Paper S1-18-2012

Understanding SAS Index

Pramod.R,
Target Corporation, Minneapolis, Minnesota

ABSTRACT

Ever imagined how it would be if the dictionary was not alphabetically arranged? Ever imagined how it would be to search a book in a library if it was not ordered by the topics? Ever tried searching a person's house in a locality without knowing the street number? All these are the examples of real life indexes which we have used/ come across.

SAS allows us to index the datasets which will help us retrieve and manipulate the data easily. Typical usage of Indexes would be while joining two tables based on one or more key variables, querying a dataset based on a filter and doing some of the by-group processing.

In this paper, I have tried to introduce the concept of Indexing in SAS and have also tried to explain the basic working model of an index (b-tree) with a simple example. This paper also covers the basic syntax, and various methods to create and manipulate indexes in SAS. In the end, I've tried to show the advantages of using the index through simple examples and compared the performance of the codes before using the index versus those after using the indexes.

INTRODUCTION

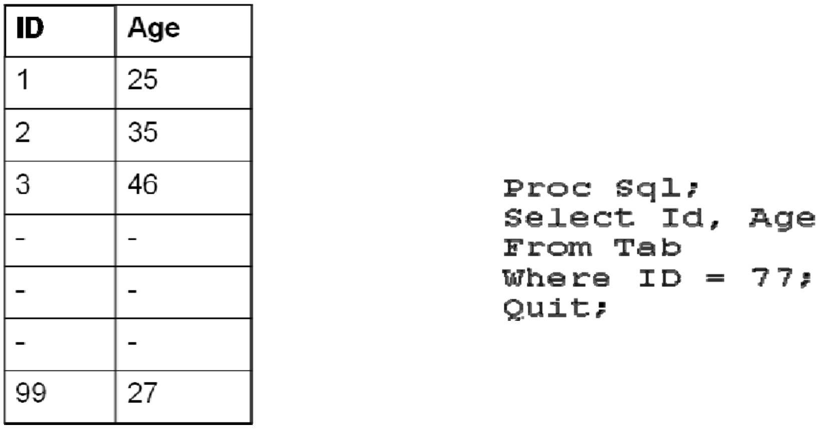
An index is a performance-tuning tool within SAS software. It allows observations with specific values to be accessed quickly from a data file. Instead of reading through a variable file to find a particular value of a variable or variables, an Index identifies the exact location of these observations. An Index is an inverted tree structure that stores values of key variables in ascending order.

Using a SAS Index can affect the query performance drastically. It could enhance the performance to a great extent or it could even degrade the performance, depending on where and how we use them. Hence it is imperative that one should not only evaluate the necessity to Index, but also choose right variable(s) to Index. For this, we would first need to understand how an index works.

WORKING OF A SIMPLE B-TREE INDEX

There are various types of Indexing algorithms available. The simplest one would be a Binary Tree Index (B-Tree Index). Below, I've illustrated the working of a simple B-Tree Index with the help of an example.

Assume that there is a table containing 100 Student IDs and their age from which you would need to fetch the age of the Student whose ID is 77. The table is in a sorted order by the ID variable as shown below. To fetch the ID and Age of the Student having the ID=77, we would write a simple SQL query as shown below.

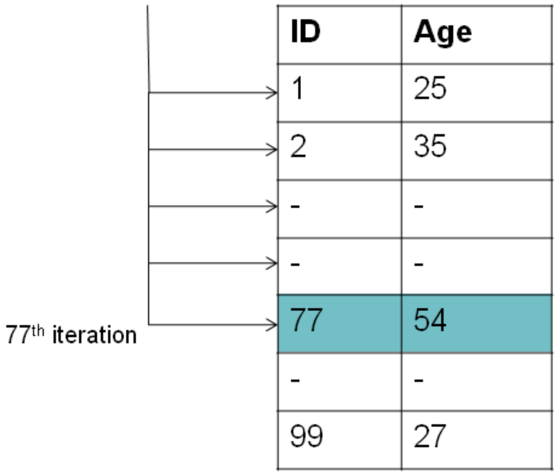


ID	Age
1	25
2	35
3	46
-	-
-	-
-	-
99	27

```
Proc Sql;
Select Id, Age
From Tab
Where ID = 77;
Quit;
```

Figure 1. Table and the query

When we run the query on the above table, when it is not indexed, the processing would occur sequentially. That means, the SAS would perform 77 Iterations before it hits the right number.

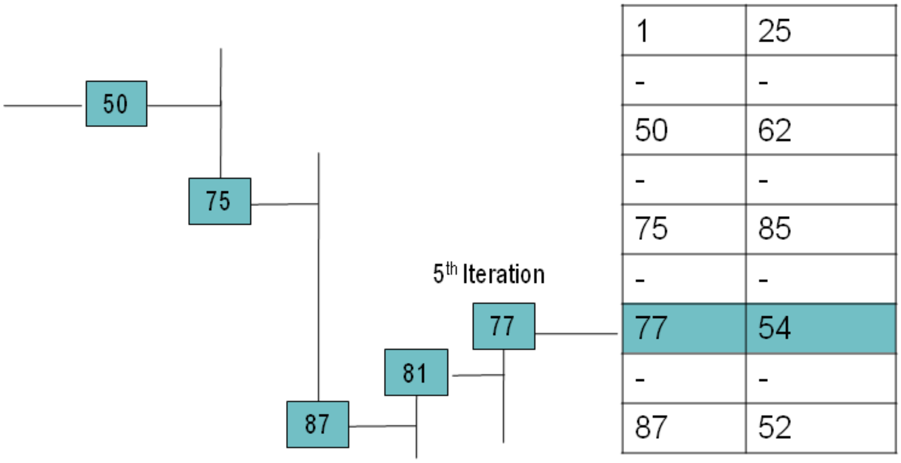


ID	Age
1	25
2	35
-	-
-	-
77	54
-	-
99	27

Figure 2. Sequential processing

It could be 77 in this scenario, versus a couple of Billion in some other dataset. It is almost impractical to have the compiler make so many iterations every time you query.

If you Index your dataset (B-Tree in this dataset), the performance is enhanced immensely because the compiler would not perform so many iterations. When it is indexed and queried, the compiler would compare if the key variable (77) is less than or greater than 50 (half of the total observations). In our case it is greater; hence it discards the values 1 to 50. Now it further divides and compares if the value 77 is greater than or less than or greater than 75 (because the midpoint from 50 to hundred is 75). Since its greater, it discards the values from 75 to hundred as well, and further compares the key variable with the midpoint from 50 to 75.. and so on and so forth. In this example, the result is retrieved in the 5th Iteration.



1	25
-	-
50	62
-	-
75	85
-	-
77	54
-	-
87	52

Figure 3. B-Tree

The basic goal of having a SAS index is to be able to efficiently extract a small subset of observations from a large SAS data set. In doing so, the amount of computer resources (CPU time, I/O's, Elapsed time, etc.) expended should be less than that of having the SAS read the entire data set sequentially.

For the above example, if the original dataset just had 5 observations (IDs 1, 2, 3, 4, 5) and if we were to search for the ID 2, then the time taken to calculate the midpoint in each of the iteration and compare them with the keyed value every time would definitely be more than that of sequentially searching for the observation with the ID 2.

SAS INDEXES

SAS employs more complex mechanisms in creating Indexes. It actually creates a separate index file (having the extension .sas7bndx) which would have the information of the indexed variable. SAS Indexes can be created in various mechanisms as described below:

- Index option in a data step

  ```
  DATA SAS-data-file-name (INDEX=
          (index-specification-1</UNIQUE><...index-specification-n</UNIQUE>>));
  ```

  Where

  - *SAS-data-file-name* is a valid SAS data set name
  - *index-specification* for a simple index is the name of the key variable
  - *index-specification* for a composite index is *(index-name=(variable-1...variable-n))*
  - the **UNIQUE** option specifies that values for the key variable must be unique for each observation.

- Proc sql

  ```
  PROC SQL;
     CREATE <UNIQUE> INDEX index-name
        ON table-name(column-name-1<...,column-name-n>);
     DROP INDEX index-name FROM table-name ;
  QUIT;
  ```

  where

  - *index-name* is the same as *column-name-1* if the index is based on the values of one column only
  - *index-name* is not the same as any *column-name* if the index is based on multiple columns
  - *table-name* is the name of the data set to which *index-name* is associated.

- Proc datasets

  ```
  PROC DATASETS LIBRARY=libref <NOLIST>;
     MODIFY SAS-data-set-name;
     INDEX DELETE index-name;
     INDEX CREATE index-specification;
  QUIT;
  ```

  where

  - points to the SAS library that contains *SAS-data-set-name*
  - *libref* the **NOLIST** option suppresses the printing of the directory of SAS files in the SAS log and as ODS output
  - *index-name* is the name of an existing index to be deleted
  - *index-specification* for a simple index is the name of the key variable
  - *index-specification* for a composite index is *index-name=(variable-1...variable-n)*.

EVALUATING THE CREATION OF THE INDEX AND THE VARIABLE TO INDEX

It is important to consider various factors before we decide to Index. I'm dividing these into two sub parts: Dataset Consideration and the Variable consideration.

Dataset Consideration:

The size of the data that is being retrieved versus the actual size of the dataset is one important consideration. If the size of the data that is being retrieved is small as compared to the total number of observations, then the performance would definitely be boosted. Smaller the number of retrieved rows from the original dataset, better the performance. However, if the number of retrieved data observations is almost equal to the actual dataset, the performance takes a terrible hit, due to the overload of CPU and I/O processing. The below table (taken from the Complete Guide to SAS Indexes book) gives us a good picture on the dataset consideration:

Subset Size	Indexing Action
1% - 15%	An index will definitely improve program performance.
16% - 20%	An index will probably improve program performance.
21% - 33%	An index might improve or it might worsen program performance.
34% - 100%	An index will not improve program performance.

Alongside with this, it is also essential to have the size of the original dataset in mind. But the concept of size is always relative and hence cannot always be determined and defined.

Variable considerations:

It is always recommended to have the variable with the high cardinality Indexed, which means that these variables should have a huge number of distinct rows in the dataset. For example, it is a good idea to Index on the Employee ID (whose cardinality is equal to the number of employees in the company) rather than indexing on the gender of the employee (which would just be 2).

Also the index would perform better if the indexed variable is already sorted before it is indexed.

Below, I shall include a couple of examples to demonstrate each of the above mentioned scenarios and concepts and show the remarkable difference in performances while using an Index versus an non indexed data.

Example 1: Simple Index

Below is a program that creates a sample dataset having 300M observations:

```
DATA TESTDATA;
Y=1;
DO I=1 TO 300000000;
       X=I+1;
       Y=Y+X;
       Z=Y+4;
       IF MOD(I,4)=1 THEN TXT='A';
       IF MOD(I,4)=2 THEN TXT='B';
       IF MOD(I,4)=3 THEN TXT='C';
       IF MOD(I,4)=0 THEN TXT='D';       OUTPUT;
END;
RUN;
```

Upon running the a simple proc sql on the un-indexed dataset to pull 3 observations from the 300M observations, it takes me about 9 mins (under some I/O, memory conditions)

```
16         PROC SQL;
17         CREATE TABLE NOINDEX AS
18         SELECT *
19         FROM WORK.TESTDATA
20         WHERE X IN (792302,36273,2361838);
NOTE: Table WORK.NOINDEX created, with 3 rows and 5 columns.

21         quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           9:01.41
      cpu time            15.96 seconds
```

However, upon indexing the dataset using the below code, the processing time has reduced about less than half a second (under the same I/O conditions)

```
PROC DATASETS LIBRARY=WORK ;
       MODIFY TESTDATA;
       INDEX CREATE X;
QUIT;
```

```
16         PROC SQL;
17         CREATE TABLE INDEX AS
18         SELECT *
19         FROM WORK.TESTDATA
20         WHERE X IN (792302,36273,2361838);
NOTE: Table WORK.INDEX created, with 3 rows and 5 columns.

21         QUIT;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.49 seconds
      cpu time            0.01 seconds
```

Example 2: Indexing on a variable that does not have high cardinality

Here I run a query to fetch a particular txt value. Below is the result that we get on a non indexed dataset while trying to fetch results for a single value of txt.

```
16         PROC SQL;
17         CREATE TABLE NOINDEX2 AS
18         SELECT *
19         FROM WORK.TESTDATA
20         WHERE TXT = 'A';
NOTE: Table WORK.NOINDEX2 created, with 75000000 rows and 5 columns.

21         QUIT;
NOTE: PROCEDURE SQL used (Total process time):
      real time           2:50.39
      cpu time            30.07 seconds
```

Upon indexing the above dataset on txt (which has a very low cardinality of 4), and running the same query upon that, we can see that the time taken has not changed much. Also the subset of the dataset that is being pulled is about 25%. Hence the chance of the Index helping the query performance is very low.

```
15          PROC SQL;
16          CREATE TABLE INDEX2 AS
17          SELECT *
18          FROM WORK.TESTDATA
19          WHERE TXT = 'A';
NOTE: Table WORK.INDEX2 created, with 75000000 rows and 5 columns.

20          QUIT;
NOTE: PROCEDURE SQL used (Total process time):
      real time           2:21.53
      cpu time            29.53 seconds
```

CONCLUSION

Indexing could prove out to be a great tool in SAS especially when there is a need of quick processing of data, or when we need to have the data given out in real time (Stored processes for example). However, one should also evaluate the data very well and then index the data because a badly constructed index can result in degradation of the performance. It is also advisable to keep trying the right combination of the indexing variables by understanding the frequency of usage of the variable and the structure of the data, in an incremental fashion until we get the optimal solution.

REFERENCES

RAITHEL, MICHAEL A. 2006. *THE COMPLETE GUIDE TO SAS® INDEXES.* CARY, NC: SAS INSTITUTE INC

ACKNOWLEDGMENTS

The Author wishes to thank Jeelani Basha, Shalini MR and Jeyvinth Rayan for willing to proofread my papers and providing valuable comments and suggestions. I would like to thank Jared Moore for all his help, right from getting information regarding the conference till helping me print this paper. Last and most importantly, I would like to thank my wife Bhargavi, who has been a constant motivator and support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Pramod R
Enterprise: Target Corporation Pvt. Ltd
Address: Apt No. 2405, 1117, Marquette Avenue
City, State ZIP: Minneapolis, Minnesota, 55403
Work Phone: 612-272-1958
E-mail: getpramod.r@gmail.com
Web: www.pramod-r.blogspot.com