International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 6 1271 – 1274

Analyatical Study and Evaluation of Database Optimization Techniques

Jayana Ahuja
Master of Computer Application Department
Sarvajanik Collage of Engineering & Technology
Surat, India
ahujajayana@gmail.com

Gayatri Kapadia
Master of Computer Application Department
Sarvajanik Collage of Engineering & Technology
Surat, India
gayatriskapadia@gmail.com

***Abstract*—** Data Management can be considered as one of most crucial part in world of technology. The primary challenge is not just maintaining data & information, but to access them in an efficient manner as well. Database optimization techniques have been designed to address these tasks that may otherwise limit the performance of a database to an extent of vulnerability. In this paper we discuss the aspects of performance optimization related to data access in transactional databases. Furthermore, we analyze the effect of these optimization techniques.

One of key part in database management is database management tools they provide query optimizer that helps to find out the efficient method for SQL query statement to access requested data by user. This optimizer chooses the plan with the lowest cost among all considered candidate plans. Optimizer uses available statistics to calculate cost, for a specific query in a given environment, the cost computation accounts for factors of query execution such as I/O, CPU, and communication.

Keywords- *Database, Optimization, Database performance, Database Management Tools*

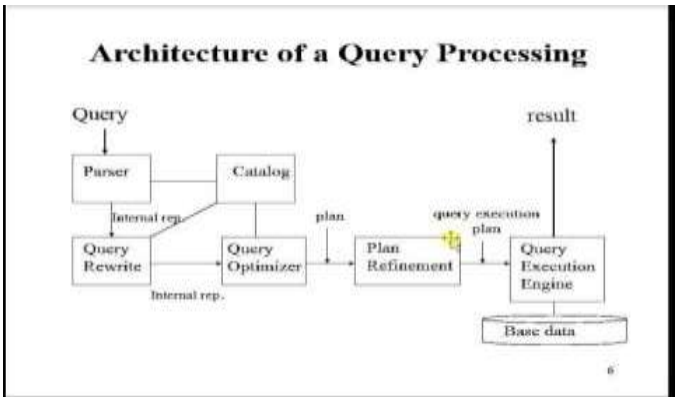
*****

I. INTRODUCTION

Due to fast changing nature of information technology, it becomes very important to have powerful computers and powerful methods to collect, store, transfer and combine huge amounts of data but at very low cost. It will be almost impossible to handle or access this large amount of data if all our database operations are not optimized. When a database associated application performs slowly, there is a 90% probability that the data access routines of that application are not optimized, or not written in the best possible way. Optimization helps to find out how to view query execution plans on your database. [1]

II. ARCHITECTURE OF QUERY PROCESSING

The Database Engine processes **queries** on a variety of data storage architectures such as local tables, partitioned tables, and tables distributed across multiple servers. The following topics cover how SQL Server processes **queries** and optimizes **query** reuse through execution plan caching. [2]



III. OPTIMIZATION OBJECTIVES

Objective of optimization process should be either to maximize the output for a given number of resources or to minimize the resources' usage for a given output. Query optimization tries to minimize the response time for a given query language and mix of query types in a given system environment[1] [11]. The total cost to be minimized is sum of following costs:

Communication Cost:
Cost of transmitting data from the site where they are stored to the sites where computations are performed and results are presented.

Secondary Storage Access Cost:
The cost of loading data pages from secondary storage into main memory.

Storage Cost:
The cost of occupying secondary storage and memory buffers.

Computation Cost:
The cost for using the central processing unit (CPU).

Typical optimizer estimate the size of the result. The result size estimation plays an important role in cost estimation because the output of the operation is an input of the next operation.

IV. PRINCIPLES OF QUERY OPTIMIZATION

1. Optimizing queries with statistics:

Query optimization with statistics uses the collected statistics on the tables and indexes in a query to select an execution plan that can process the query in the most efficient manner. As a general rule, Oracle recommends that you collect statistics on your base table if you are interested in improving your query performance. Optimizing with statistics enables a more accurate estimation of the selectivity and costs of the CONTAINS predicate and thus a better execution plan.[9]

2. Optimizing queries for response time:

A CONTAINS query optimized for response time provides a fast solution for when you need the highest scoring documents from a hint list. Besides using query hints, there are other parameters that can influence query response time such as:

- Collection of table statistics
- Memory allocation

- Sorting
- Presence of LOB columns in your base table
- Partitioning
- Parallelism
- The number term expansions in your query, etc.

3. Optimizing queries for throughput:

Query Throughput (QthD) is a measurement used to determine the performance of a database system. The throughput metric is a classical throughput measure characterizing the ability of the system to support a multi-user workload in a balanced way.

V. OPTIMIZATION TECHNIQUES:

Before you start fidgeting with individual SQL statements, it is important to note that hints are probably the last thing you should consider adding when attempting to optimize your code. There are several levels of optimization [8] that you start with the server, then the database instance, and finally go down through the database objects to individual statements. After all, the effect of any changes made to individual statements, particularly hints, may be lost when the database is fine-tuned later on.[15]

As a database developer/architect you may not want to treat the path that leads to the desk of the DBA. Fortunately, there is a bunch of things you can do to improve the runtime performance of your statements: [15]

A. Indexes

An **index** is a copy of selected columns of data from a table that can be searched very efficiently that also includes a low-level disk block address or direct link to the complete row of data it was copied from. Some **databases** extend the power of **indexing** by letting developers create **indexes** on functions or expressions. Indexes used to improve database performance and statistics[3][12][14]

B. Constraints

Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table. The whole purpose of [3] **constraints** is to maintain the data integrity during an update/delete/insert into a table. [12][15]

C. Materialized views:

A materialized view is a database object that contains the results of a query. For example, it may be a local copy of data located remotely, or may be a subset of the rows and/or columns of a table or join result, or may be a summary using an aggregate function. [6]

D. Indexing in the table column in the database

We need to create primary key in every table of the database. When we create a primary key in a table, a clustered index tree is created and all data pages containing the table rows are physically sorted in the file system according to their primary key values. Each data page contains rows which are also sorted within the data page according to their primary key values. So, each time we ask any row from the table, the database server finds the corresponding data page first using the clustered index tree and then finds the desired row within the data page that contains the primary key value[11][7]

E. Rewrite SQL statements:

- We should exclude projections that are not required.
- We should minimize the amount of work done more than once.
- We should factor sub queries that are used multiple times in the same statement.[16]
- We should use EXISTS instead of IN because the former stops processing once it has found a match.[16]
- We should use CASE and/or DECODE to avoid having to scan the same rows over and over again, especially for aggregation functions that act on different subsets of the same data.
- We should use analytic functions to do multiple or moving/rolling aggregations with a single pass through the data.
- We should avoid scalar sub queries in the SELECT-list.
- We should use joins instead of sub queries while using multiple tables, as it gives the optimizer more room to play around in.
- We should use either inner join or outer join for better result
- We should avoid implicit conversions of data types, especially in the WHERE clause.
- Write WHERE clause predicates with a close eye on the indexes available, including the leading edge of a composite index.[16]
- We should avoid, whenever possible, comparison operators such as <>, NOT IN, NOT EXISTS, and LIKE without a leading '%' for indexed columns in predicates.
- Don't abuse HAVING to filter rows *before* aggregating. We should try to avoid unnecessary sorts, including when UNION ALL rather than UNION is applicable.
- We should not use DISTINCT unless you have to use it.
- We should use PL/SQL, especially packages with stored procedures (and bind variables) and shared cursors to provide a clean interface through which all data requests are handled. Add hints once you have determined that it is right and necessary to do so.[16]

F. Appropriate Covering Index:

If we know that our application will be performing the same query over and over on the same table, we should consider creating a covering index on the table [10]. A covering index, which is a form of a composite index, includes all of the columns referenced in SELECT, JOIN, and WHERE clauses of a query. Because of this, the index contains the data we are looking for and SQL Server doesn't have to look up the

actual data in the table, reducing logical and/or physical I/O, and boosting performance. On the other hand, if the covering index gets too big (has too many columns), this could actually increase I/O and degrade performance. So we have to be very careful to create the covering indexes. [12][13]

VI. QUERY OPTIMIZATION PERFORMANCE EVALUATION

Explain plan helps to analyze how indexes are being used or where not to use indexes. To generate explain plan we required file sorts, what are the tables, columns are being queried. And we should define query structure, alter tables, columns and data types.

The EXPLAIN PLAN statement displays execution plans chosen by the Oracle optimizer for SELECT, UPDATE, INSERT, and DELETE statements. A statement's execution plan is the sequence of operations Oracle performs to run the statement.

- The row source tree is the core of the execution plan. It shows the following information:
- An ordering of the tables referenced by the statement
- An access method for each table mentioned in the statement
- A join method for tables affected by join operations in the statement
- Data operations like filter, sort, or aggregation

In addition to the row source tree, the plan table contains information about the following:

- Optimization, such as the cost and cardinality of each operation
- Partitioning, such as the set of accessed partitions
- Parallel execution, such as the distribution method of join inputs
- Here in Experimental Evaluation we have use explain plan for below query and before indexing and after indexing cost is different.

VII. RESULT

- Here in Experimental Evaluation we have use explain plan for below query and before indexing and after indexing cost is different.

Classification	Result
Before Indexing	239*4079*9=8,773,929
After indexing	1 * 1 * 9 = 9

Experimental Evaluation

select C.Name, Y.Name, Y.Population, Language from Country as C, City as Y, CountryLanguage as L where Y.Name = C.Name and L.CountryCode = Y.CountryCode and C.Name = 'Macao' ;

Generate plan:

explain select C.Name, Y.Name, Y.Population, Language from Country as C, City as Y, CountryLanguage as L where Y.Name = C.Name and L.CountryCode = Y.CountryCode and C.Name = 'Macao' ;

```
----+-------------+-------+------+---------------+---------+---------+----------------------+------+-----------------------------+ | id | select_type
| table | type | possible_keys | key  | key_len | ref               | rows | Extra                       | +----+-------------+-------+------+-
-------------+---------+---------+--------------------+------+-----------------------------
|  1 | SIMPLE | C | ALL  | NULL  | NULL| NULL| NULL             |  239 | Using where
 |  1 | SIMPLE    | Y | ALL  | NULL          | NULL| NULL| NULL              | 4079 | Using where; Using join buffer
 | 1 | SIMPLE | L| ref  | PRIMARY          | PRIMARY | 3| world.Y.CountryCode |   9 | Using index             +----+-------------
+-------+------+---------------+---------+---------+------------------
3 rows in set (0.00 sec)
```

- **Alter table country add index**
- **Alter table city add index**

explain select C.Name, Y.Name, Y.Population, Language from Country as C, City as Y, CountryLanguage as L where Y.Name = C.Name and L.CountryCode = Y.CountryCode and C.Name = 'Macao' ;

```
| id | select_type | table | type | possible_keys | key   | key_len | ref               | rows | Extra               |+----+-------------+-------
+------+---------------+---------+---------+--------------------+------+-------------------------+
|  1 | SIMPLE    | C       | ref  | c2            | c2      | 52      | const             |          1 | Using where; Using index |
|  1 | SIMPLE    | Y       | ref  | c2            | c2      | 35      | const             |          1 | Using where              |
| 1 | SIMPLE    | L       | ref  | PRIMARY       | PRIMARY | 3   | world.Y.CountryCode | 9 | Using index
```

3 rows in set (0.00 sec)

The original cost was 239 * 4079 * 9 = 8,773,929

The new cost is 1 * 1 * 9 = 9

VIII. CONCLUSION

Different techniques using different representations show that there are many other ways to represent query other than the mentioned techniques. From the above experimental analysis, we can say that the covering indexing at table level is simple method to represent the query. Still, more research scope is in Query Optimization and in this field more work can be

implemented in distributed and deductive databases. Our future work on query optimization includes comparison of optimization techniques along with the different databases.

REFERENCES

[1] Chaudhuri S. and K. Shim. Query optimization with aggregate views. In Proceedings of the 5th International Conference on Extending Database Technology, Avignon, France, March 1996.

[2] C.J. Date, "An Introduction to Database Systems", Addison

[3] Wesley, ISBN-10: 0321197844, ISBN-13: 978-0321197849,August 1, 2003.

[4] C. J. Date, "Foundation for Future Database Systems: The Third Manifesto", Addison-Wesley Professional, ISBN-10:0201709287, ISBN-13: 978-0201709285, 2000.

[5] Abdullah Dilsat : Query Optimization in Distributed Databases. Report, Middle East Technical University, December 2003.

[6] Clare Churcher, "Beginning Database Design: From Novice to Professional", Apress, ISBN-10: 1590597699, ISBN-13: 978-1590597699, January 15, 2007.

[7] Chaudhuri S. and K. Shim: An Overview of Cost-based Optimization of Queries with Aggregates. IEEE DE Bulletin, Sep. 1995. (Special Issue on Query Processing).

[8] Chaudhuri S. and K. Shim: Including group-by in query optimization. In Proceedings of the 20th International VLDB Conference, Santiago, Chile, Sept 1994.

[9] Chaudhuri S.: An Overview of Query Optimization in Relational Systems ; Pods'09, ACM New York, NY, USA, Year 1998. G Join R1 R2 G 1 Join R1 R2 Tree1 Tree3 G1 G R 2 Join R1 Tree2 Figure 3. Single Block Query Transformations International Journal of Computer Applications (0975 – 888) Volume 47– No.15, June 2012 9

[10] Sukheja Deepak and Umesh Kumar Singh : A Novel Approach of Query Optimization for Distributed Database Systems. IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1, July 2011.

[11] International Journal of Computer Applications (0975 – 888) Volume 47– No.15, June 2012 - Analysis of Query Optimization Techniques in Databases- Jyoti Mor M. Tech Student, CSE Dept. MRIU, Faridabad Indu Kashyap Assistant Professor, CSE Dept. MRIU, Faridabad R. K. Rathy, PhD. Professor, CSE Department MRIU, Faridabad

[12] Retrieved from https://www.codeproject.com/Articles/34372/Top-steps-to-optimize-data-access-in-SQL-Server on 15-June-2017

[13] http://www.dbjournal.ro/archive/2/7_Ghencea_Gieger.pdf

[14] http://www.ijltet.org/wp-content/uploads/2015/02/62.pdf

[15] http://www.techfounder.net/2011/03/25/database-optimization-techniques-you-can-actually-use/

[16] https://webdocs.cs.ualberta.ca/~zaiane/courses/cmput391-02/slides/Lect3/sld001.htm

[17] http://paper.ijcsns.org/07_book/201008/20100842.pdf