IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES
ISSN PRINT 2319 1775 Online 2320 7876
 Journal UGC CARE Listed (Group-I) Volume 11, Special Iss 01 2022

A Performance Evaluation of Indexing and Query Optimization Techniques in Relational Databases

Dr. Rasika Khandal
Assistant Professor, B.C.A. Department,
Sudha Sureshbhai Maniar College of Computer and Management,
RTM Nagpur University. Nagpur, India. Email: raut.rashu@gmail.com

Abstract:

The present research paper compares the indexing and query optimization techniques in respect to their influence on the functioning of relational databases. An important method is indexing and optimization of queries to enhance the efficiency and speed of information retrieval. The paper employs such practical evaluation on a given sample data and it compares a range of indexing techniques approaches (e.g. the B-tree, hash index), as well as optimization strategies (e.g. cost-based, heuristic-based). The descriptive statistics and hypothesis testing is used in analyzing. The outcomes give information on which methods can work better in a particular scenario, on how database developers and administrators can be advised.

Keywords:

Relational Database, Indexing Techniques, Query Optimization, Performance Evaluation, SQL, Query Execution Time, B-tree, Hash Index, Cost-based Optimization.

Introduction:

In the modern digital world, the data is expanding at a tremendous rate. Whether it is small scale projects such as databases in school to megacorporates such as banks and online stores, billions of rows of data are accessible and dealt with through relational databases. These databases store data in tables and also enable one to execute queries in order to retrieve the necessary data. But as the size of data enlarges, it turns to be slower and also hard to get the quick results. This is where indexing and query optimization is very important.

Databases: Databases can be searched faster using the method of indexing. Let us say you are reading a book and you need to find out you need to find a topic, how would you do it? You would use the index in the book to go right there without opening every single page, right. The same goes with the database index, instead of opening every row in the table and reading the table like a book until you find your required data. The indexes are of different types: B-tree index and hash index, and they all work differently based on the type of query that is to be executed.

The popular methods of relational database management system RDBMS is b-tree indexing. It packs information in a balanced tree such that the insertion, deletion, and searching are easy. B-trees are most suitable in large dataset databases that require a high frequency of range queries and dynamic updates.

Hash indexing assigns keys to specific locations in a hash table by use of hash functions. This approach can be useful when dealing with a high precision match operation e.g. obtaining a record by key e.g. an ID. It is no good choice in case of range queries or when data is frequently modified.

Indexing and query optimization working in conjunction have the potential to drastically enhance the performance of a database i.e. the database will be faster, more efficient and capable of serving more users as well as bigger amount of data.

In this research paper, the study is aimed at analyzing the impact of various indexing and query optimization strategies with reference to the performance of a relational database. This paper seeks to educate the developer, database administrator and others about the best way to do certain things as well as the students to facilitate comparison of their effect with the help of practical experiments and statistical analysis. The last objective would be to give conclusive results on how to increase the efficiency of databases through the appropriate mix of methods.

Literature Review:

In Kumar and Sharma (2020), an in-depth analysis of the work of various indexing methods in relational database systems was carried out. They discovered that good indexing such as B-tree and hash indexing can greatly save time in executing queries. In their study they focused on the importance of indexing in enhancing speed and effectiveness of large databases.

Verma and Gupta (2017) directed their attention to the cost-founded query optimization in relational databases. Their study described that cost-based techniques analyze different execution plans and they select that execution plan that has the lowest estimated cost because the plan leads to execution speedier queries. This method can prove convenient, in particular, when dealing with complex queries having additional joins and constraints.

Adaptive indexing was first mentioned by Jain and Mishra (2016). Their study towards adaptive indexing builds and optimises indexes as the database is accessed and no longer uses pre-created indexes. This adds flexibility and efficiency to the system when working in an environment where the pattern of queries constantly varies.

A comparative study of B-tree and hash indexing methods was given by Joshi and Sharma (2015). They determined that B-tree indexing is more suitable in applications where the queries are range-based and hash indexing is better suited to those applications that involve exact match queries. The research found out that the choice of indexing method must be based on which queries are being executed.

Singh and Kaur (2017) conducted a general presentation of query methods and query optimization strategies in relational database. They described different phases of a query processing such as parsing, optimization and execution. The paper has addressed the heuristic and cost-based optimization methods and has demonstrated that each of the method has its own merit based on the complexity of the query.

Patel and Shah (2019) contrasted heuristic-based optimization and cost-based optimization techniques. They decided that the heuristic was quicker to use but the cost based techniques

have better performance over the long-term since they followed all valid computable plans and then chose the best.

Chatterjee and Banerjee (2019) provided a description of indexing mechanisms in databases management systems. Their publication went through a variety of indexes that are possible such as clustered indexes, non-clustered indexes, indexes created by use of bitmaps and indexes by use of full-text and indicated the way in which each is used during different situations. They stressed on the consideration on the selection of the appropriate index in order to enhance search.

Indexing was analyzed in Deshmukh and Patil (2018) to enhance speed of execution of SQL in queries. They have proved by experiment that properly designed indexes can minimize disk I/O and increase the speed of the query especially on large databases where millions of records exist.

Raman and Chandra (2003) suggested intelligent database systems framework which is capable of selecting appropriate indexing and optimization strategies automatically. They created their early work which concerned making databases smarter through the use of decision-making algorithms and this is the actual start of current adaptive query processing systems.

Aggarwal and Goyal (2016) analyzed various query optimization methods of relational databases. They discussed a number of what database engines have done and how effective both indexing and query optimization is integral to improve performance optimum.

Objectives of the Study:

- To evaluate the impact of indexing on query performance in relational databases.
- To compare different indexing techniques such as B-tree and hash indexing.
- To study the effect of query optimization strategies on execution time.

Hypothesis:

- **H_0 (Null Hypothesis):** There is no significant difference in query performance with different indexing and optimization techniques.
- **H_1 (Alternative Hypothesis):** There is a significant difference in query performance with different indexing and optimization techniques.

Research Methodology:

The study conducted was to assess the performance of relational databases in terms of the use of various indexing and query optimization. In order to achieve this we employed a practical experimental approach. MySQL, a broadly adopted database management system, was selected as the system to be used in this study and it has various features of indexing and optimizing queries. To test this case, we generated a huge sample set of 1 million student records that includes fields student ID, name, department, marks and contact information. This huge amount of data allowed us to resemble our database settings in reality.

We executed different SQL statements on the data including the simple SELECT statements, queries involving the use of WHERE, connective, JOIN. These queries were run under various circumstances, initially the queries were run without indexing, then the B-tree indexing and hash indexing were used on the respective columns. Thereafter, we also experimented the same queries with different query optimization solutions, i.e., cost-based optimization and heuristic-based optimization.

In order to measure performance, we concentrated on the query execution time how long does each query take to provide us the result. To ascertain precision of our measurements we repeated every query several times and found the average time taken. The received data was subsequently analyzed by descriptive statistics (the minimum, maximum, and average time were determined) and t-tests (the differences in the performance of the methods were examined to see whether they were statistically significant).

Practical test and analysis by steps brought us closer to comparison between the efficiency of each indexing and optimization technique and realization that which of them will work better in different conditions.

Table 1: Descriptive Statistics:

Technique	Min Time (ms)	Max Time (ms)	Average Time (ms)
Without Indexing	1500	2000	1750
B-tree Index	300	600	450
Hash Index	250	500	375
Cost-based Optimization	220	400	310
Heuristic Optimization	350	550	450

Analysis of Descriptive Statistics:

The challenges in the collection of descriptive data in this research aid in forming our insight on how various indexing and query optimization approaches perform in terms of query execution time. The table above contains the minimum, maximum and average time (in milliseconds) that each approach required to run SQL queries on a huge dataset.

Based on the statistics it is easy to understand that non indexed queries were the slowest queries that executed. The average execution time on the present case was approximately to 1750 ms with some of the queries running as long as 2000 ms. This indicates that in absence of indexing of data, the database must scan each and every record to locate the necessary material, and this can be considered a very long process, data being voluminous.

By contrast, the performance improved substantially as we used B-tree indexing. The average time of execution became 450 ms and this indicated that the database gained the ability to find the data much quicker. B-tree index can be extremely resourceful with range queries (e.g. marks between 50-80) and this finding justifies the effectiveness of B-tree index.

Hash indexing gave an even better performance and its average time was lowest (375 ms). When an equality search is involved (e.g. student id = 12345), hash indexes tend to be very fast, since the search quickly provides a pointer to the data location. The findings indicate

that hash indexing is even more efficient than B-tree indexing in case an exact match is performed.

Regarding the query optimization, cost-based optimization approached better results in comparison with the heuristic approach. Cost-based optimization is the way that database engine analyses several plans to query and selects the one that involves the least time and compatible resources. Execution took the average of 310 ms compared to 450 ms in the case of heuristic methods which are much better results. Heuristic optimization, however, does not always give the best plan and works according to predetermined rules (as doing SELECT before doing JOIN).

- No indexing slower performance
- Hash index = optimal performance of queries on equality
- B-tree index = excellent as far as range-based requests
- Optimized based on cost = optimal executing strategy
- Heuristic optimization > no optimization at all, but not optimal

This discussion shows that optimization strategy and the suitable indexing method according to the kind of query can be very helpful regarding database performance.

Table 2: Hypothesis Testing (t-test results):

Comparison	t-value	p-value	Significant (Yes/No)
No Index vs B-tree Index	10.45	0.0001	Yes
No Index vs Hash Index	11.32	0.0001	Yes
B-tree vs Hash Index	2.10	0.041	Yes
Cost-based vs Heuristic Opt.	3.25	0.012	Yes

Analysis of Hypothesis Testing:

In order to find out whether there were any significant differences in query performance in the use of various indexing and optimization methods, we carried out the process of hypothesis testing that we ran through a t-test method. By using this test, we can verify if the gains in the performance (with average time of query execution) are related to applied techniques or this is merely a random change.

We started to hypothesize the following:

Ho: Rejected Null Hypothesis: The error of no significant difference between the performances of queries in different indexing or optimization methods.

Alternative Hypothesis (H1): Performance in query between indexing technique and optimization technique used significantly differ.

We set individual comparisons, t-tests, on important variables:

1.No Indexing compared to B-tree Indexing

It indicates that the t-value and the P-value were 10.45 and 0.0001 respectively which is significantly lower than the significance level which was 0.05. The implication of this is that there is statistically significant boost in performance that turns out to be due to the B-tree indexing. Thus, we have rejected the null hypothesis and believed the B-tree indexing provides substantially improved performance of queries.

2.Without Indexing vs Hash Indexing

In this case, t-value was 11.32 and p-value also was 0.0001. Again, this implies a very high degree of statistical difference affirming that with hash indexing, there are substantially better performances of queries as opposed to the case when no index is used.

3.The difference between B-tree Indexing and Hash Indexing

The t-value of this test was 2.10 and the p-value was 0.041 with a value that is less than 0.05. Although the difference is not so significant as compared to previous cases, it is still significant in terms of statistics. This indicates that hash indexing is a little bit faster than an index that is based on B-tree, particularly in the use of equality-based query.

4.Cost-Based Optimization vs Heuristic search

In this instance t=3.25 and p=0.012. This is again lower than the 0.05 value, which means that it will definitely achieve more performance when using a cost-based optimization rather than the heuristic rules. The dynamic selection of the preferred execution plan by the cost-based approach makes room to produce results faster.

Conclusions Overall Results:

This paper addressed the performance testing of a range of indexing and query optimization strategies applicable in the case of the relational database. After carrying out various experiments on a big dataset containing varying types of queries it was evident that indexing as well as optimization contribute significantly towards enhanced performance of query.

It was found that:

The No index query was the worst that required more time to perform.

Hash indexing returned optimal results when speed was required to a query that had an exact match (equality restrictions).

B-tree indexing also worked well too, particularly when it needed a range-based search (e.g. searching between two numbers).

In comparing query optimization techniques, the problem is clear that cost-based optimization was more effective than heuristic-based optimization since in cost-based optimization approach the optimal query execution plan is chosen dynamically.

All these improvements were found to be significant statistically as the statistic testing affirmed.

On balance, the research justifies the fact that, when the query is properly selected, the database performance can be significantly improved due to an appropriate choice of indexing

strategy and structure optimization. Database designers, developers and administrators can also find these findings useful in an attempt to speed up and streamline their systems.

Future Scope of the Study:

Although the study has come up with a significant finding, there are a number of issues that could be expanded:

- Future Experimentation on Distributed Databases: The performance factors may change under the network, storage tiers of the distributed databases, e.g., Google big query or Amazon Redshift: testing the indexing and optimization techniques can be done in future.
- Advanced Indexing Techniques: The techniques such as other types of indexes, i.e. bitmap indexes, spatial indexes and composite indexes could be used in future experiments to analyse their performance in complex queries too.
- Interaction with NoSQL Databases: Although during this work relational databases were used, in the future, it will be interesting to compare the techniques used in it with indexing approaches in trans relational databases like MongoDB and Cassandra.
- Future experiments: Dynamic workloads: In future, dynamic workloads may be used to examine how their indexing and optimization performs under realistic load, in a system like a banking or e-commerce system.
- AI-Based Query Optimization: The use of machine learning algorithm to automatically select the most optimal query plan, using past data, is an area that future research may focus on.
- Energy Efficiency Analysis: It will also be interesting to analyze the energy consumption of the database servers by the means of indexing and optimization, as it is crucial when it comes to sustainable computing.

It is by further extending the study in the above directions that future research can add even more to the area of database performance optimization.

References:

[1] A. Kumar and R. Sharma, “Performance Evaluation of Indexing Techniques in Relational Database Systems,” in *Proc. Int. Conf. on Computing, Communication & Automation (ICCCA)*, Greater Noida, India, 2020, pp. 125–129, doi: 10.1109/ICCCA.2020.9121064.

[2] S. Verma and K. Gupta, “Query Optimization in Relational Databases Using Cost-Based Techniques,” *Int. J. Computer Applications*, vol. 162, no. 7, pp. 18–22, Mar. 2017.

[3] R. Jain and P. Mishra, “Improving Database Performance Using Adaptive Indexing,” in *Proc. 6th Int. Conf. on Advances in Computing and Communications (ICACC)*, Kochi, India, 2016, pp. 78–83.

[4] D. Joshi and M. Sharma, "Comparative Study of B-tree and Hash Indexing Techniques for Relational Databases," in *Proc. IEEE Int. Conf. on Emerging Trends in Computing and Communication Technologies (ICETCCT)*, Dehradun, India, 2015, pp. 56–61.

[5] P. Singh and S. Kaur, "Query Processing and Optimization in Relational Database Management Systems: A Review," *Int. J. Recent Trends in Engineering and Research (IJRTER)*, vol. 3, no. 4, pp. 112–117, Apr. 2017.

[6] N. Patel and R. Shah, "Heuristic and Cost-Based Optimization Techniques for SQL Queries," in *Proc. IEEE Int. Conf. on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, India, 2019, pp. 312–316.

[7] S. Chatterjee and A. Banerjee, "An Overview of Indexing Mechanisms in Database Management Systems," *Int. J. Scientific and Engineering Research*, vol. 10, no. 8, pp. 104–110, Aug. 2019.

[8] A. Deshmukh and V. Patil, "Role of Indexing in Improving Performance of SQL Queries," in *Proc. 2nd IEEE Int. Conf. on Inventive Systems and Control (ICISC)*, Coimbatore, India, 2018, pp. 689–693.

[9] R. Raman and S. Chandra, "A Framework for Intelligent Database Systems," *J. Computer Science and Informatics*, vol. 6, no. 2, pp. 112–118, 2003.

[10] M. Aggarwal and H. Goyal, "A Study on Query Optimization Techniques in Relational Databases," *Int. J. Computer Science and Mobile Computing*, vol. 5, no. 6, pp. 235–241, Jun. 2016.