(IJRST) 2015, Vol. No. 5, Issue No. IV, Oct-Dec **e-ISSN: 2249-0604; p-ISSN: 2454-180X**

EFFICIENT METHOD TO FIND NEAREST NEIGHBOR WITH KEYWORDS IN MULTIDIMENSIONAL DATABASES

***Sandhya Venu, **Manju M Menon**
**M-Tech student CSE, Department Of Computer Science,*
Mohandas College of Engineering and Technology
***Assistant Professor in IT, Department Of Information Technology*
Mohandas College of Engineering and Technology

ABSTRACT

Nowadays the applications on spatial databases are tremendously increasing. The location based searches from spatial databases are widely used for both research and commercial purposes. The inputs of a spatial keyword query are a user defined location and number of keywords. The output is the object which satisfies both the location and keyword requirements. The most important part of a spatial keyword query is the geo-textual index. Extended researches are being carried out in field of geo-textual indices. This thesis proposes a parallel R-Tree construction method for geo textual indexing. Using this method we can reduce the construction time compared to the existing methods. We propose a new hybrid indexing mechanism namely Kd-B tree which has the combined advantage of kd tree and bitmap indexing. This reduces the time and space consumption remarkably. Based on the experiments conducted we can see that the time and space is considerably saved by constructing Kd-B tree.

Keywords-Spatial keyword query, Geo-textual indices, Spatial- inverted index.

I. INTRODUCTION

A spatial database is a database that is used to store and query data that represents objects defined in a geometric space. They are designed to manipulate both spatial information and the non-spatial attributes of that data. These data types are usually called spatial data types. Most spatial databases allow representing simple geometric objects such as points, lines and polygons. It provides special functions and indexes for querying and manipulating that data using something like Structured Query Language. While typical databases are designed to manage various numeric and character types of data, additional functionality needs to be added for databases to process spatial data types efficiently.

Many applications require finding objects closest to a specified location that contains a set of keywords. Conventional spatial queries focus on object's geometric properties only, such as whether a point is in a rectangle, or how close two points are from each other. We have seen some modern applications that call for the ability to select objects based on both of their

geometric coordinates and their associated texts. For example, it would be fairly useful if a search engine can be used to find the nearest hotel that offers internet, pool, spa all at the same time. Note that this is not the globally nearest hotel, but the nearest hotel among only those providing all the demanded facilities.

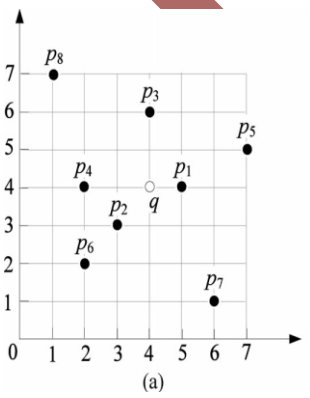
A spatial keyword query consists of a query area and a set of keywords. The answer is a list of objects ranked according to a combination of their distance to the query area and the relevance of their text description to the query keywords. Database systems use indexes to quickly look up values and the way that most databases index data is not optimal for spatial queries. Instead, spatial databases use a spatial index to speed up database operations. Spatial indices are used by spatial databases to optimize spatial queries. A number of geo-textual indices have been proposed. These indices usually combine a spatial index and a text index structure.

R-tree and inverted index are one of the best methods for spatial and textual indexing. The currently available method for geo textual indexing is spatial inverted index. Both spatial and textual information are used simultaneous for searching. R-trees are constructed for each inverted list. To improve the performance of the system, parallel R-tree construction is proposed by multitasking.

An indexing mechanism can be said to be an efficient one only if it consumes less time and space. The existing system did not fulfill this requirement fairly. So a new indexing mechanism called k dimensional tree with bitmap (Kd-B tree) is introduced. It is a hybrid indexing mechanism which combines kd tree with bitmap indices. Kd tree is used for spatal indexing and bitmap is used for textual indexing.

The rest of the paper is organized as follows. Section 2 defines the problem studied in this paper formally. Section 3 surveys the previous work related to ours. Section 4 gives the idea about proposed system. Section 5 shows the experimental results and analysis. Section 6 concludes the paper with a summary of our findings.

II. PROBLEM DEFINITIONS



p	W_p
p_1	$\{a, b\}$
p_2	$\{b, d\}$
p_3	$\{d\}$
p_4	$\{a, e\}$
p_5	$\{c, e\}$
p_6	$\{c, d, e\}$
p_7	$\{b, e\}$
p_8	$\{c, d\}$

(b)

Fig.2.1. (a) Shows the locations of points and (b) gives their associated texts.

(IJRST) 2015, Vol. No. 5, Issue No. IV, Oct-Dec **e-ISSN: 2249-0604; p-ISSN: 2454-180X**

Assume P be a set of 2D points. Each point p ϵ P, has a set of words Wp, which is called as the document of P. A spatial keyword query specifies a query point q and set of keywords, Wq. This Wq is referred to as the query documents. The spatial keyword query returns a point in Pq, which is closest to q.

$$P_q = \{p \in P \mid W_q \subseteq W_p\}.$$

III. RELATED WORKS

Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma[4] propose a hybrid index structures which combine the textual and spatial indexes into one index structure. Two basic designs introduced in are inverted file-R*-tree and R*-tree-inverted file.

R. Hariharan, B. Hore [3] propose efficient indexing strategies that process SK queries with AND semantics. This paper propose a novel indexing structure called KR*-tree that captures the joint distribution of keywords in space and significantly improves performance over existing index structures.

I.D. Felipe, V. Hristidis, and N. Rishe[6] present a method called Information Retrieval R-tree (IR2-Tree), which is a structure based on the R-tree. The IR2-Tree is an R-tree where a signature is added to each node v of the IR2-Tree to denote the textual content of all spatial objects in the subtree rooted at v.

D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen propose [7] WIR-tree, is also a variant of IR-tree. It aims at partitioning objects into multiple groups such that each group shares as few keywords as possible.

IV. PROPOSED SYSTEM

4.1 SI-index with parallel R-tree

In this work we use an efficient indexing mechanism called spatial inverted index. It is a variant of inverted index and it supports compressed coordinates. R-tree is used for spatial indexing and it is constructed for every inverted index. Distance browsing algorithm can be used to perform query processing in spatial inverted index (SI index).

We can divide the overall procedure into three modules.

Three different modules of the implementation part are

1. Inverted index construction
2. Parallel R-tree construction
3. NN search

For textual indexing, inverted indexes are the best method. Every unique word in the document has an inverted list. Each includes ids of the points, whose document has the corresponding word. These inverted lists are in a sorted order. The following figure shows the inverted list of each word in figure1.

Word	Inverted list
a	P1 , P4
b	P1,P2,P7
c	P5,P6,P8
d	P2,P3,P6,P8
e	P4,P5,P6,P7

Fig 4.1: Inverted index

Initially inverted lists are constructed for each facility in the database. Each list includes many points. Even though each point has two coordinates, convert them into only one so that gap keeping can be applied effectively. The tool needed is a space filling curve (SFC) such as Z-curve. SFC converts a multidimensional point to a 1D value such that if two points are close in the original space, their 1D value also tends to be similar. As dimensionality has been brought to 1, gap-keeping works nicely after sorting the 1D value.

Next step is to construct R-tree for each inverted list. The R-trees required for each inverted list is constructed parallel applying multitasking. The inverted list for each unique key word in the database is already constructed in the previous step. The list is sorted. This sorted list is divided based on block size B. A block is a subsequence of points in sorted list. Each block is treated as the leaf node of R-tree. Once leaf nodes are made, then the internal nodes are constructed. The division of these blocks is based on two conditions.

B is the block size and the no of points in each should lie between B and 2B-1.
The order of points in a block should match with the original order of sorted list.

Our aim is to minimize the size of resulting MBR as much as possible. There are several methods (eg: area, perimeter) to quantify the size of MBR. In our method we use area for this purpose. We calculate the cost of dividing scheme of L by adding areas of all MBRs. For convenience of notation, given an $1 \leq i \leq j \leq n$, for representing the cost of the division of the points Pi,Pi+1,...,Pj, we use C[i.j]. A[i, j] is used to represent the area of the MBR enclosing Pi,Pi+1,...,Pj. In all cases, we select smallest values as C[i,j].

$$C[i,j] = \min_{k=i+B-1}^{\min\{i+2B-2,j+1-B\}} (A[i,k] + C[k+1,j]).$$

When a spatial key word query comes, parallel search is carried out through the corresponding tree of each keyword. A point near to the query point is accessed from each tree. Keep counting how many numbers of same points is popped up continuously. Terminate the action by reporting the point once the count reaches the no of query key words.

4.2 KD-B TREE

Kd-B tree is the proposed hybrid geo-textual index structure that integrates location index and text index to process spatial keyword queries efficiently. In this proposed system Kd-tree tightly combined with bitmap index. Kd tree is used for spatial indexing and bitmap is used for textual indexing. The total no of nodes needed for creating a kd tree is equal to the total number of points in the database.

Following figure shows the architecture of the proposed system. Here kd tree is used as the indexing method for location information. For each node of the kd-tree, a bitmap index is created for indexing the text components of object contained in the node. Each node of Kd-B tree stores both spatial and textual information. This combination combines a spatial and textual index tightly such that both types of information can be used to prune the search space simultaneously during query processing.

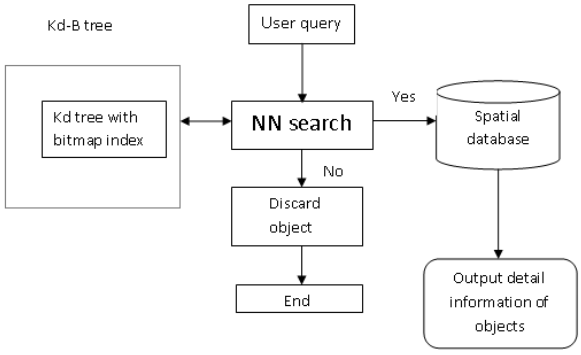


Fig 4.1:Proposed architecture

4.2.1TREE CREATION AND SEARCHING

For constructing the proposed hybrid geo textual indexing mechanism, initially all points in the database are selected. Then find the mid value among these points. The remaining points are so arranged that the values less than the mid values are placed in the left side and the values higher than the mid values are placed in the right side. The mid value is treated as the root node and the points with values less than the mid value is considered as left sub tree , and the points right to the mid value as right sub tree. Kd-tree is constructed using these points. For reducing the tree construction time the left and right sub trees are constructed parallelly by

using multitasking. Bitmap index is used for textual indexing and it uses a bit array for each bitmap index. For each point in the database, have corresponding facilities contained in it. A bitmap index is created for each point. This index indicates the presence or absence of the facilities in it. This is done by using a Boolean array for bitmap index .This array contains or 0, where 1 indicates the presence of the facility and 0 indicates the absence of the facility in the point. The corresponding bit map index of each point is included in the respective nodes of the kd tree. To do the pruning efficiently, a bit map summary is also embedded in every node except the leaf nodes. A logical bitwise OR operation is carried out on the bitmap indices of its child's for creating the bitmap summary.

A spatial keyword query contains spatial information and textual description. When a spatial keyword query comes, it checks whether the query point lies to the left or right to the current node. And also checks whether the keywords are present in the bitmap summary of the current node. The query traverses further down through the tree only if the bitmap summary contains all the query keywords. Otherwise it stops and starts searching on the other side of the tree.

V. EXPERIMENTAL RESULTS

5.1 RESULTS OF PARALLEL R-TREE CONSTRUCTION

The tree construction time mainly depend upon the no of nodes required for constructing the whole tree. There may be repetition in points in the inverted list. If all the points contain the same key words, all the points are included in the inverted list. It takes a lot of time to construct all the R-trees one by one. Since the R-trees are constructed in parallel, the construction time reduces drastically.

The following graph compares the Tree creation time and no of nodes relation for existing and proposed methods. From this we can conclude that as the no of nodes increases, the proposed system out performs the existing system.

(IJRST) 2015, Vol. No. 5, Issue No. IV, Oct-Dec e-ISSN: 2249-0604; p-ISSN: 2454-180X

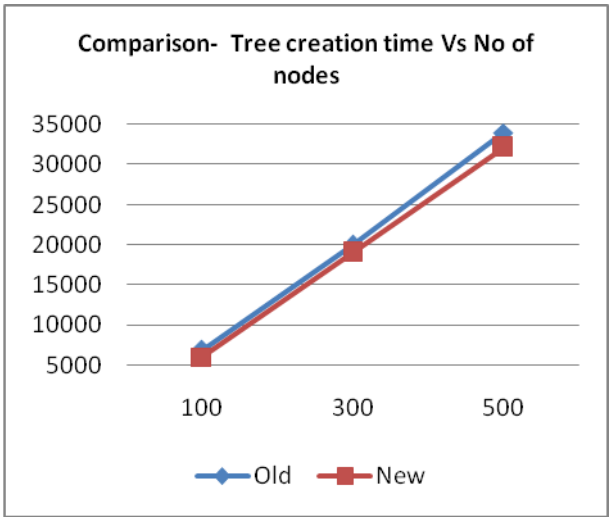


Fig-5.1-Tree creation time vs. no of nodes.

5.2 Results of Kd-B tree

Kd-B tree is a proposed indexing structure for spatial keyword queries. Kd-B tree is compared with SI-index based on time and space. The results show that Kd-B tree outperforms SI-index.

.

5.2.1 Index creation Time

The no of nodes required for constructing kd-B tree is equal to the total number of pointsin the database. In case of R-tree, even though the trees are constructed parallelly, the time required for this is more compared to Kd-B tree, because a point has to be duplicated once for every word in its text description. The analysis results comparing SI-index, SI-index with parallel R-tree construction and Kd-B tree is shown below.

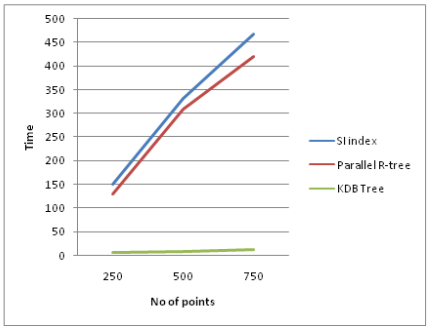


Figure 5.2: Performance evaluation based on tree creation time.

5.2.2 MEMORY COMPARISON

If a database contains N points, then only N no of nodes are required for constructing Kd-B tree. In the case of R-tree all points are contained in the leaf node. Internal nodes are carrying the location information of its child nodes. There are several such internal nodes, which consumes large memory. An experiment comparing SI-index and Kd-B tree has been conducted based on memory consumption and results are shown below.

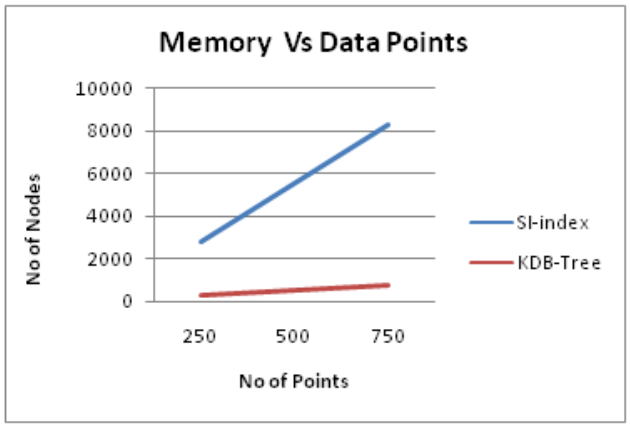


Fig-5.3-Memory Vs Data points

VI. CONCLUSION

This work provides a new geo textual indexing mechanism called Kd-B tree for spatial keyword querying. The best method among the currently available indexing mechanisms is Spatial Inverted index. It is a combined form of R-tree and inverted index. The quality of NN search mainly depends on the time and space complexity of indexing. This work proposes a parallel R-Tree construction method for geo textual indexing. Using this method we can reduce the construction time compared to the existing method in which R-Trees are constructed one after another. We propose a new hybrid indexing mechanism namely Kd-B tree which has the combined advantage of kd tree and bitmap indexing. This reduces the time and space consumption remarkably. Based on the experiments conducted we can see that the time and space are considerably saved by constructing Kd-B tree. Also it can be noted that as the number of nodes increases, the proposed method improves its performance.

REFERENCES

[1] A Guttman ‘R-trees a dynamic index structure for spatial searching’, Proc ACM SIGMOD Int Conf on Management of Data, 47-57, 1984

[2] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, “The R*-tree: An Efficient and Robust Access Method for Points and Rectangles,”Proc. ACM SIGMOD Int’l Conf. Management of Data,pp. 322-331, 1990.

[3] R. Hariharan, B. Hore, C. Li, and S. Mehrotra. Processing spatial-keyword (sk) queries in geographic information retrieval (gir systems. In SSDBM, page 16, 2007.

[4] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma. Hybrid index structures for location-based web search. In CIKM, pages 155–162, 2005

[5] A. Cary, O. Wolfson, and N. Rishe. Efficient and scalable method for processing top-k spatial boolean queries. In SSDBM, pages 87–95, 2010.

[6] I.D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In ICDE, pages 656–665, 2008.

[7] D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen. Joint top-k spatial keyword query processing. *IEEE TKDE*, 24(10):1889–1903, 2012.

[8] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørv°ag. Efficient processing of top-k spatial keyword queries. In *SSTD*, pages 205–222,2011.

[9] C. Faloutsos and S. Christodoulakis, "Signature Files: An Access Method for Documents and Its Analytical Performance Evaluation," ACM Trans. Information Systems, vol. 2, no. 4, pp. 267-288, 1984.

[10] D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa. Keyword search in spatial databases: Towards searching by document. In *ICDE*, pages 688–699, 2009.

11] A. Khodaei, C. Shahabi, and C. Li. Hybrid indexing and seamless ranking of spatial and textual features of web documents. In *DEXA*, pages 450–466, 2010.

[12] S. Vaid, C. B. Jones, H. Joho, and M. Sanderson. Spatio-textual indexing for geographical search on the web. In *SSTD*, pages 218–235,2005.

[13] Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *SIGMOD*, pages 277–288, 2006.

[14] Yufei Tao and Cheng Sheng, "Fast Nearest Neighbor Search with Keywords. ", In IEEE ,TKDE, VOL. 26, NO. 4, APRIL 2014.