Efficient Database Indexing Techniques using Cluster, Non-Cluster and Full Text Indexing

Mr. Kawhale Rohitkumar K., Miss. Prachi Patil M., Mr. Chabukswar Mandar U.

Department of Computer Science and Engineering, Shivaji University
Bharati Vidyapeeth College of Engineering, Kolhapur, India
rohit.kawhale@gmail.com
chabukswarmandar@yahoo.com

KIT College Of Engineering near Chitranagari, Kolhapur, India
prachimpatil114@gmail.com

***Abstract*— Data warehouse is the central management system having pool of datasets. The traditional data warehouse was designed in such a manner that it can efficiently manage transactional data which is highly dominated by numerical information where as in some data warehouse textual and non transactional information is encountered. The dataset which contains text data is accessed over the network on the daily basis and performance issue arises. The objective of this paper is to propose an indexing technique based on time complexity and index space complexity for data warehouse used in applications. The performance evaluation of three data warehouse queries is focused in this paper by comparing techniques used mostly with FULL TEXT indexing technique and to observe the results of variable size dataset with respect to time and space complexity.**

***Keywords*— Cluster Index, Non-Cluster Index, Full Text Index, Data Warehouse.**

I. INTRODUCTION

The traditional data warehouse was designed in such a manner that it can efficiently manage transactional data which is highly dominated by numerical information where as in data warehouse textual and non transactional information is encountered. The purpose of 'Efficient Indexing Technique for Data Warehouse' is to find out the best indexing technique among the indexing techniques like clustered indexing, non-clustered indexing, full-text indexing, etc. The objective of this paper is to propose an indexing technique based on time complexity and index space complexity for data warehouse used in application. The performance evaluation of three data warehouse queries is focused by comparing techniques. Indexing of a data warehouse is complex and if there are few indexes, the data loads quickly but the query response is slowed [1]. Although many ways of improving the performances of database systems exist, the most efficient one consists in implementing an effective data indexing mechanism. Thus, the efficient data indexing based on the type, structure, the physical organization and the cardinality of data, the type of queries and the number of competing transactions may increase the performance of the database administration systems and, implicitly, the performance of the entire information system [2].

II. BACKGROUND

A. Cluster Indexing

Clustered indexes sort and store the data rows in the table or view based on their key values. These are the columns included in the index definition. There can be only one clustered index per table, because the data rows themselves can be sorted in only one order. If a table has no clustered index, its data rows are stored in an unordered structure called a heap. A clustered index is a special type of index that reorders the way records in the table are physically stored. Therefore table can have only one clustered index. The leaf nodes of a clustered index contain the data pages.

B. Non-Cluster Indexing

Non-clustered indexes have a structure separate from the data rows. A non-clustered index contains the non-clustered index key values and each key value entry has a pointer to the data row that contains the key value. We can have a non-clustering index with an index record for each search-key value. The index record points to a bucket that contains pointers to all the actual records with that particular search-

key value. A non-clustered index is a special type of index in which the logical order of the index does not match the physical stored order of the rows on disk. The leaf node of a non-clustered index does not consist of the data pages. Instead, the leaf nodes contain index rows.

C. Full Text Indexing

Full text searching is the process of searching for words in a block of text. Full text search refers to the functionality in SQL Server that supports full text queries against character-based data. These types of queries can include words and phrases as well as multiple forms of a word or phrase. To support full text queries, full text indexes must be implemented on the columns referenced in the query. The columns can be configured with character data types (such as char and varchar) or with binary data types (such as binary and image). A full text index is made up of word tokens that are derived from the text being indexed. The needs of full-text databases are not well served by traditional database systems, since, instead of key indexing, full text requires facilities such as document ranking and indexing on text content [3].

III. PROPOSED WORK AND DESIGN

A. Factors

Following are the factors which are used to determine which indexing technique should be used on data warehouse from various existing indexing techniques.

1. Cardinality data: Distinct values in columns represent cardinality of data which can be further classified into three categories:

a) High cardinality: It can be unique ID number of employees.
b) Normal cardinality: It can be combination of first and last name of employee.
c) Low cardinality: It can be value in which male or female employee is classified.

2. Distribution: In this frequency of entries are to be considered in data warehouse to identify indexing technique because some indexes are created in less amount of time but some take considerable amount of time.

3. Value range: The minimum and maximum values are used to generate the range and count range sizes, according to which the index should be selected.

B. Proposed System Architecture

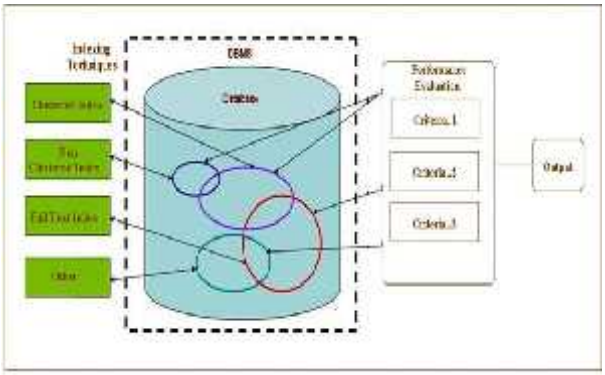

Figure 1 : Proposed System Architecture

The above Figure 1 depicts the process flow of our proposed work. The proposed system architecture consists of data warehouse on which we are going to represent the data with various indexing techniques like clustered index, non-clustered index, and full text index techniques for the efficient retrieval of information.

Although many ways of improving the performances of database systems exist, the most efficient one consists in implementing an effective data indexing mechanism. Thus, the efficient data indexing based on the type, structure, the physical organization and the cardinality of data, the type of queries and the number of competing transactions may increase the performance of the database administration systems and, implicitly, the performance of the entire information system [2].

IV ANALYSIS

In this section techniques are implemented and tested on the basis of memory consumed with dataset of different sizes. These techniques are full text index, cluster index and non-cluster index. The graphical representation shows all indexing techniques with respect to different datasets on different type of queries.

Creation time of various indexing techniques corresponding to different datasets has been calculated and on basis of these values, indexing techniques have been analyzed on factors like CPU time and creation time.

A. Process Execution Time Analysis

The process utilisation time is the combination of compilation and execution time. The above line charts shows the performance of indexing techniques. It is obtained that the full text indexing is better than cluster and non cluster indexing technique, because it requires less processing time as compare to others. In cluster indexing technique the processing is increased gradually as the number of records increased in datasets.

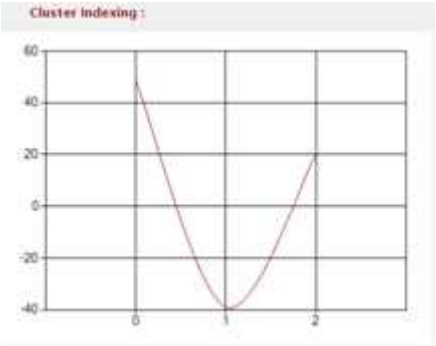

Figure 1 : Line Chart for Cluster Indexing

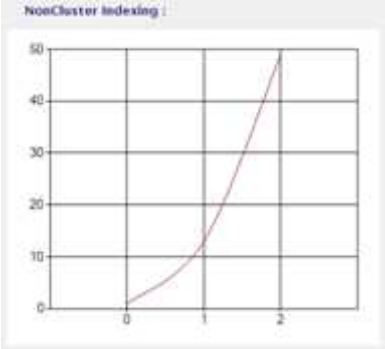

Figure 2: Line Chart for Non-cluster Indexing

Figure 3: Line Chart for Full Text Indexing

B. Execution Time

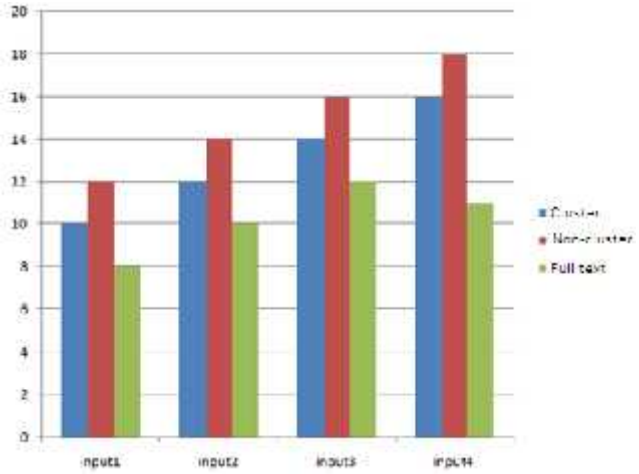

Figure 4: Bar Graph (Result versus Inputs)

The above graph gives the details of obtained experimental results of cluster, non cluster and full text indexing techniques. From the graph we analyses here that the time required for the selection of data by full text indexing is less as compared to cluster and non cluster indexing. Thus we conclude here the full text indexing is more efficient than cluster and non cluster indexing.

C. Index Memory Consumption

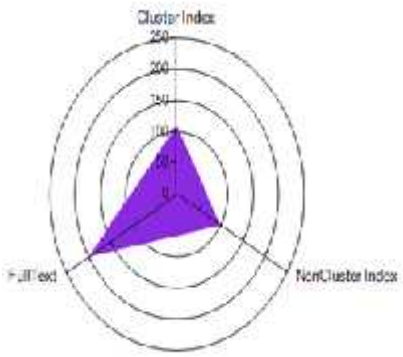

Figure 5: Pyramid chart

Cluster index takes much time than other indexes and on the contrary, it takes less memory. Non cluster takes more memory than cluster but less than full text. Full text index consumes more memory and on the contrary, it gives best performance than others.

IV. CONCLUSIONS

The ability to extract data to answer complex, iterative, and queries quickly is a critical issue for data warehouse applications. A proper indexing technique is crucial to avoid I/O intensive table scans against large data warehouse tables. The challenge is to find an appropriate index type that would improve the queries' performance.

Full text indexing provides better performance than cluster and non cluster indexes but on the contrary, it consumed much memory comparatively other techniques which is costly for large data warehouse. It can be improve by create new data structure of index which contains splitting criteria of characters and it must be memory and cost effective with timely information.

ACKNOWLEDGMENT

I express my deep sense of gratitude and appreciation towards my project guide because of his continuous inspiration and valuable guidance in throughout my work.

REFERENCES

[1] Performance Measurement of Indexing Techniques Used in Biomedical Databases – Amit Bansal, Sankalap Arora.

[2] Data Indexing Algorithms Lucian Bornaz Academy of Economic Studies, Bucharest, Romania.

[3] An Efficient Indexing Technique for Full-Text Database Systems,Justin Zobel, Alistair Moffat , Ron Sacks.