A Study on Database Indexing Techniques and Their Limitations

Aziz Nasridinov*,Young-Ho Park*
*Dept. of Multimedia Science, Sookmyung Women's University
aziz_nasridinov@yahoo.com, yhpark@sookmyung.ac.kr

Abstract

There are numerous approaches to improve the performances of database systems. The most efficient ones are considered to be implementing an effective database indexing technique. This must guarantee the balance between the memory, storage resources and processor of the database server according to the type, structure, the physical organization and the cardinality of data, the type of queries and the number of competing transactions. The challenge is to find an appropriate index type that would suite these requirements. In this paper, we provide an overview of the most used database indexing techniques such as B-tree index and Bitmap index, present an analysis of these techniques and discuss their limitations.

1. Introduction

The informational society contains large data processing information systems. The need to store, manipulate and interrogate these systems requires the implementing of high-performance database systems, capable of reacting as fast as possible to the user-submitted requests [1]. There are numerous approaches to improve the performances of database systems. The most efficient ones are considered to be implementing an effective database indexing technique. This must guarantee the balance between the memory, storage resources and processor of the database server according to the type, structure, the physical organization and the cardinality of data, the type of queries and the number of competing transactions. The challenge is to find an appropriate index type that would improve the queries' performance. In this paper, we provide an overview of the most used database indexing algorithms such as B-tree and Bitmap index, present an analysis of these algorithms and discuss their limitations.

The rest of the paper is proceeds as follows. Chapter 2 provides an overview of the most used database indexing algorithms such as B-tree and Bitmap index. Chapter 3 presents analysis of these algorithms and discusses their limitations. Chapter 4 highlights conclusions.

2. Database Indexing Techniques

In this chapter, we provide an overview of the most used database indexing techniques such as B-tree index and Bitmap index. The most used indexes in recent database systems are B-Tree and Bitmap Index [2]. These indexes represent a tree-like structure that is structurally and functionally equal.

2.1 B-Tree Index

The B-Tree Index is the default index for most relational database systems. The top most level of the index is called the root. The lowest level is called the leaf node. All other levels in between are called branches. Both the root and branch contain entries that point to the next level in the index. Leaf nodes consisting of the index key and pointers pointing to the physical location (i.e., row ids) in which the corresponding records are stored.

The term B-tree may refer to a specific design or it may refer to a general class of designs. In the narrow sense, B-tree stores keys in its internal nodes but need not store those keys in the records at the leaves. The general class includes variations such as the B+-tree and the B*-tree.

- In the B+-tree, copies of the keys are stored in the internal nodes; the keys and records are stored in leaves; in addition, a leaf node may include a pointer to the next leaf node to speed sequential access.
- The B*-tree balances more neighboring internal nodes to keep the internal nodes more densely packed. This variant requires non-root nodes to be at least 2/3 full instead of 1/2. To maintain this, instead of immediately splitting up a node when it gets full, its keys are shared with a node next to it. When both nodes are full, then the two nodes are split into three.

2.2 Bitmap Index

In bitmap index, Indexes are represented in binary format. i.e in 0's and 1's format. This index is helpful where table has low cardinality. Bitmap index stores the column values in bits. Each bit represents a single value. For example, the gender column has two possible values: Male and Female. Two bits will be used in the bitmap to capture the index on the gender column. So the more distinct the value is the more space is required to store the bitmap. Internally, the database engine, like Oracle, uses a map function to convert the bit location to the distinct value. Thus bitmap index works efficiently in case where table has low cardinality.

Hence indexes can be applied taking into consideration following points [3]:

- Table is used for intensive query processing (e.g data warehouse, OLAP systems) or transaction processing system (OLTP systems).
- Table column/columns on which index is to be applied has low cardinality (distinct column values) or high cardinality. Depending on cardinality which index is to be applied can be decided.

<Table 1> Analysis of Database Indexing Techniques

Database Indexing Technique	Advantages	Disadvantages	Implementing Commercial Systems
B-Tree Index	• It speeds up known queries. • It is well suited for high cardinality. • The space requirement is independent of the cardinality of the indexed column. • It is relatively inexpensive when we update the indexed column since individual rows are locked.	• It performs inefficiently with low cardinality data. • It does not support ad hoc queries. More I/O operations are needed for a wide range of queries. • The indexes cannot be combined before fetching the data.	• Oracle • Infomix • Red Brick
Bitmap Index	• It is well suited for low cardinality columns. • It utilizes bitwise operations. • The indexes can be combined before fetching raw data. • It uses low space • It works well with parallel machine. • It is suitable for OLAP.	• It performs inefficiently with high cardinality data. • It is also very expensive when we update index column. The whole bitmap segment of the updated raw is locked so the other raw cannot be updated until the lock is released. • It does not handle spare data well	• Oracle • Infomix • Red Brick • DB2 • Sybase

3. Analysis

The B-Tree Index is popular in data warehouse applications for high cardinality column such as names since the space usage of the index is independent of the column cardinality. However, the B-Tree Index has characteristics that make them a poor choice for DW's queries. First of all, a B-Tree index is of no value for low cardinality data such as the gender column since it reduces very few numbers of I/Os and may use more space than the raw indexed column. Secondly, each B-Tree Index is independent and thus cannot operate with each other on an index level before going to the primary source. Finally, the B-Tree Index fetches the result data ordered by the key values which have unordered row ids, so more I/O operations and page faults are generated [4].

As for Bitmap index, first of all, it performs inefficiently with high cardinality data. Secondly, it is also very expensive when we update index column. The whole bitmap segment of the updated raw is locked so the other raw cannot be updated until the lock is released. Thirdly, it does not handle spare data well.

To sum up, we can say that B-Tree Indexes should only be used for high cardinality data and predicted queries. Bitmap Indexes play a key role in answering data warehouse's queries because they have an ability to perform operations on index level before retrieving base data. This speeds up query processing tremendously. Table 1 summarizes our analysis.

4. Conclusion

In this paper, we provided an overview of the most used database indexing algorithms such as B-tree and Bitmap index, present an analysis of these algorithms and discuss their limitations. B-Tree Indexes should only be used for high cardinality data and predicted queries. Bitmap Indexes play a key role in answering data warehouse's queries because they have an ability to perform operations on index level before retrieving base data. This speeds up query processing tremendously.

This work was supported by the IT R&D program of MKE/KEIT. [10041854, Development of a smart home service platform with real-time danger prediction and prevention for safety residential environments].

Reference

[1] Bornaz, L. Optimized Data Indexing Algorithms for OLAP Systems. Database Systems Journal, Volume 1 Issue 2 (2010), 17-26.

[2] Pagh, R., Satti, R.S. Secondary Indexing in One Dimension: Beyond B-Tree and Bitmap Indexes. PODS'09 Proceedings of the twenty-eight ACM SIGMOD-SIGACT-SIGART (2009), 177-186.

[3] Fusco, F., Stoechlin, M.P., Vlachos, M. NET-FLi: On-the-fly Compression, Archiving and Indexing of Streaming Network Traffic. VLDB'10 Proceeding of the Very Large Databases (2010), 1382-1393.

[4] Vanichayoban, S. Indexing Techniques for Data Warehouses' Queries. The University of Oklahoma Press.