www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 4 April, 2013 Page No. 1047 -1049

An R-Tree Node Splitting Algorithm Using MBR Partition for Spatial Query

Dr.V.Khanaa, Dr.Krishna Mohanta
Dean Info. Bharath University Chennai 600 073
Sri Lakshmi Ammal Engineering College Chennai 73
Department of Computer Sci.&Engg.
Mail:drvkannan62@yahoo.com

Abstract
The optimization of spatial indexing is an important issue considering the fact that spatial database, in such diverse areas like geographical, CAM and image applns are growing rapidly in size and often contain in the order of millions of items.To handle these multi-dimensional data, R-tree is widely used as data structure. The node splitting algorithm used in R-tree process affects the query performance and results in an inefficient R-tree structure as it generates uneven nodes. To overcome these drawbacks, we have proposed an algorithm to balance the uneven node splitting to meet the demand of the R-tree process. The projected algorithm inserts the node into the sibling instead of splitting or re-insertion of the overflow node which paves way to reduce the overhead of splitting process, adjusting tree construction operation and number of disc accessing.

KEYWORDS
Geographical Information System, Node Splitting algorithm, Spatial database, Indexing Multi dimensional data, R-Tree.

Introduction
There has been a great deal of interest over the years in extending traditional, alphanumeric databases to handle multi-dimensional spatial data. Applications of such spatial databases have traditionally been found in Geographical Information Systems (GIS) and Computer Aided Design (CAD) packages. In a GIS, for example, that contains maps of all the roads, lakes and rivers of a country X, there is a need to facilitate queries such as requesting all the roads within Y miles of city Z. More recently, spatial data structures have been used to aid in the retrieval of images from large image databases by shape similarity. In order to process these types of queries quickly, an efficient indexing mechanism for spatial data objects is required, according to their location in space. A number of structures have been proposed for handling multi-dimensional point data. Antoine Guttmann was one of the first persons to propose them. In 1984, Guttmann published a book[Gutt84] in which he presented a data structure called R-Tree (Rectangle Tree) that represents data objects by intervals in several dimensions.An example of such a data structure is the R-tree and its variants, where the data space is successively decomposed into (hyper) rectangles. Where the primitive index region is an polygon. All these data structures are suited for handling spatial data dynamically. In addition to retrieval, they allow runtime insertion and deletion of objects in the database. If some of the current databases are considered to be large, future databases are expected to be huge. For example, the U.S. Bureau of the Census has been building the TIGER database to store a detailed map of the country; its size is currently approximately 19 Gb. In the near future, NASA's Earth Observation System database is expected to include more than 1010 Mb of image data. The volume of such databases containing millions of data objects necessitates the storage of the index structure on disk. R-tree is a data structure used for indexing multi-dimensional information and used for spatial access methods such as storing, retrieving and applying query process on spatial data It is a height balanced tree similar to B-tree with index records in its leaf nodes containing

pointer to data objects Leaf node in an R-tree contain index entry of the form (I, tuple-identifier) where tuple-identifier refers to a tuple in the database and I is an N- dimensional rectangle Non-leaf nodes contain entries of the form (I, child-pointer) where child-pointer is the address of a lower node in the R-tree and I covers all rectangles in the lower node‘s entries Every node contains between m and M index records unless it is the root, where m is the minimum number of records and M is the maximum number of records where m <= ⌐M/2¬ All leaves appear at the same level

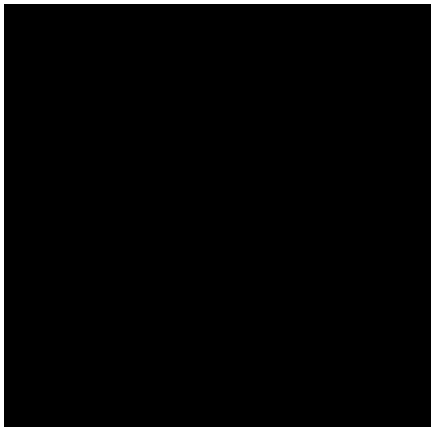
Terminology

The R-tree is an object hierarchy which is applicable to arbitrary spatial objects which is formed by aggregating their minimum bounding boxes and storing the aggregates in a tree structure. The aggregation is based, in part, on proximity of the objects or bounding boxes. The number of objects or bounding boxes that are aggregated in each node is permitted to range between m <= (M /2) and M, thereby leading us to use the prefix (m, M) to characterize a particular R-tree and mirroring the effect of a B-tree. The root node in an R-tree has at least two entries unless it is a leaf node in which case it has just one entry corresponding to the bounding box of an object. MBR Node Splitting A node splitting algorithm is a key issue in R-tree construction. This splits the available number of objects into the requirement of MIN and MAX limit provided for the given R-tree. The two metrics that affects the query performance of an R-tree are the total area which is the area of the node‘s MBR, and the overlap area which is the area of the overlapped part of two nodes. The larger the total area is, the more dead spaces there are, and the higher the probability of unnecessary accesses is. Overlap causes accesses to multiple nodes when a query object falls into the overlapped region, which increases number of page accesses.

Outlier MBR Handling It is the situation where a data is numerically distance from the rest of the data. In GIS it is denoted as an object which is deviate markedly from other objects of the sample in which it occurs. Several Outlier handling techniques have been developed since R-tree was published. Among them R*-tree outlier handling method uses the forced reinsertion. That is, if a disk page overflows, some objects are removed from the page and new objects are reinserted into the index. The R*-tree algorithm selects outlier by calculating the distance between the center of the MBR and center of the objects in the MBR, the objects whose distance are very far from the center of the MBR while comparing with the other objects, those objects are removed and the new objects are reinserted into the MBR

METHODS

Guttmann’s quadratic node splitting In here, it picks two records that may cause the worst split if put into the same node. Using these two records as seeds, the algorithm respectively finds a record that may affect the splitting quality the most and assigns it to the appropriate node until all records are assigned. If there are just enough records are unassigned to make one of the two nodes to satisfy the lower bound of records number, the rest of records will be assigned to that node directly.



R*-tree’s node splitting algorithm

If the leaf node is full, this algorithm starts to work. The first m records make one group and the rest of the records make the other group. There are three metrics guiding the splitting: area-value, margin- value and overlap-value. It first chooses a splitting axis, in which axis the splitting is to be performed. The next step is to split the records along the chosen axis.

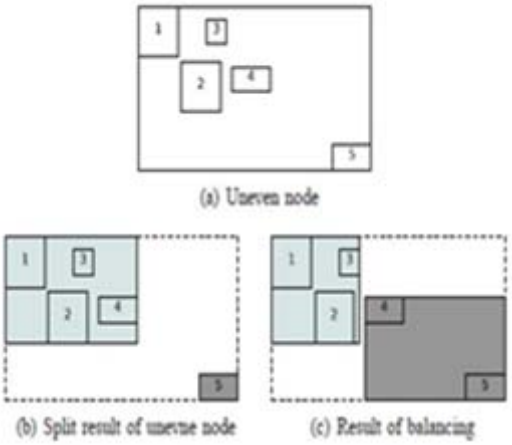
The metrics used for splitting are
Area value
Margin value

Overlap value

Optimal node splitting algorithm In this algorithm they gave two node splitting algorithms. The first one is a basic node splitting algorithm which partitions a full node into two making a metric the best. The second algorithm is an improvement of the basic one, called SHIFT method, to gain a high occupancy of disk pages. The metrics used for splitting are Area Perimeter

Linear node splitting algorithm

The main objective of the algorithm is to reduce the CPU time of an R-tree construction as well as keep the quality of
the resulting R-tree good enough. The algorithm partitions the records into two groups along each axis according to the distance between a record‘s MBR and the minimum and maximum coordinates on the axis. It then chooses a split axis by examining the number of records in each group. Then the partition along the selected axis is the final splitting result. More superior in terms of time required to split a node. Node splitting using MBR Partition Policy In this algorithm, they try to partition the records such that each resulting node has a shape as square as possible. The splitting axis is chosen by using j-Long method, if the chosen MBR is j-Rectangle then took the dimension j as the splitting axis, else they compute the axis for splitting by counting the number of records parallel to x-axis and number of records parallel to y-axis, the
axis in which the greater records are available is chosen as the splitting axis,if the axis chosen is wrong then overlap occurs in a high probability. According to the chosen axis the splitting is performed.



Conclusion:

In this paper, we introduced our node splitting method that is simple and efficient. We described our method in the order of we developing it that is first a basic method, then a policy to fix some uneven situation, and at last an improvement. We compared our method with other methods first using a simple example with some analysis, which showed that our method would split a overflowed node well. Then several experiments were conducted to compare our method and the most commonly used methods. The results showed that our method is efficient.

REFERENCES:

1. Yan Liu, Jinyun Fang and Chengde Han, ―A New R-tree Node Splitting Algorithm using MBR Partition Policy‖, Proceedings of the 17th International Conference on Geoinformatics, pp.1-6, 2009.

2. A. Guttman, ―R-trees: A Dynamic Index Structure for Spatial Searching‖, Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, pp.47-57, 1984.
3. N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, ―The R*-tree: An Efficient and Robust Access methods for Points