# Simple Rules to Remember When Working with Indexes

Kirk Paul Lafler, Software Intelligence Corporation, Spring Valley, CA

## Abstract

SAS users are always interested in learning techniques related to improving data access. One way of improving information retrieval from a SAS data set or table is to create and use an index. An index consists of one or more columns that are used to uniquely identify each row within a table. Functioning as a SAS object containing the values in one or more columns in a table, an index may be defined as numeric, character, or a combination of both. This presentation will emphasize the rules that users should know when creating and using indexes to make information retrieval more efficient.

## Introduction

Given the large number of books and articles on SQL and SQL-related topics, how strange it is not to find more material related to indexes and their impact on WHERE clause processing. Certainly, these topics deserve additional attention in order to assist SQL users' achieve a greater understanding in applying these powerful features in their database applications.

Indexes are designed to improve the speed in which subsets of data are accessed. Rather than physically sorting a table (as performed by the ORDER BY clause or the BY statement in PROC SORT), an index is designed to set up a logical data arrangement without the need to physically sort it. This has the advantage of reducing CPU and memory requirements. It also reduces data access time when using WHERE clause processing. This paper presents elements essential to achieving a better understanding of indexes and their effect on WHERE clause processing.

## Tables Used in Examples

The data used in all the examples in this paper consists of a selection of movies that I've viewed over the years, along with its actors. The Movies table consists of six columns: title, length, category, year, studio, and rating. Title, category, studio, and rating are defined as character columns with length and year being defined as numeric columns. The data stored in the Movies table is illustrated below.

**MOVIES Table**

| | Title | Length | Category | Year | Studio | Rating |
|---|---|---|---|---|---|---|
| 1 | Brave Heart | 177 | Action Adventure | 1995 | Paramount Pictures | R |
| 2 | Casablanca | 103 | Drama | 1942 | MGM / UA | PG |
| 3 | Christmas Vacation | 97 | Comedy | 1989 | Warner Brothers | PG-13 |
| 4 | Coming to America | 116 | Comedy | 1988 | Paramount Pictures | R |
| 5 | Dracula | 130 | Horror | 1993 | Columbia TriStar | R |
| 6 | Dressed to Kill | 105 | Drama Mysteries | 1980 | Filmways Pictures | R |
| 7 | Forrest Gump | 142 | Drama | 1994 | Paramount Pictures | PG-13 |
| 8 | Ghost | 127 | Drama Romance | 1990 | Paramount Pictures | PG-13 |
| 9 | Jaws | 125 | Action Adventure | 1975 | Universal Studios | PG |
| 10 | Jurassic Park | 127 | Action | 1993 | Universal Pictures | PG-13 |
| 11 | Lethal Weapon | 110 | Action Cops & Robber | 1987 | Warner Brothers | R |
| 12 | Michael | 106 | Drama | 1997 | Warner Brothers | PG-13 |
| 13 | National Lampoon's Vacation | 98 | Comedy | 1983 | Warner Brothers | PG-13 |
| 14 | Poltergeist | 115 | Horror | 1982 | MGM / UA | PG |
| 15 | Rocky | 120 | Action Adventure | 1976 | MGM / UA | PG |
| 16 | Scarface | 170 | Action Cops & Robber | 1983 | Universal Studios | R |
| 17 | Silence of the Lambs | 118 | Drama Suspense | 1991 | Orion | R |
| 18 | Star Wars | 124 | Action Sci-Fi | 1977 | Lucas Film Ltd | PG |
| 19 | The Hunt for Red October | 135 | Action Adventure | 1989 | Paramount Pictures | PG |
| 20 | The Terminator | 108 | Action Sci-Fi | 1984 | Live Entertainment | R |
| 21 | The Wizard of Oz | 101 | Adventure | 1939 | MGM / UA | G |
| 22 | Titanic | 194 | Drama Romance | 1997 | Paramount Pictures | PG-13 |

The data stored in the ACTORS table consists of three columns: title, actor_leading, and actor_supporting, all of which are defined as character columns. The data stored in the Actors table is illustrated below.

**ACTORS Table**

| | Title | Actor_Leading | Actor_Supporting |
|---|---|---|---|
| 1 | Brave Heart | Mel Gibson | Sophie Marceau |
| 2 | Christmas Vacation | Chevy Chase | Beverly D'Angelo |
| 3 | Coming to America | Eddie Murphy | Arsenio Hall |
| 4 | Forrest Gump | Tom Hanks | Sally Field |
| 5 | Ghost | Patrick Swayze | Demi Moore |
| 6 | Lethal Weapon | Mel Gibson | Danny Glover |
| 7 | Michael | John Travolta | Andie MacDowell |
| 8 | National Lampoon's Vacation | Chevy Chase | Beverly D'Angelo |
| 9 | Rocky | Sylvester Stallone | Talia Shire |
| 10 | Silence of the Lambs | Anthony Hopkins | Jodie Foster |
| 11 | The Hunt for Red October | Sean Connery | Alec Baldwin |
| 12 | The Terminator | Arnold Schwarzenegge | Michael Biehn |
| 13 | Titanic | Leonardo DiCaprio | Kate Winslet |

## Understanding Indexes

What exactly is an index? An index consists of one or more columns in a table to uniquely identify each row of data within the table. Operating as a SAS object containing the values in one or more columns in a table, an index is comprised of one or more columns and may be defined as numeric, character, or a combination of both. Although there is no rule that says a table must have an index, when present, they are most frequently used to make information retrieval using a WHERE clause more efficient.

To help determine when an index is necessary, it is important to look at existing data as well as the way the base table(s) will be used. It is also critical to know what queries will be used and how they will access columns of data. There are times when the column(s) making up an index are obvious and other times when they are not. When determining whether an index provides any processing value, some very important rules should be kept in mind. An index should permit the greatest flexibility so every column in a table can be accessed and displayed. Indexes should also be assigned to discriminating column(s) only since query results will benefit greatest when this is the case.

## Simple Rules to Know About Indexes

When an index is specified on one or more tables, a join process may actually be boosted. The PROC SQL processor may use an index, when certain conditions permit its use. Here are a few things to keep in mind before creating an index:

- If the table is small, sequential processing may be just as fast, or faster, than processing with an index
- Do not create more indexes than you absolutely need
- If the page count as displayed in the CONTENTS procedure is less than 3 pages, an index may no be warranted
- If the data subset for the index is not small, sequential access may be more efficient than using the index
- If the percentage of matches is 15% or less then an index may be appropriate
- The costs associated with an index can outweigh its performance value – an index is updated each time the rows in a table are added, deleted, or modified.

Sample code will be illustrated next to demonstrate the creation of simple and composite indexes using the CREATE INDEX statement in the SQL procedure.

### *Creating a Simple Index*

A simple index is specifically defined for one column in a table and must be the same name as the column. Suppose you had to create an index consisting of movie rating (RATING) in the MOVIES table. Once created, the index becomes a separate object located in the SAS library.

**SQL Code**

```
PROC SQL;
  CREATE INDEX RATING ON MOVIES(RATING);
QUIT;
```

**SAS Log Results**

```
PROC SQL;
  CREATE INDEX RATING ON MOVIES(RATING);
NOTE: Simple index RATING has been defined.
QUIT;
```

### *Creating a Composite Index*

A composite index is defined for two or more columns in a table and must have a **unique** name that is different than the columns assigned to the index. Suppose you were to create an index consisting of movie category (CATEGORY) and movie rating (RATING) in the MOVIES table. Once the composite index is created, the index consisting of the two table columns become a separate object located in the SAS library.

**SQL Code**

```
PROC SQL;
  CREATE INDEX CAT_RATING ON MOVIES(CATEGORY, RATING);
QUIT;
```

**SAS Log Results**

```
PROC SQL;
  CREATE INDEX CAT_RATING ON MOVIES(CATEGORY, RATING);
NOTE: Composite index CAT_RATING has been defined.
QUIT;
```

## Index Entries and Pointers

An index file is stored in the same SAS library as its associated data file. Having the same name as its data file, it is represented as a separate entity known as an INDEX member type. An index file contains entries organized hierarchically with entries being connected by pointers and organized in ascending order. Each entry contains a unique value corresponding to the column's frequency distribution and one or more unique observations, referred to as the record identifier (RID). Space that is occupied by deleted values are automatically reclaimed and reused by the index. A sample index containing entries representing the index file for the movie rating (RATING) is illustrated below.

| Value | RID |
|---|---|
| G | 21 |
| PG | 2, 9, 14, 15, 18, 19 |
| PG-13 | 3, 7, 8, 10, 12, 13, 22 |
| R | 1, 4, 5, 6, 11, 16, 17, 20 |

## Index Limitations

Indexes can be very useful, but they do have limitations. As data in a table is inserted, modified, or deleted, an index must also be updated to address any and all changes. This automatic feature requires additional CPU resources to process any changes to a table. Also, as a separate structure in its own right, an index can consume considerable storage space. As a consequence, care should be exercised not to create too many indexes but assign indexes to only those discriminating variables in a table.

Because of the aforementioned drawbacks, indexes should only be created on tables where query search time needs to be optimized. Any unnecessary indexes may force the SAS System to expend resources needlessly updating and reorganizing after insert, delete, and update operations are performed.

## Optimizing WHERE Clause Processing with Indexes

A WHERE clause defines the logical conditions that control which rows a SELECT statement will select, a DELETE statement will delete, or an UPDATE statement will update. This powerful, but optional, clause permits SAS users to test and evaluate conditions as true or false. From a programming perspective, the evaluation of a condition determines which of the alternate paths a program will follow. Conditional logic in PROC SQL is frequently implemented in a WHERE clause to reference constants and relationships among columns and values.

To get the best possible performance from programs containing SQL procedure code, an index and WHERE clause can be used together. Using a WHERE clause restricts processing in a table to a subset of selected rows. When an index exists, the SQL processor determines whether to take advantage of it during WHERE clause processing. Although the SQL processor determines whether using an index will ultimately benefit performance, when it does the result can be an improvement in processing speeds.

## Conclusion

Indexes can be used to allow rapid access to table rows. Rather than physically sorting a table, an index is designed to set up a logical arrangement for the data without the need to physically sort it. Not only does this have the advantage of reducing CPU and memory requirements, it also reduces data access time when using WHERE clause processing. As was presented, by adhering to a few important rules about creating indexes, SAS users can use an index and a WHERE clause together to improve a query's performance and processing speeds.

## References


Lafler, Kirk Paul (2007), *"Simple Rules to Remember When Working with Indexes,"* Proceedings of the 1st Annual SAS Global Forum (SGF) 2007 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2006), *"A Hands-on Tour Inside the World of PROC SQL,"* Proceedings of the 31st Annual SAS Users Group International Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2005), *"Manipulating Data with PROC SQL,"* Proceedings of the 30th Annual SAS Users Group International Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2004). *PROC SQL: Beyond the Basics Using SAS*, SAS Institute Inc., Cary, NC, USA.

Lafler, Kirk Paul (2003), "*Undocumented and Hard-to-find PROC SQL Features*," *Proceedings of the Eleventh Annual Western Users of SAS Software Conference*.

Lafler, Kirk Paul (1992-2005). *PROC SQL for Beginners*; Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (1998-2005). *Intermediate PROC SQL*; Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2001-2005). *Advanced PROC SQL*; Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2002). *PROC SQL Programming Tips*; Software Intelligence Corporation, Spring Valley, CA, USA.

*SAS® Guide to the SQL Procedure: Usage and Reference, Version 6, First Edition (1990)*. SAS Institute, Cary, NC, USA.

*SAS® SQL Procedure User's Guide, Version 8 (2000)*. SAS Institute Inc., Cary, NC, USA.

## Acknowledgments

The author would like to thank John Xu and Delayne.H.Stokke, MWSUG Co-Chairs, for asking me to present at and to prepare this paper for MWSUG 2007. Thank you for a great conference!

## Trademark Citations



## About the Author

Kirk Paul Lafler is consultant and founder of Software Intelligence Corporation and has been programming in SAS since 1979. As a SAS Certified Professional and SAS Institute Alliance Member (1996 – 2002), Kirk provides IT consulting services and training to SAS users around the world. As the author of four books including *PROC SQL: Beyond the Basics Using SAS* (SAS Institute. 2004), Kirk has written more than two hundred peer-reviewed papers and articles that have appeared in professional journals and SAS User Group proceedings. He has also been an Invited speaker at more than two hundred SAS International, regional, local, and special-interest user group conferences and meetings throughout North America. His popular SAS Tips column, "Kirk's Korner of Quick and Simple Tips", appears regularly in several SAS User Group newsletters and Web sites, and his fun-filled SASword Puzzles is featured in SAScommunity.org. Comments and suggestions can be sent to:

Kirk Paul Lafler
Software Intelligence Corporation
World Headquarters
P.O. Box 1390
Spring Valley, California 91979-1390
E-mail: KirkLafler@cs.com