Volume 8, No. 5, May-June 2017

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

A Review Paper on Information Retrieval Techniques for Point and Range Query in Database System

Monika Yadav
Dept. of Computer Science and Applications,
Kurukshetra University, Kurukshetra, India

Abstract: Information retrieval (IR) is a science of searching for information. A query is used to extract information from database where fast query processing is the main issue. The mechanisms like index and hash table can be used to solve this problem but the main challenge is to find a proper index which improves query performance. This paper describes various indexing and hashing techniques in terms of query support, structure and application. B-tree, B+-tree, bitmap index are discussed for point query followed by spatial indexing techniques like quad tree, k-d tree, R-tree, R+-tree, R*-tree for range query and multidimensional data.

Keywords: B-tree, B+-tree, Bitmap index, Hashing, Index, Spatial indexing.

1. INTRODUCTION

Information retrieval is the tracing of information from stored data. Many universities, public libraries and web search engines uses information retrieval system.
Early use of computer for IR- In 1945, the idea to search for relevant information in computers was made familiar in the article '*As We May Think by Vannevar Bush'* [1]. In 1948, a machine which was capable of searching for text references associated with subject code was created called 'UNIVAC'. In 1970s large scale retrieval system came into use. After this IR as a research discipline focus on two important developments:
a) How the documents to be indexed?
b) How these documents to be retrieved?
In IR system first all need to know about what is data? So, data is defined as characters which may be recorded on magnetic, optical or mechanical recording media. The two types of storage data are:-
a) Structured data(SD)
b) Unstructured data(USD)
Structured data is applied to databases and unstructured data is applied to everything else. SD is used to access SQL (structured query language) and this provides a better way to manage data. In Sql, data is retrieved with the help of query; a query is request for information from database. A database
query can either be a select query or an action query. Action query performs insertion, updation and deletion on data. SQL is the standard for query language and its extensions are MySQL, AracleSQL, NuoDB. Other type of query is web search query i.e. a query that is typed in search engines such as Google, Yahoo or Bing etc. When user enters a query in web search then search engine uses an algorithm to determine results. For fast query processing indexing techniques are used in search engine for information retrieval. Indexing is crucial part of IR system. Traditional indexing techniques are only for point query. For range and multidimensional data, spatial indexing is used. K-D tree, Quad tree, R-tree, R+-tree, R*-tree are popular techniques of spatial indexing.

This paper is organized as follows. In section 2 indexing and its techniques are introduced. Section 3 describes bitmap index. In section 4 spatial indexing and its techniques are introduced. Section 5 concludes the paper.

2. INDEXING

A database index helps in improve speed of data retrieval operations in a database table. If database use indexes then there's no need to look each row in table each time a database table is accessed. Index in database works same as index used in a book. The use of query in indexing gives much better performance in IR. Indexing is also used for sorting the data. Index should be chosen properly so that it takes small space and supports ad-hoc and complex queries. There are different types of index used in databases i.e. clustering index, primary index and secondary index. Sparse and dense index are types of primary index. The popular techniques of indexing are-

2.1 B-TREE

B-tree is a self- balancing tree data structure which was proposed by Rudolf Bayer in [2]. It performs searches, insertions and deletions in logarithmic time. B-tree is generalization of binary search tree in which a tree node can contain more than two children. B-tree of order m satisfies following properties:
a) Each node in B-tree has at most m children and at least m/2 children.
b) The root has minimum two children if it is not a child node.
c) All leaf nodes are at the same level.
d) A non leaf node have m children contains m-1 keys.
In B-tree each node contain set of keys and pointers. It is a shallow but wide data structure store millions and billions of items. Its internal nodes have variable number of child nodes with predefined range. Each time data is inserted and deleted from a node its child node changes. In order to maintain B-tree, non leaf nodes may be join or split. In B-tree records are stored at internal nodes as well as at leaf nodes which shows as in Fig1.

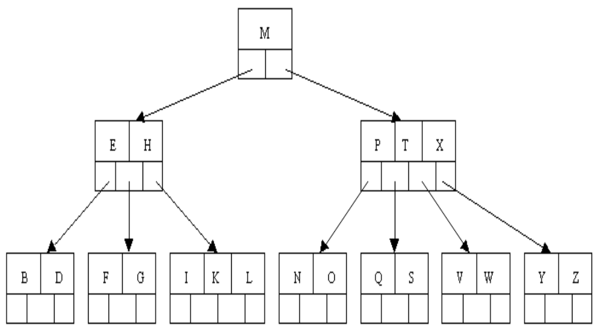


Fig1- B-TREE INDEX [3]

Mostly MySQL storage engine uses B-tree. There is no redundancy present is B-tee because every search key value stores only once. The main problem occur when internal nodes increase in B-tree then depth of tree also increases and performance of B-tree decreases. B-tree uses for linear data and its complexity is O(log n). Apple's file system, HFS+ and some Linux file systems use B-tree.

2.2 B +-TREES INDEX
B+-tree is a tree structure in which every path from root node to leaf node of tree is of same length [4]. It is a multilevel index and nodes in B+ tree are represented similar as B tree. There is no paper on B+-tree but a survey on B-tree also covers B+-tree. Data in B+-tree is stored only on leaf nodes in sorted order which is shown in Fig 2. The complexity of B+-tree is O (log n). In B+-tree there is no need of file organization to maintain the performance. If order of B+-tree is b(capacity of node) then number of children to a node (m) is b/2<=m<=b and number of keys in a node are at least b-1 and at most b. Main problem in B+-tree is that it adds space overhead with complexity O (n). To remove this space overhead a variant of B+-tree is used that is

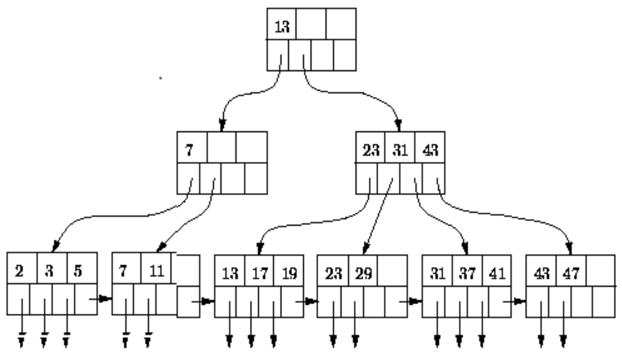
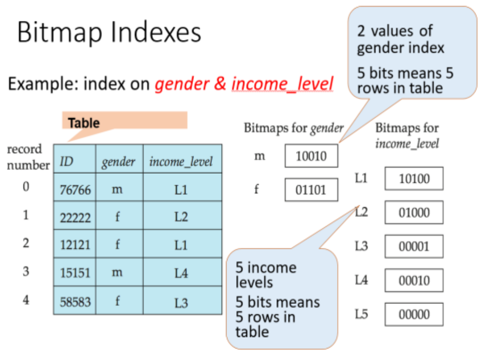


Fig 2- B+-TREE INDEX [5]

compact form. It uses the empty space of siblings before overflow occurs in a node. Some database system such as IBM DB2, Informix, Microsoft SQL Server and SQLite support B+-tree.
The performance factors in B –tree and B+-tree are index size and height. They affect performance in following way:
a) Longer the index size, less values can fit in a node, and hence more height of tree.
b) More height of tree, more disk access is needed.
c) More disk access results less performance.

3. BITMAP INDEX

For high cardinality columns in database previous indexing techniques are more suitable. But if these techniques are used for low cardinality columns in database then performance of query processing will degrade, so Bitmap index is used for low cardinality data. For example if a database contains gender as an attribute then only two values are possible-male and female. Moreover, this gives better performance as compared to B-tree when queries use combinations of AND/OR operators.


Bitmap Indexes
Example: index on gender & income_level
Table
2 values of gender index
5 bits means 5 rows in table
5 income levels
5 bits means 5 rows in table

record number	ID	gender	income_level
0	76766	m	L1
1	22222	f	L2
2	12121	f	L1
3	15151	m	L4
4	58583	f	L3

Bitmaps for gender	
m	10010
f	01101

Bitmaps for income_level	
L1	10100
L2	01000
L3	00001
L4	00010
L5	00000

Fig 3-BITMAP INDEX [7]

Bitmap is represented by bits and number of bits in bitmap index is equal to number of records in a table. Fig 3 represents two bitmap indexes for gender, one for male and second for female and whole table is accessed by L1, L2, L3, L4 and L5 bitmaps. Bitmap index offers important function for saving space and time as well as very good query performance in a large data warehouse [6].
Bitmap index is used for selection in database for example; a selection query like find women whose income is in range 15000-20000. Bitmap index also counts the number of tuples. For example how many
men have income level L4. In this, results are obtained from bitmap index without even accessing
the relation. The problem with bitmap index is that it can be used only when a relation column contains low distinct value.

4. HASHING

One disadvantage of indexing techniques is that it should access an index structure to find information, or should use binary search and these lead to more I/O operations [4]. So, hashing is used to avoid this. In this, a hash function is used for locating; inserting and delete records and maps search key value to buckets. Records are stored in buckets. These buckets suffer from overflow so use additional buckets. In hashing another problem is collision. There is various methods to handle collisions i.e. open addressing and second is chaining. In open addressing, linear probing and double hashing can be used. The two types of hashing are-
a) Static hashing- In this type of hashing there is no change in number of buckets. When a search key value is put, hash function always computes same address [8]. So, static hashing result more collisions.
b) Dynamic Hashing- In this type of hashing buckets vary at dynamically. When there is overflow occurs then more buckets can be added. Some techniques of dynamic hashing are;

i) Extendible hashing- Extendible hashing was described by Ronald Fagin in [9]. All modern file system use this type of hashing for example- Global File System, ZFS and the SpadFS file system. It is used for binary representation of hash values to access for directory with array size 2^d and d=global depth. When a disk block is full then it splits into two blocks. Here local depth store with each bucket and this tells about number of bits on which bucket contents are based. The main advantages of extensible hashing are that its performance does not degrade when file size grows and space overhead is negligible. Minor disadvantage is that it needs directory to be searched before access.

ii)Linear hashing- Linear hashing is a dynamic hash algorithm. It was proposed by Witold Litwin in [10], and later popularized by Paul Larson [11]. In this type of hashing only one bucket grow or shrink at a time. There are two main advantages of linear hashing over extensible hashing. One is it does not use a bucket directory and second is when overflow occurs it is not always the overflow bucket that is split. By creating a chain of pages in overflow bucket, overflow can be handled. The hash function in linear hashing change dynamically and at most two hash functions use in linear hashing at a given instant of time.

5. SPATIAL INDEXING

Techniques like B-tree and B+-tree, bitmap index all used for only one dimensional data. So there is a need of special indexing in which more than one dimensional data can be indexed. In spatial indexing spatial data is indexed and this data includes geographic data such as maps. The spatial indexing data answers the following query-

a) Nearness query: - It searches query like finding all hospitals that lie within a given distance. Nearest neighbor query can also be searched.

b) Range query:- It searches range query i.e. find all stationary shops within the geographical boundaries of given town.

This type of indexing also handles queries of insertion and union of regions. In spatial index there is a need of some techniques so that one can access spatial data in less time with accuracy. So some technique of spatial indexing are-

5.1 k-d TREE

k-d tree stores k-dimensional points. It is a space partitioning data structure. It was proposed by John Louis Bentley in [12]. In tree structure like binary tree each non leaf nodes are divide into two children in one dimension. Similarly k-d tree divides the space into two partitions, that’s why it is also called

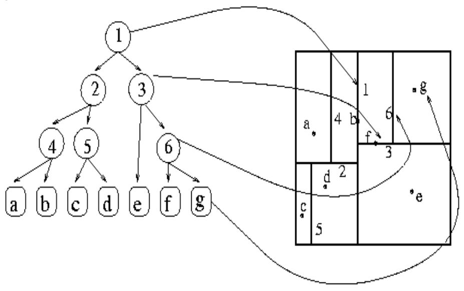


Fig 4- K-D TREE INDEXING [13]

multidimensional BST. Each level of k-d tree divides the space into two parts and this partitioning is performed along one dimension at the node at the highest level of the tree, along another dimension in nodes at the subsequent level and so on. This stops when a node contain less than a given maximum number of points [4].In Fig 4, k-d tree have maximum number of points in leaf nodes is one and data is stored at leaf nodes. The extension of k-d tree is k-d-B tree which allow multiple child nodes for internal nodes. This reduces the height of tree and these types of trees are mainly used for secondary storage.

5.2 QUADTREE

Quad tree was proposed by Raphael and J.L.Bentley in [14].The difference between k-d tree and quad tree is in division of space. The root node contains the entire space in quad tree and internal nodes divide the space into four equal size quadrants. So each non leaf node has four child nodes. The maximum points in a leaf node are fixed. In fig 5 the maximum point is set to one.

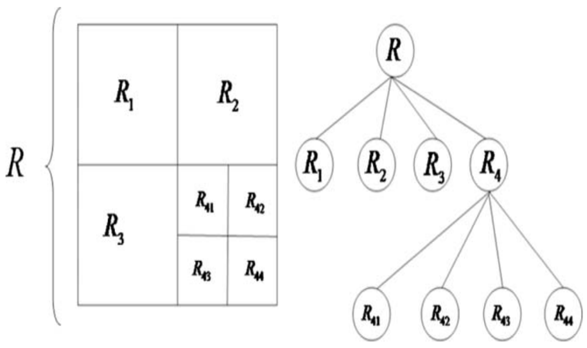


Fig 5-QUAD TREE [15]

K-d tree and quad trees are not balanced and they have many empty leaves so there performance degrades.

5.3 R TREE

It was proposed by Antonin Guttmann in [16]. Here R means rectangle. In R-tree space is divided into rectangles. R-tree helps in store spatial objects i.e. restaurant locations and it answers quickly to queries. In this type of tree nearby objects are grouped and these groups represented by MBR (minimum bounding rectangle) which helps in reduce the coverage. It is also a balanced tree and all leaf nodes are present at same level. Fig 6 represents R-tree index structure by making minimum bounding rectangles. Let M be the maximum number of entries fit in one node and m<=M/2 be the minimum number of entries in a node then R-tree satisfy following properties [16]-

a) Each leaf node index records are between m and M.

b) In each leaf node index is (I, tuple-identifier) where I is smallest rectangle that spatially includes n-dimensional data object.

c) Each non-leaf node children are between m and M unless it is root.

d) In each non leaf index is (I, child-pointer) where I is smallest rectangle that spatially includes the rectangles in child node.

e) In R-tree root node contains at least two children unless it is leaf.

f) All leaves in R-tree ate at same level.

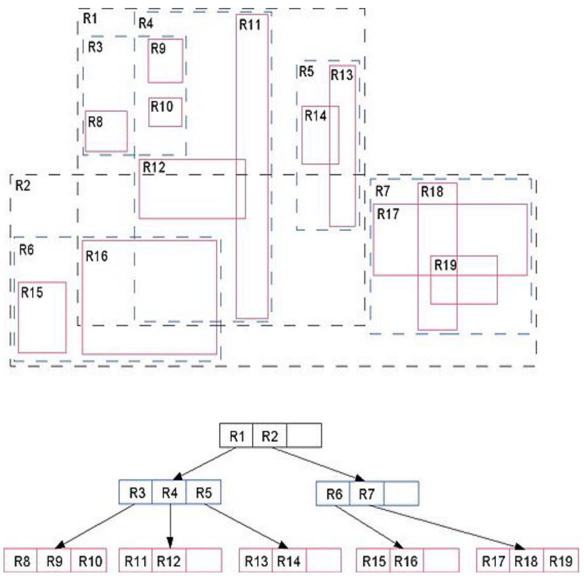


Fig 6-R-TREE INDEX STRUCTURE [17]

In spatial data two things are important –
a) Coverage- It is outlined as total space of all the rectangles related to the nodes of that level. Minimum coverage helps in reduces empty space.
b) Overlap- It is outlined as total space contained within two or more nodes. Minimal overlap reduces search paths to the leaf nodes.
So an efficient R-tree needs both minimum coverage and overlap. But in R-tree when a node falls in overlapping region then more than one searching path may be possible and the performance degrades. So there is a need of another technique in which zero overlapping can be possible.

5.4 R+-TREE
R+-tree is a variation of Guttman's R-tree(R+-tree) that avoids overlapping rectangles in intermediate nodes of the tree [18]. In this, the grouping of rectangles is different from R-tree. The MBR does not intersect one another in R+-tree. A single path is followed and saves the disk space when compare to R-tree. Fig 7 shows that there is no overlapping of rectangles.

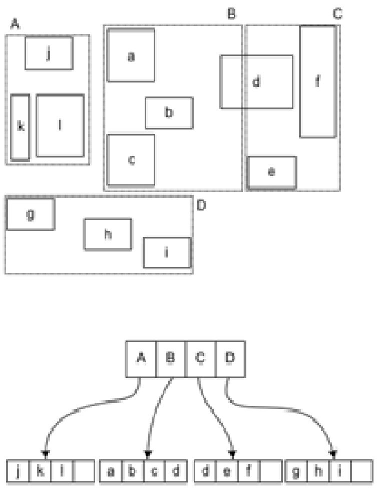


Fig 7-R+-TREE

R+-tree. But due to duplication of rectangles in R+-tree, its structure sometimes can be large.

5.5 R*-TREE [19]
This type of tree also stores point and spatial data similar to R-tree. R*-tree is very attractive due to-
a) It supports both point and spatial data at the same time.
b) Its implementation cost is slightly higher than R tree.
R*-Tree also reduces both overlap and coverage by two ways:-
i) By using revised node split algorithm.
ii)Due to forced reinsertion at node overflow.
In R*- tree deletion and reinsertion of data provides more appropriate location than its original and that's why it produce more well groups in nodes. But its complexity increases due to reinsertion.

6. CONCLUSION

This paper presents various information retrieval techniques by which any query can be searched and retrieved in less time with accuracy. Many indexing schemes are discussed in this paper and some of them are used in real world for query optimization. Some index structure support point query and one dimensional data and some of them support range query and multi dimensional data. B-tree, B+-tree supports point query but R-tree and its variants supports range query and multidimensional data. R*-tree supports both point and range query. Complexity can also be optimized by changing the existing index structure like in compact B+-tree. One can make new index structure by combining good properties of any two index structure like k-d-B tree combines properties of k-d tree and B-tree.

REFERENCES

1. wikipedia.org. [Online]. https://en.wikipedia.org/wiki/Information_retrieval
2. E.McCreight R. Bayer, "Organization and Maintenance of Large Ordered Indexes," Springer-Verlag, vol. 1, pp. 173-189, September 1972.
3. http://cis.stvincent.edu. [Online]. http://cis.stvincent.edu/html/tutorials/swd/btree/btree.html
4. Henry F.Korth,S.Sudarshan A. Silberschatz, Database System Concepts, 5th ed.: McGraw-Hill, 2006.
5. http://www.cburch.com. [Online]. http://www.cburch.com/cs/340/reading/btree/
6. Z.Zuping,H.I.Housien Z.Q.Abdulhadi, "Bitmap Index as Effective Indexing for Low Cardinality Column in Data Warehouse," International Journal of Computer Applications, vol. 68, no. 24, pp. 38-42, April 2013.
7. https://mjromeo81.com. [Online]. https://mjromeo81.com/2017/01/28/introduction-to-bitmap-indexes/
8. www.tutorialspoint.com. [Online]. https://www.tutorialspoint.com/dbms/dbms_hashing.htm
9. J. Nievergelt, N. Pippenger, H. Raymond Strong R. Fagin, "Extensible Hashing- A Fast Access Method for Dynamic Files," ACM Transactions on Database Systems , vol. 4, no. 3, pp. 315-344, September 1979.

10. W. Litwin, "Linear Hashing : A New Tool For File and Table Addressing," IEEE, pp. 212-223, 1980.
11. Wikipedia The Free Encyclopedia. [Online]. https://en.wikipedia.org/wiki/Linear_hashing
12. J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," ACM, vol. 18, no. 9, pp. 509-517, September 1975.
13. http://groups.csail.mit.edu. [Online]. http://groups.csail.mit.edu/graphics/classes/6.838/S98/meetings/m13/
14. J. L. Bentley R. A. Finkel, "Quad Trees : A Data structure for Retrieval on Composite Keys," Springer-Verlag, vol. 4, pp. 1-9, March 1974.
15. http://electronicimaging.spiedigitallibrary.org. [Online]. http://electronicimaging.spiedigitallibrary.org/article.aspx?articleid=1100865
16. A Guttman, "R-Trees:A Dynamic Index Structure For Spatial Searching," ACM SIGMOD, pp. 47-57, June 1984.
17. https://en.wikipedia.org. [Online]. https://en.wikipedia.org/wiki/File:R-tree.jpg
18. N.Roussopoulos, C.Faloutsos T.Sellis, "The R+-Tree:A Dynamic Index for Multi-Dimensional Objects," in Proceeding of 13th International Conference on Very Large Data Bases, 1987.
19. H.P. Kriegal, R. Schneider, B. Seeger N. Beckmann, "The R*-tree: an efficient and robust access method for points and rectangles ," Proc. ACM SIGMOD International Conference on Management of Data, pp. 322-331, 1990.