Applying R Trees In Non Spatial Multidimensional Databases

Jinka Sravana , Suba. S

M.Tech, Dept of Computer Science, Vasavi College of Engineering, Hyderabad, India.
Assistant Professor, Dept of Computer Science, Vasavi College of Engineering, Hyderabad, India.
Email: jinkasravana@gmail.com, subasuseela@gmail.com

ABSTRACT: In this paper we propose that R trees which is a spatial access method has the ability for indexing on multiple dimensions. The traditional Database Management Systems (DBMSs) like Oracle, Mysql do not perform well when multidimensional Non Spatial data is given since these DBMSs follows one dimensional indexing at different levels and it effects the retrieval time of queries poorly has sequential scan of the database is done. R Tree families are the most efficient among the indexing structures in data accessing methods in case of spatial data The multicolumn index structure available in the present-day DBMSs follows a single dimension indexing at multiple levels. Spatial access methods like R-Trees are capable of indexing on multiple dimensions R trees have a better performance in case of spatial data. Retrieval time of queries can be improved considerably compared traditional DBMSs methods.

Keywords: R trees, Non spatial data, multidimensional databases, retrieval, Database Management Systems

1. INTRODUCTION

Day-by-day the data in the databases increases rapidly, due to this reason organizing the spatial queries for a spatial access method in Database Management Systems (DBMSs) had became an important task. B trees cannot handle these queries efficiently when two or more of these attributes are to be integrated to form an index structure of a relation. Since B trees follows single dimensional indexing at multiple levels. R trees which is a spatial access methods has the capability of indexing on multiple levels. R tree families are the most efficient among the indexing structures in data accessing methods in case of spatial data. Making use of R trees in non spatial data will help in multidimensional indexing of the data. B-Trees cannot store new types of data. Specifically people wanted to store geometrical data and multi-dimensional data. The R-Tree provided a way that is they store such type of data

2. LITERATURE

In this section we discuss about spatial data, non spatial data, R tree, its structure and advantages of R tree compared to B trees.

2.1 SPATIAL DATA

The data that indicates the Earth location (latitude and longitude, or height and depth) of these rendered objects is the spatial data common example of spatial data can be seen in a road map. A road map is a two-dimensional object that contains points, lines, and polygons that can represent cities, roads, and political boundaries such as states or provinces. Using these attributes can make it easier to answer queries like "find all tanks whose speed is 10 km and oriented to north" or "find all enemy tanks in a certain region"

2.2 NON SPATIAL DATA

Nonsapitaldata(also called ***attribute*** or ***characteristic*** data) is that information which is independent of all geometric considerations. For example, a person's height, mass, and age are non-spatial data because they are independent of the persons location. There are different types of spatial indexing methods , they are:

- Grid
- Z-order
- Quad tree
- Oct tree
- UB- tree
- R-tree

2.3 R TREE

R tree which is a spatial access method and is a height balanced tree like B tree. In R trees spatial data objects are represented by minimum bounding rectangles(MBR) see figure a. R trees was implemented for accessing the multidimensional data ,it can also be used for single dimensional databases. R tree mainly depends on the splitting of the node. In R trees there are two different types of splitting can be done. We can use either quadratic split or linear split.

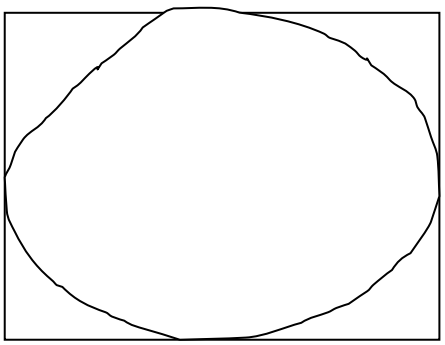

Figure a Representation of MBR

2.3 PROPERTIES OF R TREES

- The number of nodes in the tree is between m and M where M is the maximum number of nodes.
- The M value may differ for leaf and non leaf nodes
- All leaf nodes are at the same level.
- R-Tree is a height-balanced tree similar to a B-Tree.
- Leaf nodes contain pointers to data objects.

- Structure is designed in such a way that a spatial search requires visiting only a small number of nodes.
- R-Trees can organize any-dimensional data by representing the data by a minimum bounding box.
- A node can have many objects in it.
- The leaves point to the actual objects.

2.4 ADVANTAGES OF R TREES

- The retrieval time of given queries can be faster compared to B trees.
- The reason behind this is R trees are a multiple dimension where as B trees are one dimension.

3. ALGORITHM

3.1 INSERTION

The R tree insertion algorithm is explained as follows:

1. Algorithm Insert(type Entry Ei, type Node RNi)Insert a new Entry Ei in an R-tree with root RNi.
2. Traverse the tree from root RNi to the leaf to correct leaf.
3. In case ties select the node who's MBR (minimum bounding rectangle) has the minimum area.
4. If the selected leaf Li can accommodate Ei
5. Insert Ei into Li
6. Update all MBRs in the path from the root to Li, so that all of them cover Ei.mbr
7. Else if Li is already full
8. Let Si be set consisting of all Li entries and the new entry Ei select as seeds belonging to e1i, e2i.
9. Let Si be the set consisting of all Li entries and the new entry Ei
10. Select as seeds two entries e1i,e2i belongs to si where the distance between e1i and e2i is the maximum among all the pairs of entries from Si
11. From the two nodes L1i, L2iwhere the first contains e1i, and the second contains e2i.
12. Examine the remaining members of Ei one by one and assign them to L1i or L2i, depending on which of the MBRs (minimum bounding rectangle) of these nodes will require the minimum area enlargement so that it can cover this entry.
13. If a tie occurs
14. Assign the entry to the node whose MBR has the smaller area.
15. End if.
16. If a tie occurs again
17. Assign the entry to the node that contains the smaller number of entries.
18. End if.
19. Update the MBRs of nodes that are in the path from root to Li, so as to cover L1i and accommodate L2.
20. Perform splits at the upper levels if necessary.
21. Suppose if the root has to be split in such situation, create a new root.
22. Increase the height of the tree by one
23. End if.

3.2 SEARCH :

The search algorithm in R Tree is some s what similar to t the B Tree search method. In this it follows the certain method that is if an R Tree is given let the root node be k, then let us find all records whose rectangles overlap a given search rectangle P. Let us denote an entry in the node as M(MI,MP), where MI denotes the smallest rectangle bounding the sub-tree where as MP is the pointer to the sub tree. SearchLeaf(t, s)

1. for each entry E in t
2. do if EI overlaps s
3. then output E

Searching an R-Tree is unlike searching an B-Tree, All internal nodes whose minimal bounding rectangles intersect with the search rectangle may need to be visited during a search. So a worst case performance is O(N) instead of O(logN). Intuitively, we want the minimal bounding rectangles stored in a node to overlap as little as possible so that we need to search as little nodes as possible. We can apply the searching of an R-tree to find objects that overlap a search object, say o, by the following steps. Search Obj(t, o)

1. Bounding box of the search object o
2. Search Sub Tree(t,s) and revise the above Search Leaf(t,s) as follows:

Search Leaf (t,s)

1. for each entry E in t
2. do if EI = s
3. then if EP = o
4. then output E

In this way the search an node takes place.

4. IMPLEMENTATION OF R TREE

In this section let us discuss how the R tree is implemented. At first I had taken the dataset which consisting of 53,145 records columns represents the trajectory id x,y .By using the insertion algorithm I had inserted the given into it. First we have to form the rectangle coordinates that is, in the given x and y values we have to take first 4 values in that we have take the x minimum and y minimum values. Similarly we have to consider the x maximum and y maximum values. In this way the rectangle coordinates will be formed.. In this the insertion of a given dataset is done. Later I had searched for given rectangle coordinates. Similarly the same dataset has been inserted into oracle and Mysql. Later searching has been done in Mysql ,Oracle .In MySql and Oracle queries has been written related to the dataset.

5. RESULTS

By inserting the given dataset in R tree I had got the results and the result is represented in figure b .Later I took a small dataset consisting of 34 records which contains count reference ,outlet, time and inserted in mysql

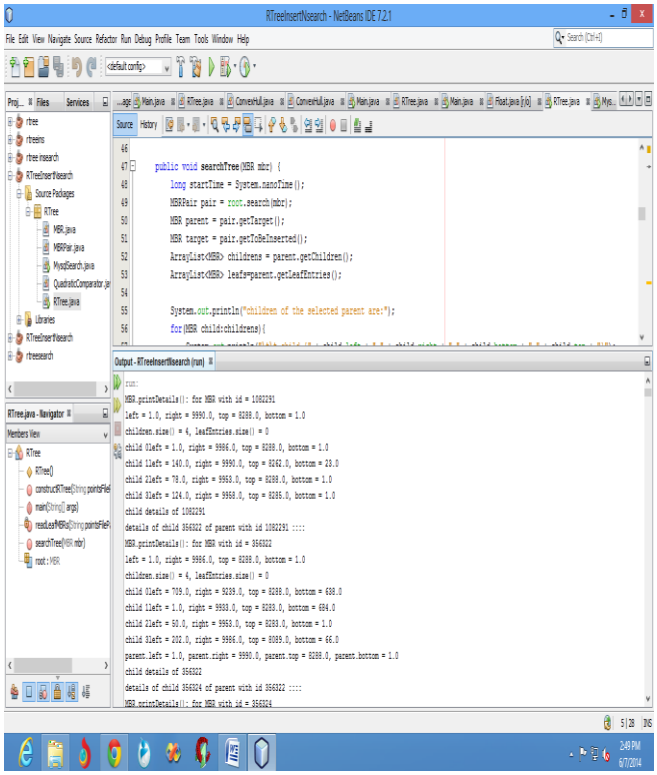

Figure b showing insertion of R tree

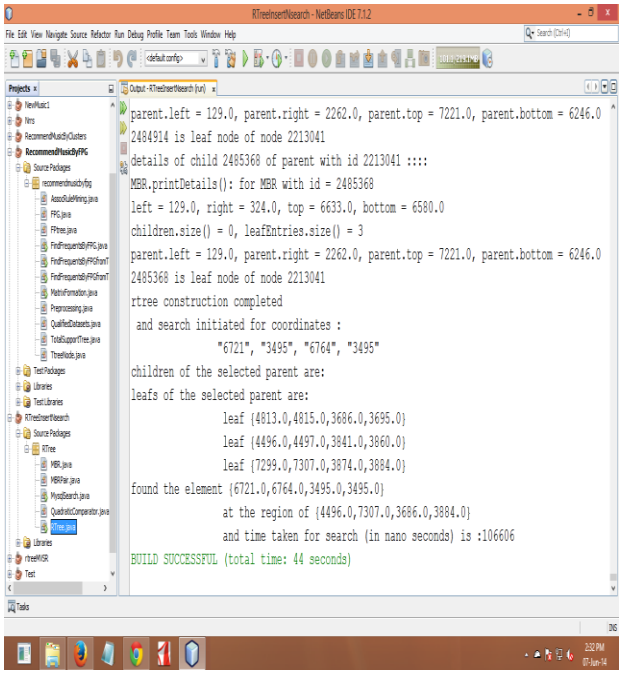

Figure c showing R tree insertion with time

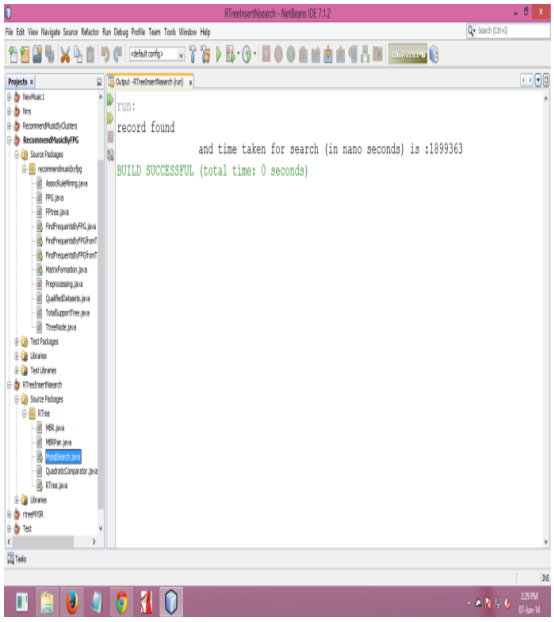

Figure d showing insertion in mysql

6. CONCLUSION

In this paper we described how R tree is better compared to B trees finally I can conclude that by using R Tree we can reduce the retrieval of queries compared to MySql The R-tree structure has been shown to be useful for indexing spatial data . Since R Trees is a multidimensional database. By using this performance can be reduced.

REFERENCES

[1]. A. Guttman, —R-Trees: A Dynamic Index Structure for Spatial Searching,Proc. ACM SIGMOD ’84, pp. 47-57, 1984.

[2]. D. Greene, —An Implementation and Performance Analysis of Spatial Data AccessMethods,Proc. Fifth IEEE Int’l Conf. Data Eng. (ICDE ’89), pp. 606-615, 1989.

[3]. K. Chakrabarti and S. Mehrotra, —Efficient Concurrency Control in Multi-Dimensional Access Methods,Proc. ACM SIGMOD ’99, pp. 25-36, 1999.

[4]. J.K. Chen, Y.F. Huang, and Y.H. Chin, —A Study of Concurrent Operations on R-Trees, Information Sciences, vol. 98, nos. 1-4, pp. 263- 300, May 1997.