

Use of the Where clause in SAS* 6.07

Robert Zdanky, Manager, Grain Business Unit, ATSF Railway Company

ABSTRACT

The talk will explain to the beginning user how using a WHERE clause with indexing will save CPU time. By using a where clause the SAS user does not need to use a data step to select subsets. The paper will also talk about WHERE clause operators such as: Between And, Contains, Is Null Is Missing, Like Same and.

OVERVIEW

We will see in this tutorial how much CPU time can be saved by using the WHERE statement and INDEXING. All examples in this tutorial were run against a data set which contained 906,932 observations, 18 variables with a length of 112. All programs were run in batch mode on an MVS system.

Indexing with the WHERE statement saves enormous amounts of CPU time.

A. INDEXING

Indexing establishes a set of pointers in the data set. For a discussion how indexes work see the reference. The code to create a simple index is easy.:

```
PROC DATASETS LIBRARY=BASE;  
MODIFY GRNDATA;  
INDEX CREATE STCC5;
```

Stcc5 is a five digit commodity code; every commodity hauled on the railroad carries a STCC code. An index variable in conjunction with a WHERE

statement can cut CPU time by 80%. But creating an index is not a cost free operation, it took 45 CPU seconds to create the index. To create the data base from a flat file took 62 CPU seconds. Also the indexed file needed 8 per cent more space for storage. Guidelines on when to use the index will be provided later in the paper..

Two useful options are the NOMISS and UNIQUE. The NOMISS option eliminates missing observations from the index, though the observations remain in the data set. UNIQUE forces the index to contain unique observations.

One final note, SAS software does not always use the index variable. It may decide it is more efficient not to use the index. The system option MSGLEVEL=I will indicate if the index variable is being used.

B. WHERE Statement

The WHERE statement like the IF selects observations from SAS data sets that meet certain conditions. Unlike the IF, the WHERE statement can be used in PROCs to select observations. This means one does not need a separate data step to subset the observations. Also the WHERE only brings in the remaining variables after the observation is selected.

The theory would lead one to think the WHERE in a PROC would be more efficient than a combination of a data step and a PROC. The evidence indicates in a **nonindexed environment** using a WHERE in a PROC gained one only 4 or 5%. The real big efficiency gains come from using an index variable vs. a nonindex variable.

In order to show how the comparisons were made, the coding from the first examples will be shown. In this example, STCC5 is an indexed variable and 2% of the observations were selected.

```
I.  A  DATA TESTING;
      SET BASE.GRNDATA;  WHERE
      STCC5 IN('01137',
      '44111');  CPU TIME 1.08

      B  PROC MEANS DATA=TESTING
          NWAY NOPRINT;
          CLASS ORGCITY DSTCITY;

          VAR CARS;
          OUTPUT OUT=SUMIT
          SUM=CARS;  CPU TIME 0.84

          TOTAL CPU TIME FOR BOTH
          STEPS  1.92

      C  PROC MEANS
          DATA=BASE.GRNDATA NWAY
          NOPRINT;

          WHERE STCC5 IN('01137',
          '44111');
          CLASS ORGCITY DSTCITY;
          VAR CARS;
          OUTPUT OUT=SUMIT
          SUM=CARS;  CPU TIME FOR
          THE PROC WITH THE WHERE
          1.69 SECONDS;
          0.23 SECONDS SAVED OR A
          GAIN OF 12%.
```

In example II, the same exact coding was used, except the variable STCC5 was not indexed. the results were:

```
II  A DATA STEP      8.31 seconds
      B PROC MEANS    0.84 seconds
          TOTAL      9.31 seconds
II  C PROC MEANS
      WITH A WHERE    8.80 seconds
```

0.51 seconds saved or a gain of 5%

Notice the gain in efficiency between I A and II A. It took 1.08 seconds with an indexed variable vs. 8.31 seconds a gain of 87%.

It would take 6 passes through the data to break even on a CPU basis. The CPU savings were about 7.23 seconds per pass and it took 45 cpu seconds to create the indexed data set. $45/7.23=6$ on a rounded basis.

More examples of using indexed vs. non indexed variables are presented below. The first line shows the SAS code which selected the observations.

```
WHERE STCC5='01137' AND
ORGSTATE='KS';
% OF OBSERVATIONS SELECTED 0.5
CPU TIME INDEXED 0.44
NONINDEXED 6.56 % GAIN 93
```

```
WHERE STCC5 IN('01137','01136')
AND ORGSTATE='KS';
% OF OBSERVATIONS SELECTED 0.7
CPU TIME INDEXED 0.63
NONINDEXED 8.59 % GAIN 93
```

```
WHERE STCC5 IN('46111','45111');
% OF OBSERVATIONS SELECTED 42
CPU TIME INDEXED 16.35
NONINDEXED 12.76 % GAIN -28
```

Even when nonindexed variables are added to the selection criterion, index variables performed extremely

well. When 42% of the observations were selected, the indexed variables performed worse than the nonindexed variables. SAS Institute advice of not using an index if your selections exceed 1/3 is sound advice.

Before we discuss special WHERE clauses, there is a problem if the IN statement is used with the WHERE statement.

The SAS code used to test the IN and WHERE was of the form:

```

      W H E R E      S T C C 5
IN('01137','30719',....,'20821');

```

Instead of using the WHERE the IF used in the data step:

```

      I F      STCC5='01137'      O R
STCC5='30719'      O R      ....
STCC5='20821';

```

Length of Selection List refers to the number of elements in the WHERE or IF statements. Indexed variable indicates if an Indexed variable was used. Type of selector shows whether an IF or a WHERE statement was used. CPU time is in seconds. The results were:

Length of Select List	Ind. Var.	% of Obs.	Type of Select.	CPU TIME
23	Yes	8.1%	Where with in	4.13
23	Yes	8.1%	IF	9.12
10	Yes	3.6%	Where with in	1.78
10	Yes	3.6	IF	8.02
23	No	8.1%	Where with in	25.76
23	No	8.1%	IF	9.14
10	No	3.6%	Where with in	13.97
10	No	3.6	IF	8.09

When using an indexed variable, the WHERE was the best way to go. But when using nonindexed variables, the IN statement was inefficient when the number of elements exceeded 8 or 9.

C. SPECIAL WHERE OPERATORS

The IN statement can be used with an IF. There are a number of operators which can only be used with the WHERE. They are:

"BETWEEN AND" simply selects a range for numeric variables.

```

      W H E R E A T M I L E S B E T W E E N 501 A N D
1000;

```

"CONTAINS" looks for a string in a character variable and is case sensitive.

WHERE ORGCITY CONTAINS 'KAN'; One must be careful when using the 'CONTAINS' operator. In the above example, we were looking for all rail traffic that originated in Kansas City. Not only was Kansas City picked up but other cities such as: Spokane, Kankakee, Kanopolis, etc. were selected. Nine other cities besides Kansas City were selected.

"IS NULL" "IS MISSING" selects missing numeric or character variables.

```

      W H E R E C A R S I S M I S S I N G

```

"LIKE" is similar to the "CONTAINS" but looks for the character string in a particular location. The underscore '_' represents one position, while the percent sign '%' represents

any number of positions.

WHERE ORGCITY LIKE 'KAN%';
The above statement will still select Kansas City, Kankakee, and Kanopolis , but not Spokane.

"==" will select observations that spelled like a particular character string.

WHERE SHIPPER =='CARGILL';
This operator is great for fields that do not have hard edits. In this example, we are looking for misspellings of Cargill, one of our shippers. The computer would not reject "Cargil" because it may be a new shipper. The "==" would pick it up. Be warned, humans can misspell names beyond all recognition. The "==" operator will not pick up all variations.

D CONCLUSIONS

The WHERE statement in conjunction with an index variable saves enormous amount of CPU time. In this example only 6 passes through the data would justify setting up an index variable.

Although using the WHERE in a nonindexed environment saves little CPU time, coding using the WHERE makes it much easier to follow the programs logic.

Don't use a WHERE with an IN statement if the selection list exceeds 8 elements and the selection variable is not indexed.

REFERENCES

1. Painter, Sally "Efficient Use

of SAS Data Set Indexes in SAS Applications" Proceedings of the Sixteenth Annual SAS Users Group International Conference p 408-411

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

Robert Zdanky
Santa Fe Railroad
4515 Kansas Avenue
Kansas City, KS. 66106
913-551-4184