# Nearest Neighbor Search with Keywords in Spatial Databases

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

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

[2] Head of Department, Dept of CSE, RCERT Chandrapur, MH, India

**Abstract - In real world, there are billions of rows in a spatial database. If someone want to search for a location or place, it searches all the rows and return the result. Practically there can be only few rows in the database which are of importance to use. As with many pioneering solutions, the IR2-tree has a few drawbacks that affect its efficiency. The most serious issue among all is that the number of false hits can be really very large when the object of final result is far away from the query point, or the result is empty. In such cases, the query algorithm would need to load the documents of many objects, causing expensive overhead as each loading necessitates a random access. So if search is performed only in the used data subspace, the execution time would be saved. We propose such system which can implement this efficiently with the help of R-tree and Nearest neighbor algorithm using inverted Index spatial R-Tree to solve this problem.**

**Keywords -** *Nearest Neighbor Search, R-Tree, Spatial Database, Spatial Query.*

## 1. Introduction

Nearest neighbor search (NNS) also called as proximity search, similarity search or closest point search, is an optimization problem for finding closest points or similar points.. In a set of points Nearest neighbor search which gives the nearest neighbor of a query point, is an important and widely studied problem in many fields, and it has variety of applications. We can search closest point by providing keywords as input; it can be spatial or textual. Many functions of a spatial database are useful in different ways in specific contexts. For example, in a geography information system, range search can be used to find all restaurants in a certain area, while nearest neighbor retrieval can discover the restaurant closest to a given address [1].Some of the solutions to the NNS problems are as follows:

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

### 1.1 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 and polygons. R-tree might be to store spatial objects such as locations of restaurant or the polygons that typical maps are made of: streets, buildings, outlines of lakes, coastlines, etc. and then find answers quickly to queries such as "Find all museums within range of my current location", or "find the nearest gas station". The R-tree can also accelerate nearest neighbor search for various distance metrics, including great-circle distance.

Data in R-trees is organized in pages that can have variable number of entries (up to some pre-defined maximum fill, and usually above a minimum fill). Each entry within a non-leaf node stores two pieces of information: a way of identifying a child node, and the bounding box of all entries within this child node. Leaf nodes store the data required for each child, often a point or bounding box representing the child and an external identifier for the child node. For point data, the leaf entries can be just the points themselves.

### 1.2 Minimum Spatial Bounding

The minimum bounding rectangle (MBR) also called as bounding box or envelope, is an expression of the maximum extents of a 2-dimensional object (e.g. point, line, polygon) or set of objects within its (or their) 2-D (x, y)coordinate system, in other words min(x), max(x), min(y), max(y). The MBR is a 2-dimensional case of the minimum bounding box. In geometry, the minimum/smallest bounding or enclosing box for a point set (*S*) in *N* dimensions is the box with the smallest measure (area, volume or hyper volume in higher dimensions) within which all the points lie.

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

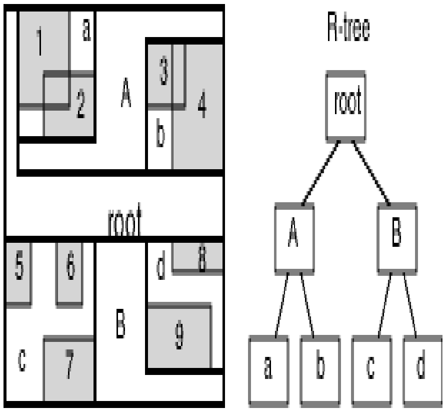


**Figure 1:** R-Tree index on MBR

## 2. Existing Work

Some noticeable work done in Nearest neighbor search (NNS); an optimization problem for finding closest (or most similar) points.

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.

V. Hristidis and Y. Papakonstantinou[10], presented DISCOVER, a system that performs keyword search in relational databases. It works in three step. First it generates the small set of candidate networks that guarantee that all *MTJNT*'s will be produced. Then the greedy algorithm produces a near-optimal execution plan to evaluate the set of candidate networks. Finally, the execution plan is executed by the DBMS.

G.R. Hjaltason and H. Samet[11] proposed data to find the highest query results by accessing a bottom portion of the IR2-Tree. This work has the subsequent contributions. The matter of top-n spatial keyword search is outlined. The IR$^2$-Tree is an economical categorization structure to store spatial and matter data for a group of objects. Economical algorithms also are bestowed to take care of the IR$^2$-Tree, that is, insert and delete objects.

## 3. Proposed Work

The objectives behind the proposed work are as follows:

- To find the nearest location of a query.
- To reduce delay in searching.
- To enhance accuracy for the result of a query.

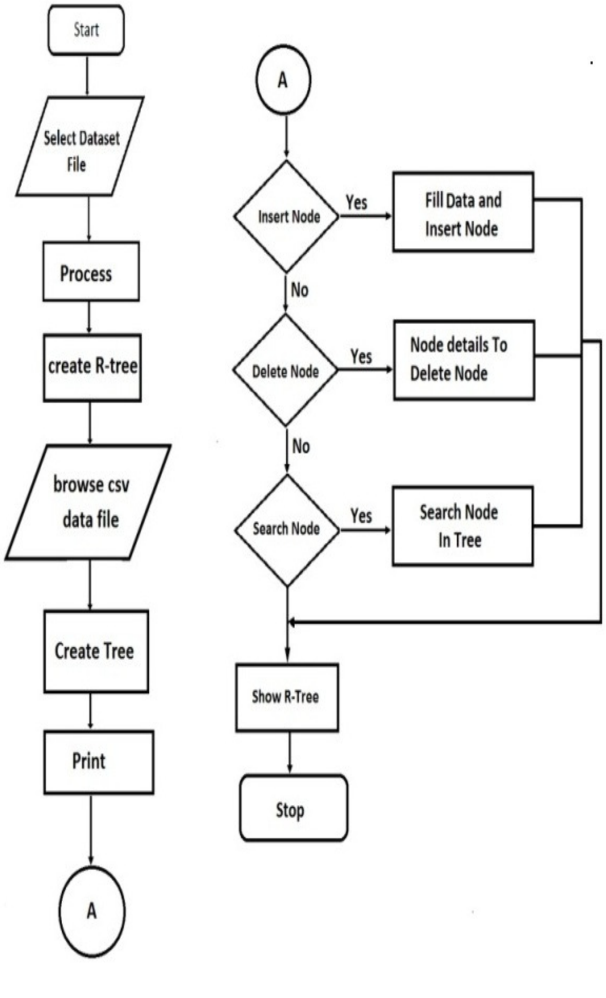


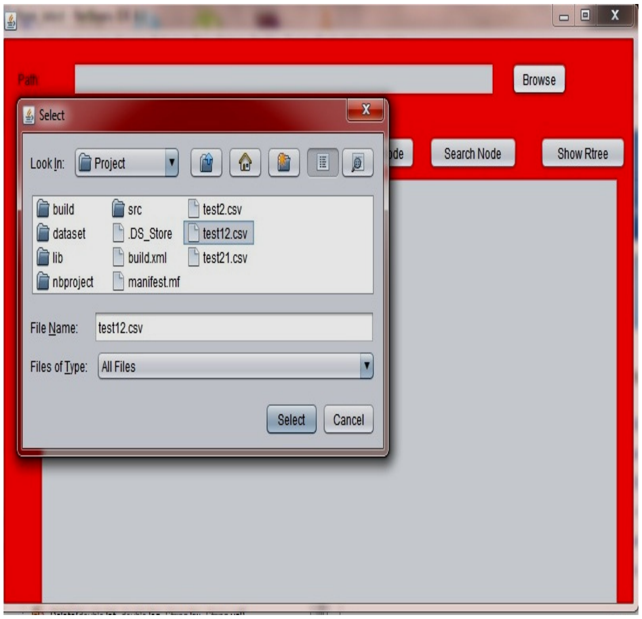
**Figure 2:** Flowchart of proposed work

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 which can be represented as 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 contain attributes that provide more information about the entity that is being represented .After gathering dataset we create an indexes on those datasets.

The idea behind R tree data structure is to group nearby objects and represent them with their minimum bounding rectangle in the next higher level of the tree. Since all the 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 single object; at higher levels the aggregation of an increasing number of objects.

## 4. Experimentations

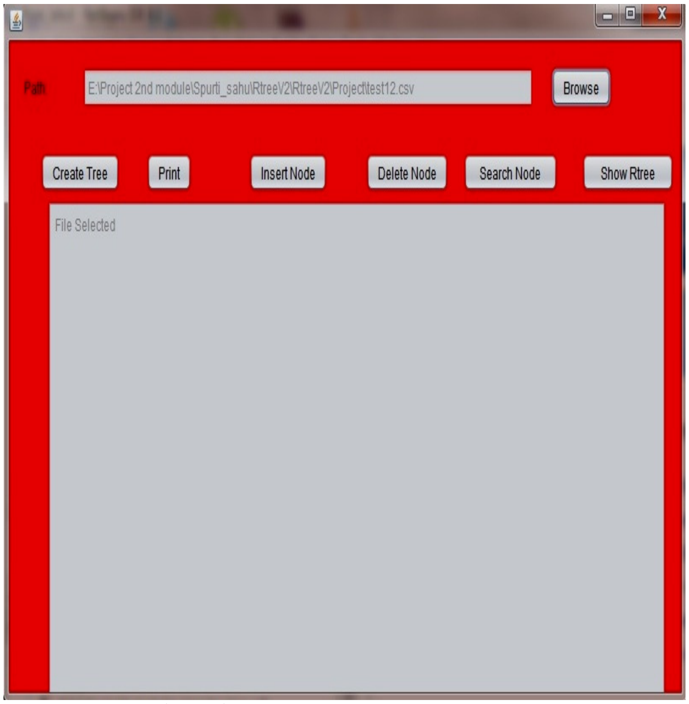
First collect datasets containing points and save them in the database. Then we choose this .txt file for processing as shown in fig. 3 by creating tables which consist of various fields. The points are to be inserted in those fields, called dataset generation. We can import dataset file in which database is made. Figure 4 shows the output after processing the datasets.



File Name: test12.csv

Files of Type: All Files

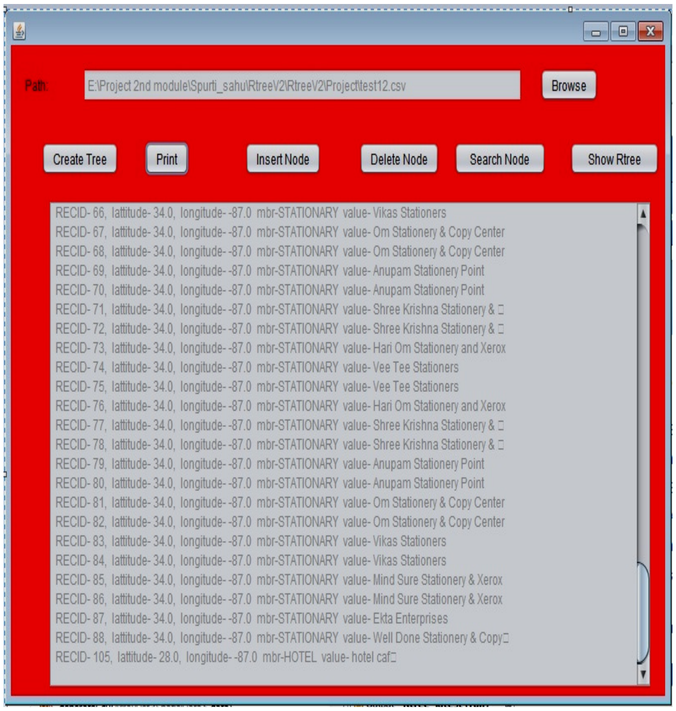
**Figure 3**: Importing location dataset file for processing



Path: E:\Project 2nd module\Spurti_sahu\RtreeV2\RtreeV2\Project\test12.csv

```
File Selected
```

**Figure 4:** Processing location datasets



Path: E:\Project 2nd module\Spurti_sahu\RtreeV2\RtreeV2\Project\test12.csv

```
RECID- 66,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Vikas Stationers
RECID- 67,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Om Stationery & Copy Center
RECID- 68,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Om Stationery & Copy Center
RECID- 69,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Anupam Stationery Point
RECID- 70,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Anupam Stationery Point
RECID- 71,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Shree Krishna Stationery & □
RECID- 72,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Shree Krishna Stationery & □
RECID- 73,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Hari Om Stationery and Xerox
RECID- 74,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Vee Tee Stationers
RECID- 75,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Vee Tee Stationers
RECID- 76,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Hari Om Stationery and Xerox
RECID- 77,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Shree Krishna Stationery & □
RECID- 78,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Shree Krishna Stationery & □
RECID- 79,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Anupam Stationery Point
RECID- 80,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Anupam Stationery Point
RECID- 81,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Om Stationery & Copy Center
RECID- 82,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Om Stationery & Copy Center
RECID- 83,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Vikas Stationers
RECID- 84,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Vikas Stationers
RECID- 85,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Mind Sure Stationery & Xerox
RECID- 86,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Mind Sure Stationery & Xerox
RECID- 87,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Ekta Enterprises
RECID- 88,  lattitude- 34.0,  longitude- -87.0  mbr-STATIONARY  value- Well Done Stationery & Copy□
RECID- 105,  lattitude- 28.0,  longitude- -87.0  mbr-HOTEL  value- hotel caf□
```

**Figure 5:** R-tree created on datasets

The below figure shows the structure of R-tree. In this the root node is MBR i.e Stationary and Hotel, and in the child node we have put the locations. All the nearby objects aregroup together and they are represented with their minimum bounding rectangle.

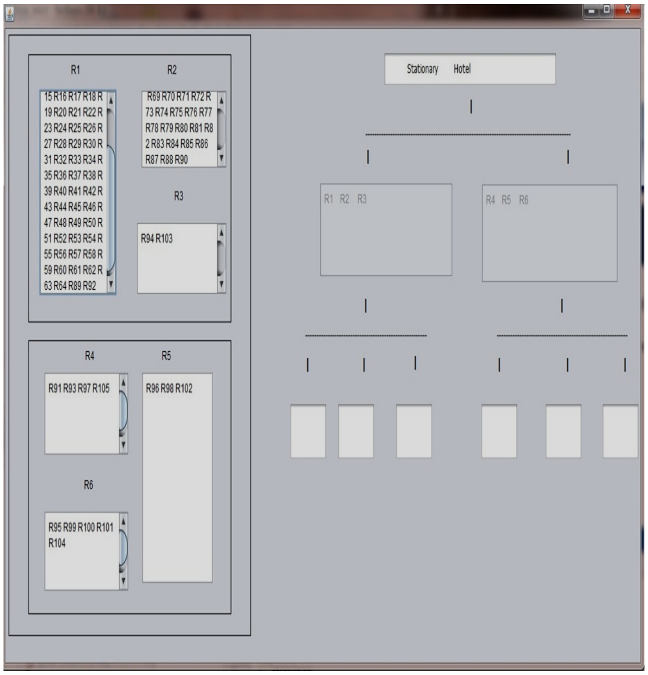
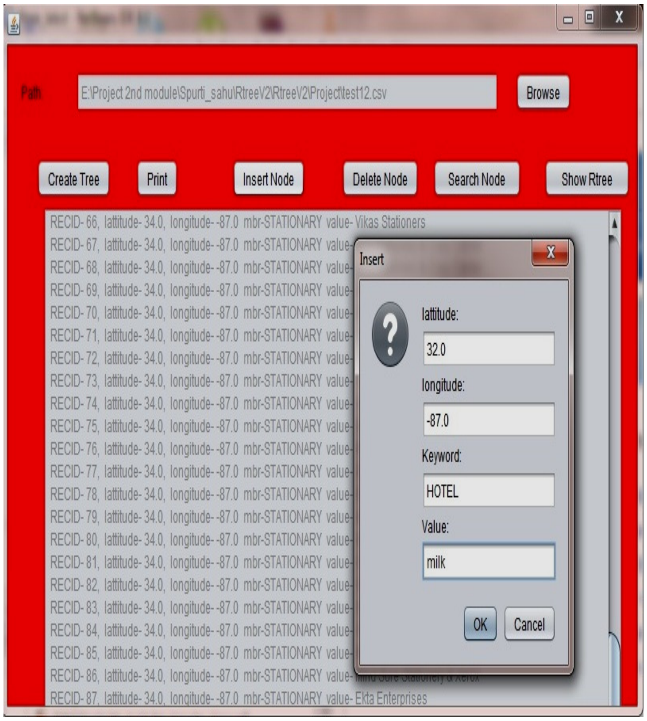


**Figure 6:** Displaying R-Tree



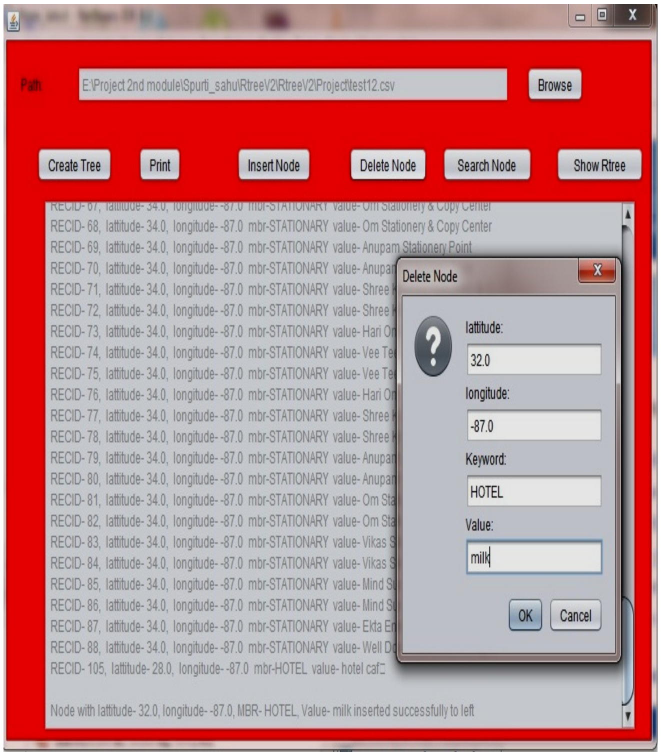
Path: E:\Project 2nd module\Spurti_sahu\RtreeV2\RtreeV2\Project\test12.csv

```
RECID- 66, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Vikas Stationers
RECID- 67, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 68, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 69, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 70, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 71, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 72, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 73, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 74, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 75, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 76, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 77, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 78, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 79, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 80, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 81, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 82, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 83, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 84, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 85, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 86, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value-
RECID- 87, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Ekta Enterprises
```

lattitude: 32.0
longitude: -87.0
Keyword: HOTEL
Value: milk

**Figure 7:** Insert node in database

For inserting any node in the database we have to enter location's latitude, longitude, keyword and value that describe the location as shown by fig 7. After this, these information are updated into the database.



Path: E:\Project 2nd module\Spurti_sahu\RtreeV2\RtreeV2\Project\test12.csv

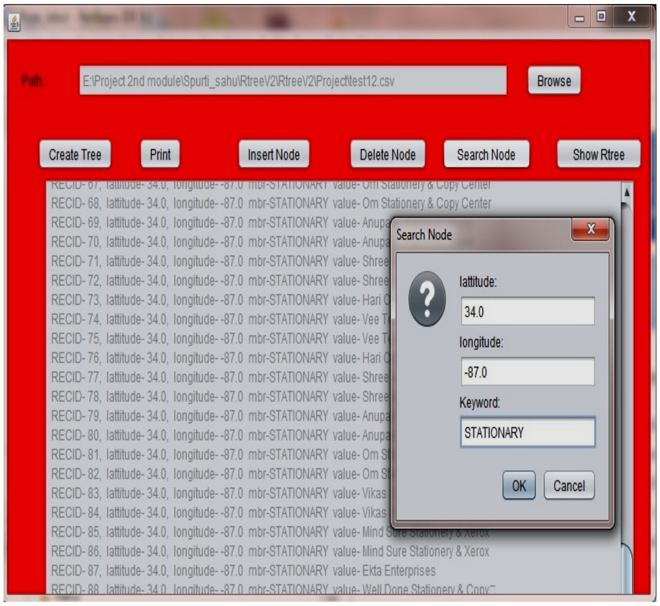
```
RECID- 67, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Om Stationery & Copy Center
RECID- 68, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Om Stationery & Copy Center
RECID- 69, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Anupam Stationery Point
RECID- 70, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Anupar
RECID- 71, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Shree
RECID- 72, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Shree
RECID- 73, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Hari Or
RECID- 74, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Vee Te
RECID- 75, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Vee Te
RECID- 76, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Hari Or
RECID- 77, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Shree
RECID- 78, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Shree
RECID- 79, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Anupar
RECID- 80, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Anupar
RECID- 81, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Om Sta
RECID- 82, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Om Sta
RECID- 83, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Vikas S
RECID- 84, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Vikas S
RECID- 85, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Mind S
RECID- 86, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Mind S
RECID- 87, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Ekta E
RECID- 88, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Well D
RECID- 105, lattitude- 28.0, longitude- -87.0 mbr-HOTEL value- hotel caf□

Node with lattitude- 32.0, longitude- -87.0, MBR- HOTEL, Value- milk inserted successfully to left
```

lattitude: 32.0
longitude: -87.0
Keyword: HOTEL
Value: milk

**Figure 8:** Delete Node in database

For deleting the node we well enter the values for latitude, longitude, keyword and value of the location, then it will search the node in the database and if the node is found then the node will be deleted from the database as shown by fig 8.



Path: E:\Project 2nd module\Spurti_sahu\RtreeV2\RtreeV2\Project\test12.csv

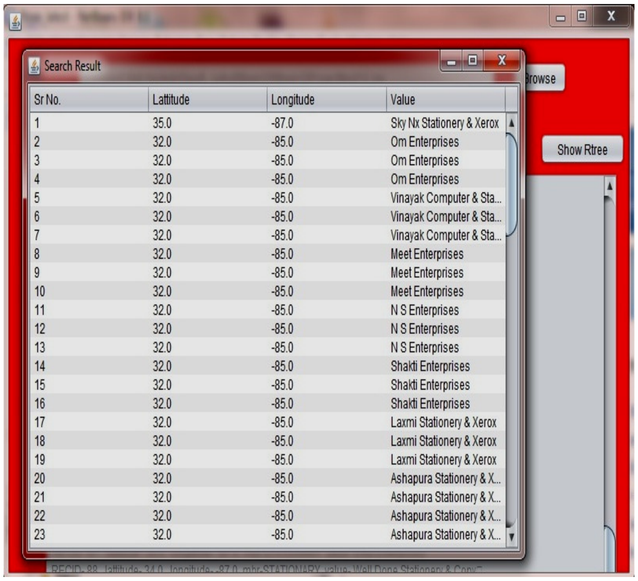
```
RECID- 67, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Om Stationery & Copy Center
RECID- 68, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Om Stationery & Copy Center
RECID- 69, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Anupa
RECID- 70, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Anupa
RECID- 71, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Shree
RECID- 72, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Shree
RECID- 73, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Hari C
RECID- 74, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Vee T
RECID- 75, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Vee T
RECID- 76, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Hari C
RECID- 77, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Shree
RECID- 78, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Shree
RECID- 79, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Anupa
RECID- 80, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Anupa
RECID- 81, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Om St
RECID- 82, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Om St
RECID- 83, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Vikas
RECID- 84, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Vikas
RECID- 85, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Mind Sure Stationery & Xerox
RECID- 86, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Mind Sure Stationery & Xerox
RECID- 87, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Ekta Enterprises
RECID- 88, lattitude- 34.0, longitude- -87.0 mbr-STATIONARY value- Well Done Stationery & Copy□
```

lattitude: 34.0
longitude: -87.0
Keyword: STATIONARY

**Figure 9:** Searching Location in database

For analyzing the query particular location will be searched, for this latitude longitude and keyword of the location will be entered and in the output nearest location of that query will be displayed as shown by fig 10.

.



| Sr No. | Lattitude | Longitude | Value |
|---|---|---|---|
| 1 | 35.0 | -87.0 | Sky Nx Stationery & Xerox |
| 2 | 32.0 | -85.0 | Om Enterprises |
| 3 | 32.0 | -85.0 | Om Enterprises |
| 4 | 32.0 | -85.0 | Om Enterprises |
| 5 | 32.0 | -85.0 | Vinayak Computer & Sta... |
| 6 | 32.0 | -85.0 | Vinayak Computer & Sta... |
| 7 | 32.0 | -85.0 | Vinayak Computer & Sta... |
| 8 | 32.0 | -85.0 | Meet Enterprises |
| 9 | 32.0 | -85.0 | Meet Enterprises |
| 10 | 32.0 | -85.0 | Meet Enterprises |
| 11 | 32.0 | -85.0 | N S Enterprises |
| 12 | 32.0 | -85.0 | N S Enterprises |
| 13 | 32.0 | -85.0 | N S Enterprises |
| 14 | 32.0 | -85.0 | Shakti Enterprises |
| 15 | 32.0 | -85.0 | Shakti Enterprises |
| 16 | 32.0 | -85.0 | Shakti Enterprises |
| 17 | 32.0 | -85.0 | Laxmi Stationery & Xerox |
| 18 | 32.0 | -85.0 | Laxmi Stationery & Xerox |
| 19 | 32.0 | -85.0 | Laxmi Stationery & Xerox |
| 20 | 32.0 | -85.0 | Ashapura Stationery & X... |
| 21 | 32.0 | -85.0 | Ashapura Stationery & X... |
| 22 | 32.0 | -85.0 | Ashapura Stationery & X... |
| 23 | 32.0 | -85.0 | Ashapura Stationery & X... |

RECID- 88 lattitude- 34.0 longitude- -87.0 mbr-STATIONARY value- Well Done Stationery & Copy

**Figure 10**: Search Result of location

## 5. Result Analysis

By experimentation, we evaluate the practical efficiency for solutions to the nearest neighbor search with keywords. Real spatial static datasets are used that catalogs vast amount of spatial data objects. We employ a number of pre-processing steps to arrive at a more structured database. The downloaded database is heterogeneous in nature. For each database, we identify and remove the real-valued fields and return the spatial and text field. Now the database is more structured and homogeneous containing only spatial and textual data.

We use different size of datasets in our experiment and calculate its execution time(in millisecond) and standard deviation as shown in fig 11

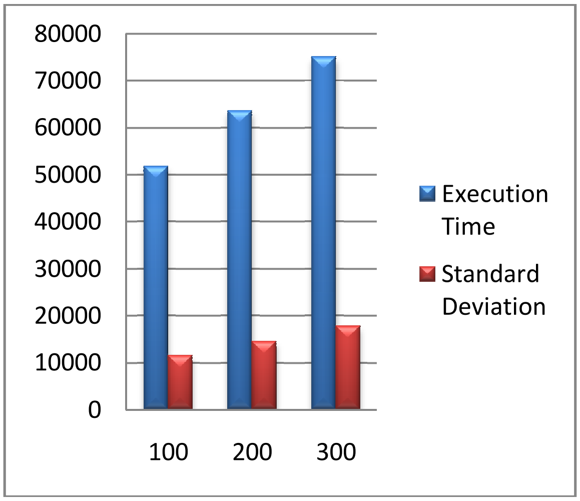


**Figure 11**: Execution time& Standard deviation for different records

## 6. Conclusion

The objective of our work is to increase the speed of query processing and give the real time answer of spatial query. As we have seen many applications intended for a search engine that is able to support new form of query that are combined with keyword search. The existing system does not give real time answer for such type of query. For this purpose we have proposed to use R-tree and spatial inverted index which is readily incorporable into commercial search engine that applies massive parallelism, implying its immediate industrial merits.

## References

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

[2] 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..

[3] 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.

[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] D. Zhang, Y.M. Chee, A. Mondal, A.K.H. Tung, and M. Kitsuregawa, “Keyword Search in Spatial Databases: Towards Searching by Document,” Proc. Int’l Conf. Data Eng. (ICDE), pp. 688-699, 2009.

[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] 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.

[9] 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.

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

[11] G.R. Hjaltason and H. Samet, “Distance Browsing in Spatial Databases,”ACM Trans. Database Systems, vol. 24, no. 2, pp. 265-318,1999.