Combining R-Tree and B-Tree to Enhance Spatial Queries Processing

Marwa A. M. Abd Elwahab[1], Khaled M. Mahar [2], Hatem Abdelkader [3], Hatem Awad Khater[4]

[1,2]College of Computing and Information Technology
Arab Academy for Science and technology, Alexandria
marwa.abdelwahab@aast.edu
khmahar@aast.edu

[3]Information Systems Department faculty of Computer and Information Systems, Monofia University, Alexandria
hatem6803@yahoo.com

[4] Research & Development Department, Egyptian Naval Forces
hatem.a.khater@googlemail.com

***Abstract*—The search process in the spatial database consumes much time for the user. The size of the database determines the efficacy and speed of information retrieval. The potency of spatial query in Geographic Information Systems (GIS) is of paramount importance. It depends heavily on the query processing algorithms. When the database size increases, the processes of data retrieval gets complicated. This paper employs two types of data structures; Rectangle-Tree (R-tree) and the Balanced-Tree (B-tree) which is called aRB-Tree. These two types have been formerly merged by researchers to facilitate the retrieval of data as quickly as possible while providing efficient results. The same idea has been adopted in this paper where the aRB (aggregate RB) accelerates executing complex queries and minimizes the time consumed in searching, editing, deleting and updating any record in the spatial database. In this paper a web application was developed to test the efficiency of using both data structures. This web application utilises a medical institution database that contains locations such as addresses of physicians, clinics, hospital, labs, and radiology centres. Results of aRB-Tree are more sufficient and accurate than using each index alone.**

***Keywords*— Spatial indexing; aRB- Tree; GIS; R-tree; B-Tree;**

I. Introduction

Geospatial database uses Geographic Information System (GIS) to locate and access data quickly and efficiently [1][2].

Geographic databases are used for both storage of Spatial and Non-Spatial attributes. It can be used for indexing data structures to provide fast response to spatial queries. Spatial database consists of a collection of records representing spatial objects; each record has a unique identifier which used for retrieving ([3]-[5]).

Spatial data objects consist of one point or more (group of thousands polygons) which has been distributed randomly through the space. It is hard to store groups of those objects in one relational table with static record size [3]. It is possible to store spatial data about objects in relational database, but it does not support retrieval efficiently to spatial objects. Data base systems need to index concepts that retrieve data items quickly according to their spatial locations. There are many indexing structures that had been used for solving multidimensional data problem like R-tree and B-tree.

In particular, R-tree indices are performed on widespread spatial attributes [4]. On the other hand, the B-tree indices are performed on non- spatial attributes. Spatial data known as geometric or shape data is stored in spatial databases.

Attribute data are used to describe the characteristics of spatial features. There are two models for storing the spatial data: raster model and vector model. Raster model has cells, the cells have a value that corresponds to the attribute of the spatial feature at that location; while vector model is a representation of geographic data in any of the following forms: points, lines or polygon.

Geospatial data is divided into two main groups; the first group is documentary data that describes data: it is the combination between numbers, letters, video clips or images and it represents the whole map. The second group is the geometrical data which describes a location with the coordinates x and y; it represents numerical data just the entities. Making complete geographic database requires two major files: documentary file and geometrical file; Geospatial data is defined as geographically referenced data that describes both the locations and characteristics of spatial features such as roads, land parcels, and vegetation stands on the Earth's surface, it is also called geospatial data and it is stored in spatial databases; Multidimensional trees are used in order to build indices for these data. Attributes of spatial objects are still one-dimensional, so that this non-spatial part can be stored in relational databases with references to the spatial data.

GIS platforms are used to implement spatial databases and to handle spatial data, in particular image data. There is a need to increase the high quality of satellite images because it directly affects the quality of data stored in spatial databases [6].
Location based services (LBS) are services that use the geographical data and its position. Here are the four essential

components for LBS to work: Localization, map, Client hardware, point of interest and communication. In this research LBS can be used for locating the medical place based on the user current location. Component is the most important one in which the user searches for a medical institution within the range that he is located in.

In this paper, a web application is developed as a location based services. This web application is based on cloud computing features. The application implements actual geodatabase containing thousands of spatial data objects which represent medical sites in Egypt such as clinics and addresses of physicians, hospitals, laboratories and radiology centres. The main purpose of this paper is to enhance the quality of spatial queries. This will be performed through merging two data structures namely B-Tree and R-Tree .The system integrates the R-tree and B-tree models to facilitate executing complex queries and to minimize the time consumed in searching, editing, deleting and updating any record in the spatial database compared with the B-Tree. In the aggregate RB-tree, the extents of all regions are stored in R-tree. Each (leaf/non-leaf) entry of the R-tree is associated with a pointer to a B-tree that stores historical aggregate data about the entry.

The rest of the paper is organized as follows: section II discusses the background and related work, section III presents the proposed framework, section IV discusses the prototype implementation and performance evaluation, and finally the last section V contains the conclusion and the future work.

II. Background and Related Work

GIS is a system of hardware and software used for storage, retrieval, mapping, and analysis of geographic data. The main practitioners in GIS are the operating personnel and the geodata. Spatial features are stored in a coordinate system (latitude/longitude, state plane, etc.), which references a particular place on the earth. Descriptive attributes in tabular form are associated with spatial features. Spatial data and associated attributes in the same coordinate system are layered together for mapping and analysis [7].

The huge size of the geodatabase and the long processing steps to search these databases require a technique for easier and faster time search. Indices can be used for this purpose; indices will increase the speed of search in databases. Alphabetical indexing would reduce the time to extract data since it prevents data redundancy.

Spatial index can be defined as the kind of data structure which has been arranged in some sequence for the position and shape of space entities or spatial relationships between entities [1]. Another definition of spatial index is “the primary key to improve the efficiency of spatial query” [9].

R-tree: A dynamic index structure for spatial searching. R-tree spatial index evolved from the B-tree spatial index. The core idea is to use the Minimum Bounding Rectangle to represent space objects and its index space in order to reduce computation & storage memory [2].

A. R-Tree Overview

Based on the previous research, there are a number of spatial data index methods proposed. One example of these methods is the R-tree. R-tree is the most widely accessed method used. It is also known as dynamic R-tree. [4]. R-tree is used in spatial database as a spatial access method. Spatial database is the core of GIS. The aim of spatial access method is improving query performance by incorporating an additive data structure since the high query performance is one of the key features of successful GIS systems [4].

R-tree is composed of the root, intermediate node, and leaf node. Each leaf node is created in the form of Minimum Bounding Rectangle (MBR) [8]. Leaf node does not store the actual spatial objects, but it stores the minimum bounding rectangle of the actual spatial objects. Figure 1 shows some rectangles, organized to form a corresponding R-Tree.

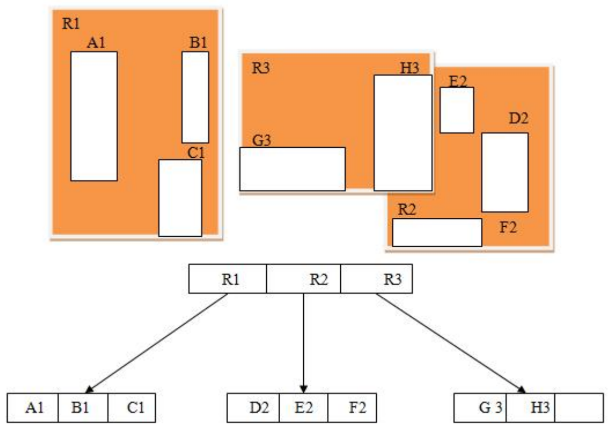

Fig. 1 R-tree Rectangles

1) *R-Tree Steps:* Query algorithm of R-tree is similar to B-tree. It uses top-down query from the root node, but MBR of R-tree will overlap to result in the non-uniqueness of query path. Since the B-tree uses only the address, the B-Tree results will depend on the existence of the required address. R-tree searches the database using the miles space from the required address.

The Google map has been used in the search process thus when the user searches for a specific address, the system uses R-Tree spatial index which searches the addresses map and returns the results to the user on the map. If the users asked for more details concerning the addresses, the system will use B-

Tree for this purpose. B-Tree searches the system database and plots theses results on the map in front of the user.

The user searches for a specific address in the system database. If the system finds this data, it directly compares it with the current database, then if this data is found; finally it will retrieve the data that is in the database on the map. However, if the search data was not found in the current database or web database, there will not be any data to be given to the user.

2) *Usage of R-Tree*: The R-tree algorithm is one of the indexing algorithms. It is flexible since it is organized according to the data index structure [8]. This makes it flexible and adjustable. The index can be established without the prediction of the entire spatial extent because it is a natural extension of B-Tree and they have similar structure and characteristics. One of the main usages of R-Tree is that it can be integrated with traditional relational database. This is the reason many R-Tree spatial databases are chosen for spatial index. However, when the R-Tree node entries exceed M (the maximum storage for each node index entries), the node must be split.

B. *B-Tree Overview*

B-tree is a data structure for storing data with countless run times for insertion and deletion. B-tree structure is shown in Figure 2. B-tree is a special type of trees whose properties make them useful for storing and retrieving information.

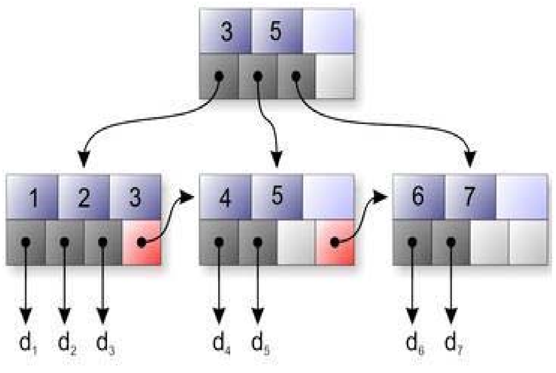

Fig. 2 B-tree Structure [10]

B-trees are multi-way trees in which each node contains a set of keys and pointers. B-tree with four keys and five pointers represents the minimum size of any B-tree node. The longest path between the root and a leaf will always be O (logm n) [where m is the order of the B-tree and n is the number of nodes].

B-Trees are dynamic so the height of the tree grows and contracts as records are added or deleted. B-trees have a pointer structure as they can be used in learning the recursive pointer algorithms [10].

A B-Tree is made of nodes, where each node contains a "left" pointer, a "right" pointer, and a data element. The "root" pointer points to the topmost node in the tree. The left and right pointers point to smaller "subtrees" on either side. A null pointer represents a B-tree with no element [10].

1) *B-Tree Steps:* B-tree has three main properties: all leaves are on the same level, the root has K descendants and finally other internal node has K descendants [m/2] [10].

2) *Usage of B-Tree*: Main usage of B-Trees is database applications. They are the basis for the Virtual Sequential Access Method (VSAM) since they allow fast retrieval of data. Hash trees are an alternative to B- Trees for the organization or large files of data [10].

C. *aRB-Tree Overview*

In the aRB-tree, the extents of all regions (in this case r1, r2… r4) are stored in an R-tree. Each (leaf/non-leaf) entry of the R-tree is associated with a pointer to a B-tree that stores historical aggregate data about the entry. Figure 3 below illustrates the aRB-Tree structure [11].

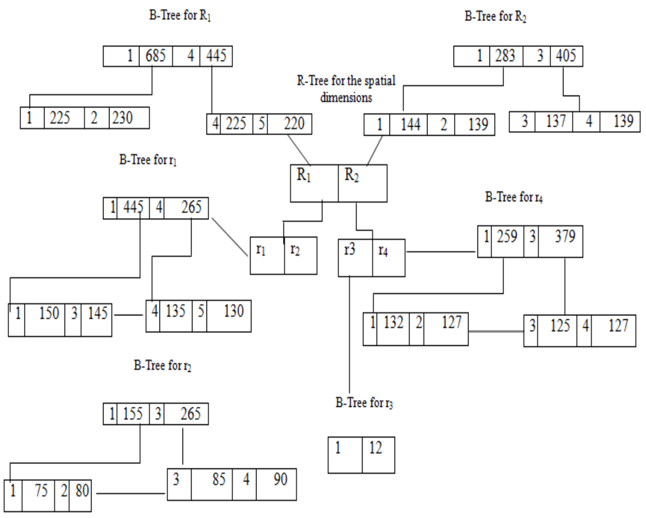

Fig. 3 The aRB-tree Structure [11]

Different researchers used R-Tree and B-Tree as a geodatabase index. Table I shows a summary of previous researches done in the same field:-

Table I. Use of R-Tree and B-Tree in some related work

Reference	Index name	Object
[3]	R-Tree	Analysing the overlap between brother nodes and multipath.
[2]	R-Tree	Managing multimedia retrieval systems
[12]	R-Tree	Clustering uncertain objects described by probability density functions
[13]	R-Tree	Integrating R-Tree with LOD feature which is a 3D scene
[14]	R-Tree	**Developing strategies for accelerating multidimensional query processing**
[15]	B-Tree	**Managing mobile ad hoc network**
[16]	B-Tree	Managing flash memory storage
[17]	B-Tree	**Solving the problem of the route match delay**

Research in [11] introduced the combination of the R-Tree and B-Tree index in the form of aRB-Tree index with sketches for approximate query processing to solve the problem of distinct counting. The aRB- Tree can be used to calculate the summarized information about the moving objects in a specified area during query period; such as the number of mobile users covered by a sect, etc. In this case the corresponding Spatio-temporal total query will return the total number of phone calls made by all users in a specified region during a particular time, since the previous indices consume space and produce inefficient results.

After testing the usage of aRB- Tree with sketches, results were more accurate and using aRB- Tree reduced the space consumption and the query time return. This research also uses the combination of the two indices.

III. Proposed Framework

This section discusses mainly the proposed framework that contains the aRB-tree module, the reason to use R-Tree and B-Tree, and the system analysis and design phase details.

The proposed model has two main data structures; the first is the R-Tree implementation and the second one is B-Tree implementation. They are merged to facilitate the search process to the user and help him to get the full information. The user enters the required address and defines the range of the miles to search within. The user communicates with the web application interface.

A. *Reason to use R-Tree and B-Tree*

The goal of the integration and use of R-tree and B-tree is to provide full information to the user in terms of appearance all the Addresses he is looking for in detail and facilitate the implementation of complex queries and reduce the time spent when he searches for any record in the database.

The web application interface will start communicating the system through the first module. This module starts working by searching the required address in the Google map, and then the second module searches the system database. The medical database information is collected manually by searching the internet, searching hospitals address, asking doctors and pharmacists. If the address was found the third module will plot the addresses in the Google map and show it back to the user. The Google map is used to plot the results as points, each point represents an address. Figure 4 shows the system framework.

The second data structure is the B-Tree module; it works the same way as the aRB-Tree but the B-Tree searches uses the address only without the miles and of course B-Tree will search only the system database and it will not check the Google map.

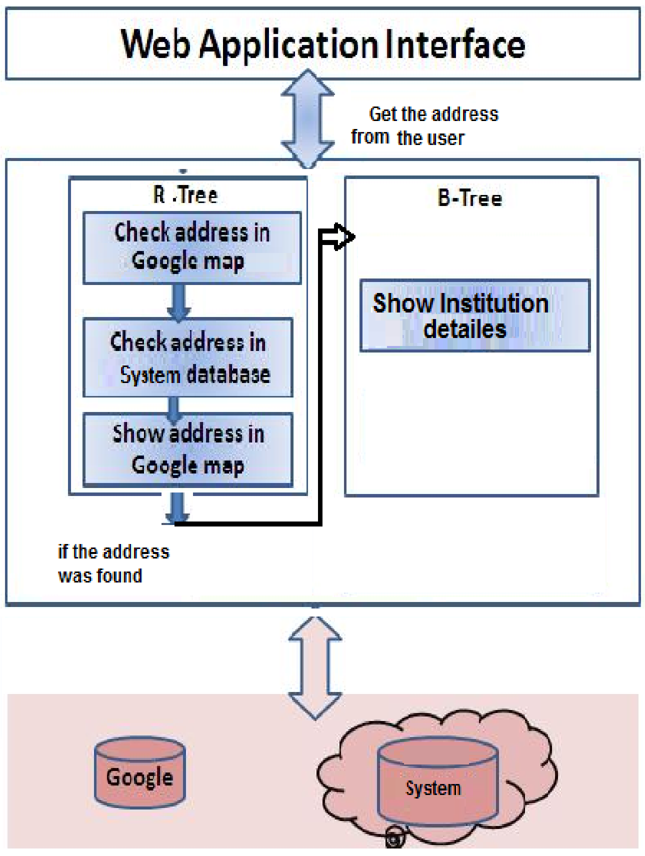
Web Application Interface
Get the address from the user
R .Tree
Check address in Google map
Check address in System database
Show address in Google map
B-Tree
Show Institution detailes
if the address was found
Google
System

Fig. 4 System Framework

B. *Analysis and Design*

Before the system implementation, the system is designed to define the main processes in different phases. The following two diagrams will show the main activities of the system using flowchart and a use case. Figure 5 represents the Address Search and Display Flowchart of the system.

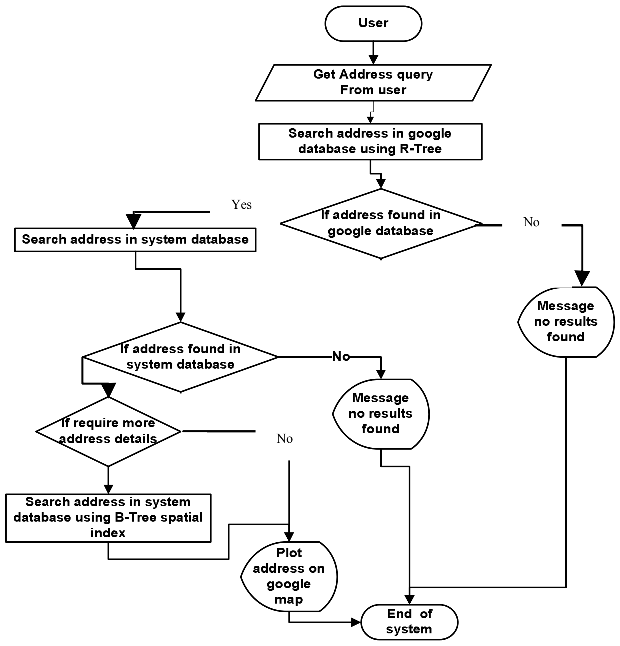
User
Get Address query From user
Search address in google database using R-Tree
If address found in google database
Yes
No
Search address in system database
Message no results found
If address found in system database
No
Message no results found
If require more address details
No
Search address in system database using B-Tree spatial index
Plot address on google map
End of system

Fig. 5 Address Search and Display Flowchart

The main functions of the proposed framework are described through a use case diagram as shown in Figure6. Seven use cases can be identified which are: search address, compare address, set search query, get address, plot address point, set search query, and search address details, as shown in Figure 6.

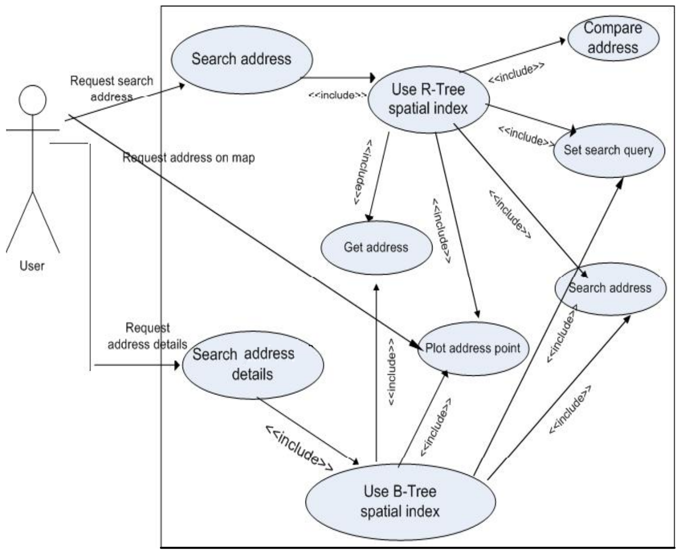

Fig. 6 System Use case

The spatial database indices works in a proximity technique. Proximity analysis is one way of analysing locations of features by measuring the distance between them and other features in the area. The distance between point A and point B may be measured as a straight line or by following a networked path, such as a street network. When the user searches for the following address [9 midan el Tahrer Street] (which will be written in Arabic) then the search will start from the right to the left starting with the number of the street, the name of the street, and finally the city name. The system search for address using x and y coordinates of the address that is represented in the Google map. On the other hand, if the address is written in English, then the search will start from the left to the right starting with the country, city name, street name, and finally street number. The database structure that is used in the implemented system consists of location, government, site, area, doctor, specification, and provision category this database structure is shown in Figure7.

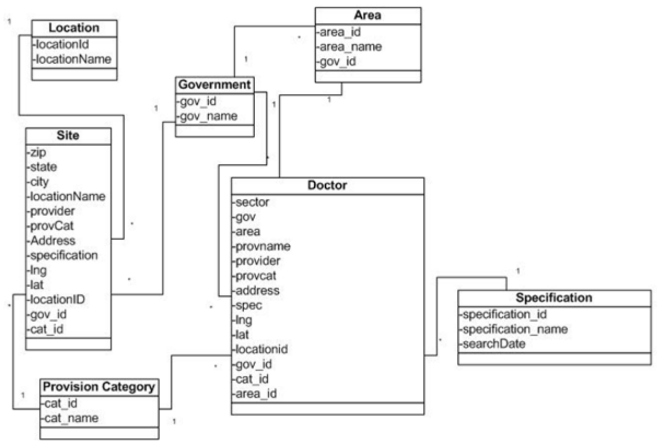

Fig. 7 Database Structure

IV. Prototype Implementation and Performance Evaluation

The server specification in our experiment is processor: quad Core; Ram: 4 GB. The applications used in the paper are the following: Google Map (java API), C#.net is the language that was used to write the two data structures using Microsoft Visual Studio 2008 ,Microsoft SQL server 2008 for building our database and ASP Spider.Net for Cloud Computing[18].

In our experimentation the user utilizes the aRB-Tree in which he can retrieve the spatial data using R-Tree and if he wants more details about the searched institution then he can switch to use the B-Tree. That is, in the first part of aRB-Trees, spatial indexing(R-tree) is used to retrieve the location of the institution while in the second part the attribute-indexing or non-spatial indexing (B-tree) is used to retrieve other data such as the working hours. They are merged and implemented on our real database which has been created using Microsoft SqlServer 2008 which contains thousands records about pharmacies and hospitals, laboratories and physicians addresses, and all existing disciplines in Egypt. This database is linked with Google Map to get the information about locations; the web application and the database are implemented over the cloud; the medical institution DB created in this prototype is just a case study and the idea of merging the two trees can be applied in other domains as well.

In the system the user interaction with the system by default the user will search using R-Tree. If the user wants more details about the searched institution address, he can click on more details option, so the system will use the B-Tree index after this the user will write the address in the address box. When the user shows the B-Tree screen and writes the address, the results will be shown on the webpage and will be plotted on the Google map beside the results. When the user shows the aRB-Tree screen, he must specify the miles within his search and writes the address. The results will be shown on the webpage and the results will be plotted on the Google map beside the results.

The aRB-tree structure algorithm works as follows: first it searches the address in the database referring to the web. There are two probabilities; either it finds this address and directly compares it with the database system or it retrieves the data that is in the database on the map using R-Tree and searches for the coordinates of this address that exist in the database in the system using B-Tree. However, if the data in the database is found in the system but was missing in the Google's map, no data will be retrieved.

The user searches for a specific query, example of this query is "ش التحرير 9" (9 Eltahrir Street). The query represents the required address to be found on the map such as a physician's address or a hospital address. In case of searching for a hospital located just 5 miles from a particular street, the aRB-tree structure takes the name of the street that is selected and searches around it, taking into consideration the distance requested (e.g. 5 miles) and draws a circle on the map with a radius of miles and looks in this region and then shows the search results on the map in the form of circles. Within those circles, the aRB- Tree splits the map to a set of rectangles given to each rectangle of them code or index to make it easier to search and then compares them with the information found in the database. Finally it shows on the map by clicking on any site.

Figure 8 shows the total matched locations of the two indices R-tree and aRB-Tree. This figure represents to what extent the system displays the user's results as related to his query. The results produced by the aRB- Tree are more specific and efficient compared to those produced by R-Tree. Table II shows examples of queries used during performance evaluation.

Table II. Query Examples

Query no.	**Example**
Query no.1	9 Eltahrir Street
Query no.2	El- Zamalek
Query no.3	El- Mohandesin
Query no.4	El- Maady

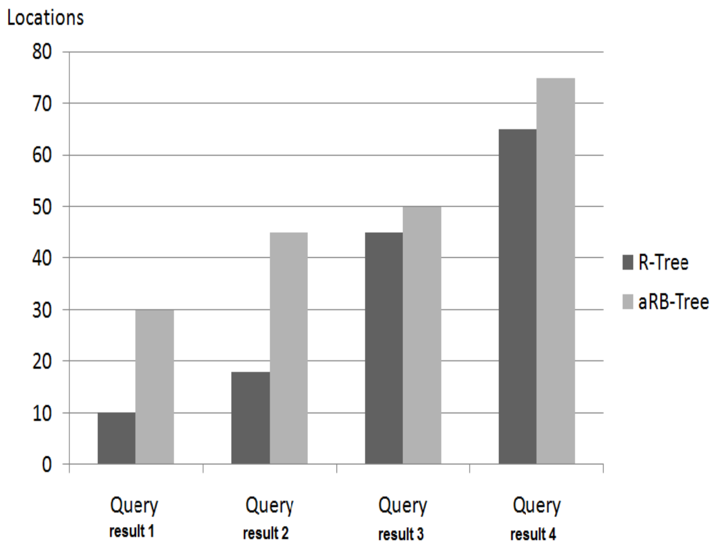

Fig. 8 Total Matched Locations

B-Tree structure searches for the coordinates of a given address that only exists in the database in the system. If B-Tree finds the coordinates, it directly shows them on the map, through the id number. If the coordinates of this address was not found in the system's database, no data will be retrieved. The purpose of using B-Tree index is showing details of the results given to the user.

However, B-Tree has a drawback since it searches only in the system's database and thus yields fewer results. Index should write the information accurately and should also be very specific so that the output is faster. For example, when asked to search for the city of Alexandria, the search results will be more and take more time. But if the user entered "Alexandria - Abu Qir", the results will be less, more accurate and takes a shorter time.

B-Tree searches only for a specific required address. It searches a medical database in which the address may be stored in the database or not. The B-tree structure was tested in six trials with six different addresses, the last address which is "ش التحرير 9" (9 Eltahrir Street) is the most specific address used. The keyword in searches affects the search results in the B-Tree since it searches for either a specific address stored in the database or not. The probability of the appearance of specific address increases when searching in a specific range. Addresses like "Alexandria الاسكندرية", "Roshdy رشدى"are considered to be a wide area with a less probability to appear in the search results.

V. CONCLUSION AND FUTURE WORK

For dealing effectively with geospatial data, it should be taken into consideration that each application requires a special index so that items can be retrieved quickly. The traditional indexing strategies are now unsuitable for multidimensional geospatial databases. As previously shown, aRB-tree structure meets these needs and it can help to enhance the search process more than B-tree does. The important key is to be able to retrieve any data quickly and effectively according to its spatial location. This paper implements two indices that are famous for being used in spatial database research. A web application was implemented over a cloud for providing a location based service in which the user searchers for a specific location. The web application uses aRB-Tree to search a created medical institution database that contains addresses of physicians, hospitals, clinics, etc. The user asks for specific medical information using a search query, the system uses R-Tree index for retrieving the results and plotting these results on the map. However, if the user asked for more details about theses results, the system will use the B-Tree index for this purpose. The system is implemented and under validation now. The results show that the aRB- Tree is more accurate and efficient compared to those of R-Tree.

For Further research, researchers can test other spatial indices as R* and R+ They can try the indices results on different databases and not necessarily the medical type. Finally the paper is limited to two experiment design, total response time and total matched location.

REFERENCES

[1] Z. Wei , W. Wanzhen, Y. Xingguang, X. Gang, "An optimized query index method based on R-tree," in *Proc Fourth International Joint Conference on Computational Sciences and Optimization, pages:1007-1010,* 2011.

[2] Y. Lifang; L. Rui; H. Xianglin; L. Yueping, "Performance of R-tree with slim-down & Reinsertion Algorithm," in *Proc International Conference on Signal Acquisition and Processing*, pages:291-294, 2010.

[3] Z. Shaohui, C. Zhanwei , "The research of Hilbert R-tree spatial index algorithm based on Hybrid clustering," in *Proc International Conference on Electronic & Mechanical Engineering and Information Technology*, *pages:3495-3497,* 2011.

[4] Guttman, "R-trees: A Dynamic Index Structure for Spatial Searching", in *Proc ACM SIGMOD International Conference on Management of Data*, Boston, Ma, pages: 45-57, 1984.

[5] Z. Yon-Gui. Yon, Q. Song, Y. Fu-Ping "An Query processing for continuous K-nearest neighbor based on R-tree and Quad tree," in *Proc Computer Science and Information Technology (ICCSIT), pages:35- 40*,2010.

[6] G. Jun K. Shengnan, "A new node-split algorithm in R-tree based on spatial clustering," *IEEE Seventh International Conference on Fuzzy Systems and Knowledge Discovery* (FSKD 2010) *Vol.5,* pages: 2081- 2085, 2010.

[7] (2010) The ESRI website.[Online].Available: http://www.esri.com/

[8] J. GONG, Sh. KE,"3D spatial query implementation method based on R-tree," in Proc Remote Sensing, Environment and Transportation *Engineering (RSETE), pages: 2828- 2831*, 2011.

[9] L. Weihua, W. Yonggang, T. Xiaojun, and Y. Yan "An improvement of index method & structure based on R-tree," *IEEE Proc International Conference on Computer Science and Software Engineering*, Vol.4, pages: 607- 610, 2008.

[10] Belbaraka, M."Algorithms for generating and coding B-Trees", *ProQuest Dissertations and Theses (PQDT), 1996.*

[11] Y. Tao, G. Kollios, J. Considine, F. Li, and D. Papadias "Spatio-Temporal Aggregation Using Sketches," *in ICDE, 2004.*

[12] Kao, B. Lee, S. Lee, F. Cheung, D." Clustering Uncertain Data Using Voronoi Diagrams and R-Tree Index,"*IEEE Transactions on Knowledge and data engineering, Vol.22 No.9, 2010.*

[13] Shengnan, K." Integrating R-tree and Levels of Detail," *Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2011.*

[14] Yu,B. Kim,H. Choi,W.Kwon,D., " Parallel Range Query Processing on R-Tree with Graphics Processing Unit," *Ninth IEEE International Conference on Dependable, Autonomic and Secure Computing, 2011.*

[15] Yongfei Ye, Y. Liu,M.Sun,X. Yan,G., " Group Key Management Scheme Based on B-Tree Topology in Ad Hoc Networks," *IEEE 2nd international conference on computer science and network technology (ICCSNT), pages: 1676-1680, 2012.*

[16] Wu, C., Lin, Y. "A Concurrency Buffer Control in B-Trees for Flash-Memory Storage Systems," *IEEE embedded systems letters, Vol.4, issue: 1, pages: 9-12, 2012.*

[17] Du,H., WEI,W, YANG,J. "A B-Tree Algorithm for Partitioned Index based on CIDR List in High-end Router," *IEEE international conference of information technology, computer engineering and management sciences(ICM*), *Vol.2,pages: 124-128,* 2011.

[18] (2013) The aspspidaer website.[Online].Available: http://www.aspspider.com/