***I.J. Wireless and Microwave Technologies*, 2011, 4, 40-46**
Published Online August 2011 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijwmt.2011.04.06

Available online at http://www.mecs-press.net/ijwmt

A New R*Q-tree Spatial Index based on the Delaying Selection and Clustering Splitting

Quanyou Song[a,*1], Wan-gao Li[b,*2]

[a] *Department of Information and Communication Engineering, Henan Vocational and Technical College of Communications, Zhengzhou, China*
[b] *Network Management Center, Henan Institute of Engineering, Zhengzhou, China*

Abstract

The original R*Q-tree is an important class of query index in the spatial database, but when carry on frequent insertion and deletion, the efficiency is very low, because its construction cost is much more than that of the other query methods. So the new R*Q-tree using the delaying and Clustering splitting technology has been proposed for improving the shortcomings of the original R*Q-tree. When the data objects are inserted into the new R*Q-tree, the nodes may overflow. During the process of insertion, rather than split the node immediately, it attempts to insert the nodes into the adjacent neighboring nodes by using of the delaying selection splitting technology until they are full. If the adjacent neighboring nodes are full, it will use the clustering technology to split the nodes, data items will be reorganized between the neighboring nodes and the splitted nodes. The new R*Q-tree can ensure the premise of the query performance and the cost of structure greatly reduced, at the same time, significantly the space utilization of the index structure is improved. Finally, the experimental analysis and appraisals show that the operation efficiency of the new R*Q-tree is enhanced. And in the new R*Q-tree and the promoter region, there are no duplicated elements, so the structure of the new R*Q-tree can be simplified.

Index Terms: Delaying Selection Splitting; Clustering Splitting; R*Q-tree

1. Introduction

The original R*Q-tree is one of the most popular spatial index structure, which can be widely used in all kinds of prototype system researches and commercial fields. The original R*Q-tree is a dynamic tree construction of depth balance, whose query is similar to that of the R*-tree [1]. The query performance of the original R*Q-tree is mainly affected by the two aspects of coverage and overlapping. Oversized coverage can result in large dead space and reduce the query efficiency [2-3]. Excessive overlapping will search too many query paths, in the worse situation, it needs to traverse the whole path.

* Corresponding author:
E-mail address: [*1] 61915600@163.com; [*2] wgli@haue.edu.cn

In fact, in the process of structuring the original R*Q-tree, there exists the question about the objects to be inserted into which node and how to split the present spatial distribution. The spatial distribution will change with the insertion of the objects, it is the question for the original R*Q-tree to make the original decision unreasonable. The main idea is to offer a revision mechanism in the process of building the query constructor. The delaying selection splitting and clustering splitting of dynamic R*Q-tree will be proposed in this paper, which further widen the neighboring request of the nodes, so the improved neighboring mechanism is proposed, and it can largely improve the efficiency of spatial utilization.

2. The Idea of the New R*Q-tree

The new R*Q-tree improves the original R*Q-tree on splitting algorithm, the other algorithms of the new R*Q-tree are the same as that of the original R*Q-tree.

The main idea of the new R*Q-tree is to guarantee the neighboring performance in contraction, when the neighboring mechanism is improved, the neighboring request can be widen. Delaying selection and clustering splitting technique include delaying selection splitting and clustering splitting [4]. Delaying selection splitting is that when the nodes are overflowed, we do not split the nodes immediately, but insert the objects into the not full neighboring nodes, and centers of the intensive area of MBR in the nodes are nearest to these objects. Clustering splitting is that when the inserted nodes and neighboring nodes are all full, we can reorganize the entries between the node and its neighboring nodes.

3. Clustering Algorithm

We use a usual K-Means algorithm in this paper, O(k*n*d*t) for time complexity, k for clustering number, n for the number of the objects, d for the data dimension, t for the times of algorithm iteration. Firstly, k objects can be chosen from n objects in this algorithm at random as the center of clustering, and then each object can be distributed into the classic clustering according to the nearest distance rule in every iteration, and refresh the center of clustering. Iteration never changes until the center of clustering doesn't change. When the entries of the R*Q-tree are clustered by K-Means algorithm, it can define the distance between the center of the clustering and the entries. For the clustering including n points in d dimension, the center of which can be defined as the average.

For the clustering including n rectangles $R_1,..,R_n$ in d dimension, each rectangle stands for $R_i(P_{iL},P_{iH})$, P_{iL} for the left-bottom point, P_{iH} for the right-upper point, the center will be defined as the following:

$$\overline{P}\left(\frac{\sum_{i=1}^{n} Area\left(R_i\right)\left(P_{iL}x_1 + P_{iH}x_1\right)/2}{\sum_{i=1}^{n} Area\left(R_i\right)}, \dots, \frac{\sum_{i=1}^{n} Area\left(R_i\right)\left(P_{iL}x_d + P_{iH}x_d\right)/2}{\sum_{i=1}^{n} Area\left(R_i\right)} \right) \tag{1}$$

4. The New R*Q-tree Construction

The insertion process of the traditional R*Q-tree is that:

(1) Find the inserted leaf-nodes.
(2) If the leaf-nodes are not all full, then the entries [mbr,oid] will be inserted into the corresponding leaf-node page, and at the same time adjust the father rectangle.
(3) If the leaf-nodes of the inserted objects have been full, then they can be splitted. Create a new node; M+1 entries will be distributed in the two nodes. When the new R*Q-tree searches for the target leaf-node, it directly uses the clustering algorithm to define the distance and choose the similar nodes, ChooseSubtree for the detail algorithm. ChooseSubtree can be used by iteration or recursion to find the target leaf-node. The main topics of the new R*Q-tree are insertion algorithm and splitting algorithm.

4.1. Delaying selection splitting technique

Using the delaying selection splitting technique in the insertion algorithm, when the nodes1 are overflowed, we can not split the nodes1, select some furthest objects distance from the center of the intensive area in MBR corresponding to the nodes1, insert the selected objects into the neighboring nodes2 of the MBR whose centers of the intensive area are nearest to them, at the same time, the neighboring nodes2 are not full, until the nodes1 are not overflowed. This technique can delay the nodes to be splitted, and achieve a smaller MBR of the nodes1 after reinserting, reduce the number of splitting and the cost of index, and so call it as the delaying selection splitting technique [5-7].

Algorithm 1 (insert the objects into leaf-node--Insert Entry)

a) Input: leaf for leaf-node, e for the inserted object, k for the number of the neighboring leaf-nodes.
b) Output: none
c) If the leaf is not full, e will be inserted into the corresponding leaf-node page, and then return.
d) If the leaf is full, k leaf-nodes which are the nearest distance from it will be found in its neighboring brother nodes.
e) The objects in other leaf nodes can be found by distance consequence in the not full k leaf nodes.
f) If it can find the not full leaf node, calculate the distance between e, the data objects in the leaf and the center of the intensive area in MBR corresponding to the already full leaf, select the furthest object distance from the center of the intensive area in MBR corresponding to the leaf, insert it into the corresponding neighboring leaf-node page of the MBR whose center of the intensive area is nearest to this selected object, at the same time, the neighboring leaf-node is not full, if the object is e, insert it according to the above method, else process the furthest object according to the above method, insert e into the corresponding leaf-node page, and return.
g) If it can not find the not full leaf node, the clustering splitting algorithm to reorganize the data-items will be complemented in leaf and k neighboring nodes.

4.2. Clustering splitting technique

The clustering splitting tries its best to avoid splitting, and results in the worsening new R*Q-tree constructer, however, the clustering splitting uses the clustering technique to deal with the nodes splitting, so the improved neighboring mechanism can be offered. When the inserted nodes and neighboring nodes are all full, then they will be splitted by using of the clustering technique to reorganize data-items, so call it as the clustering splitting.

Algorithm 2 (leaf-nodes splitting algorithm)

a) Input: leaf for the leaf-node, m for the number of the inserted data objects, k for the number of the neighboring brother leaf-nodes *b)* Output: more leaf-nodes by clustering the entities in the leaf and k leaf-nodes.

c) When the inserted objects are fewer, calculate the distance between the objects and the centers of the intensive area in MBR corresponding to the nodes, select the nodes of these MBR whose centers of the intensive area are nearest to these objects, cluster the inserted objects and the data objects in the nodes by using the K-Means algorithm, and output m+1 or much more clusters

d) When the inserted objects are much more, cluster the M*(k+1)+m entries by using the K-Means algorithm, and output k+2 or much more clusters. Refresh the entries of the leaf and k brother leaf-nodes

According to the number of the clusters, create one or more leaf-nodes; insert these leaf-nodes into the farther node of these leaf-nodes.

The time cost of the algorithm includes: n*choosesubtree+ n1*entry+n2*clustering splitting, n1+n2=n, n2= [n/ (M*k)], the cost of the clustering splitting mainly includes IO, O(k*n*d*t) (the time complexity of clustering), t (the number of iteration). When k=0, the novel R*Q-tree is R*-tree. When $k\geq$ [n/M], the new R*Q-tree is able to organize the dynamic data. The delaying selection and clustering splitting technique can ensure that:(1)the clustering splitting can get optimization entries, reduce the coverage and overlapping of the nodes, improve the query efficiency;(2)delaying selection splitting delays the nodes splitting, reduces the number of the splitting.(3) space utilization can get largely improved. In fact, with the addition of the number of the neighboring nodes k, it can largely delay the splitting and reduce the number of nodes splitting, improve the space utilization, but at the same time it adds the cost of the clustering splitting.

5. Performance Appraisals

In order to exam the performance of the new R*Q-tree, we build an environment based on Java, and also build the original R*Q-tree algorithm. The Two groups of data can be used in the experiment: (1) the data set can be generated: including 50000 rectangle objects, coordinate is the float between 0.01 and 1.0. (2) the real data set: including 16704 polygon objects.

In the experiment, the page size set for 2kb, the volume of the leaf-node and the middle-node for 50, the packing factor for 70%. For the structure query, a group of data can be inserted into the tree by random consequence, and obtain the time and space utilization by figuring the average of 100 trees. For a query, it needs to complete 1000 times to get the average.

The construction of the index structure is the main topic of the robustness and scalability of index structure. From the Fig.1, we can get the conclusion that the new R*Q-tree is superior to the traditional R*Q-tree in the cost of structure, at the same time, it also reduces the cost of structure than that of the R*-tree.

From the Fig.2, we can see that the query performance of the new R*Q-tree is superior to that of the conventional R*Q-tree, and the query performance of the novel R*Q-tree is the same as that of the R*-tree. Its k-NN query performance is also superior to that of the R*-tree and is beyond the conventional R*Q-tree. With the increasing number of spatial database, the query performance of the new R*Q-tree is superior to that of the conventional R*Q-tree and R* -tree.

The space utilization of the spatial data index tree is one of the main sign of the spatial index technology [5]. Due to the spatial database with the character of having a mass of data, the index tree structure must guarantee that each of leaf-node can store the spatial data as it can. As to the reduction of the cost of space, the follow formulas can compare the novel R*Q-tree, the conventional R*Q-tree and R*-tree.

S= (the number of indexing tree storing spatial data objects/the total leaf-node number of indexing tree) ×100%

The follow figures explain the space utilization of the different index structures.

The new R*Q-tree is beyond the conventional R*Q-tree in space utilization. From the Fig.3, we can see that with the addition of the number of neighboring nodes k, the space utilization can be improved greatly. The new R*Q-tree introduces the delaying selection and clustering splitting technology. Once the spatial data-items satisfy the condition, the insertion into their brother nodes will take place, which can not only reduce the number of nodes splitting, but also improve the space utilization.

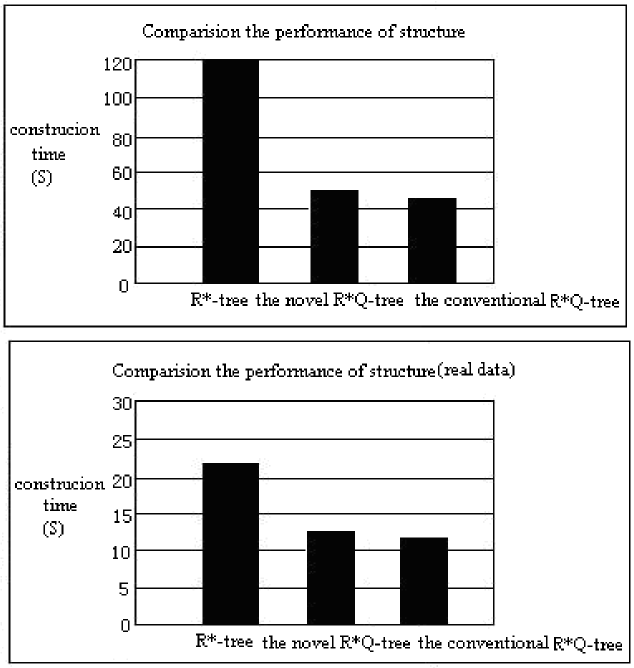


Fig. 1. Structural performance evaluation of different index structures

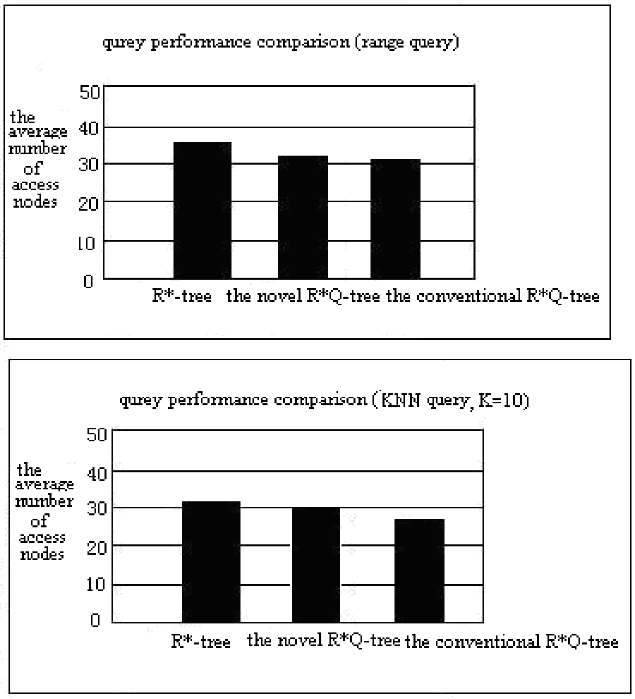
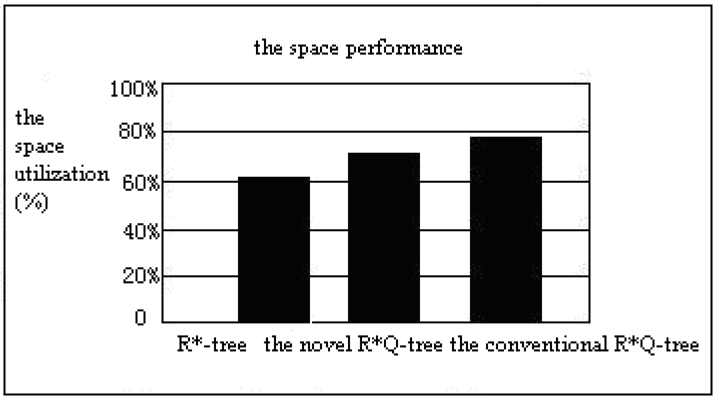


Fig. 2. The query performance of different index structures



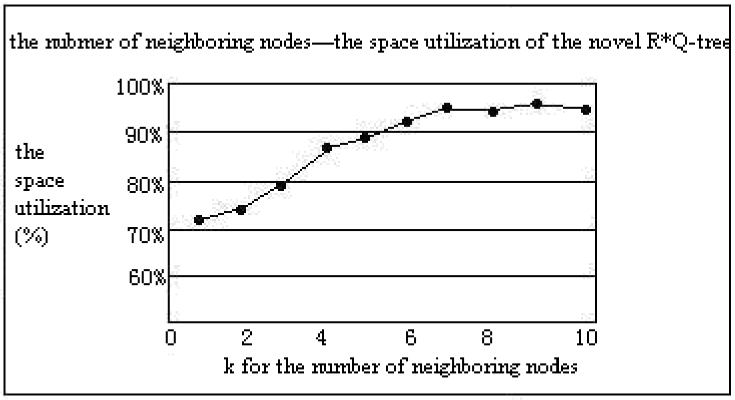


Fig. 3. The space utilization of different index structures

In conclusion, the new R*Q-tree has the same query performance as that of the R*-tree, the constructional cost is beyond that of the conventional R*Q-tree, so it has the better space utilization.

6. Conclusions

The original R*Q-tree is the most popular query index of spatial database, but not suitable for the situation of frequent insertion and deletion, because its construction cost is many times more than that of the other query methods. So the new index structure of the dynamic R*Q-tree is proposed based on the delaying selection and clustering splitting, therefore it is called the new R*Q-tree. The delaying selection splitting does not split the node when the node overflows in the new R*Q-tree, but tries to insert these overflowed objects into the neighboring and not full node according to the nearest distance, until insert the inserted object into the reasonable node. The clustering technique is to split the nodes and reorganize the entries between the node and its neighboring nodes. The new R*Q-tree can reduce the construction overheads with the guarantee of the query performance, and the space utilization of the index structure is improved.

References

[1] A.Guttman, “R-trees: A Dynamic Index Structure for Spatial Searching,” Proc. ACM SIGMOD, 1984, pp.47–57.

[2] Shupeng Chen, Xuejun Lu, Chenghu Zhou, "*Geography informational system*," Beijing science press, 1999. (in chinese)
[3] Guobin Li,Yongli Tang, "*Spatial database technology*," Electronics industry press, 2009. (in chinese)
[4] Xiaofeng Lei, Kunqing Xie, Xingxing Jin, "A Novel Implementation of Dynamic n R-tree Based on Lazy Splitting and Clustering," Computer Science, Vol.34, No 4, 2007, pp.102–125.
[5] Ming Huang, Zhe Chen, "Research on the spatial index based on improvement RQ-tree," Journal of Heilongjiang Institute of Technology, Vol.19, No3, 2005, pp.18–20.
[6] Xincai Wu, "*Geography informational system theory and method*," Electronics industry press, 2009. (in chinese)
[7] Jianhua Qiu, Xuebing Tang, Huaguo Huang, "An index structure based on quad-tree and R*-tree–R*Q-tree," Computer applications, Vol.23, NO.8, 2003, pp. 124–126