LAYERED R-TREE: AN INDEX STRUCTURE FOR THREE DIMENSIONAL POINTS AND LINES

Yong Zhang*, Lizhu Zhou*, Jun Chen**
*Department of Computer Science and Technology, Tsinghua University
Beijing, China, 100084
zy_thu@yahoo.com
**National Geomatics Center of China
No. 1 Zizhuyuan, Baishengcun, Beijing, China, 100044
junchen@gps.ceic.gov.cn

KEY WORDS: R-tree, Layered R-tree, Three Dimensional, Quad-tree

ABSTRACT

The applications of Geography Information System have grown very fast. To deal with the massive and complex spatial data, many spatial index methods have been provided to accelerate the search of the spatial objects. The R-tree is the most famous and practical spatial index. We extend it to three dimensions with the technology of Quad-tree to deal with three-dimensional points and lines. We provide a method named Layered R-tree, which partition the space into layers paralleled to one of the coordinate plane, in each layer, the spatial objects are organized by R-tree. We find that this kind of methods is suitable to the three-dimensional points and lines in the real world.

1 INTRODUCTION

Current DBMSs efficiently organize, access, and manipulate enormous quantities of data for traditional applications in banks, airlines, government agencies, hospitals, and other large organizations. However, today new non-traditional applications (Figure 1) (Sellis, 1987, Evangelidis, 1995, Faloutsos, 1987, Faloutsos, 1996, Lu, 1992), with growing mountains of data, require innovative solutions to storage and access problems. A spatial database consists of a collection of tuples representing spatial objects (the basic property is spatial location, such as latitude and longitude and height above the earth), and each tuple has a unique identifier that can be used to retrieve it (Guttman, 1984). The applications of spatial databases are consisted of geographic information, medical image databases, multimedia database and traditional databases, etc (Faloutsos, 1996).

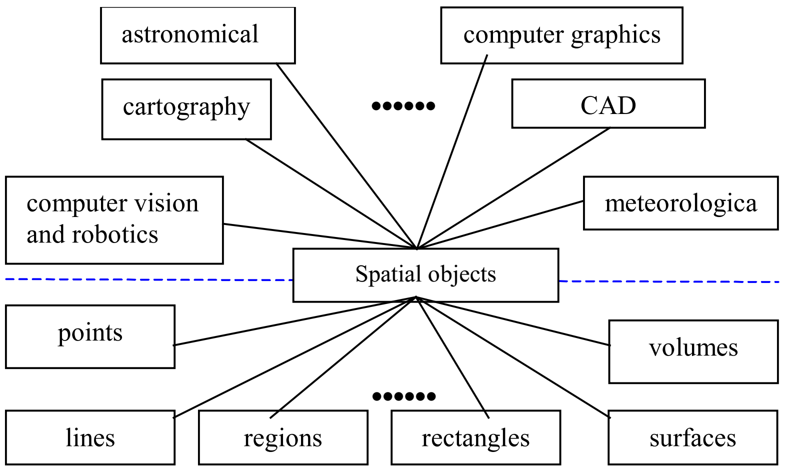

Figure 1 The structure of spatial information

Access methods in spatial database refer to spatial indexes. In databases, each record with n attributes can be considered as a point in n-dimensional space (Guting, 1994). The spatial indexes can be looked as one of the special multi-dimensions indexes. There have been a number of proposals for spatial accessing, but none of them has been integrated into a commercial general purpose DBMS. There are two reasons (Evangelidis, 1995), one is the volume of data is usually very large, the other is the distribution of data varies more drastically.

The typical spatial queries in spatial database are *range query* (Given a rectangle, retrieve all the elements that intersect it) and *point query* (A special case of the range query, the query rectangle degenerates to a point) (Kamel, 1994, Beckmann, 1990). Corresponding to the spatial queries, there are two basic access methods (Seeger, 1988, Lu, 1992, Beckmann, 1990): PAM (point access methods) and SAM (Spatial access Methods). But all of the access methods are based on the traditional indexes such as *stored arrays, binary trees, B-trees, and hashing, etc.* (Lu, 1992).

Our approach is motivated by an examination of R-tree-based index structures (Berchtold, 1996). One major reason for using R-tree-based index structures is that we have to index not only point data but line segments. In contrast to most other index structures (such as k-d-B trees (Robinson, 1981), grid files (Nievergelt, 1984), and their variants (See e.g. Seeger, 1990)), R-tree-based index structures do not need point transformations to store spatial data and therefore provide a better spatial clustering. But they suffer from a poor exact match performance and often from inefficient range queries because cells may overlap considerably in a dynamic setting. If R-trees are built using the dynamic insertion algorithms, the structure may provide excessive space overlap and "dead-space" in the nodes that result in bad performance (Henrich, 1989, Sellis, 1987, Beckmann, 1990).

Quad trees (Beckley, 1985) are an extension of the one-dimensional binary tree to a two-dimensional structure. Each node in the tree has four sons to partition its scope of the universe. The root for instance partitions the entire universe into NE(1), NW(2), SW(3), and SE(4) quadrants, with each son having one quadrant as its scope. Quad-trees are usually discussed for the two-dimensional case, but can be extended to arbitrary dimensions.

Based on the R-tree and Quad-tree, we provide a new method – Layered R-tree, which partition the space into layers, then organize spatial objects in each layer. We will discuss it at length in the next sections

The remainder of the paper is organized as follows: In section 2, some preliminary concepts are presented. Section 3 describes the architecture of the Layered R-tree. Section 4 states the detail algorithms of layered R-tree. The last is the conclusion.

2 PRELIMINARY CONCEPTS

The spatial queries (Shekhar, 1999) are often processed using *filter* and *refine* techniques. Approximate geometry such as the minimal orthogonal bounding rectangle of an extended spatial object is first used to filter out many irrelevant objects quickly. Exact geometry is then used for the remaining spatial objects to complete the processing.

The spatial access technologies can be divided into two kinds: *transform* and *sub-regions* (Sellis, 1987, Kamel, 1994, Lu, 1992, Seeger, 1988, Henrich, 1989) (Figure 2).

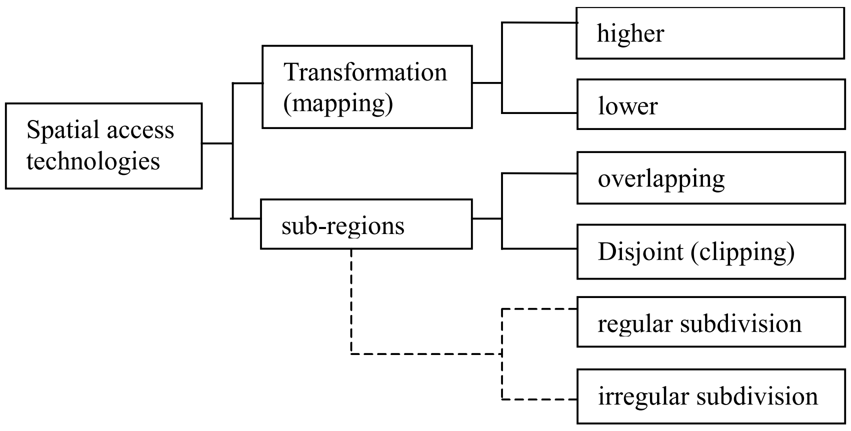

Figure 2 The classification of spatial access methods

We based on the second technique – sub-regions, and overlapping. The space is divided into several layers, in each layer every spatial object is represented by its Minimum Boundary Box (MBR).

3 THE ARCHITECTURE OF THE LAYERED R-TREE

This section gives out the algorithms of Layered R-tree that utilize the characters of the real data. In reality, the three dimensional points and lines are almost in the one height scope, or the data is clustered. So we can participation the space into several layers (Figure 3, Figure 4).

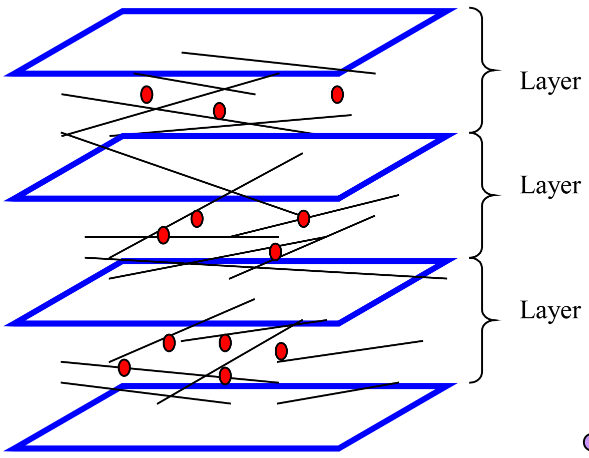

Figure 3 Splitting the three dimensional space into layers

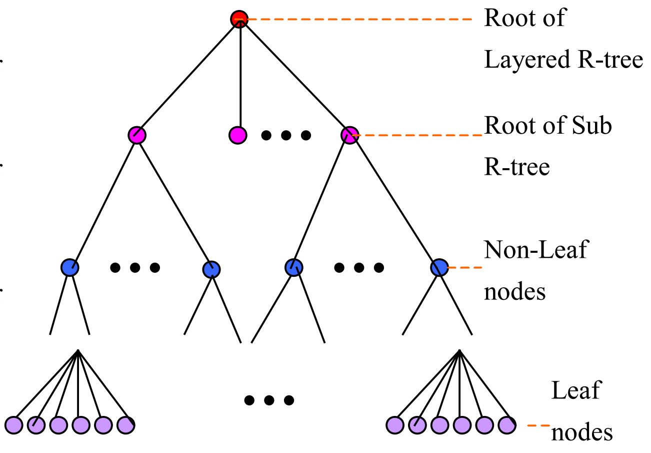

Figure 4 The structure of Layered

The layered R-tree can be partitioned into several parts: the root node of layered R-tree, the root nodes of sub R-tree, Nol-Leaf nodes and Leaf nodes.

The structure of the root of the layered R-tree is:

```
struct typLRStructure {
        double LayerMinBound, LayerMaxBound; //define the boundary of each layer;
        typNode Roots(10); //the pointers point to the root nodes of the underlying R-trees;
};
```

The structure of the nodes of the R-trees (Here the nodes include the root node, the non-leaf nodes and the leaf nodes of the R-tree):

```
struct typNodeStructure {
        typMBR MBR; // the minimum boundary rectangle ( here should be cuboid);
        int TotalItems; //the total number of the items in the Node
        bool IsLeaf; //TRUE: a leaf node; False: root node or non-leaf node;
        typItem Items(10); //the points point to the child nodes;
        typNode pLeft, pRight; //points to the nodes in the same level;
        typNode pParent; //points to the parent;
        int SN; //the sequence of the parent in the node which it belongs to ;
};
```

4 THE OPERATIONS AND SEARCH OF LAYERED R-TREE

4.1 INSERT, DELETE and UPDATE

The operations to the data items include INSERT, DELETE and UPDATE that are listed detailed in the following:

- INSERT: Before this operation, the minimum boundary cuboid should be calculated which is a binary group ((x1, y1, z1), (x2, y2, z2)). Then z1,

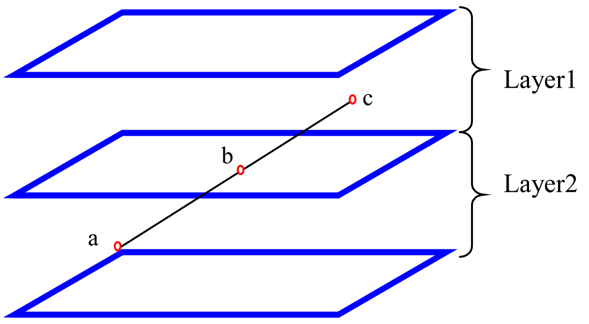

Figure 5 The line passing through layers is divided into several

z2 are compared with the boundary layers, if it falls in one of the layers, then it is inserted into the root node of the R-tree corresponding to this layer; if it passes through several layers, then it is divided into several parts (as Figure 5.). Then the next process is the same as in R-tree.

- DELETE: When a date item is removed from the Layered R-tree, first the data item should be found in the Layer-R-tree (the search is listed below). The found item is removed from the Layered R-tree. The next process is similar to that of R-tree. When the influence of the delete is spread to the root of the corresponding R-tree, if there isn't any data item in the R-tree, then the pointer corresponding to the R-tree is set to NULL.
- UPDATE: It is the combination of INSERT and DELETE.

4.2 Search

The search algorithm descends the Layered R-tree from the root in a manner similar to R-tree. Given a Layered R-tree whose root node is L, find all index records whose cuboids overlap a search cuboid S.

1) (Search Root) Check the two points in the search cuboid S, then decided which layers should be selected, one or more layers can be selected, which is decided by the volume of the search cuboid. Then search all the R-trees corresponding to layers. Suppose the roots of these layers are T1, T2, … and so on.
2) (Search subtrees) If T_i is not a leaf, check each entry E in T_i to determine whether E->MBR overlaps S. For all overlapping entries, invoke Search on the tree whose root node is pointed to by E->Items.
3) (Search leaf node) If T_i is a leaf, check all entries E to determine whether E->MBR overlaps S. If so, E is a qualifying record.

5 CONCLUSION

In this paper, we summarize main of the access methods in spatial databases. To three dimensional points and lines we provide a new spatial index – Layered R-tree, which partitions the three dimension space into layers, then uses the R-tree in each layer. We implement the algorithm using Visual C++. From the perspectives of building, search, insert, delete, we get better performance. Now the layers are predefined, we will extend it to dynamic to apply to the various distributions of the data, and compare it to other spatial indexes.

ACKNOWLEDGEMENTS

This research is supported by Natural Science Foundation of China (NSFC) under the grant number 69833010.

REFERENCES

Beckley, D.A., Events, M.W., Raman, V.K., Multikey Retrieval from K-D Trees and Quad-trees, ACM SIGMOD Conference, Austin, Texas, pp. 201-301.

Beckmann, N., Kriegel, H.P., Schneider, S., Seeger, B., 1990. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. ACM SIGMOD Conference. Atlantic City, NJ, pp. 322-331.

Berchtold, S., Keim, A.D., Kriegel, H.P., 1996. The X-tree: An Index Structure for High-Dimensional Data, Proceedings of the 22nd VLDB Conference, Mumbai (Bombay), India.

Evangelidis, G., Lomet, D., Salzberg, B., 1995. The hB$^{\Pi}$-tree: A Modified hB-tree Supporting Concurrency, Recovery and Node Consolidation, Proceedings of the 21st VLDB Conference, Zurich, Switzerland.

Faloutsos, C., Sellis, T., Roussopoulos, N., 1987. Analysis of object oriented spatial access methods, ACM SIGMOD Conference, SanFrancisco, California, pp. 426-439.

Faloutsos, C., Gaede, V., 1996. Analysis of n-dimensional Quadtrees Using the Hausdorff Fractal Dimension, Proceedings of the 22nd VLDB Conference, Mumbai(Bombay), India, pp. 40-50.

Guting, R. H., 1994. An Introduction to spatial database systems. VLDB Journal, 3(4): pp. 357-399.

Guttman, R., 1984. "R-Tree: A Dynamic Index Structure for Spaital Searching," Proc. SIGMOD Conf. Ann. Metting, Boston, ACM, pp. 47-57.

Henrich, A., Six, H.W., Widmayer, P., 1989. The LSD tree: spatial access to multidimensional point and non point objects. Proceedings o f the Fifteenth International Conference on Vary Large Data Bases, Amsterdam, pp. 45-54.

Kamel, I., Faloutsos, C., 1994. Hilbert R-tree: An Improved R-tree Using Fractals, Proceedings of the 20th VLDB Conference, Santiago, Chile.

Lu, H.G., Ooi, B.C., 1993. Spatial Indexing: Past and Future, IEEE Data Engineering Bulletin, Volume 16, Number 3, September, pp. 16-21.

Nievergelt, J., Hinterberger, H., Sevcik, K.C., 1984. The Grid File: An Adaptable, Symmetric Multikey File Structure, ACM Trans. On Database Systems, Vol. 9, No. 1, pp. 38-71.

robinson, J.T., 1981. The K-D-B-tree: A Search Structure for Large Multidimensional Dynamic Indexes', Proc. ACM SIGMOD Int. Conf. On Management o f Data, pp. 10-18.

Seeger, B., Kriegel, H.P., 1988. Techniques for Design and Implementation of Efficient Spatial Access Methods, Proceedings of the 14th VLDB Conference, Los Angeles, California pp. 360-371.

Seeger B., Kriegel H.-P., 1987. The buddy Tree: An Efficient and Robust Access Method for Spatial Data Base Systems, Proc. 13th Conf. on Very Large Database, Brighton, England, pp. 507-518.

Sellis, T., Roussopoutos, N., Faloutsos, C., 1987. The R+-tree: A Dynamic Index For Multi-Dimensional Objects, Proceedings of the 13th VlDB Conference, Brighton, pp. 507-518.

Shekhar, S., Ravada, S., Liu, X., 1999. Spatial Databases-Accomplishments and Research Needs, IEEE Transactions on Knowledge and Data Engineering, Vol. 11, No. 1, January/February.