IOSR Journal of Engineering (IOSR JEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
PP 07-09

An Overview of R-Trees and Its Variants in Spatial Data Searching

Bhagyashree Ambulkar[1], Rajeshree Ambulkar[2]
Asst. Professor, Department of Computer Science, G. H. Raisoni Institute of Information Technology, Nagpur
Asst. Professor, Department of Information Technology, Priyadarshini College of Engineering, Nagpur

Abstract: *Indexing plays a vital role in query processing for fast retrieval of results. Queries using indexing executes fastly and provide noteworthy performance enhancement. The R-tree allows to index multidimensional objects. It arranges data in a treelike structure, with bounding boxes at the nodes. Bounding boxes indicate the next level of the data that is connected to the sub tree underneath. The R-Tree variants are categorized as the original R-Tree, hybrid of R-Tree and an extension of R-Tree to support several applications. In this paper, we overview the R-Tree and its variants R^+ Tree, R^* Tree, Hilbert R-tree, Spatio-temporal indexing by RST-trees spatial databases are presented.*

Keywords: *R-Tree, R-Tree variants, Spatial Index, Multidimensional data*

I. Introduction

Spatial Data management is one of the significant research field of computer science.A spatial database is improved to store and query spatial data representing objects which are defined in a geometric space [1].The applications which use these type of data enforce storage of large volume of data. So, efficient handling of these type of data is essential. Traditional indexing technique is useful with one dimensional data but it is difficult to use same technique for multi-dimensional data. So there is need of effective technique to handle the multidimensional data. In order to support spatial objects in a database system several issues should be taken into consideration such as: spatial data models, indexing mechanisms, efficient query processing, cost models.[2] One of the most significant methods to access the data effectively and efficiently through indexing is the R-tree structure proposed by Guttman in 1984. Spatial indexing plays an important role in applications like Geographical Information System, Computer Aided Designing and Multimedia Information System, etc.
Many R-Tree variants has evolved for better performance in variety of aspect such as query retrieval, insertion cost, application specific and so on. R-Tree is the most widely used spatial index structure for its implementation simplicity and performance efficiency.

II. R-Tree Index Structure

R-Tree index structure is similar to B-Tree index structure even though the data stored in the index is somewhat different. R-Tree is the height-balanced tree (all leaf nodes are at same level).The R-tree organizes data in a tree-like structure called an R-tree index. The index uses a *bounding box,* which is a rectangular shape that completely contains the bounded objects. Bounding boxes can surround data objects or other bounding boxes. Bounding box has a set of coordinates of equal dimension as the bounded object. It is useful to choose the small size of bounding box for performance reason. The minimum bounding box is always most efficient one.

In the R-Tree index structure, the objects are represented as Minimum Bounding Rectangles (MBRs). R-Tree comprisesroot node, non-leaf and leaf nodes. The nodes in R-Tree are bounded in a maximum and minimum limit. Figure 1 shows data objects enclosed by set of bounding boxes and other bounding boxes.

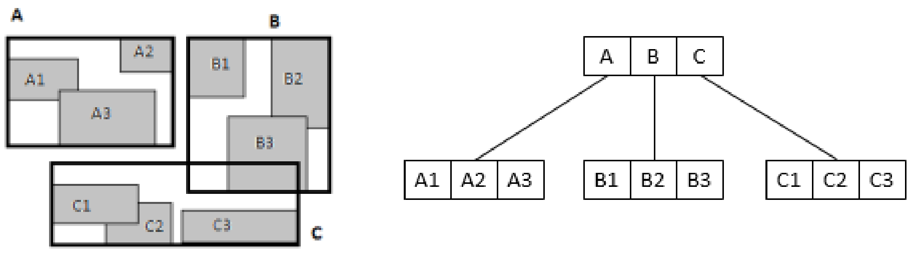

Figure1: R-Tree Indexing Structure Example

Each leaf node contain unique identifier, minimum bounding rectangle (MBR) that covers the object and pointer to the actual database record. The non-leaf node has the MBR which covers lower level of children and pointer to the children. For the valid R-Tree structure, following properties must be satisfied:

a. A node in R-Tree can hold maximum limit M and lower limit m such that $2 \le m \le M/2$.
b. Each leaf node contains between m and M index records unless it is the root.
c. Each leaf node should contain between m to M children otherwise it is the root.
d. The root node should have at least two children unless it is a leaf.
e. All leaf nodes must appear on the same level.

In the R-Tree, data can be inserted, deleted and updated. When a node extents the maximum capacity, the node is again divided in such way that each node having the number of objects lies between the minimum and maximum limit.R-tree based indexing allows concurrent searching and updating operations. Also, there is no need of periodic reconstruction.

DML Operations on R-Tree:

Searching: Searching a record in R-Tree within a given rectangle is most frequently used query. The tree is always start navigation form root node to leaf node. The non-leaf records which overlap the given rectangle are selected during navigation for further search. The searching is carried out at each level till the leaf nodes are reached. The leaf nodes which overlap with the given rectangle are the qualified records. The index records which found in an overlapping rectangle are the qualified records.

While inserting records to the R-Tree, the properties of the R-Tree must satisfied before and after the insertion.

Insertion: During insertion operation, new index records are added to the leaves. It is fine if the number of entries are in between minimum and maximum bound. If number of entries extended the maximum limit, then nodes are split in such way that the entries in each node lies between minimum and maximum limit and the generated changes propagate up the tree.

Deletion: Deleting a record also should satisfy the above mentioned properties for valid R-Tree index before and after deletion. To delete a particular record, first search algorithm is used to find the record then after that record is deleted and changes are then circulated to the root. There are chances of node having less number of entries then minimum limit. In this situation, the nodes are merged to satisfy the properties.

Updating: Updating a record involves deletion of the old values and insertion of new values.

III. R-Tree Variants

In all R-tree variants, tree traversals for any kind of operations are executed exactly the same way as the original R-tree. Basically, the variations of R-trees differ in the way they perform splits during insertions by considering different minimization criteria instead of the sum of the areas of the two resulting nodes [1].

R+-Tree

The R+-Tree variant is used to increase search performance particularly for point queries. It reduces the search cost by reducing the number of disk accesses needed to get the result [6]. Overlapping and coverage region are the two important concerns for search performance. The main concept of R+-Tree is to have zero overlap. If any data rectangle is overlapped with another data rectangle at lower level then it is decomposed into a collection of non-overlapping sub-rectangles whose union makes up the original rectangle. In insertion operation, the rectangle is added to more than one node by splitting into sub-rectangles during overlapping. The deletion operation also requires to search the all sub rectangles and delete all of them. Due to decomposition, overlapping is avoided and the multi-path search made easy here. Splitting should be circulated both up and down direction for the constancy of the tree.

R*-Tree

The R*-tree [Beck90] is an improved version of the R-tree that takes into consideration a more complex criterion for the distribution of bounding rectangles through the nodes of the tree.R*-Tree is prevailing performance wise structure that is many use for performance comparison. The R*-Tree disobey the limitation for the number of pairs per node and also follow forced reinsertion.In forced reinsertion technique, when node is overflows, p entries are extracted and reinserted in the tree. R*-trees has one more features is it takes into account additional criteria except the minimization of the Sum of the areas of the produced MBRs.It helps R*-tress to increase its performance.These criteria are the minimization of the overlapping between MBRs at the same level, and also minimization of the perimeter of the produced MBRs.

The insertion operation is not for free as it is CPU demanding since it applies a plane-sweep algorithm.

Hilbert R-tree

The Hilbert R-tree it is a hybrid structure by combination of R-trees and B+ -trees. Hilbert R-tree is aB+ -tree with geometrical objects being characterized by the Hilbert value of their centroid. Thus,Leaves and internal nodes are augmented by the largest Hilbert value of their contained objects or their descendants.For insertion of new object at each level the Hilbert values of the alternative node are checked and thesmallest one that is larger than Hilbert value of object under the insertion is followed.In addition, another heuristic used in case of overflow by Hilbert R-trees is the redistribution of objects in sibling nodes [2]. In other words, in such a case up to s siblings are checked in order to find available space and absorb the new object.Hilbert R trees were proven to be the best dynamic version of R-tress and the variant is vulnerable performance wise to large object.

Spatio-temporal indexing by RST-trees

Spatio-temporal indexing by RST-trees usesa Spatio-temporal indexing and is based on the R*-tree.Spatio-temporal data models entities having spatial properties that vary as a function of time .The best example is a cadastral system where it is required to record not only the current state of the land and history of all changes of boundaries of the land parcels. Such a repository would be able to answer queries that contain both spatial and time dimensions: "What was the information recorded in 1999 about the pieces of land that made the object of a legal dispute between person A and person B in 1980?"
This query can be projected onto four dimensions: two time dimensions (the required information must have been valid in 1980 and must have been current in the database in 1999) and two spatial dimensions (the object of a legal dispute might be the intersection between two non-disjoint two-dimensional regions owned by the persons A and B). Processing such a query might involve two separate indexes: one for the spatial data and one for the temporal data. However, an RST-tree can accommodate both types. The spatial data is modeled in a way identical to the R*-tree.

IV. Conclusion

R-Tree is the most broadly used spatial index structure serving various applications which uses multi-dimensional data. Many variants of R-Tree have been proposed so that some aspects of indexing may be improved or done efficiently. This paper presents an overview of the index structures of R-Tree and its variants R+ Tree, R* Tree, Hilbert R-tree, Spatio-temporal indexing by RST-trees. The R-Tree variants have been proposed to improve the indexing or done efficiently.

References

[1]. https://www.tutorialspoint.com/Spatial-Databases
[2]. "R-trees Have Grown Everywhere", YANNIS MANOLOPOULOS, ALEXANDROS NANOPOULOS, APOSTOLOS N. PAPADOPOULOS Aristotle University of Thessaloniki, Greece and YANNIS THEODORIDIS University of Piraeus, Greece
[3]. R-Trees – A Dynamic Index Structure for Spatial Searching by Marios Hadjieleftheriou, Yannis Manolopoulos, Yannis Theodoridis, Vassilis J. Tsotras
[4]. A State-of-Art in R-Tree Variants for Spatial Indexing
[5]. https://www.ibm.com/support/knowledgecenter/en/SSGU8G_12.1.0/com.ibm.spatial.doc/ids_spat_071.htm
[6]. https://pdfs.semanticscholar.org/d237/bbb195007c1f9c91395488df0d62641f0e40.pdf
[7]. Timos K. Sellis, Nick Roussopoulos and Christos Faloutsos, ―The R+-Tree: A Dynamic Index for Multi-Dimensional Objects‖, *Proceedings of 13th International Conference on Very Large Data Bases,* pp.507-518, 1987.