU time taken to subset the data sets using the following six methods:

WHERE with IN statement on indexed data
WHERE with IN statement on non sorted non indexed data
WHERE without IN statement on indexed data
WHERE without IN statement on nonsorted and non indexed data
Subsetting with the use of PUT statement for Table lookup on indexed data
Subsetting with the use of PUT statement for Table lookup on non sorted and non indexed data

WHERE with IN Statement

The code used to create data subsets by this method is:

```
libname abc 'sr1:[scratch]' ;
data xyz ;
set abc.89DATA ;
where proc in ('77400','77405','77410','77420','77425','77430','77261','77262','77263') ;
run ;
```

WHERE without IN statement

The code used in this method is

```
libname abc 'sr1:[scratch]' ;
data XYZ ;
set abc.89DATA ;
where proc = '77400' or proc = '77405' or proc = '77410' or proc = '77420' or proc = '77425' or proc = '77430' or proc = '77261' or proc = '77262' or proc = '77263' ;
run ;
```

Use of PUT statement for Table lookup

The code in this method was:

```
proc format ;
   value $keepprc
      '77400','77405','77410','77420','77425','77430','77261','77262','77263' = 'keep' ;
run ;
libname abc sr1:[scratch] ;
data XYZ ;
set abc.subset89 ;
```
if put (proc, $keepprc.) = 'keep'; 
run;

RESULTS

The results are summarized in Table 1.

<table>
<thead>
<tr>
<th>Method</th>
<th>Indexed Data</th>
<th>Nonsorted, Non Indexed Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHERE with IN</td>
<td>0:12.49</td>
<td>7:02.14</td>
</tr>
<tr>
<td>WHERE w/o IN</td>
<td>0:12.47</td>
<td>7:22.03</td>
</tr>
<tr>
<td>Subsetting IF</td>
<td>5:47.64</td>
<td>4:27.68</td>
</tr>
</tbody>
</table>

The table shows the CPU time taken to do the selection of the 9 codes from the indexed and non indexed data. We found that the most efficient way to subset, if time is the critical factor, is to index the data and use the WHERE option without the IN statement. This method used a total CPU time of 0:12.47. But this gain in time is offset by the fact that the indexed data uses a large storage space. If the data we are working with is non indexed, then the most efficient way to subset the data is to use the subsetting IF statement. This method is the least efficient way when the data set is indexed. We also tried to use indexes to merge data sets but it was not very efficient and so we have not included the results in this paper.

DISCUSSION

This paper looks at various methods of subsetting large data sets. The need to do this arises if time is of importance or if space is a problem. For example, if from the data set used in this paper only few of the procedure codes are to be analyzed, it is a waste of resources to read all the observation. The same program which would run in a few minutes on a subsetted data set takes a couple of hours to run on the large data set. If a couple of iterations are run, the time savings from using small data sets are tremendous. Indexing the data set on the most frequently used subsetting variable is the most efficient way to extract a few observations. On the other hand, indexing takes up a lot of storage space and this consideration has to be taken into account when deciding which method to use. One conclusion of this paper is that if indexing is not done then the most efficient way to subset data is the PUT table lookup method. It is both faster and also easy to read and maintain.

Indexes are an efficient way of subsetting data only if the data set we are working with is stable and does not change. If the data is dynamic then indexes cannot be used. Also, recreating indexes is very costly and so if index is to be used, say, just once a year then it is not worth the expense - both of time and computer resources. There are some other problems with indexes also and we think SAS institute is working on them and hope to solve them in version 6.07.

REFERENCES


SAS is a registered trademark of SAS Institute Inc in the USA and other countries. ® indicates USA registration.
VMS is a trademark of Digital Equipment Corp., Maynard, MA, USA.