Techniques for Performance Enhancement of SQL Queries in Relational Databases

Praveena M.V.[#1], Dr.Ajeet A. Chikkamannur[#2]
#1 Assistant Professor, Department of Computer Science &Engineering, DrAmbedkar Institute of Technology,Bangalore, VTU, Karnataka, India
#2 Professor, Department of Computer Science & Engineering,R L Jalappa Institute of Technology, Doddaballapur, Bangalore,VTU, Karnataka, India

Abstract

Information is the most important and critical component of any enterprise. It is a processed form of data and communicated to a user to make decisions.The demand for reliable database systems is on the rise with the consolidation and standardization of information systems. In today's highly competitive world, every organizations require skilled database professionals to manage their information system. The employees need to know the basics of data and databases for its effective utilization. Therefore, the demand and requirement for skilled database professionals is increasing. The database manipulation in relational databases is generally handled through structured query language (SQL) queries. Composing and fine tuning of SQL queries for large and complex databases is an incredibly difficult task. We propose several techniques to enhance the performance of SQL queries written and used in several relational databases to get optimal results. SQL queries plays a vital role in finding solutions in organizational databases in effective and efficient ways.

Keywords - *Database, Performance, Query, RDBMS, Relation, SQL*

I.INTRODUCTION

In our everyday life in the modern era, everybody come across databases and database systems quite frequently. Several activities on a day-to-day basis require interaction with one or the other databases. Databases play an important role in almost every walks of life where computers are used, including business, electronic commerce, medicine, engineering, education, agriculture, manufacturing, to name a few. A database basically representsthe aspects of the real world. It is a logical collection of data with some inherent meaning. It is planned, designed and built with data for a specific purpose. A database can be generated and maintained manually or by a group of application programs written specifically for that particular task or by a database systems.

A database management system (DBMS) is a collection of programs that enables users to build and manage a database. It is a software system that allows users to access and manipulate data contained in the database. The major goal of DBMS is to provide an environment that is both convenient and efficient to use in storing and retrieving information from the database. DBMS is a general purpose software system used to define, create, manipulate, protect and maintain database. It is also used to share databases among various users. In a database environment, the primary resource is the database itself and the secondary resource is the DBMS and its related software's. DBMS is used to control redundancy, restrict unauthorized access, provides backup and recovery, and enforcing standards.

A relational database management system (RDBMS)is a special type of DBMS that is used for manipulating relational databases. It provides us with extra special features and allows us to get most out of the database. The RDBMS is asoftware tool to build, edit, update, delete and store database. It is the modern practical approach to utilize and manipulate database. Oracle, SQL Server, MySQL, PostgreSQL, DB2, Sybase, Informix are some of the popular examples for RDBMS.RDBMS facilitates us to manipulate the database using SQL. A comprehensive relational database language SQL has statements for data definition, query and update. SQL has facilities for defining views, security and authorization, and transaction controls. It also consists of rules to embed SQL statements into general purpose programming languages. It uses the terms table, rows and columns for the formal relational data model terms relation, tuple and attribute respectively.

II. RELATED WORK

There exist several techniques to improve SQL query performances. Some of the systems need high performance and may compromise on reliability. Hence it is very much essential to improve and fine tune SQL query performance in recent information systems. In the

performance improvement by NoSQL approach [1], the authors proved that the use of NoSQL will increase high performance in RDBMS's. Any RDBMS has very high reliability and performance because it supports SQL standard. In improvement by measuring statement coverage mechanism [2], the authors abstracted the problems of unifying software tests that manage SQL queries. The testing database using SQL statement coverage measurement has been improvised. Here the structure of the database had been considered instead of whole information. Also they explored the alternative paths to measure test SQL query coverage to improve performance.

In fine tuning the RDBMS, the type and size of the workload is a key consideration [3]. The authors claims that the autonomic and self-fine tuning RDBMS's had the capacity to maintain their performance by auto-recognition of workload.

III. PROPOSED TECHNIQUES

With new RDBMS and agile hardware tools, SQL queries runs faster with less response time. Even though there is a lot of score for enhancing query performance.

Case Study – Consider a database with the following tables:

Customer – Table Schema

CustId	Fname	Lname	City	State	Phone	AccDate

The Index names for "Customer" table are CustIdPK (primary key), IxState, IxPhone, IxName and IxCity. Where IxNamerefers to Fname and Lname, IxCity refers to City and State respectively.

Orders – Table Schema

OrderId	CustId	OrderDate	OrderCost	Shipped

The Index names for "Orders" table are OrderIdPK (primary key), IxCustDate. Where IxCustDate refers to CustId and OrderDate.

The proposed techniques to improve the SQL query performance involves ten steps.

A. Databases optimizer with statistics

The database statistics is basically index related details and index distribution information stored in catalogs. Database optimizer is the brain of any RDBMS. The optimizer analyzes SQL queries and determines the efficient and effective action plan. It acts like an expert system. It also performs complex actions on several set of information. Different SQL commands are used for different databases to update statistics.

Consider the following query – "Select * from Customer Where City = 'Bangalore' and Phone = '417 – 218 – 9058'". The Where clause in the query contains two indexes, with one field each. But, database optimizer can use only one index per table. It is better to use IdPhone index because it returns least number of rows, runs faster and also a least expensive path. If the database statistics are not updated properly, the optimizer may choose either of the index. Hence update statistics properly for better usage.

B. Optimized index creation:

The SQL optimizer mainly depends on indexes in tables. Indexes will never decrease the performance of SQL queries, hence it is important to have a right balance and mixture of indexes in any tables. Estimating the unique column values for a field is very important while creating indexes. Consider an example, in our case study, IdCity is not a good index. If someone wants to search for customer in a particular city, it can generate several rows, then searched sequentially will result in slow response time.

The composite index in RDBMS, contains more than one field, for example "IdCustDate" in our case study. Another one called Clustered index, similar to a phone directory, which arranges data by last name. Different RDBMS's use different terminologies to support indexes.

C. Avoid correlated sub queries and coding loops:

The query uses values from the other query such as parent query is known as correlated sub query. These queries runs in row by row fashion, hence decreases the query execution performance. Using join operations, we can refactor these queries and use efficiently.

Consider a scenario where thousands of queries to be executed in sequence with loops for inserting values into the database table. Avoid such loops in the code and change such queries using INSERT or UPDATE commands with multiple columns and rows. This optimization increases the query performance drastically.

D. Avoid methods on RHS

Consider the following example –"Select * from Customer Where YEAR (AccDate) == 2018 and MONTH (AccDate) == 09". Rewriting the above query by avoiding the methods on the right hand side of the query as shown below will drastically increase the performance– "Select * from Customer Where AccDate BETWEEN "09/01/2018' and "09/30/2018".

E. Careful SELECT option

Select is a commonly used command in SQL queries. Consider a table with several columns and thousands of rows. If the application really requires few columns, then there is no meaning in querying for the entire data using select command. It definitely results in

a waste of resources. Therefore, instead of using "Select * from Customer", use "Select LName, City, Phone from Customer".

F. Expected growth prediction

As we know, the indexes have a negative effect on SQL queries. Hence, in order to reduce its effect, it is better to specify an appropriate value for the fill factor during index creation. Different RDBMS's have different terms for expected growth like PCTFREE is used in Oracle and DB2, FILL FACTOR is used in MS SQL Server, Informix and Sybase RDBMS.

G. Execution plan using EXPLAIN

The execution plan using EXPLAIN command is very useful in enhancing the query performance. For example, "EXPLAIN PLAN for <MyQuery>" is used in Oracle, "SET EXPLAIN " is used in Informix, "Set SHOWPLAN_ALL on <MyQuery>" is used in MS SQL Server and Sybase. To execute EXPLAIN in databases, a third party tools can be used like WinSQL Professional.

H. Avoid temporary tables

The usage of temporary tables in SQL queries increases the complexity. If the query can be written in simple ways without temporary tables, then the use of these tables can be avoided. In some unavoidable circumstances, the wise use of these tables can be suggested. After its usage, the best practice is to delete temporary tables compulsorily rather than automatic deletion after termination.

I. Alternate for foreign keys:

The foreign keys ensures high data integrity, but decreases query performance. Instead of using foreign keys in SQL queries, use metadata information stored in the system tables to enforce integrity rules. With this, the customer can apply rules at application layer, which results in increase in query execution performance.

J. Split data into multiple drives

When the size of the data increases in the database, accessing it from one hard disk consumes more time and cumbersome also. In order to gain faster access, split database into several physical hard drives. Even the tables can also be split into several disks. This results in gaining significant speed in retrieving data.

IV. CONCLUSION

Even a minor change into a query may have positive or negative effect on the reliability and performance.By looking at the present scenario, every database developers need to know how to fine tune SQL queries to enhance their performance. SQL in any RDBMS has high performance impact because of its standard management. The proposed techniques are helpful to enhance SQL queries in any RDBMS's. In order to obtain optimal results, these techniques are useful. However, the developers and database administrators carefully use and adapt these techniques to improve their query formation.

REFERENCES

[1] Yong-Lak Choi, Woo-SeongJeon, Seok-Hwan Yoon, " Improving Database System Performance by Applying NoSQL", J Inf Process Syst, Vol.10, No.3, page 355-364, September 2014.

[2] Maria Jose Suarez-Cabal, Javier Tuya, "Improvement of Test Data by Measuring SQL Statement Coverage", IEEE proceedings of the Eleventh Annual International Workshop on Software Technology and Engineering Practice (STEP'04), 2004.

[3] Dr M. Senthil, K. Sivaraman, "Performance Improvements In DBMS", International Journal of Advance d Computer Technology, Vol. 2, No. 4, page 12-15, 2016.

[4] Law Y., Wangl H., Zaniolo C., "Query Languages and Data Models for Database Sequences and Data Streams" Proceedings of the 30th VLDB Conference, Canada, vol.30, page 492-503, 2004..

[5] Bahmani A. H., Naghibzadeh M., Bahmani B., "Automatic database normalization and primary key generation" IEEE CCECE/CCGEI, May 5-7, Niagara Falls, Canada, page 11-16, 2008..

[6] Ajeet A Chikkamannur, Handigund S. M.," A concoct semiotic for recursion in SQL", International Journal of Emerging Trends and Technology in Computer Science, Vol 2, Issue 3, page 204-210, ISSN 2278-6856, 2013.

[7] Dr. Shivanand M. Handigund, Bhat S., " An Ameliorated Methodology for the Abstraction of Object Class Structures for an Information Systems", CETS 2010, International Conference on E-business Technology & Strategy, Ottawa, Canada, page 362-367, 2010.

[8] Elmasri, Navathe, "Fundamentals of Database Systems", 5th Edition, Pearson Education, 2008.

[9] Ramkrishnan R., Geherke J., "Database Management Systems", McGraw Hill International Editions, Third Edition. 2004.

[10] O'Neil P., O'Neil E., "Database: principles, programming and performance", 2nd Edition, Morgan Kaufmann, 2001.

[11] Date C. J., Darwen H., "Guide to The SQL Standard", third edition, Addison Wesley Publishing Company.

[12] Sumeet Bajaj, RaduSion, "CorrectDB: SQL Engine with practical query authentication", 39thInternational Conference on International Conference on Very Large Data Bases, Trento, Italy, page 529-540, 2013.

[13] Ogden W. C., "Implications of a cognitive model of database query: comparison of a natural language, formal language and direct manipulation interface", ACM SIGCHI Bulletin 51, Volume 18, Number 2, page 51-54,1986.

[14] Parlikar A., Shrivastava N., Khullar V., Sanyal S., "NQML: Natural Query Markup Language", IEEE Proceeding of NLP - KE '05, page 184-188, 2005.