ISSN:0975-9646

A Proposed Hybrid Spatial Indexing: QX Tree

Jayanaranjan Dash
School of Computer Engineering
KIIT University, Bhubaneswar, India

Dipa Patra
School of Computer Engineering
KIIT University, Bhubaneswar, India

Chittaranjan Pradhan
School of Computer Engineering
KIIT University, Bhubaneswar, India

Abstract—**Out of different spatial indexing structures available for accessing spatial data, none of them is suitable for high dimensions. This is because the performance of the spatial indexing structures become poorer with the increase in dimension. Thus there is a need for a better spatial indexing structure for the same. Here we have proposed a hybrid indexing structure by combining the Quad Tree and X Tree. We have considered the X Tree over R Tree used in the previous hybrid indexing structure, QR Tree. This is due to the better performance of X Tree over the R Tree in case of highly overlapped data.**

Keywords-Spatial indexing, Hybrid spatial indexing, Quad tree, X tree, R Tree, QR Tree, QX Tree.

I. INTRODUCTION

Spatial index can be defined as the data structure according to a certain order, which is based on the position and the shape of spatial objects or a certain spatial relation exists between spatial objects. Spatial index is a supplementary measure between the space objects and space operation algorithm, whose main objective is to screen and filter the spatial data[1].

The most important requirements for these data structures are the ability to provide fast access to large volumes of data and preserve spatial relationships, such as nesting and neighborhood for indexed objects. Several tree-like access methods were proposed for spatial objects[2]. Quadtree[3] is one of the first data structures for high dimensional data, which was developed by Finkel and Bentley in 1974.

According to H.Samet[4], "A class of hierarchical data structure whose common property is the recursive decomposition of space is known as Quadtree". A Quad tree is a rooted tree like structure whose internal node has exactly four children. It is mainly used to partition a space by recursive subdividing method, which turns the space into four quadrants. The four quadrants are treated as four child of the tree labeled as NW,NE, SW and SE. It indicates the quadrant they represent. Figure 1 gives the overview of Quadtree.

Spatial index is an important process to improve the performance of the spatial Database. There are a lot of indexing methods are proposed like Quad Tree, KD Tree[8], R Tree[6], X Tree[5] etc. Due to increase in spatial data and the number of working dimensions, the performance of above indexing methods decrease gradually. To solve this problem, researchers started to propose hybrid spatial indexing techniques by combining advantages of different spatial indexing techniques.

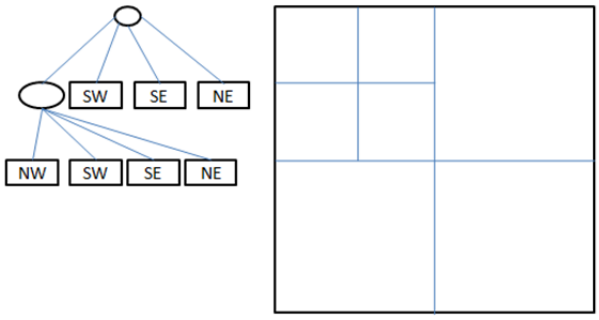

Figure 1. Structure of Quad Tree

QR Tree[2] is a hybrid indexing method which combines the properties of both Quad Tree and R Tree to give a better performance for spatial data present in higher dimension. In QR Tree, a given space is first divided according to Quad Tree with a maximum depth d. Then each individual divided sub space contain their corresponding R Trees as shown in Figure 2.

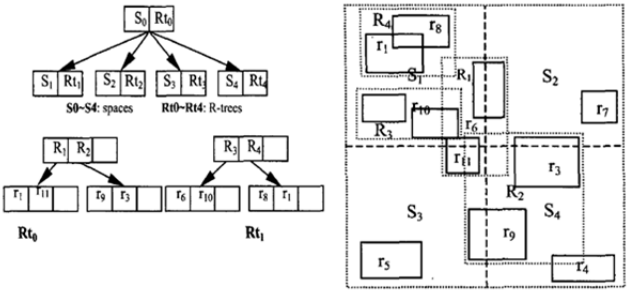

Figure 2. Structure of QR Tree in 2D Space

The QR Tree gives a faster searching performance than both Quad Tree and R Tree at higher dimensions. But the main demerits of R Tree based indexing is that the performance becomes poor when data is highly overlapped. The more increase in dimension, the more possibility of data overlapping. So, here we propose another hybrid spatial indexing method which combines the Quad Tree and X Tree.

The other spatial indexing method is X tree[5], which is a variant of very popular R tree[6]. The data structure is based on the B Tree[7] indexing method. The main disadvantages of R tree based indexing method is the poor performance with respect to the dimension increment. The data overlapping directly proportional to increase of dimension, which has a further negative impact on query

processing. For a simple point query, we have to follow a multiple path in R tree based indexing.

The X tree is a spatial indexing method which support efficient query processing of data at high dimension. It supports both point data as well as extended spatial data. X tree uses the overlapping concept in terms of regions. It avoids the overlapping as minimum as possible. It also uses an extended variable size directory nodes called as supernodes. The X tree uses the available main memory more efficiently.

X tree is like a hybrid of linear array based directory and R tree based directory. It is designed in such a way that X tree automatically organize the directories hierarchical as possible.

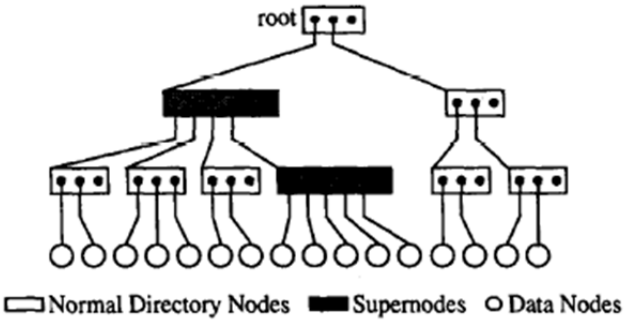

Figure 3. Structure of X Tree

Figure 3 describes the overall structure of X tree. The data nodes of X tree contain MBR(Minimum Bounded Region), pointers to the actual data, and the directory nodes contain MBRs along with pointers to the sub MBR. The X tree consists of three different types of nodes; i.e. supernodes, data nodes and the normal directory node. Supernodes can be defined as large directory having variable size. The main objective of supernodes is to avoid splits inside the directory.

II. The Proposed Hybrid Spatial Indexing: QX Tree

A. Definition

QX tree can be defined as a spatial data structure which combines the features of Quadtree and the X tree as shown in Figure 4. The number of X tree can be found out as:

$$n = \sum_{i=0}^{d-1} (2^k)^i \qquad (1)$$

where, d = depth of quad tree and k = number of dimension.

Quad tree divide the entire index space (S) into n sub parts i.e. $S_0,S_1,S_2...S_{n-1.}$ Each part is a d-level sub space. The sub spaces are disjoint to each other; i.e. no sub space overlaps with other subspace at any level.

Each of X tree ($Xt_0,Xt_1 \ldots Xt_{n-1}$) associates itself with n node and n sub spaces of Quad tree. A spatial object P is belong to $S_{i.}$ That implies:

- P is completely inside $S_{i.}$
- S_i is the smallest subspace to completely contain P.

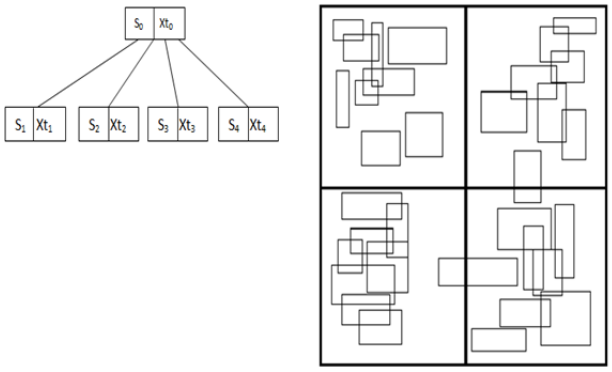

Figure 4. Structure of QX Tree

B. Search Algorithm

Let a searching rectangle area named R is given to search for all spatial data within or on the rectangle R. We must perform searching operations on X Trees associated with subspaces which intersect with R. If the sub space associated with the root node intersects with R and intersects with the index space of corresponding X tree, then we search in the X tree.

For every sub node, we determine whether its corresponding sub space are intersecting with R or not. If not, then the node and sub tree is the end. If yes, then it intersects with corresponding X tree index space.

```
QX_Search(X, R) /*Searching all data in the search area
    R in the QX tree rooted at X*/
    {
        S: Sub space within X;
        if S is not overlapping with R then return;
                if X.MBR overlap with R then
                        X_Search(X.MBR,R); /*X Tree
                                Search */
                for each child node of X Do
                        QX_Search(X.child,R);
    }
```

C. Insert Algorithm

To insert a data, we should first confirm that which sub space it belongs to and their corresponding nodes. Then insert data in corresponding X tree.

```
QX_Insert (X, obj) /*insert object obj into the QX tree
    having root X*/
    {
        if X is a leaf node of Quad Tree, Then
                X_Insert(X.MBR,obj); /*Calling X tree
                Insert Algorithm */
        else
                {
                Found = false; /*determining whether
                subspaces contain obj*/
                for every child node of X Do
                        S=the sub space associated with
child of X
                if S contain obj entirely then,
```

```
                    {
                        found=True;

        QX_Insert(X.child,obj);
                        break;
                    }
                if not found then X_Insert(X.MBR,obj);
                }
    }
```

D. Delete Algorithm

To delete a data item, we first confirm the subspaces it belongs to and their corresponding nodes. Then delete the data from corresponding X tree.

```
QX_Delete(X, obj) /*Item to be deleted is obj from the QX
    tree rooted at X*/
    {
    if X = leafnode of quad tree then
        X_delete(X.MBR,obj); /*Calling the X tree Delete
        algorithm*/
    else
        {
        found=false; /*Determine if one of subspace
        contain obj */
        for every child node of X DO
        {
        S=the subspace associated with child of X;
        if S contain obj entirely then
                found=True;
                QX_Delete(X.child,obj);
                break;
        }
        if not found then X_Delete(X.MBR,obj);
        }
    }
```

III. CONCLUSION

In this paper, we have analyzed an existing hybrid index structure named QR tree. But we found that the performance decreases with the overlapping data in higher dimension. So we have proposed a new hybrid spatial indexing method and termed it as QX Tree, which combines both Quad Tree and X tree. Because X tree gives better performance for overlapped data as compare to R Tree. Here we only present the algorithm of the new hybrid spatial index structure. Further it can be implemented with suitable programming language and should test its vulnerability by comparing several other indexing method.

REFERENCES

[1] Guobin Li, Lin Li, "A Hybrid Structure of Spatial Index Based on Multi-Grid and QR-Tree, International Symposium of Computer Science and Computational Technology, 2010, pp. 447-450.

[2] Yu-Chen Fu, Zhi-Yong Hu, Wei Guo, Dong-Ru Zhou, "QR-tree: a Hybrid Spatial Index Structure:, IEEE International Confeence on Machine Learning and Cybernetics, 2003, Vol. 1, pp. 459-463.

[3] R. A. Finkel, J. L. Bentley, "Quad Trees: A Data Structure for Retrieval on Composite keys", Springer Acta Informatica, 1974, Vol. 4, No. 1, pp. 1-9.

[4] Hanan Samet, "The Quadtree and Related Hierarchical data Structures", ACM Computing Surveys, 1984, Vol. 16, No. 2, pp. 187-260.

[5] Stefan Berchtold, Daniel A. keim, Hans-Peter Kriegel, "The X-tree: An Index Structure for High-Dimensional Data", ACM International Conference on Very Large data Bases, 1996, pp. 28-39.

[6] Antonin Guttman, "R-trees: A Dynamic Index Structure for Spatial Searching", ACM SIGMOD International Conference on Management of Data, 1984, Vol. 14, No. 2, pp. 47-57.

[7] Douglas Comer, "Ubiquitous B-Tree", ACM Computing Surveys, 1979, Vol. 11, No. 2, pp. 121-137.

[8] Jon Louis Bentley, "Multidimensional Binary Search Trees used for Associative Searching", ACM Communications, 1975, Vol. 18, No. 9, pp. 509-517.