International Journal of Database Theory and Application
Vol. 3, No. 1, March, 2010

A Tutorial on Spatial Data Handling

Rituparna Sinha[1], Sandip Samaddar[1], Debnath Bhattacharyya[2], and Tai-hoon Kim[3]

[1]Heritage Institute of Technology
Kolkata-700107, INDIA
rituparna.snh@gmail.com
Hannam University,
Daejeon, South Korea
debnathb@gmail.com
[3]University of Tasmania, Australia
taihoonn@empal.com

Abstract

Spatial data is data related to space. In various application fields like GIS, multimedia information systems, etc., there is a need to store and manage these data. Some datastructures used for the spatial access methods are R tree and its extensions where objects could be approximated by their minimum bounding rectangles and Quad tree based structures where space is subdivided according to certain rules. Also another structure KD Tree is used for organizing points in a k dimensional space.

This paper makes review on some of these Hiearchical datastructures used for handling point data. It focuses on PR Quad Tree and KD Tree.The insertion procedure of these structures is reviewed and analyzed and also a comparison between them is drawn..

Keywords: *Spatial Data, Quad Tree, PR Quad Tree, KD Tree.*

1. Introduction

Conventionally, database systems were designed to manage and process alpha-numeric data represented by character strings, numerical values, date and Boolean or logical expressions There was no provision in conventional relational database systems to support the storage and processing of spatial data represented by points, lines, polygons and surfaces. Spatial Databases were used which supports spatial Data Types.

Spatial information system requires an efficient spatial data handling [9,10]. The large amount of information and complexity of characteristics of data have given rise to new problems of storage and manipulation.

Modern applications are both data and computationally intensive and require [11] the storage and manipulation of voluminous traditional (alphanumeric) and non traditional data sets, such as images, text, geometric objects, time series, audio, video. Examples of such emerging application domains are: geographical information systems (GIS), multimedia information systems, time-series analysis, medical information systems, on-line analytical processing (OLAP) and data mining. These applications impose diverse requirements with respect to the information and the operations that need to be supported. Therefore from the database perspective, new techniques and tools need to be developed towards increased processing efficiency.

Some data structures adopted in most research on spatial access methods, are R trees [6,11] and their extensions R+ trees and R* trees [7], where objects could be approximated by their minimum bounding rectangles [8]. The other approach where space is subdivided according to certain rules is the quad tree data structures.

Another data structure for storing a finite set of points from a k-dimensional space is KD - Tree. It was examined in detail by J.Bentley [1].

2. Introduction to Quad Tree

Quad tree is a 2-D spatial data structure that successively partitions a region of space into 2^2 quadrants or cells. Cells are successively subdivided into smaller cells. The main idea of quad tree structure is repeatedly divide a geometrical space into quadrants and the strategy it follows is similar as divide and conquer strategy. A quad tree has four children; each corresponds to each direction, namely NE, NW, SW and SE. The most studied quad tree approach to region representation is termed as region quad tree, which is proposed by Klinger. The region quad tree is based on a regular decomposition of the space of equal size. An object is put into a quad tree node corresponding to a quadrant if and only if it is inside that quadrant but is not inside any of its children. It is a variation of Maximum block representation where the blocks must be disjoint and must have a standard size, which is in power of 2. It is developed with the intension of representing homogeneous parts of an image in a systematic way. As one property of decomposition is to partition the space infinitely, so any desired degree of resolution is achieved. It is helpful in the field of image processing and database management. If the objects are represented by points, i.e. zero sized elements e.g. the locations of all the cities in a particular state, and then either point quad tree or PR quad tree is used. In the case of PR quad tree the total space is decomposed into quadrants, sub quadrants, until the number of points in each quadrant is within a certain limit. This limit is known as node capacity of the PR quad tree. But, in case of Point quad tree, the space is decomposed into quadrants, sub quadrants, till a point exists in a quadrant.

3. PR Quad Tree

3.1. Introduction to PR Quad Tree

PR quad tree (P for point and R for region) : It was proposed by Hanen Samet[11,12].It is an adaptation of the region quad tree to point data which associates data points with quadrants. The PR quad tree is organized in the same way as the region quad tree (which will be discussed later). The difference is that leaf nodes are either empty or contain a data point and its coordinates. A quadrant contains at most one data point. Data points are inserted into PR quad trees in a manner analogous to that used to insert in a point quad tree - i.e., a search is made for them. Actually, the search is for the quadrant in which the data point says A, belongs (i.e., a leaf node). If the quadrant is already occupied by another data point with different x and y coordinates, say B, then the quadrant must repeatedly be subdivided (termed splitting) until nodes A and B no longer occupy the same quadrant. This may result in many subdivisions, especially if the Euclidean distance between A and B is very small. The shape of the resulting PR quad tree is independent of the order in which data points are inserted into it. Deletion of nodes is more complex and may require collapsing of nodes - i.e., the direct counterpart of the node splitting process outlined above.

In binary search trees, the structure of the tree depends not only upon what data values are inserted, but also in what order they are inserted.

In contrast, the structure of a Point-Region quad tree is determined entirely by the data values it contains, and is independent of the order of their insertion.

In effect, each node of a PR quad tree represents a particular region in a 2D coordinate space. Internal nodes have exactly 4 children (some may be empty), each representing a different, congruent quadrant of the region represented by their parent node.

Internal nodes do not store data. Leaf nodes hold a single data value. Therefore, the coordinate space is partitioned as insertions are performed so that no region contains more than a single point. PR quad trees represent points in a finitely-bounded coordinate space.

Consider the collection of points in a 256 x 256 coordinate space as in figure 1.

A	(100,125)
B	(25,-30)
C	(-55,80)
D	(125,-60)
E	(80,80)

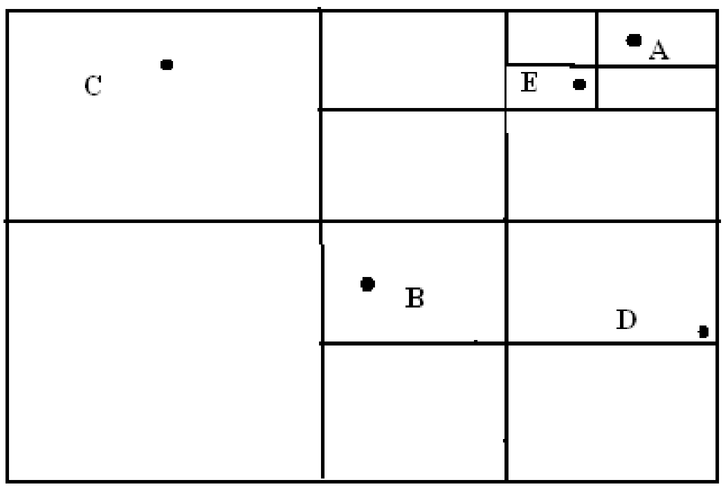


Figure 1. Planer representation of PR Quad Tree

The subdivision of the coordinate space shows how it will be partitioned as the points are added to a PR quadtree.

3.2. Insertion in PR Quadtree

Data points are inserted in a PR Quad Tree in a manner analogous to Point Quad Tree.The algorithm for insertion is as follows. Data points are inserted in a PR Quad Tree in a manner analogous to Point Quad Tree.

The **Algorithm** for insertion is as follow:
A is the point to be inserted.
Root refers to the root node of the tree.
Current refers to the node most recently traversed in the tree
sqdt is the sub-quadrant in which the point to be inserted lies

```
Suppose a point A is to be inserted.
Step1:  (Check for null tree)
            If root =null i.e tree is empty
                      then set Root ←A
                      go to Step 9
           End if
Step2:   current = root
Step3:   if current is not a leaf node
                       sqdt= sub-quadrant in which A lies w.r.t the quadrant represented
                       by  the current node.
           Step 3a: if child node of current corresponding to the sqdt sub-quadrant is empty
                              then go to step8.
                       End if
           Step 3b: set parent=current
                              set current=child node of current corresponding to sqdt sub-quadrant.
                              Go to step 3
           End if
Step 4:  Temp=current
Step 5:  sqdt=subquadrant of parent in which current lies
Step 6:  Create a node (n1) referring the sqdt subquadrant.
          Make n1 the child node of parent corresponding to sqdt subquadrant
                      Set current=n1
                      Partition the quadrant corresponding to current node equally
                       into 4 subquadrants
                             set sqdt = subquadrant in which A lies after partition
          Step 6a:  if sqdt is same as subqquadrant corresponding to temp node 's data
                            then parent=current
                            go to step6.
                     End if
Step7:  Insert temp as the NE or SE or SW or SE child of current node based on the
          sub-quadrant current node in which temp's data lies.
Step8: Insert A as the NE or SE or SW or SE(based on sqdt) child of current node.
Step9: End
```

3.3. Analysis of the Algorithm

The above procedure can be illustrated by the following example in figure 2. And 3and 4. Obviously inserting the first point, A, just results in the creation of a leaf node holding A.

Inserting B causes the partitioning of the original coordinate space into four quadrants, and the replacement of the root with an internal node with two nonempty children:

A	(100,125)
B	(25,-30)

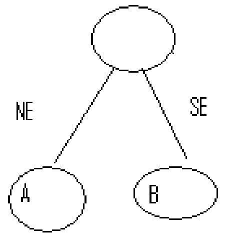
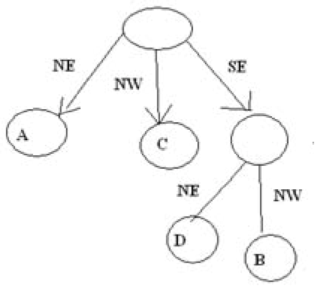
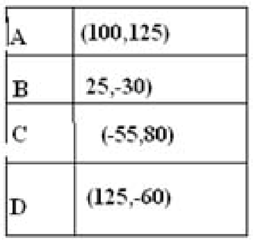


Figure 2. Insertion of A and B in PR Quad Tree

The display above shows the SE and NE corners of the regions logically represented by each node, and the data values stored in the leaf nodes. In an implementation, nodes would not store information defining their regions explicitly, nor would empty leaf nodes probably be allocated.

Inserting C does not cause any additional partitioning of the coordinate space since it naturally falls into an empty leaf:





A	(100,125)
B	25,-30)
C	(-55,80)
D	(125,-60)

Figure 3. Insertion of C in the PR Quad Tree.

Inserting D will cause the partitioning of the SE quadrant in order to separate B and D.

Suppose the value E (80, 80) is now inserted into the tree. It falls in the same region as the point A, (0, 0) to (128, 128).

However, dividing that region creates three empty regions, and the region (64, 64) to (128, 128) in which both A and E lie. So, that region must be partitioned again. This separates A and E into two separate regions (see illustration on the slide "Coordinate Space Partitioning") If it had not, then the region in which they both occurred would be partitioned again, and again if necessary, until they are separated. Inserting E results in the following tree.

A	(100,125)
B	(25,-30)
C	(-55,80)
D	(125,-60)
E	(80,80)

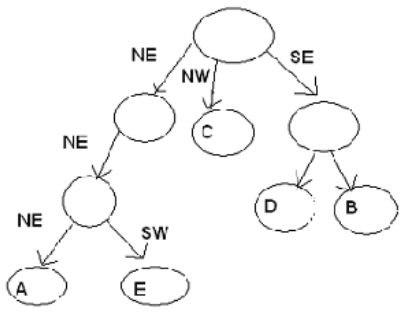


Figure 4. Insertion of D and E in PR Quad Tree.

Insertion proceeds recursively, descending until the appropriate leaf node is found, and then partitioning and descending until there is no more than one point within the region represented by each leaf. If we change the order of insertion i.e. A, B, D, C then E.,we can say that in a PR Quad Tree

1.The shape of the tree is entirely independent of the order in which the data elements are added to it.
2. In the worst case it may happen that the points are so closely spaced that the number of partitions increases and the tree level goes on increasing and results in a imbalanced skewed tree**.**

4. The KD-Tree (Organization of point data)

4.1 Introduction to KD Tree

A KD-tree is a data structure for storing a finite set of points from a k-dimensional space. It was examined in detail by J.Bentley [1]. In [1] Bentley described the KD-tree as a k-dimensional binary search tree. A node in the tree (Figure below) serves two purposes: representation of an actual data point and direction of a search. A discriminator D (P), whose value is between 0 and k-1 inclusive, is used to indicate the key on which the branching decision depends. A node P has two children, a left son L (P) and a right son R (P). If the discriminator value of node P is the jth attribute (key), then the jth attribute of any node in the L(P) is less than jth attribute of node P, and the jth attribute of any node in the H(P) is greater than or equal to that of node P.

The basic form of the KD-tree stores K-dimensional points. This section concentrates on the two-dimensional (2D) case shown in figure 5 and 6. Each internal node of the KD-tree contains one point and also corresponds to a rectangular region.

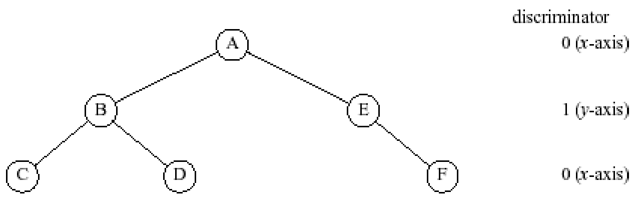


Figure 5. The structure of a KD-tree.

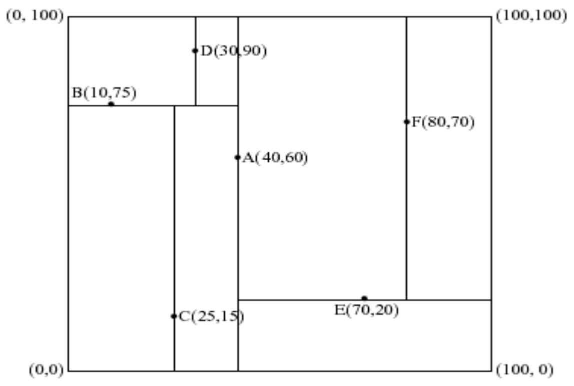


Figure 6. The planar representation of KD Tree.

The root of the tree corresponds to the whole region of interest. The rectangular region is divided into two parts by the x-coordinate of the stored point and by the y-coordinate alternately. A new point is inserted by descending the tree until a leaf node is reached. At each internal node the value of the proper coordinate of the stored point is compared with the corresponding coordinate of the new point and the proper path is chosen. This continues until a leaf node is reached. This leaf also represents a rectangular region, which in turn will be divided into two parts by the new point.

In [2] also it is described how the nearest neighbor algorithm is applied to KD trees. KD trees are most often utilized as a primary storage structure, but the **extended K-d tree** [18] has been modified for secondary storage This structure uses the K-d structure as a directory to data stored in pages, but only the leaves of the directory contain page pointers. Each internal node stores the discriminator field, although all nodes at the same level have the same discriminator.

A second K-d tree hybrid, called the **K-D-B tree** [16,17,4], has evolved for very large indexes KD- B trees combine features of B-trees and K-d trees. This structure has two types of nodes or pages, which are called region pages, and point pages Region pages define region boundaries and page ID's for each associated region. Point pages contain indexes to point data base records. As in a B-tree, a point is inserted and splitting may ripple all the way to the root, thus the tree is always perfectly balanced and grows from the leaf towards the root. The splitting, however, uses the discriminator concept when deciding which way to split a region.

A third K-d tree hybrid 1s defined as an **optimized K-d tree** for finding best matches [19] This hybrid is a static structure Discriminators are not tagged by node position in the tree, in fact, nodes at the same level need not have the same discriminator The algorithm to build the optimized tree chooses the key with the largest spread as the discriminator, and the median of the discriminator key values as the partition

4.2. The Algorithm for Insertion in a KD Tree

In 1975 Bentley described the algorithm for insertion as follows. A node P is passed which is not in the tree**.**

```
Step1 → (Check for null tree)
        If root =null i.e tree is empty
           Then set Root ←A
           Go to Step 7
        End if
Step2 →Set current = root
Step3 →Set Discriminator=x
Step4 → if current is not null
           If discriminator=x then
                If x coordinate of P less than x coordinate of current node
                      Parent=current
                      Current=left child (current)
                Else
                      Parent=current
                      Current=right_child (current)
                Endif
                Discriminator=y
           Else
```

```
                    If y coordinate of P less than y coordinate of current node
                          Parent=current
                          Current=left_child (current)
                    Else
                          Parent=current
                          Current=right_child (current)
                    Endif
                    Discriminator=x
              End if
              Go to step 4
          End If
Step 5→ Create a new node ndnew representing the point P
Step 6→ if discriminator =x then
              If y coordinate of P less than y coordinate of parent node
                    Set left child (parent)= ndnew
              Else     Set right_child (parent)= ndnew
           End if
          Else
              If x coordinate of P less than x coordinate of current node
                    Set left child (parent)= ndnew
                   Else     Set right_child (parent)= ndnew
              End if
          End If
Step 7→End
```

4.3. Analysis of the Algorithm

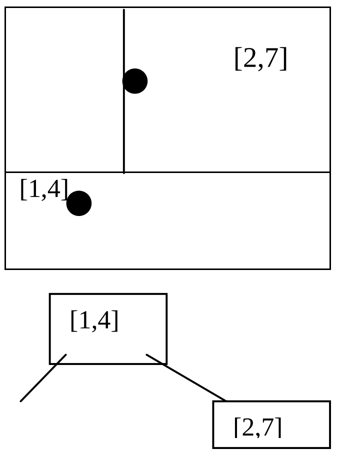


Figure 7. Planer representation and nodes.

From this algorithm we can say that:

1) The shape of the tree depends on the order in which the points are inserted. This can be illustrated by the following example.

Suppose we are inserting the points [1,4], [2,7], [5,2], [7,8] in this order. So first [1,4] becomes the root node shown in figure 7. The [1,4] node splits along the y=4 plane. Now [2,7] point comes and compares its y coordinate value with [1,4] node's y coordinate and is positioned left in the planer representation and becomes right child of [1,4]. Now [2,7] node splits along X=2 plane.

Now insertion of [5,2] will be on the left of [1,4], since by comparing y coordinates of root [1,4], this node becomes left child of [1,4].And for [7,8] we start with root and compare its y coordinate value with roots y coordinate value and then moving down ,the node's x coordinate value is compared with x coordinate value of [2,7],and becomes the right child of [2,7] shown in figure 8.

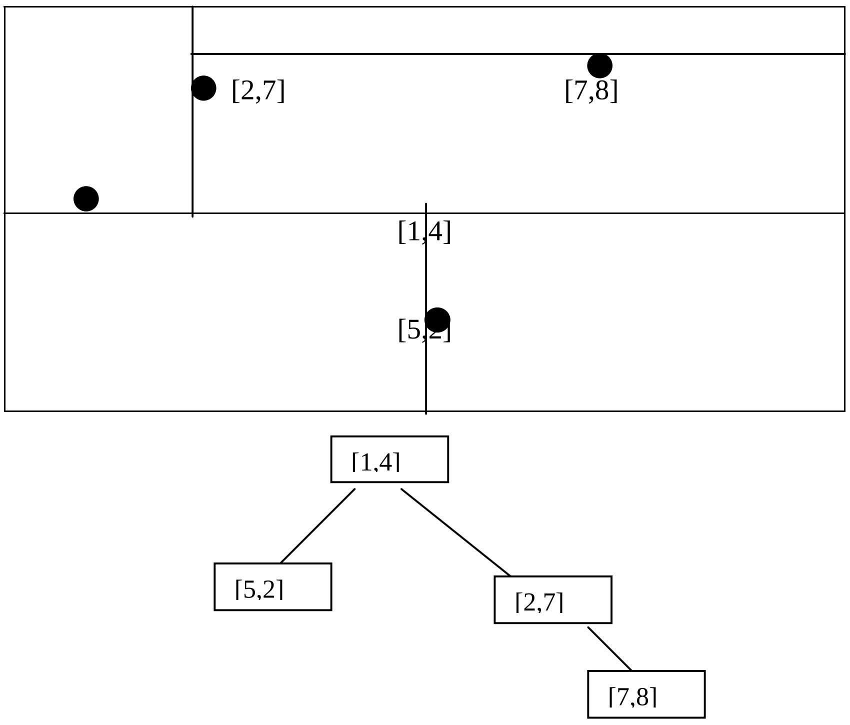
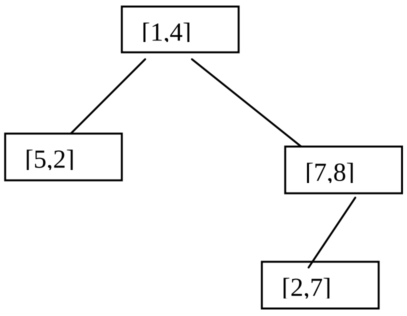


Figure 8. Planer representation and nodes.

Now if we change the order of insertion i.e. we insert [1,4], [7,8],[2,7] then [5,2] we find the tree is as shown below



So we can conclude that

1) the shape of the tree depends on the order in which the insertion is taking place.

2) In the worst case the number of levels in the tree becomes equal to the number of points that is inserted.

Suppose that points [2,5] , [3,6] ,[4,7],[6,8] are to be inserted. Then the tree that is formed in figure 9.

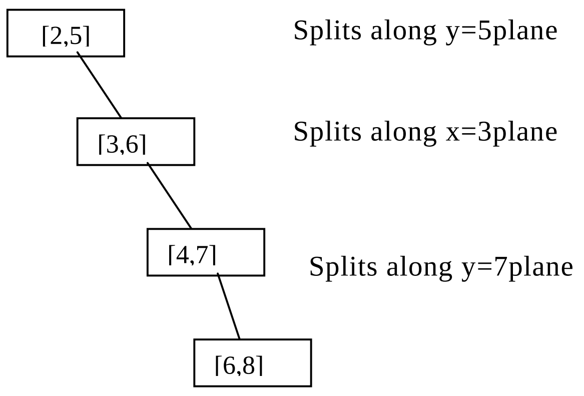


Figure 9. The KD-tree.

5. Comparison Study

1)**PR Quad** Tree is a tree data structure used to partition a 2 dimensional space by recursively subdividing it into 4 quadrants or regions
Whereas a **KD Tree** is a k dimensional tree data structure used for organizing points in a k dimensional space.

2) In a **PR Quad** Tree each node of a PR quad tree represents a particular region in a 2D coordinate space.
Whereas every node in a **KD Tree is a** k dimensional point.

3) In a **PR Quad** Tree non-leaf nodes do not store data. Internal nodes have exactly 4 children (some may be empty), each representing a different, congruent quadrant of the region represented by their parent node.
Whereas in a **KD Tree** even non-leaf nodes is a k dimensional point. Every non leaf node generates a splitting hyper plane that divides the space into 2 subspaces

4) In **PR Quad tree** only leaf nodes contain data points
Whereas in a **KD Tree** every node is a k dimensional point

5) **In a PR Quad Tree** the structure of the tree is determined entirely by the data values it contains, and is independent of the order of their insertion**.**

Whereas in a **KD Tree** the shape of the tree depends on the order in which the points are inserted.

6) **In a PR Quad Tree** dense data regions require more partitions and therefore the quad tree will not be balanced in this situation
Whereas In a **KD Tree in** the worst case the number of levels in the tree becomes equal to the number of points that is to be inserted.

6. Conclusion

We have shown in this paper the methods of the storage and manipulation of voluminous spatial data sets, which were really a problem in conventional Database.
Among spatial data set we have considered only point data and its storage and access methods. We have focused mainly on one Quad Tree based structure, PR Quad Tree and another Hierarchical Data structure KD Tree used for organizing point data. There are also other structures like point Quad Tree and some extended version of KD Tree, R Tree which are not discussed in detail.
We have mainly stressed on the insertion procedure of these structures. The deletion procedure is also equally important which is not covered in this paper.

References

[1] Bentley, J.L, “Multidimensional binary search trees used for associated searching”, Comm. Of the ACM 18 (1975), pp. 509-517.
[2] Andrew W Moore, “An introductory tutorial on KD trees”, Technical Report No. 209, Computer Laboratory, University of Cambridge, 1991.
[3] Mark H. Overmars and Jan van Leeuwen,” Dynamic Multi-Dimensional data structures based on Quad and KD tress”, Acta Informatica 17, (1982) pages 267-285
[4] Ralf Hartmut Güting, “An Introduction to Spatial Database Systems”, The VLDB Journal — The International Journal on Very Large Data Bases, Volume 3 , Issue 4 (October 1994), Pages 357 - 399
[5] Hanen Samet, “The Quadtree and Related Hierarchical Data Structures”, ACM Computing Surveys (CSUR) Volume 16 , Issue 2 (June 1984),Pages: 187 - 260
[6] Guttman,.”R-Trees: A Dynamic index structure for spatial searching “, in proceedings ACM SIGMOD, 1984 pp, 47-57.
[7]N.Beckmann; H.P. kriegel, R.Schneider, B.Seeger, ”The R* Tree-An efficient and Robust Access Method for points & Rectangles”,in proceedings ACM SIGMOD 90 International Conference on management of data, 1990.
[8]T.Brinkhoff;H.P Kriegel,B.Seeges,”Efficient Processing of spatial join using R-Trees”, in proceedings ACM SIGMOD’93 International Conference on Management, 1993.
[9] S.K. Chang, E.Jungert, Y.Li,”The Design of Pictorial Database based upon the theory of symbolic projections”, in preceedings 1st International Symposium on large Spatial databases,santa Barbara,USA, 1989,pp.303-323
[10] E.Nardelli, G.Projetti, ,”Time & space efficient secondary memory representation of Quadtrees ”information system vol 22.N 1, 1997, pp 25-37
[11] Apostolos N. Papadopoulos and Yannis Manolopoulos ,“Nearest Neighbor search, A Database Perspective”, Springer, January 2005.
[12] H. Samet: "The Design and Analysis of Spatial Data Structures", Addison-Wesley, Reading MA, 1990.
[13] H. Samet; "Applications of Spatial Data Structures", Addison-Wesley, Reading MA, 1990
[14] R.A. Finkel and J.L. Bentley ,”Quad Trees: A Datastructure for retrieval of composite keys” , April 8, 1974.
[15] Overmars, M H and Van Leeuwen, Jan Overmars, M H and Van Leeuwen, Jan “Multikey retrieval from K-d trees and QUAD-trees”, International Conference on Management of Data, 1985, Pages: 291 - 301
[16] RobInson, J T ,”The K-D-B-Tree A Search Structure for Large, Multidimensional Dynamic Indexes”, 4CM-SIGMOD 1981 International, Conference 0" Management -o-f Data Association.
[17] Van Leeuwen , J and Overmars, M H ,” Stratified Balanced Search Trees”, Acta, Informatica, Vol 18, 1983 Pp. 345-359.

[18] Chang, J M and Fu , KS,” Dynamic Clustering Techniques for Physical Database, Design “,International Conference on Management of Data, 1980, Pages: 188 - 199

[19] Friedman, J H , Bentley, J L , and Finkel, RA ,” An Algorithm for Finding Best Matches in Logarithmic Expected Time”, ACM Transactions on Mathematical Software vol 3, No 3, 1977, pp. 209-226.

Authors

Rituparna Sinha, has completed her M.Tech from National Institute of Technical Teachers Training & Research (NITTTR,Kolkata) in the Department of computer science and Engineering. She has worked as a Lecturer in Mallabhum Institute of Technology ,in the Department of Information Technology. Presently she is working as a Lecturer of IT Department in Heritage Institute of Technology. She has 4 years of teaching experience. Her research interest field is on medical imaging.

Sandip Samaddar has completed his M.Tech from National Institute of Technical Teachers Training & Research (NITTTR,Kolkata) in the Department of computer science and Engineering. He has worked as a Lecturer in Mallabhum Institute of Technology, in the Department of Information Technology. Presently He is working as a Lecturer of IT Department in Heritage Institute of Technology. He has 3 years & 5 months of teaching experience. His research field of interest is on bio-informatics.

Debnath Bhattacharyya, M.Tech in Computer Science and Engineering from West Bengal University of Technology, Kolkata. Currently, he is associated as a Professor with the Multimedia Engineering Department at Hannam University, Daejeon. He has 15 years of experience in Teaching and Administration. His research interests include Bio-Informatics, Image Processing and Pattern Recognition. He has published 67 Research Papers in International Journals and Conferences and 6 Text Books for Computer Science.

Prof. Tai-hoon Kim, M.S., Ph. D (Electricity, Electronics and Computer Engineering), currently, Professor of Hannam University, Korea. His research interests include Multimedia security, security for IT Products, systems, development processes, operational environments, etc. He has 14 Years of experience in Teaching & Research. He has already got distinctive Academic Records in international levels. He has published more than 100 Research papers in International & National Journals and Conferences.