Paper 040-2008

Understanding WHERE-Clause Processing and Indexes

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

Abstract

SAS® users can quickly access selected observations (rows) of data in SAS data sets using WHERE clause processing and indexes. This presentation explores valuable techniques for achieving improved query performance using indexes. Attendees learn what an index is, the purpose of an index, how an index is created, factors that determine when an index is warranted, how WHERE clause processing can take advantage of an index, and techniques for tuning indexes for better performance.

Introduction

Given the large number of books and articles on SQL, how strange it is not to find more material associated with indexes and WHERE clause processing. Certainly, these topics deserve special attention to assist SQL users' better apply these powerful features in their programs, because good or bad usage can make all the difference in query outcomes.

Indexes can be used to improve the access speed to desired table rows. 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.

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
- If the page count as displayed in the CONTENTS procedure is less than 3 pages, avoid creating an index
- Do not to create more indexes than you absolutely need
- 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 approximately 15% or less then an index should be used
- 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 below on creating 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 rating (RATING) and movie category (CATEGORY) 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 RATING_CAT ON MOVIES(RATING, CATEGORY);
      QUIT;
```

SAS Log Results

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

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, an index can consume considerable storage space. Consequently, care should be exercised not to create too many indexes but assign indexes only when they are needed and only for 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. And even worse, the PROC SQL optimizer may accidentally use an index when it should not.

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 (2010), *"Exploring Powerful Features in PROC SQL,"* SAS Global Forum (SGF) Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2009), *"Exploring DICTIONARY Tables and SASHELP Views,"* South Central SAS Users Group (SCSUG) Conference (November 8th – November 10th, 2009), Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2009), *"Exploring DICTIONARY Tables and SASHELP Views,"* Western Users of SAS Software (WUSS) Conference (September 1st – September 4th, 2009), Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2009), *"Exploring DICTIONARY Tables and SASHELP Views,"* PharmaSUG SAS Users Group Conference (May 31st – June 3rd, 2009), Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2006), *"Understanding WHERE-Clause Processing and Indexes,"* Proceedings of the 2006 South Central SAS Users Group (SCSUG) 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 2007 Western Users of SAS Software (WUSS) Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (1992-2008). *PROC SQL for Beginners*; Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (1998-2008). *Intermediate PROC SQL*; Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2001-2008). *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

I would like to thank Steve Popernack, Programming Beyond the Basics (Advanced Tutorials) Section Chair, for accepting my abstract and paper. I'd also like to thank Dr. LeRoy Bessler, Alix Riley, and Craig Wildeman for a terrific conference.

Trademark Citations

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.

About the Author

Kirk Paul Lafler is consultant and founder of Software Intelligence Corporation and has been using SAS since 1979. Kirk provides IT consulting services and training to SAS users around the world. As a SAS Certified Professional, Kirk has written four books including PROC SQL: Beyond the Basics Using SAS, and more than four hundred peer-reviewed papers and articles. He has also been an Invited speaker and trainer at more than three hundred SAS International, regional, local, and special-interest user group conferences and meetings throughout North America. Kirk's current interests include writing technical books and ebooks, conducting SAS training around the world, serving on the sasCommunity.org Advisory Board; contributing SAS- and SQL-related topics; writing and supporting "Kirk's Korner of Quick and Simple Tips" for numerous SAS User Group newsletters and websites; and sharing his fun-filled SASword Puzzles 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