A Survey Paper on Variants of R-tree Data Structure

Bhoomika Panchal, Vinit Kumar Gupta
Hasmukh Goswami College of Engineering, Ahmedabad, India

***Abstract*- R-tree is one of the two basic index structures for spatial databases with various variants evolved of it. Depends upon the applications requirement individual or combination of variants is utilize to achieve performance. Various performance optimization parameters are index structure, query support, data type support, etc. Overlapping is major disadvantage of R-tree which degrades search performance. Though here we present variant of R-tree which will reduces the search complexity.**

***Index Terms*- R-tree, variants, index structure, search complexity, query performance, overlapping, Minimum bounding rectangles (MBRs)**

I. INTRODUCTION

Spatial data can commonly be found in diverse applications including Cartography, Computer Aided Design, Computer Vision, Robotics and many others [1]. The amount of available spatial data is of an ever increasing storage & retrieval. Index is a data structure that provides linear time lookup. An index is a data structure that is used for improving performance of Search Complexity. To get single value from the data store containing N objects, the number of operation in worst case is $\Omega(n)$. So, when data store contains millions of object, and we want to get value from them, Searching is very useful and common operation. So, Search complexity should be minimized. Index is one of the data structures that can be used to improve this.

Main idea of indexing is to optimize the query search. For any type of search or to get information we perform a query and query is processed by database system or search engine internally process query [11] on database of different content. To support multi dimensional indexing, Multiway trees are introduced. A Tree can have more than two children is called multiway Tree. Multiway Trees are much powerful for search operation. Examples of multiway tree are R+-tree, R*-tree, Hilbert R-tree, QR+-tree, BR-tree.

The paper is organized as follows in section 2 structure of R-tree are described. Section 3 variants of R-tree are discussed. Section 4 comparison between different index structures based on their performance and their application. In section 5 conclusion and future scope.

II. STRUCTURE OF R-TREE

R-tree is dynamic index structure which support multidimensional data to be stored. It was proposed in 1984 by Antonin Guttman[14]. R-tree is high balanced structure tree. It performs the operations like insertion, deletion, searching.

It overcomes the limitation of B Tree, which able to store only single dimension. Whereas, R-Tree supports multi dimensional data. However, it has the limitation of too many overlapping regions and too many disk accesses because of overlapping regions. To overcome this drawback, many variants of R-trees were proposed like R+-Tree [2], R*-Tree [3], Hilbert R-Tree [14], X-Tree [12], BR-Tree [13].

R-Tree [1] has proposed three different splitting approaches: Linear Split, Quadratic Split and Exhaustive Split. But among these, it was proved that Quadratic Split is best. Figure show structure of R-tree and relation exist between its rectangles [1].

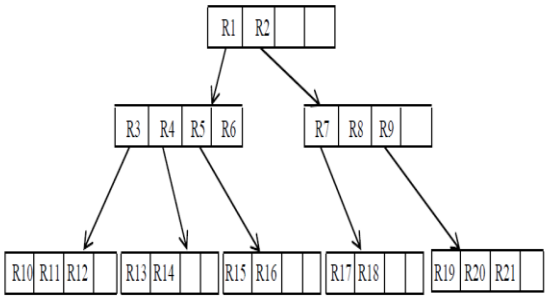

Fig 2.1, Structure of R-tree [1]

Here we perform the example of search operation. For Search operation, we had given a search Point N. So, we start at root node and locate all child nodes whose rectangle B intersects C. Search in the sub trees of those child nodes. And when we get to the leaves, return particular entry.
Figure 2.2 shows an example of R-Tree search operation.

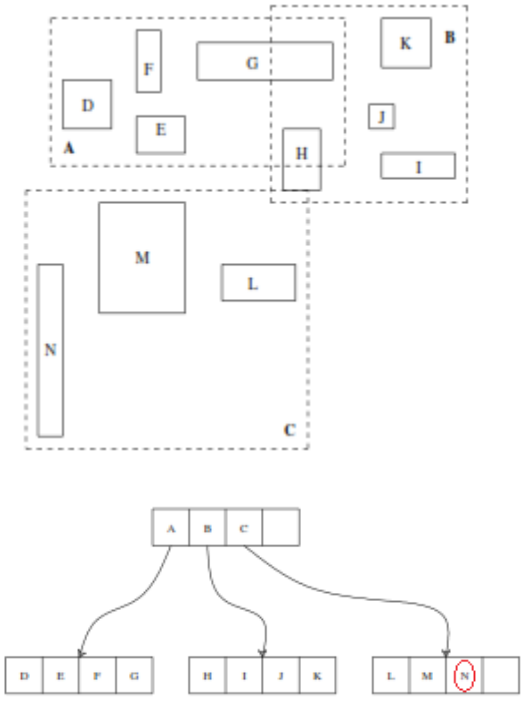

Fig 2.2, Example of R-tree for searching [2]

III. VARIANTS OF R-TREE

A. R+-tree

R+ tree is a variant of the R-tree proposed by Timos K. Sellis, Nick Roussopoulos, Christos Faloutsos in 1987 [2]. It differs from R-tree, where an entry of internal nodes does not overlap with each other. For Search performance, it has a improved approach compare to R-Tree. It takes fewer paths to find a single entry compare to R-Tree.

In this data structure retrieval performance could be improve because it avoid the visiting multiple path during point search query [5]. Figure 3.1 shows the structure of R+ tree. Which demonstrate object d is store in two leaf nodes B and C. so, due to clipping there is no overlapping done at same level of tree.

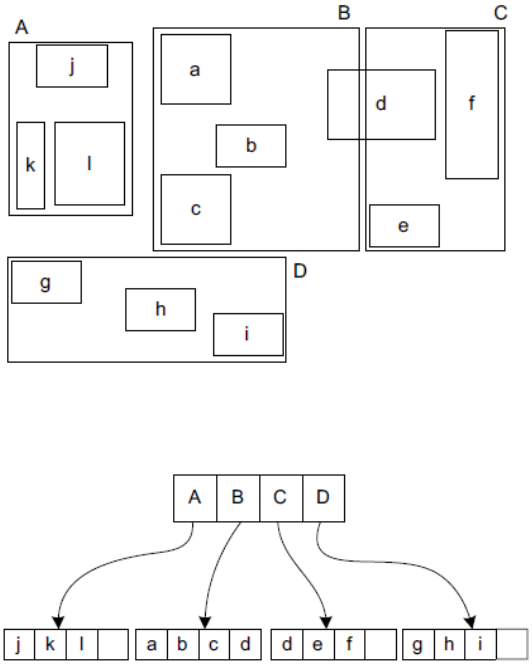

Fig 3.1, Example of R+tree [2]

B. R-tree*

R*-tree [3] is also variant of R-tree its results shows that it performs better than R-Tree and R+-Tree in query processing. It has mainly four advantages: (i) Minimizes the area covered by directory rectangle (ii) Minimizes the overlap between directory rectangles (iii) Minimize the margin of directory rectangle, and (iv)Higher storage utilization. It's one of the costly operation is Reinsertion, which removes the number of entries and reinsert it. However, it reduces the split operation.
An Analysis of R* Tree shows that It is not adequate for indexing high dimensional [7] data sets. The R*-tree which incorporates a combined optimization of area, margin and overlap of each enclosing rectangle in the directory.

C. Hilbert R-tree

It had proposed a new R tree structure that outperforms all the older ones. The heart of the idea is to facilitate the deferred splitting approach in R-trees. This is done by proposing an ordering on the R-tree nodes. This ordering has to be 'good', in the

Data Structure	Characteristics	Limitations	Complexity	Application
R Tree [1]	• Multi Dimensional Indexing. • Handles Spatial Data. • Height balanced Tree. • Used for Range Query.	• Multiple paths from root to leaf level, which degrades the search performance. • Few Large rectangles increase the degree of overlap.	• Not utilize space more efficiently. • Not have worst case time complexity.	• Real world application like navigation system etc [10].
R+ Tree [2]	• Overlapping of internal nodes is avoided. • Speeds up the search.	• Data Redundancy. • Since rectangles are duplicated, R+ Trees are larger than R Tree for same data set. • Construction and maintenance of R+ Tree is complex than all other variants.	• Non overlapping data utilize space efficiently than R-tree.	• Multidimensional data object.
R* Tree [3]	• Minimization of area covered by each MBR. • Minimization of overlap between MBR. • Minimization of MBR margins. • Maximization of storage utilization.	• For Insertion, R* Tree is more complex than Linear Split of R Tree. • Implementation Cost is slightly higher than R Tree. • No concept for moving object. • Not designed for spatial-temporal objects.	• Implementation cost is more than other R-tree. • Variants but robust in data distribution than other ugly structures.	• Application with data in form of points and rectangles.
Hilbert R Tree [4][14]	• Instead of Split, Add to Siblings. • Storage is increased. • Unnecessary Split is avoided. • Search cost give 28% saving above R*-tree.	• Preprocessing is required for sorting Hilbert values.	• Search cost give 28% saving above R*-tree.	• Cartography, Computer Aided Design (CAD), computer vision and robotics etc [14].

Table 1.Comparision of R-tree variants Index Structure

sense that it should group 'similar' data rectangles to gather, to minimize the area and perimeter of the resulting minimum bounding rectangles (MBRs) [14].

It has chosen the so-called '2D-c' method, which sorts rectangles according to the Hilbert value of the center of the rectangles. Given the ordering, every node has a well-defined set of sibling nodes; thus, we can use deferred splitting. By adjusting the split policy, the Hilbert R-tree can achieve as high utilization as desired.

The structure of Hilbert R-tree is based on Hilbert space-filling curve. In the figure 3.2 shows different three space-filling curves.

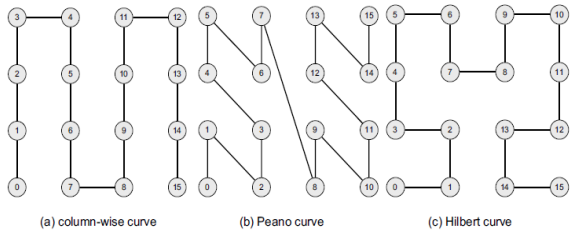

Fig 3.2, Structure of three space-filling curve [5]

IV. COMPARISION OF R-TREE VARIANTS INDEX STRUCTURES

A. Query type

There is different type of queries

I. Point Queries: Given a point in the space, find all objects that contain it.

II. Region Queries: Given a region (query window), find all objects that intersect it.

B. Complexity

Each index structure is compare base on two parameter of complexity which is time and space. Though, all variant contain different terms and condition according to that the complexity id define.

C. Application

All variants are used for some specific application which is used for improve performance.

V. CONCLUSION

R-tree data structure is used for multidimensional data. So, some of the pitfalls of it is overcome by its variants which are R+-tree, R*-tree, Hilbert R-tree, etc. this all data structure overcome the search performance by sorting or managing disaster data which is in different region.

In future, this data structure can be improved by reducing the search complexity as well as overcome the wastage of memory.

ACKNOWLEDGEMENT

I would like to express my deep sense of gratitude to my guide, Asst Prof.VinitKumar Gupta for his valuable guidance and useful suggestions. I would like to thank Asst Prof. Indr jeet Rajput also for his precious suggestion.

REFERENCES

[1] Guttman, Antonin. R-trees: a dynamic index structure for spatial searching. Vol. 14. No. 2. ACM, 1984.

[2] Sellis, Timos, Nick Roussopoulos, and Christos Faloutsos. "The R+-tree: A dynamic index for multi-dimensional objects." (1987).

[3] Beckmann, Norbert, et al. The R*-tree: an efficient and robust access method for points and rectangles. Vol. 19. No. 2. ACM, 1990.

[4] Patel, Parth, and Deepak Garg. "Comparison of Advance Tree Data Structures." arXiv preprint arXiv:1209.6495 (2012).

[5] Manolopoulos, Yannis, et al. R-Trees: Theory and Applications: Theory and Applications. Springer, 2010.

[6] Prof. A. Kemper, R tree, University of Passau, Sebastian Kabisch, 18 June 2003, Moderation by Thomas Bernreiter

[7] Multi Dimensional Indexes, www.csd.uoc.gr/~hy460/pdf/005.pdf

[8] Zhu, Qing, Jun Gong, and Yeting Zhang. "An efficient 3D R-tree spatial index method for virtual geographic environments." ISPRS Journal of Photogrammetry and Remote Sensing 62.3 (2007): 217-224, Elsevier, 2007

[9] Xiong, Xiaopeng, and Walid G. Aref. "R-trees with update memos." Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on. IEEE, 2006.

[10] Wu, Chin-Hsien, Li-Pin Chang, and Tei-Wei Kuo. "An efficient R-tree implementation over flash-memory storage systems." Proceedings of the 11th ACM international symposium on

Advances in geographic information systems. ACM, 2003.

[11] Tao, Yufei, and Dimitris Papadias. "The mv3r-tree: A spatio-temporal access method for timestamp and interval queries." (2001).

[12] Berchtold, Stefan, Daniel A. Keim, and Hans-Peter Kriegel. "The X-tree: An index structure for high-dimensional data." Readings in multimedia computing and networking 451 (2001).

[13] Hua, Yu, Bin Xiao, and Jianping Wang. "BR-Tree: a scalable prototype for supporting multiple queries of multidimensional data." Computers, IEEE Transactions on 58.12 (2009): 1585-1598.

[14] Kamel, Ibrahim, and Christos Faloutsos. "Hilbert R-tree: An improved R-tree using fractals." (1993).