www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 5 May 2016, Page No. 16584-16588

Nearest Neighbor Browsing and Search with Keywords in Spatial Databases

Sphurti S. Sao[1], *Dr. Rahila Sheikh*[2]

[1]M. Tech Student IV Sem, Dept of CSE, RCERT Chandrapur,
MH, India RCERT Chandrapur, MH, India

[2]Assistant Professor, Dept of CSE, RCERT Chandrapur,
MH, India RCERT Chandrapur, MH, India

Abstract— ***Users may search for different type of things from anywhere. But Search results depend on the user entered query which has to satisfy their searched properties that is stored in the spatial database. Due to rapid growth of users it becomes essential to optimize search results based on nearest neighbour property in spatial databases. Conventional spatial queries, such as range search and nearest neighbour retrieval, involve only geometric properties of objects which satisfies condition on geometric objects. Nowadays many modern applications aim to find objects satisfying both a spatial condition and a condition on their associated texts which is known as Spatial keyword search. For example, instead of considering all the hotels, a nearest neighbor query would instead ask for the hotel that is closest to among those who provide services such as pool, internet at the same time. For this type of query a variant of inverted index is used that is effective for multidimensional points and comes with an R-tree which is built on every inverted list, and uses the algorithm of minimum bounding method that can answer the nearest neighbor queries with keywords in real time.***

Keywords – Spatial Database, nearest neighbour search, R-tree, Keyword search, spatial queries.

1. INTRODUCTION

Nearest neighbor search (NNS) also known as proximity search, similarity search or closest point search, is an optimization problem for finding closest points or similar points. Closeness is typically expressed in terms of dissimilarity function. The less similar are the objects, the larger are the function values.

Various solutions to the NNS problem have been proposed. The quality and usefulness of the algorithms are determined by the time complexity of queries as well as the space complexity of any search data structures that must be maintained. The informal observation usually referred to as the curse of dimensionality states that there is no general-purpose exact solution for NNS in high-dimensional Euclidean space using polynomial preprocessing and poly-logarithmic search time. Some of the solutions to the NNS problems are mentioned below:

- Linear Search
- Space Partitioning
- Locality sensitive hashing
- NNS in spaces with small intrinsic dimensions
- Projected radial search
- Compression / Clustering based search and more

A spatial database or geodatabase is a database that is optimized to store and query data that represents objects defined in a geometric space. Most spatial databases allow representing simple geometric objects such as points, lines and polygons. Some spatial databases handle more complex structures such as 3D objects, topological coverage, linear networks. While typical databases are designed to manage various numerics and character types of data, additional functionality needs to be added for databases to process spatial data types efficiently. These are typically called geometry or feature [1].

Features of Spatial Database:

Database systems use indexes to quickly look up values and the way that most databases index data is not optimal for spatial queries. Instead, spatial databases use a spatial index to speed up database operations.

In addition to typical SQL queries such as SELECT statements, spatial databases can perform a wide variety of spatial operations. The following operations and many more are specified by the Open Geospatial Consortium standard [1]:

- *Spatial Measurements*: Computes line length, polygon area, the distance between geometries, etc.
- *Spatial Functions*: Modify existing features to create new ones, for example by providing a buffer around them, intersecting features, etc.
- *Spatial Predicates*: Allows true/false queries about spatial relationships between geometries. Examples include "do two polygons overlap" or 'is there a residence located within a mile of the area we are planning to build the landfill?'

- *Geometry Constructors*: Creates new geometries, usually by specifying the vertices (points or nodes) which define the shape.

- *Observer Functions*: Queries which return specific information about a feature such as the location of the center of a circle.

The importance of spatial databases is reflected by the convenience of modeling entities of reality in a geometric manner. For example, locations of restaurants, hotels, hospitals and so on are often represented as points in a map, while larger extents such as parks, lakes, and landscapes often as a combination of rectangles. Many functionalities of a spatial database are useful in various ways in specific contexts. For instance, in a geography information system, range search can be deployed to find all hospital in a certain area, while nearest neighbor retrieval can discover the hospital closest to a given address. Today, the widespread use of search engines has made it realistic to write spatial queries in a brand-new way.

Conservative spatial queries, such as range search and nearest neighbor retrieval, involve only conditions on objects' numerical properties. We have seen some modern applications that call for the ability to select objects based on both of their geometric coordinates and their associated texts. The major drawback of these straightforward approaches is that they will fail to provide real time answers on difficult inputs. A typical example is that the real nearest neighbor lies quite far away from the query point, while all the closer neighbors are missing at least one of the query keywords.

The spatial database is being referred to the database which contains geographical information like coordinate of any location or point. The higher areas are shown by rectangles and planes etc. Finding the particular location in such database from particular point is difficult task. The location search along with keyword is again a tedious job. If you want to search a restaurant which serves bread, nuggets and brandy from such database from your point will need to write spatial queries. Such queries need proper indexing otherwise it will drastically affect the performance of searching. So the traditional method such as IR tree carries same drawback which we are trying to sort out at some extent.

The main objective of our application is to derive the best searching scheme for spatial database. As spatial database consists of multidimensional points, rectangles, planes etc, It becomes very difficult to find the best solution to our search query.

In this paper, we are going to design a system which will help to find the nearest neighbor location of a query with the help of R-tree and minimum bounding method where spatial database consists of large spatial objects, and to find the correct result it will take more time. So to obtain the search result in less time the proposed system will use R-tree indexing structure. By using indexing structure the time required for searching will be less. And also the accuracy of the system will be our priority and for that reason we use minimum bounding method with R-tree.

2. RELATED WORK and LITERATURE SURVEY

Nearest neighbour search (NNS), also known as closest point search, similarity search. It is an optimization problem for finding closest (or most similar) points. We can search closest point by giving keywords as input; it can be spatial or textual.

Yufie Tao and Cheng Sheng [2], developed a new access method called the spatial inverted index that extends the conventional inverted index to cope with multidimensional data, and comes with algorithms that can answer nearest neighbor queries with keywords in real time. we design a variant of inverted index that is optimized for multidimensional points, and is thus named the spatial inverted index (SI-index). This access method successfully incorporates point coordinates into a conventional inverted index with small extra space, owing to a delicate compact storage scheme. Meanwhile, an SI-index preserves the spatial locality of data points, and comes with an R-tree built on every inverted list at little space overhead.

Cao et al. [3], proposed collective spatial keyword querying, they present the new problem of retrieving a group of spatial objects, each associated with a set of keywords. We develop approximation algorithms with provable approximation bounds and exact algorithms to solve the two problems.

Lu et al. [4], combined the notion of keyword search with reverse nearest neighbor queries. propose a hybrid index tree called IUR-tree (Intersection-Union R-Tree) that effectively combines location proximity with textual similarity. Based on the IUR-tree, we design a branch-and-bound search algorithm.

Cong et al.[5], proposed the concept of prestige-based spatial keyword search. The central idea is to evaluate the similarity of an object p to a query by taking also into account the objects in the neighborhood of p.

G. Cong, C.S. Jensen, and D. Wu [6] proposed a approach that computes the relevance between the documents of an object p and a query q. This relevance score is then integrated with the Euclidean distance between p and q to calculate an overall similarity of p to q. The few objects with the highest similarity are returned. In this way, an object may still be in the query result, even though its document does not contain all the query keywords.

I.D Felipe, V. Hristidis and N. Rishe [7], object texts are utilized in evaluating a boolean predicate, i.e., if any query keyword is missing in an object's document, it must not be returned. Neither approach subsumes the other, and both make sense in different applications. As an application in our favor, consider the scenario where we want to find a close restaurant serving "steak, spaghetti and brandy", and do not accept any restaurant that do not serve any of these three items. In this case, a restaurant's document either fully satisfies our requirement, or does not satisfy at all.

Y.-Y. Chen, T. Suel, and A. Markowetz [8], have studied efficient query processing in geographic web search engines. They discussed a general framework for ranking search results based on a combination of textual and spatial criteria, and proposed several algorithms for efficiently executing ranked queries on very large collections. They integrated their algorithms into an existing high-performance search engine query processor and evaluated them on a large data set and realistic geographic queries. Their results show that in many cases geographic query processing can be performed at about the same level of efficiency as text-only queries.

V. Hristidis and Y. Papakonstantinou [9], presented DISCOVER, a system that performs keyword search in relational databases. It proceeds in three step. First it generates the smallest set of candidate networks that guarantee that all *MTJNT*'s will be produced. Then the greedy algorithm creates a near-optimal execution plan to evaluate the set of candidate networks. Finally, the execution plan is executed by the DBMS.

3. PROPOSED METHODOLOGY

We are trying to create an application in which we have to import spatial datasets into relational database for processing. Our system is based on R-tree and performs searching operations on it. In this paper, we are going to discuss about R-tree and various operations perform on it. The main purpose of this application is to find the nearest location of the input query.

Spatial data, also known as geospatial data, is information about a physical object that can be represented by numerical values in a geographic coordinate system. Spatial data represents the location, size and shape of an object on planet such as a building, lake, mountain or township. Spatial data may also include attributes that provide more information about the entity that is being represented. After gathering dataset we create an indexes on those datasets.

R-Tree: R-trees are tree data structures used for spatial access methods, i.e., for indexing multi-dimensional information such as geographical coordinates, rectangles or polygons.

The key idea of the data structure is to group nearby objects and represent them with their minimum bounding rectangle in the next higher level of the tree; the "R" in R-tree is for rectangle. Since all objects lie within this bounding rectangle, a query that does not intersect the bounding rectangle also cannot intersect any of the contained objects. At the leaf level, each rectangle describes a single object; at higher levels the aggregation of an increasing number of objects.

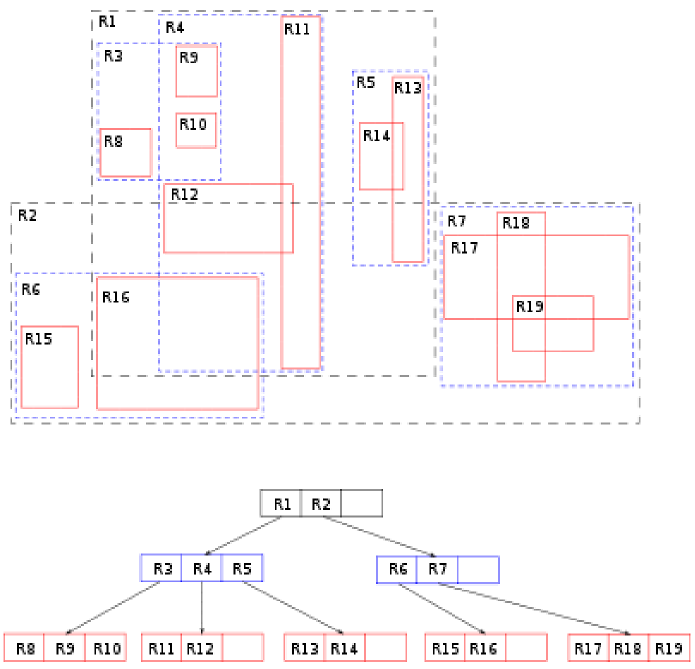

Figure 1: Example of R-tree

Similar to the B-tree, the R-tree is also a balanced search tree organizes the data in pages, and is designed for storage on disk. Each page can contain a maximum number of entries, often denoted as M. It also guarantees a minimum fill (except for the root node), however best performance has been experienced with a minimum fill of 30%–40% of the maximum number of entries. The reason for this is the more complex balancing required for spatial data as opposed to linear data stored in B-trees.

The key difficulty of R-trees is to build an efficient tree that on one hand is balanced (so the leaf nodes are at the same height) on the other hand the rectangles do not cover too much empty space and do not overlap too much (so that during search, fewer sub-trees need to be processed). For example, the original idea for inserting elements to obtain an efficient tree is to always insert into the sub-tree that requires least enlargement of its bounding box. Once that page is full, the data is split into two sets that should cover the minimal area each. Most of the research and improvements for R-trees aims at improving the way the tree is built and can be grouped into two objectives: building an efficient tree from scratch (known as bulk-loading) and performing changes on an existing tree (insertion and deletion).

R-trees do not guarantee good worst-case performance, but generally perform well with real-world data. While more of theoretical interest, the (bulk-loaded) Priority R-tree variant of the R-tree is worst-case optimal, but due to the increased complexity, has not received much attention in practical applications so far.

When data is organized in an R-tree, the k nearest neighbors of all points can efficiently be computed using a spatial join. This is beneficial for many algorithms based on the k nearest neighbors, for example the Local Outlier Factor. De-Li-Clu, Density-Link-Clustering is a cluster analysis algorithm that uses the R-tree structure for a similar kind of spatial join to efficiently compute an OPTICS clustering.

Operations on R-tree:

When dynamic structure R-Tree is designed, you can efficiently complete operations on spatial database, such as search, insert, delete, node splitting, updates and other operations. The application will work in the way as shown in below figure.

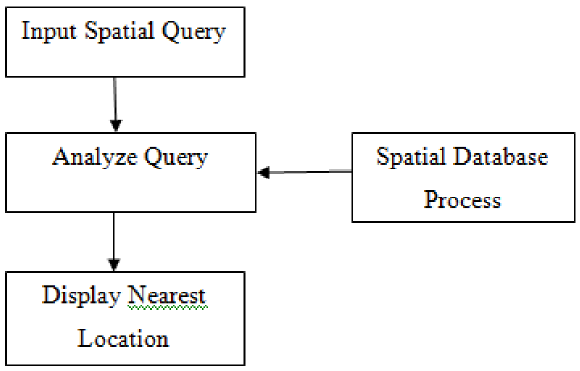

Figure 2: Flow Diagram of the System

A) Search:

Search algorithm accomplishes the following task, given an R-Tree whose root node is T, find all index records whose rectangles overlap a search rectangle S. We denote an entry in a node as E(EI, EP), where EI represents the smallest rectangle bounding the sub-tree or the spatial object, EP is the pointer to the sub-tree or the spatial object.

```
SearchSubTree(t, s)
1. If t is not a leaf
2.then for each entry E in t do
3. if EI overlaps S
4. then SearchSubTree(EP, s)
5. else SearchLeaf(T, s)
SearchLeaf(t, s)
1. for each entry E in t
2. do if EI overlaps s
3. then output E
```

Searching an R-Tree is unlike searching an B-Tree, All internal nodes whose minimal bounding rectangles intersect with the search rectangle may need to be visited during a search.

We can apply the searching of an R-tree to find objects that overlap a search object, say o, by the following steps.

```
SearchObj(t, o)
1. s←bounding box of the search object o
2. SearchSubTree(t,s)
```

and revise the above SearchLeaf(t,s) as follows:

```
SearchLeaf(t,s)
1. for each entry E in t
2. do if EI = s
3. then if EP = o
4.then output E
```

B) Insertion:

Like insertion in B-Tree, inserting new data tuple into R-Tree may cause splitting nodes and the splits propagate up the tree. Furthermore, an insertion of a new rectangle can increase the overlap of the nodes. Choosing which leaf to insert a new rectangle and how to split nodes during re-balancing are very critical to the performance of R-Tree

Algorithm Insert: Insert a new index entry E into an R-Tree T.

```
Insert(E, t)
1. L←ChooseLeaf(E, t) > select a leaf node L where to place E
2. If L need not split
3. then install E
4. else SplitNode(L)
5. AdjustTree(L)
ChooseLeaf(E,t)
1. N←t
2. while N is not a leaf
3.do choose the entry F in N whose rectangle FI needs least enlargement to include EI
4.N←FP
5. return N
```

C) Deletion:

Algorithm Delete: Remove an index record E from an R-Tree

```
Delete(E, t)
1. L←FindLeaf(E, t)
2. If L is null
3. Then return
4. Remove E from L
5. CondenseTree(L)
6. If the root node has only one child.
7. then make the child the new root.
FindLeaf(E, t)
1. if t is not a leaf
2. then for each entry F in t
3. do if FI overlaps EI
4. then FindLeaf(E, FP)
5. else for each entry F in T
6. do if FI = EI & FP=EP
7. then return T
```

Analyzing Spatial Queries

Next, after creating the indexes we will analyze the spatial query, the query will be searched. The latitude longitude and keyword of the location will be entered and in the output nearest location of the query will be displayed. The minimum bounding method (MBM) performs a single query, but uses the minimum bounding rectangle to prune the search space. Specifically, starting from the root of the R-tree for dataset, MBM visits only nodes that may contain candidate points.

4. CONCLUSION

The proposed system will use the indexing structure R tree. The system will use the group nearest neighbor technique for spatial queries, which uses minimum bounding method.

This method will use the minimum bounding rectangle to prune the search space.MBR are frequently used as an indication of the general position of a geographic feature or dataset for either display, first approximation spatial query or spatial indexing purpose. The system is working well as it shows the nearest location based on query without taking too much time.

REFERENCES

[1] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, "Processing Spatial-Keyword (SK) Queries in Geographic Information Retrieval (GIR) Systems," Proc. Scientific and Statistical Database Management (SSDBM), 2007.

[2] Yufei Tao and Cheng Sheng, "Fast Nearest Neighbor Search with Keywords", IEEE transactions on knowledge and data engineering, VOL. 26, NO. 4, APRIL 2014.

[3] X. Cao, L. Chen, G. Cong, C.S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M.L. Yiu, "Spatial Keyword Querying," Proc. 31st Int'l Conf. Conceptual Modeling (ER), pp. 16-29, 2012.

[4] J. Lu, Y. Lu, and G. Cong, "Reverse Spatial and Textual k Nearest Neighbor Search," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 349-360, 2011

[5] X. Cao, G. Cong, C.S. Jensen, and B.C. Ooi, "Collective Spatial Keyword Querying," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 373-384, 2011.

[6] G. Cong, C.S. Jensen, and D. Wu, "Efficient Retrieval of the Top-k Most Relevant Spatial Web Objects," PVLDB, vol. 2, no. 1, pp. 337- 348, 2009.

[7] I.D. Felipe, V. Hristidis, and N. Rishe, "Keyword Search on Spatial Databases," Proc. Int'l Conf. Data Eng. (ICDE), pp. 656-665, 2008.

[8] Y.-Y. Chen, T. Suel, and A. Markowetz, "Efficient Query Processing in Geographic Web Search Engines," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 277-288, 2006.

[9] V. Hristidis and Y. Papakonstantinou, "Discover: Keyword Search in Relational Databases," Proc. Very Large Data Bases (VLDB), pp. 670-681, 2002.

[10] T.Miranda Lakshmi , A.Martin , R.Mumtaj Begum, Dr.V.Prasanna Venkatesan, "An Analysis on Performance of Decision Tree Algorithms using Student's Qualitative Data", I.J.Modern Education and Computer Science, 2013, 5, 18-27 Published Online June 2013 in MECS