LOCATION-BASED NEAREST NEIGHBOUR SEARCH USING GRID-BASED SPATIAL INDEXING

AUNG ZAW MYINT

UNIVERSITY OF COMPUTER STUDIES, YANGON

FEBRUARY, 2022

Location-based Nearest Neighbor Search Using Grid-based Spatial Indexing

Aung Zaw Myint

University of Computer Studies, Yangon

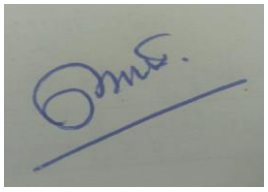
A thesis submitted to the University of Computer Studies, Yangon in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

February, 2022

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.



4 - 2 - 2022 ……..……..

Date Aung Zaw Myint

ACKNOWLEDGEMENTS

I would like to give my sincere thanks to people who have assisted and guided me during the Ph.D study.

First and foremost, I especially thank Dr. Mie Mie Khin, Rector, University of Computer Studies, Yangon, for overall supporting my thesis. I greatly appreciate the help I have received from her. I would like to mention my earnest appreciation to Dr. Mie Mie Thet Thwin, former Rector of the University of Computer Studies, Yangon, for her kind support and encouragement.

Secondly, I would like to express my deepest gratitude to my supervisor, Professor Dr. Khin Mo Mo Tun, Pro-Rector, the University of Information Technology, for her tremendous support and guidance during my doctoral study. I would like to express my sincere gratitude to my supervisor for giving much of her time. I was motivated by her excellent guidance and creative ideas to improve my research skills. I admire her courage, her positive outlook, her ability to offer advice. I am very lucky to be working under her supervision.

My heartfelt thanks and respect go to Dr. Thin Lai Lai Thein, Professor, Dean of the Ph.D. 10th batch, for her excellent guidance, patience, and taking care of me throughout the Ph.D. life. I would also like to express my respectful gratitude to Daw Aye Aye Khine, Associate Professor, Head of English Department, for her valuable support and editing the correct usage of language in my thesis.

In addition, I would also like to extend my appreciation to Prof. Dr. Khine Khine Oo, who gave me kind support, motivation, encouragement, and valuable suggestions in this thesis.

I would also like to thank all my teachers for mentoring, encouraging, and recommending the dissertation. I am also grateful to all talented and friendly colleagues from Ph.D 10th batch for their friendship and accompany during my graduate student life.

Finally, I want to express my sincerest and deepest gratitude to my father, mother, sisters, and brother for their unconditional love, support, and encouragement. I would not have been where I am today without their endless love and tremendous support. My dissertation is dedicated to them.

ABSTRACT

With the rapid development of mobile technology, location-based services (LBS) have become essential useful services to provide geographic information to mobile users. Location-based services (LBS) support the relevant geo-location information services that are navigation services, emergency services, information services, tracking, and management services. The mobile users extract the desired location information from their location position by using LBS content provider. LBS content provider needs to facilitate location information services for the users' satisfaction. Nearest neighbour search is the important part for location-based services to retrieve geo-location objects. To respond quickly location data, an index structure is essential for constructing spatial database. Therefore, many researchers have also been studied the spatial data structure for location-based nearest neighbour search.

The focus of the research is to develop location-based nearest neighbour queries on the spatial database. To retrieve the nearest location objects effectively, an indexing technique is required to construct the data efficiently. Most of the location-based services content providers use spatial databases to provide useful information that depends on the application areas of location-based services (LBS).

The system offers the users the geolocation objects based on user's location position. The geolocation objects possess the location position and its attribute data. The system supports location-based keyword search and specific service type search for client users. Most of users prefer to find location objects what are existing nearby and within the particular distance from their current location. Therefore, The system develop for client users that provides range query and k-nearest neighbour query (kNN) that can retrieve location data accurately.

This system is to provide location information efficiently according to the user preferences services from the specific location. This research uses nearest neighbor search algorithms for range query and k-nearest neighbor (kNN) query. Users can choose query types both range query and kNN query to retrieve geo-information data. In addition, the system supports to user not only spatial keyword search but also category-based search from user desired location. In the system, the spatial access method is necessary that is to support efficient storing and retrieving of geolocation objects in the spatial database. Using indexing techniques in the spatial database allows the clients to retrieve data quickly. The system uses R-tree indexing technique

based on the Grid index structure for accessing objects effectively. R-tree index structure has been widely used in the research projects of spatial databases among variants access methods. R-tree index structure is a dynamic index structure for the spatial database. R-tree also supports various queries such as finding all records in a particular range and the objects which cover a certain area. The system is designed to support spatial queries efficiently and it also supports speed up computational performance.

In this research, location data from townships in Yangon Region were used to develop the proposed system. The proposed index structure constructed the geo-data to support nearest neighbour queries efficiently. The geo-data are two-dimensional objects which compose latitude and longitude values and their attribute values. The location objects are enclosed in a minimum bounding box rectangle which is the coordinate of the lower-left corner and the coordinate of the upper-right corner. The main property of R-tree is to minimize the area of each enclosing rectangle in the index structure. For this reason, the proposed system uses R-tree that can be optimized for query processing in the spatial database. The implementation of this research developed on the client-server model. The main service of the server system is implemented on the cloud services for web hosting and creating the data in the cloud database.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

CHAPTER 1

INTRODUCTION

With the advanced technology in Geographic Information System (GIS) and wireless communication technologies, Location-based services (LBS) provide geo-location information to mobile users. Nowadays LBS is mainly used in companies, commercial sectors, and government organizations that integrated location-based services into their mobile applications. With a vast amount of smart-aware devices such as mobile phones and notebook computers, mobile users prefer to access geo-data anywhere and anytime from the service provider. LBS can give geospatial information to mobile consumers depending upon their current location and each specific location. The mobile users may be finding nearby places such as restaurants, cinemas, ATMs, and so on.

Geospatial data are data that are related to location information and its attribute information. Geospatial data are stored in spatial database. Thus, LBS applications need to build a spatial database, which includes geo-information data. Spatial database provides geo-location information objects such as finding the places of interest object to a given location. Spatial database system keeps in storage server which acts as a back-end system.

1.1 Spatial Database for Location-Based Services

Spatial databases have been used for data management and data analysis over the past decade. Spatial database management system has many challenges that are used in location-aware applications and decision support systems based on geographic location. Most spatial database applications have been developed only for conventional relational databases that cannot handle the data complexity of points, lines, and polygons. Hence depending on the use of data types and operations need to focus on efficiently manipulating, retrieving, and storing spatial data. Geo-spatial applications usually store spatial data and non-spatial in spatial database. Thus spatial indexing techniques are being required to be effective to access spatial information.

Nowadays, with the growth of LBS development, the vendors of LBS need to provide user's effective geospatial information. Spatial databases in mobile environments have better performance for mobile devices. It can be high-speed data processing time, network response time, efficient memory management, and network transmission. But it may be an excessive amount of space and long queries. Therefore there is a need to use an efficient spatial indexing method to solve these problems.

1.2 Spatial Access Techniques

Spatial access techniques are to support querying spatial objects sufficiently. Furthermore, these techniques are also used for efficient spatial data predictions. Spatial data may be usually large and data is quite often complex. A spatial access method needs to use spatial index methods. Without the spatial indexing method, it may search every object in the spatial database. Therefore, a spatial index is needed to efficiently find the data that users want without having to scan the entire database. However, most spatial data sets may be located the big amount of data relying on the memory [4].

In this research, the proposed system helps in reducing the workload of both server and client with minimum response time and with minimum computing cost. Mobile device processing units, memory, and power specifications are limited. Mobile clients must be designed to balance these resources between the client and the server.

Location-based services (LBS) as a promising field of study bring new opportunities and challenges to accessing, sharing, and dissemination of geographic information. The client users can access the required information according to the around locations and surroundings through pull and push mode.

1.3 Objectives

The objectives of this research are:

- to create an index structure for retrieving geolocation objects in the spatial database,
- to give useful information for user's preferences services,
- to provide useful information efficiently from anywhere and anytime,
- to perform range query and kNN query for distance keyword search and nearest neighbour search,

- to minimize the search space and storage space by using R-tree and grid indexing techniques,
- to retrieve k nearest neighbour results quickly and efficiently.
- to create an effective system for location-based services by using index structures: R-tree and Grid index

1.4 Motivation

Location-based Services applications are one of the special interests of the users in the mobile computing environment. Mobile users can get location information by using location-based service applications. Most mobile users need compatible LBS applications on web applications and mobile applications. In LBS, the location-based query returns the result according to the user's request. Nearest neighbour search that is a sub-class of the location-dependent query is an important function for LBS. Query processing is a key role of location-based service providers to provide the geolocation information sufficiently to mobile users. Most mobile users request the places of services from their desired location nearby. LBS content providers need to implement users' preferred query types such as nearest neighbour query and range query.

Location-based keyword searching and categories-based searching are the necessities of mobile devices users. However, mobile devices in mobile computing environments confront limitations, and challenges (e.g., energy consumption, out of memory space, long queries). LBS consumed the power energy usage of GPS or Wi-Fi in the located device. Power consumption is a major constraint in smart-aware devices. Therefore, it needs effective query processing in location information retrieving. Concerning limitations and problems in this environment, spatial indexing methods are needed for retrieving the nearest results efficiently. To implement an LBS system, the database system must be able to store, update, and query the location of large numbers of geolocation objects. This poses new challenges to database technology. User reliable services are needed supporting what kind of location-information users need and their nearby services are available.

1.5 Organization of Thesis

The dissertation is composed of six chapters, references, and a list of publications. Chapter one introduces the challenge of spatial databases for LBS. This chapter explains briefly Location-based services (LBS). Spatial Access techniques are also described. The Objectives of the thesis are also described and the motivation of the thesis is given in this chapter. Chapter two will discuss literature reviews that have been done in GIS and Location-based services (LBS). It will also discuss the basic Components & Functionality of LBS. It will discuss applications of Location-based Services. Chapter three describes a detailed description of indexing approaches in spatial processing that are used on the server-side for indexing. Chapter four shows the system design and implementation of the system for nearest neighbour query. Chapter five explains the experimental evaluation of the proposed system. Limitations of the system are discussed and the conclusion for this research is described in chapter six.

CHAPTER 2
LITERATURE REVIEW

This chapter describes a brief of Geographic Information System and location-based Services with some research works. Reviews of location-based queries and nearest neighbour search are also discussed in this chapter. In detail, spatial query methods on the spatial database are also described with the related works.

2.1 Geographic Information System

Geographical Information Systems (GIS) has become a real-world phenomenon. The growing number of mobile phones devices with internet access enables the users to get real-time information from anywhere. Information and communication technology is a combination of different technologies and is therefore interrelated.

Geographic information systems or GIS are used to collect, store, forecast, and edit all types of surface-related data. As such, it supports all types of query results related to geographic information science. Now it is tremendously applied in resource planning, spatial data management, asset management, transportation, environmental science, and observation. The geographic Information System is a system that allows users to make querying, analyze geolocation information, modify the map, and display all related operations. GIS is becoming an indispensable system for combining different maps and remote sensing information that is used to generate different models in real-time environments. Geographic information systems are the science of applied geographic concepts, applications, and systems [1].

Geographic information systems can be used for resource management, city planning, asset management, environmental impact assessment, mapping, scientific research, criminology, logistics, sales, and marketing. For example, agricultural planners can use geographic data to combine soil, terrain, and rainfall data to determine the size and location of biologically relevant areas, making them ideal for location-specific crop planning. You can decide where to go. The final output may include land ownership, transportation, infrastructure, labor availability, and an overlay of the distance to the market center.

2.1.1 Location-based Services (LBS)

Location-based services (LBS) are a popular service that uses navigation technology provided by mobile wireless networks. LBS is a wireless service that uses Geographic Information Systems (GIS) and other satellite navigation platforms to identify information for mobile device users.

LBS are currently used in emergency services, road navigation, fleet navigation, and various other application areas that require location information. Other location-based services offer the positioning service for various applications combined with GIS and Global Positioning System (GPS). Figure 2.1 shows the convergence technologies of location-based services.

For example, if someone is in an unknown location and the user needs to know the nearest hospital, hotel, or other when requesting a service provider on a mobile device, the operator confirms the user's location. The user-preferred services are stored in the database and the services provider responds to the nearest one. Grab Services and Uber services offer the user essential services which provide nearby taxis according to the user's request.

Therefore, LBSs are primarily provided by cellular networks with wireless technology and offer a variety of services. Mobile phones with GPS capability will locate and send to the operator. Therefore, several mobile operators around the world are providing location-based services for a wide range of applications. Increasing the number of players in this area creates healthy competition, which will later be modified in the interests of both consumers and operators. In addition, operator premium rate billing provides users with the opportunity to explore the absolute potential of mobile services. In this way, LBS provide a comfortable life and ease of use for mobile phone users.

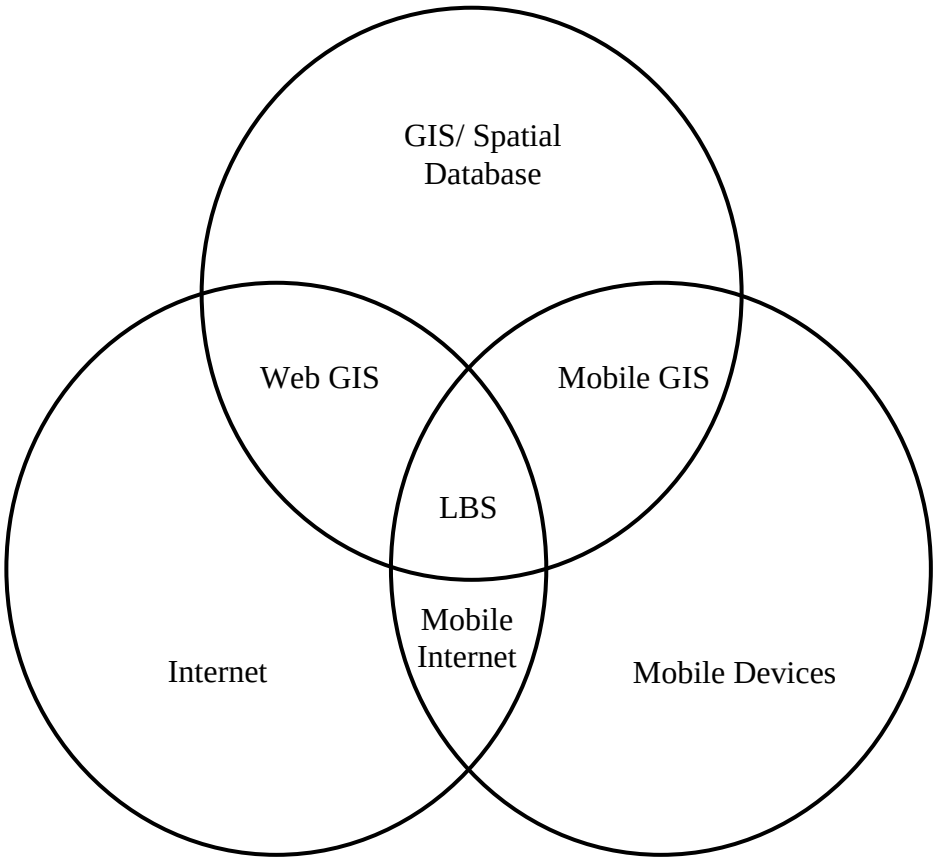


Figure 2.1 Convergence Technologies of LBS

Location-based services (LBS) are used to locate the user's position by providing mobile devices, mobile communication networks [2]. LBS provides valuable information to the users. The most user-requested services are where the nearest places and the most suitable route to go to the desired place. The popularity of location-based services is the most important and useful asset for users in almost every industry. However, location-based application variations fall into different categories such as navigation, emergency assistance, tracking, advertising, billing, management, games, and leisure.

Location-based services allow mobile users to find location information and provide a variety of location-related services. Location-based services (LBS) use cellular networks to provide other value-added services, primarily based on the user's location.

2.1.2 Basic Components and Functionality of LBS

Several key components of location-based services are smart-aware mobile devices, positioning capabilities, communication networks, service and application provider, data and content provider, and user. Figure 2.2 introduces the components of location-based service.

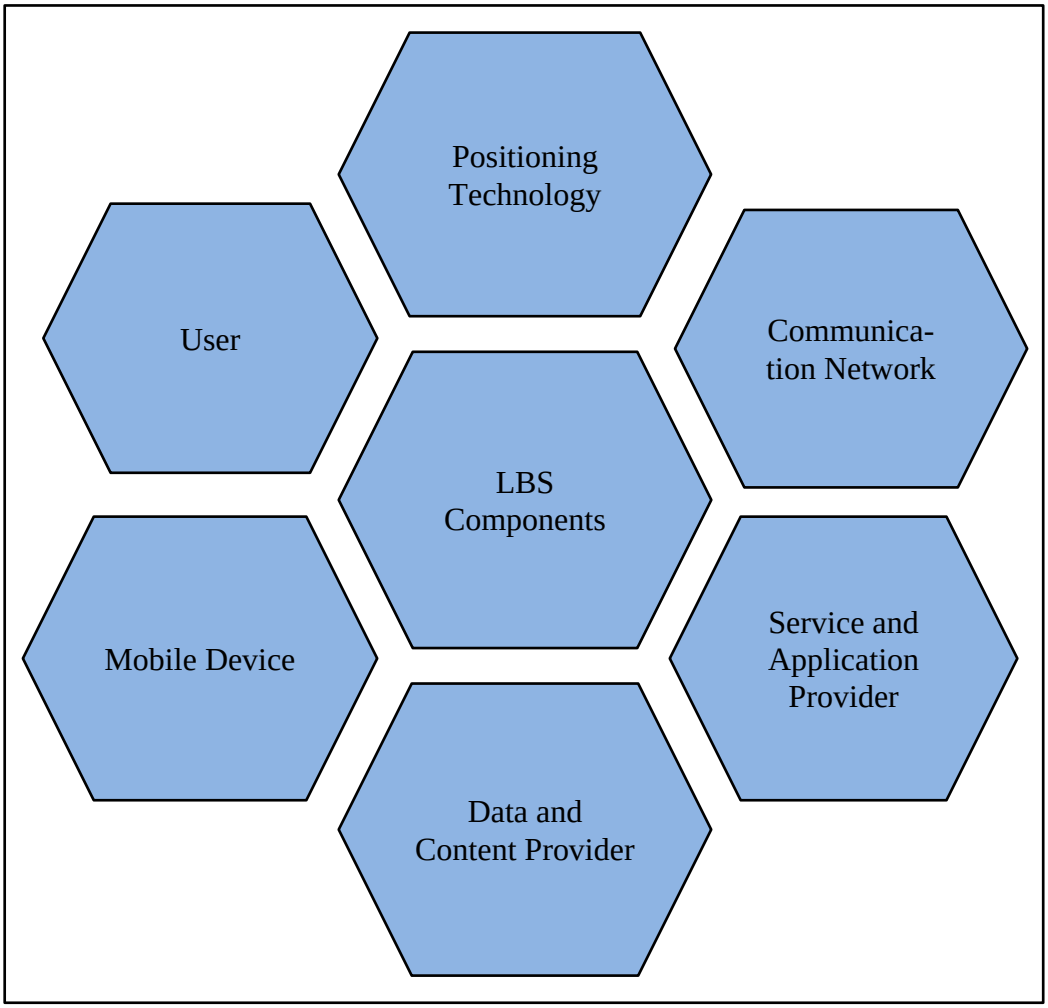


Figure 2.2 Components of Location-based Services

1. Mobile device: Mobile devices refer to all types of smart-aware devices that can access various forms of information. The devices may be specific and multi-purpose. The mobile client user needs to request the desired information according to the specific location.
2. Positioning: The positioning component establishes the location of the device. The mobile device is used as the positioning system for indoor and outdoor. GPS is often used outdoors to determine to track and often used

to improve positioning accuracy degradation. For indoor LBS applications, other technologies such as WiFi, Bluetooth, and RFID (Radio-Frequency Identification) are available. [2].

3. Communication network: Cellular network and Wireless network deliver the data transfer and service request of the user's mobile device to the content provider and send to the user's device the requested information.
4. Data and content provider: The service provider serves as submitting the user requested information and returns the user's needed information. The service provider offers different services according to the client's requirements.
5. Service and Application Provider: The service provider supports the applications and other related services and components that are solved the user's request and respond to the user.
6. User: The person who is using location-aware smart devices to get reactive (pull) information and proactive (push) information.

2.1.3 Architecture of LBS

The majorities of the location-based services applications have client-server structural design and can be abstracted into three foremost parts which are client, server, and wireless communication to hook up client and server. The client is accountable for the distribution of the user's demand and the position of the mobile device to the server, and the server is accountable for offering services based on the position of the mobile device. The client can make assistance to data acquisition by gathering information. The server will put the data collected from the field into the database and will then offer functionality for all clients based on the database [3]. The structural design of LBS is revealed in Figure 2.3.

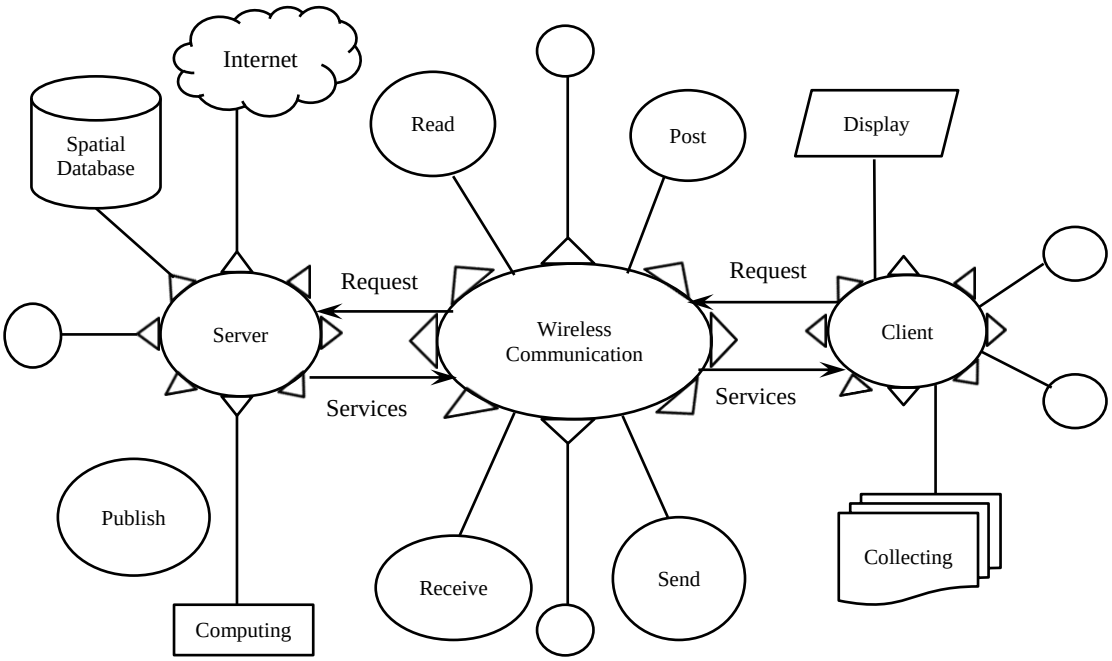


Figure 2.3 Client-Server Models of LBS

Most LBS applications have client-server architecture. The client is responsible for sending the user's request and the geographic location of the mobile device to the server, and the server is responsible for providing services based on the geographic location of the mobile device. The server puts the information collected from the fields into the database and serves all clients based on the database. Figure 2.4 shows the function of the server system.

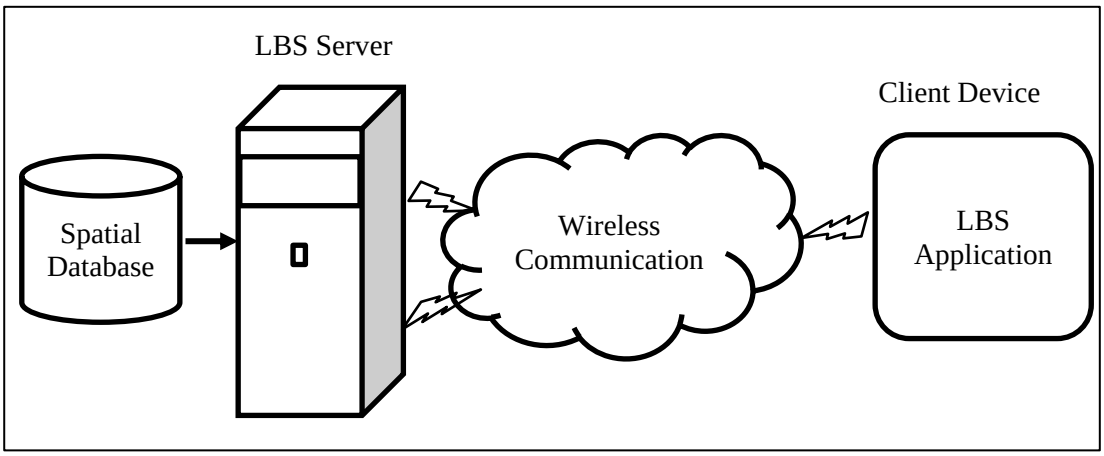


Figure 2.4 Server System

LBS applications provide client users of mobile devices which request queries to the server. The main idea of location-based services is that a mobile device sends its current location to the server. The server retrieves the relevant geo-location

information and responds to the client. Google Maps are being used to develop in the research field of geo-spatial applications. LBS integrate the location of mobile devices with other information to add value to users. Figure 2.5 shows the connection between the server and the client device.

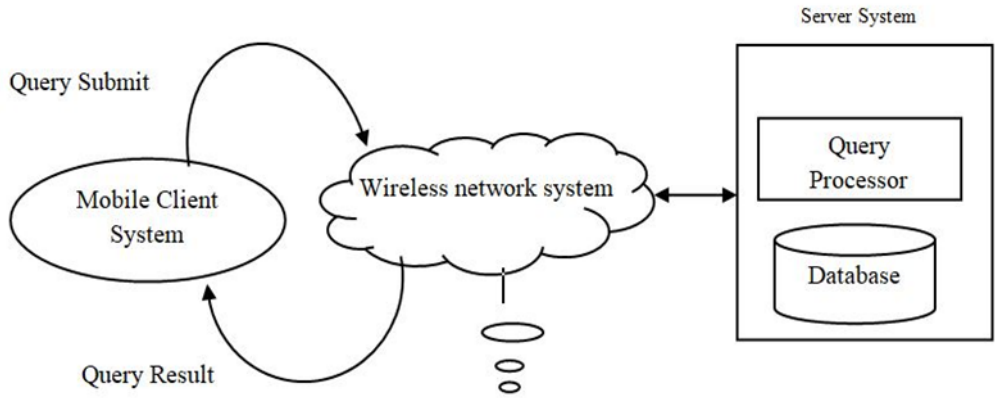


Figure 2.5 Client Systems for Mobile Devices

Most mobile clients use the Android platform and ios platform. A typical Android application consists of activities and services. Activities are the basic application components that provide a view of the user interface, and services are application components that do not provide the user interface but perform background tasks. The mobile client can get its current location via GPS built-in devices. The mobile client reports to the server its current location information. It sends the query of user interests to the server.

The LBS design includes several components such as maps, GIS, and location collection services. GIS providers provide geospatial capabilities for many LBSs, including map information, map visualizations, and directory services. Google Map API is used to process a map in an LBS system such as Map View, which displays the map, it requires Google's Map API key. It is needed to activate the Google Maps API v2 service at "https://maps.googleapis.com/ maps/api". The API key can be obtained from the API Access section of the API Access page. The server transfers the data to the client. To perform this task, the server gathers and analyses the locations of the objects on the map. The server identifies and analyzes the location of objects on the map and structures them into indexes. The server is an essential component for building efficient spatial data in LBS applications. The server provides various spatial

information (digital road map, etc.) and spatial information (route guidance instructions, etc.) according to the request of the client. It also provides a variety of geospatial query processing capabilities, such as finding the closest neighbor (such as a restaurant) to a particular location. These limitations require an efficient main memory spatial processing algorithm and an intelligent user interface. To reduce the time it takes to find a feature that matches a spatial query on the server-side, it needs an efficient spatial index method. The functions of the client-server system and wireless communication are shown in Figure 2.6 and the functions are described as follows:

1. Client functions: Clients send requests and location data for processing to the server. The client can contribute to the acquisition of information by information collection in the field.
2. Server functions: Server must be a high-performance computer system because the hosting server acquired several requests from the clients. In addition, the hosting must be responded to the clients. Therefore, the infrastructure of the hosting server should be used higher computing products depending on the number of the clients' requests.
3. Wireless Communication functions: Wireless communication is performed by delivering information and receiving data communication from the endpoint in real-time [3].

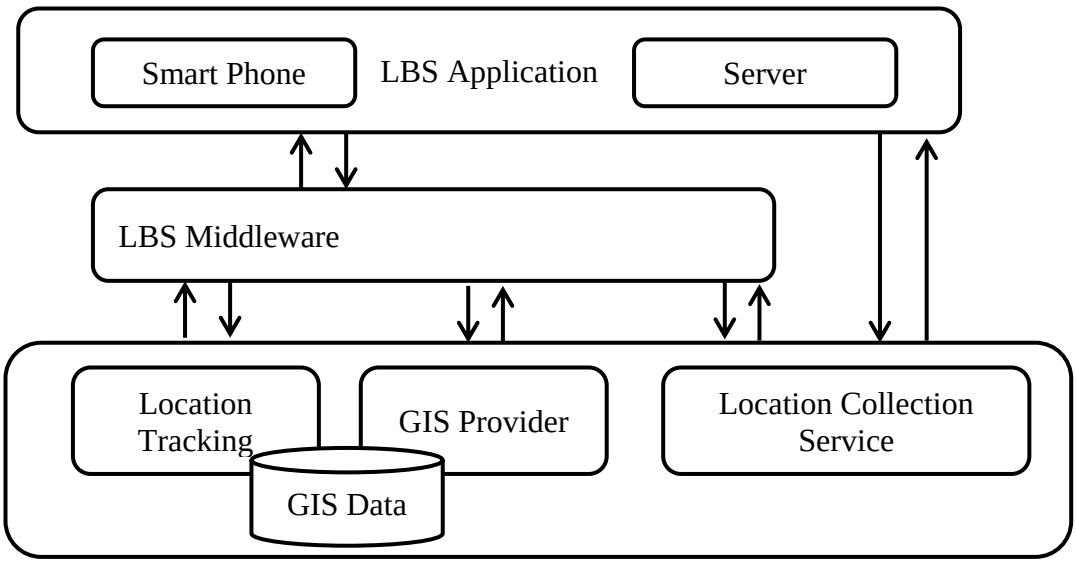


Figure 2.6 Function of The Client-server System

2.1.4 Reactive (Pull) and Proactive (Push) of LBSs

Reactive (pull) services are explicitly called for establishing by the user and it takes a service session between the client application and the LBS server [3]. A Reactive LBS application is triggered by the client based on the location and sends the queries request to the system to search information. LBS content provider responds to the user's required results. Thus, reactive LBSs are required to synchronize the clients and the service providers as shown in Figure 2.7.

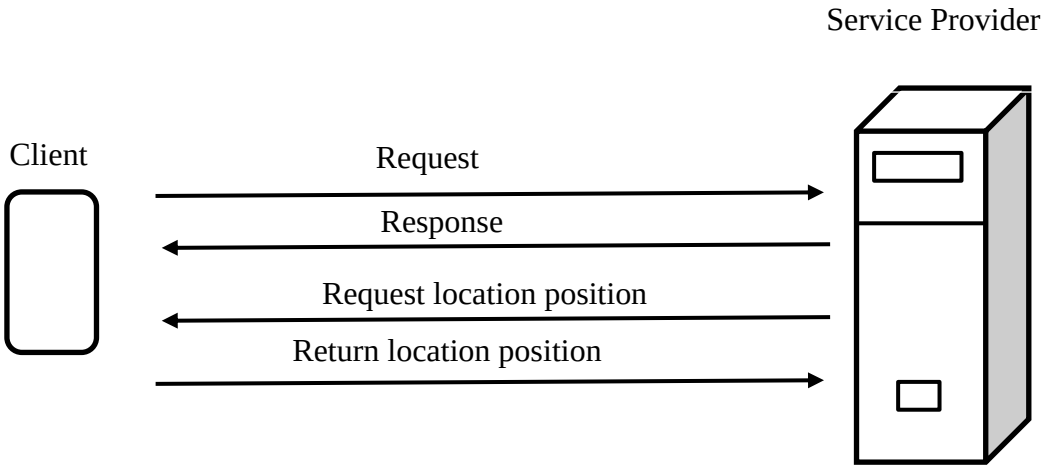


Figure 2.7 Reactive LBSs (Pull Services)

Proactive (Push) services are automatically initialized as soon as a predefined event occurs that no need to explicitly request by the user, but the user has to register formally. Figure 2.8 shows the service provider provides the related services or gives notification when the client turns on these services.

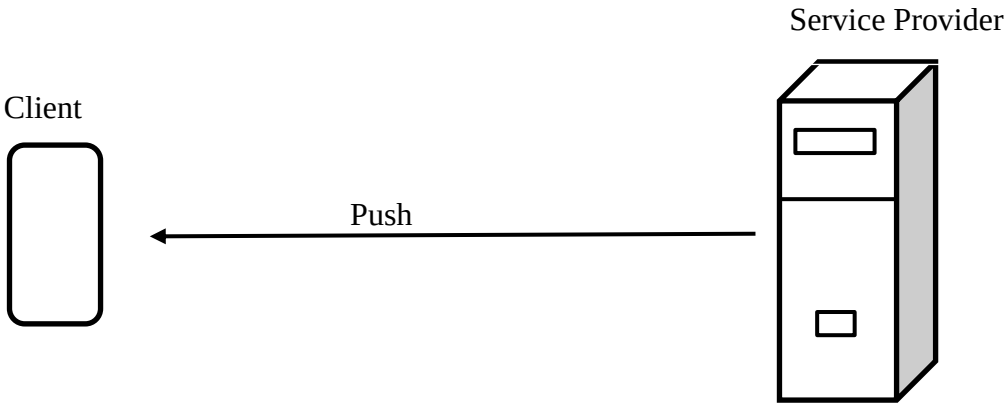


Figure 2.8 Proactive LBSs (Push Services)

2.1.5 The Usage of Location-based Services in Several Areas

The applications of LBSs are useful and essential in all industries areas. The applications areas can be divided into different categories which include directory assistance, fleet management, emergency, asset tracking, directions, points of interest, and location-sensitive buildings[2]. LBS applications can categorize into four main types which are maps and navigation, tracking services, information services, and applications. Figure 2.9 shows the key areas of LBS applications. The following application services are provided by LBS.

1. Emergency Services: It assists to determine the mobile user's location in case of any emergencies. When the mobile user has happened in an emergency (injury, criminal attack, accident), the service can define the user's location on the map. The device transfers the geolocation position to the service provider that responds quickly and efficiently.
2. Navigation Services: Navigation services send the best optimal route to the mobile user that on the location position of the user. Navigation services require a mobile network and Global Navigation Satellite System(GNSS) which is to extract the exact location position.
3. Information Services: The mobile users can access the geo-location information which gives nearest services like hotels, restaurants, clinics, and cafes from anywhere. The mobile users can get the real-time traffic congestion and optimal route finding to the destination by using the service provider.
4. Tracking and Management Services: Tracking service is applied to the mobile customer and the corporate sector. Vehicle tracking is essential for monitoring the real-time movement of vehicles and can also be applied to fleet management for delivery and better providing user requirements. The fleet management system is essential for all industries that enable the customers to better manage customized solutions.
5. Billing Services: location-based billing is used by the mobile operator which enables to offer the users for getting wireless service depending on the specific location with different rate plans [2].

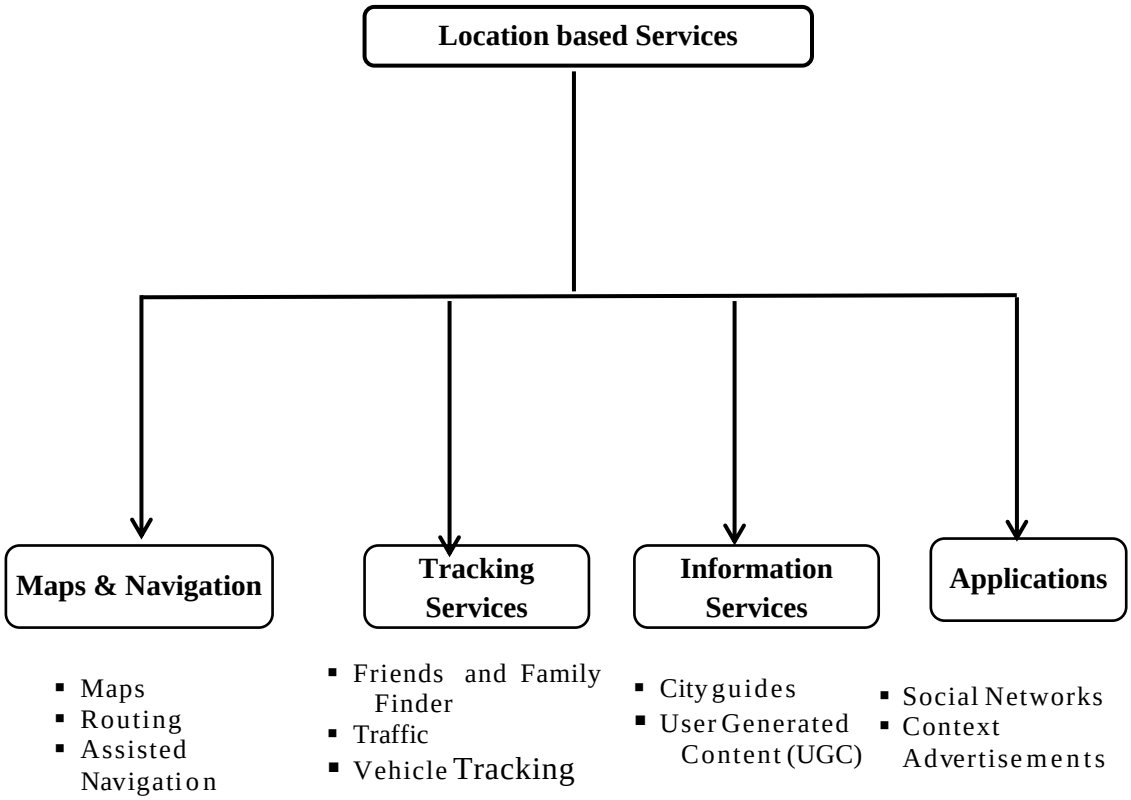


Figure 2.9 Types of LBS

2.2 Location-based Queries

As mobile device usage grows, location-based service providers provide different types of location-based queries. Mobile users are searching for relevant information from the surrounding area anytime, anywhere. Traditional queries may have different query execution structures. If the user is on the move, it may be necessary to identify location-related attributes to respond to location-based requests. [5]. Location-dependent queries can be divided into two main parts. The first one is the type of user or object, and the second is the query type. The user type represents the state of the device location when issuing the query and the retrieved object. Query types and categories are related to the state of the query, whether continuous or non-continuous [6]. Location-based query processing uncovers location information for mobile users and extracts location-based information. Many studies have focused on protecting the privacy of mobile users. Many other studies have focused on reducing server load when handling the mobile query. Many research papers describe different approaches to efficient spatial query processing [7].

2.2.1 Range Query

A range query is a query that retrieves the data that is related to other features at a specified distance. Range queries can be abstracted into this equation: (Dmin <distance (A, B) <= Dmax), Where Dmin and Dmax are distance values, distance is a function that finds the distance between A and B. (A, B) <= Dmax can be assumed as a special case when Dmin = 0.

Range queries can be static distance search or moving distance search. If the search area is the rectangle, it is described as a window query. It also may be static or dynamic [7]. It depends on the region of interest that is moving or not. Table 2.1 describes query classification for range queries and their variants.

Table 2.1 Query Classification

Query Type	**Variants**
Range Queries	Static and moving range queries
	Window queries
	Within distance queries DS, DD, SD, SS
	Inverse range queries
Nearest Neighbor queries	Nearest neighbor (NN) queries
	Incremental Nearest Neighbor queries
	Aggregate Queries, Approximate Distance Query
	Distance Join Queries, Group NN Queries
	Closest pair query
	Continuous query
	Reverse NN Query
	Spatio-temporal query
Navigation	Trip and In route query
Point Queries	Closest, Whereat, When at

2.2.2 Nearest Neighbour Query

Nearest Neighbor (NN) queries are responsible for retrieving the feature that is closest to a specific location. kNN query retrieves k objects besides the closest one from the query point that describes in Figure 2.10. The spatial database is used to perform static queries and dynamic queries. Inverse k-NN queries can fetch objects at specified positions within k-nearest neighbors. A constrained NN query is a range-constrained query of the retrieved object in the database. [7].

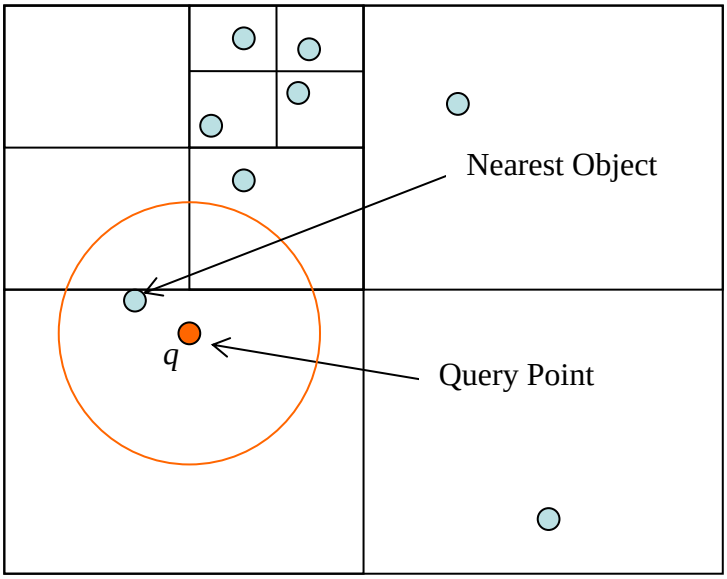


Figure 2.10 Nearest Neighbour Query

N. Roussopoulos et al. [9] presented the method for scanning an R-tree along branches and boundaries to retrieve NN objects to the query point and generalize it to find the k nearest objects. This task describes two key metrics which are optimistic and pessimistic search ordering strategies and pruning. The optimistic metric minDist is defined as the minimum distance between a query point and the nearest point. minDist is the lower bound of any k-NN distance. The pessimistic metric minmaxDist is defined as the farthest distance at which NN request points can exist. The main idea is to visit from the root according to the minimum distance in depth-first search.

K. Cheung et al. [10] proposed the NN search improvement transformations using R-tree. The NN search improvement is used two heuristics metrics which are minDist and minmaxDist. In addition, it was found that these two metrics do not increase to prune. Therefore, no calculation of minmaxDist is required. This is at least as powerful as the previous trim function algorithms. G. Hjaltason et al. [11] suggested a best-first search paradigm for nearest neighbor querying. Unlike depth-

first search, the next extended MBR always has a minimum distance (minDist) to query all subtrees that are being accessed. Thus, the priority queue is maintained to support entries in the MBR that are expanded using minDist as ordering. First, the route is placed in the seniority queue. At each stage, the top-level element of this seniority queue is placed for processing, and its child MBR is placed as a candidate. This search process ends when there is an item at the top of the queue. It only visits the MBR it needs. Therefore, it is necessary to preserve the size of the index entry heap and the minimum object ordering distance.

2.2.3 Navigation Query

Navigation queries can take network traffic into account and get the appropriate path to the destination for mobile users. These types of queries use one of the following formats: Predicates apply to many objects. For example, the closest adjacent query or predicate is checked for all objects related to other objects. Continuous queries are evaluated until interrupting the user chosen. Persistent queries are considered the state of moving objects [7].

2.3 Spatial Data and Spatial Database Management System

Spatial data is data that can be viewed, manipulated, and analyzed using related attributes that indicate locations on a map. This spatial attribute is usually provided in pairs of coordinates that take the position and shape of the particular feature to measure and represent graphically [8].

Spatial data is collected and stored in two basic formats called vectors and rasters. The vector format is the geographic objects that are a point, line, or polygon. Vector data can be stored as part of a terrain base that can provide a spatial reference framework for data collection and analysis. The data in the topographic-based dataset includes geodetic and survey control points, as well as all the features commonly found in common topographic maps, such as roads, rivers, urban areas. Application vector data is data on the state of the natural environment (i.e, environmental protection areas, forest resource inventories, air, and water quality), land, and its resources' human activity and utilization for network of conversion and registration, transportation, and public interest business.

2.3.1 Spatial Database Management System

The Spatial Database Management System (SDBMS) is designed for users to easily create geo-data, SDBMS applications are used in systems such as urban transportation and remote sensing. Databases have been used in business and management applications traditionally. Common data types that occur in such spatial database applications use integers, floating-point numbers, characters, currency units, and dates. This set of data types is difficult to operate in real spatial applications. Therefore, recent studies on database systems focused on the efficient storage and management of complex data.

Spatial data can be found on the roadmap. A roadmap contains 2D objects that can represent cities, roads, and boundaries such as states and regions. A roadmap is a visualization of geographic information. All objects that are located on the earth's surface are projected onto a 2D display while maintaining the relative position and distance from the rendered object. The data describes latitude, longitude, height, and depth that are the rendering objects. GIS is used to visualize spatial data related to the earth [8]. The main advantages of this approach to spatial data management are:

1. Traditional GIS users have access to a complete spatial information system based on industry standards with an open interface to data (such as SQL).
2. Spatial data is now stored in the enterprise-wide Database Management System (DBMS), allowing many enterprise applications to be spatially enabled.
3. Reduce system management complexity by eliminating the hybrid architecture of the GIS data model.
4. Enables seamless integration of GIS data, provides applications that meet the increasingly demanding analytics, and reports the needs of a growing geospatial user community.

The functionality of the generic DBMS includes a user-accessible catalog with metadata, a DBMS library management system, data abstraction and independence, data security, activity logging, and auditing. The database schema design technique that works to clarify the organization of data is called normalization. DBMS normalization minimizes data redundancy and dependencies by modifying an existing schema to reduce the data repetition of the database’s size and define the relationships between them. DBMS Output is a packaged SQL that allows users to view debug

information and output and send messages from subprograms, packages, PL / SQL blocks, and triggers. Oracle database system initially developed the file transfer package that provides instructions for copying binary files within a database and transferring binary files between databases. Figure 2.11 shows the simplified database management system how the DBMS takes functions between the users and or querying and the storage device which relies on storing the data.

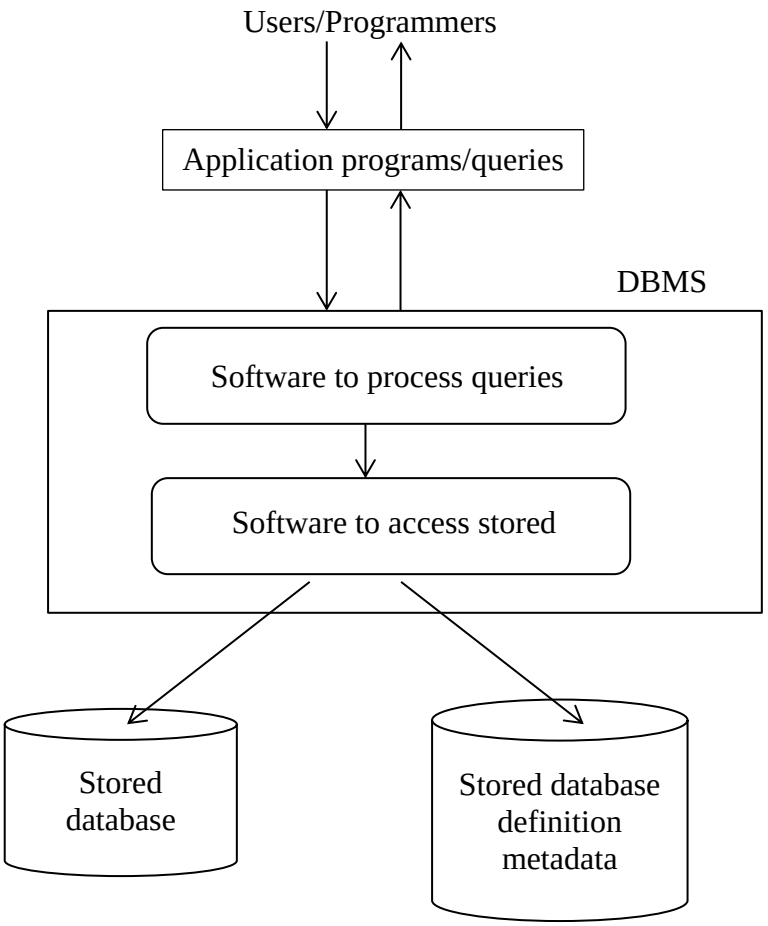


Figure 2.11 A Simplified Database System Environment

2.4 Location-based Keyword Queries

Location-based queries refer to answer location information that exists around the mobile user's location. LBS queries have been provided in a vital role in real-world mobile applications. The spatial query is a set of queries that has query type with location coordinates. With the support of location-based queries types, the mobile users need to input and search for spatially and textually relevant geolocation features.

2.4.1 Top-k Keyword Queries

Top-k queries retrieve the *k* objects which are ordering scores from the distance of query point and the query keywords that are the textual description of relevance. Most LBS application domains support the top k query answers that end users are most interested in querying objects [12]. Top-k query processing is applied in many research domains that implement query optimization, spatial indexing techniques, and query languages. In the common use of spatial indexing techniques, R-tree index is built into the relational database for sorting objects based on their scores. Information retrieval R-tree (IR2 tree) integrates the signature file with the R-tree to respond to keyword search. Each node in the IR2 tree has at least one signature that represents the object associated with the text of every leaf node[13].

R-tree with an inverted file drives for inverted file and spatial proximity searching. The inverted file is applied to retrieve text relevancy and R-tree is applied to search location proximity. This framework includes the computing top-k query that uses both textual data and location proximity to prune the search space [14]. Location-aware keyword queries return ranked objects. The text description matches near the location of the query keyword. Location-aware top-k Prestige-based text search (LkPT) query, is proposed to get the top k spatial web objects ranked according to both prestige-based relevance and location proximity. This query occurs in many types in essence mobile applications such as google map service [15].

Chen et al. [16] provided a comprehensive survey of the status of geo-text indexes and propose benchmarks that allow performance comparisons of spatial keyword queries. They considered the support for boolean kNN queries, top-k queries, and range queries.

Zhang. [17] proposed bR tree that is the modified of the R-tree. An m-closet keyword query (mCK query) is aimed at querying the nearest object that matches the query keywords and minimizes the diameter of their distance. Priori-based search strategies based on the bR tree to prune search space. The collective spatial query employs different semantics so that the group of resulting objects covers the query keyword and minimizes cost. Top-k keyword query for direction-aware is to find a set of trajectories that is not only geographically close but also the query needs

meaningfully. They provided new similarity functions, hybrid index structures, efficient search algorithms, and further optimizations for efficiently answering queries [18]. Classification techniques for top-k query processing based on multiple dimensions are shown in table 2.2.

Table 2.2 Classification of Top-k Query Processing Techniques

Query Model	1. Top-k Selection 2. Top-k Join 3. Top-k Aggregate
Data & Query Certainty	1. Certain Data Exact Methods 2. Certain Data Approximate Methods 3. Uncertain Data
Data Access	1. No Random 2. Both Sorted and Random 3. Sorted and Controlled Random Probes
Implementation Level	1. Query Engine 2. Application Level 1. Indexes/Materialized Views 2. Filter-Restart
Ranking Functions	1. Monotone 2. Generic 3. Unspecified

2.4.2 Continuous K-Nearest Neighbour Search

K-nearest neighbor (k-NN) query is the basic type of query in spatiotemporal databases. The k-NN query retrieves the k-nearest neighbor closest to the query position. A continuous k-NN (CkNN) query is a k-NN query that runs continuously and updates the results whenever the movement of the objects changes the query

results. In addition, spatiotemporal database servers are expected to handle lots of moving objects and lots of consecutive queries [22].

Most users require the system which is to respond to the query as soon as possible or to process the query efficiently. However, the load of location-dependent service is heavier as it becomes more popular, and the response time increases since more mobile objects are monitored and more continuous queries are registered in the system. A grid cell level-based continuous k-NN query processing algorithm, called CkNN for short. It utilizes the main memory to store the movement of objects [23]. CkNN processes newly registered queries by k-NN search algorithm. It searches the k-NN of queries according to the partition of grid cell level [44]. During query processing, CkNN tries to minimize the cost of checking grid cells and moving objects. While processing static continuous k-NN queries, CkNN employs an incremental update and query processing algorithm. The incremental algorithm makes the most of information obtained in the last query processing phase and attempts to reuse the data produced in query processing as moor as possible. Comprehensive implementation results show that CkNN is superior to the continuous k-NN query processing approach in all problem settings. Huang et al. [24] considered efficient processing of multiple k-NN queries. They have shown different approaches for reusing main memory caches and previously calculated queries. He also reported on an empirical study of proposals using data from the actual road network and place of interest.

2.4.3 Nearest Neighbour Query for Location-aware Mobile Network

With the growth of interest in mobile networks, there is a need to consider querying the nearest object from any mobile nodes in the network. In LBS, it is common for a node to issue a query to retrieve information about a given location held by a mobile node. This query finds kNN which is the k closest sensor data items (location-dependent sensors) associated with a particular location from the specified query point. The kNN query method has been proposed in various environments such as peer-to-peer networks, wireless sensor networks [25]. Figure 2.12 shows an example of the function of kNN query in the mobile network where the construction worker gets the nearest information about victims and damage of buildings.

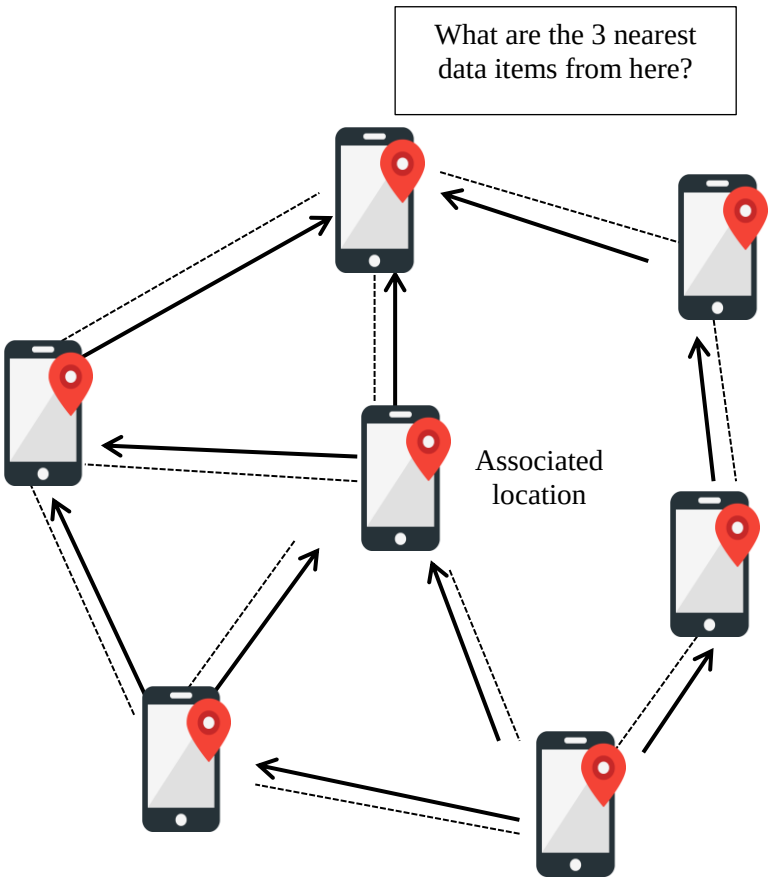


Figure 2.12 Example of a k-NN Query in Mobile Network

Query issuing nodes can get k-NN from nodes in a small area, so if the node caches the required data items, users can effectively search for the data items. Therefore, effectively caching data can downgrade search network traffic congestion and response time.

The Filled Area (FA) method kept the query processing overhead low by reducing the search area. In this method, the data item remains on the node near the location where the item is associated, the node caches the data item near its location, and the query issuer gets the k-NN from the nearby node. Through extensive simulation, they have verified that to achieve low overhead and high accuracy in query results.

Network bandwidth limits and dynamic topologies change as mobile nodes move. Frequent exchange of beacon messages by mobile nodes detect accurately to change in the location of neighboring nodes results in increased traffic, frequent packet loss, and inaccurate query results. If a node does not exchange beacon messages frequently, traffic will be reduced, but the node will not know the exact location of its neighbors. The k-NN query processing technology assumes that

location data is available in the database and focuses on improving performance. A k-NN query processing method for maintaining traffic and query results in mobile networks. The writer also addressed the dynamic adjustment of k-NN boundaries to address the spatial irregularities and mobility of ad hoc nodes. Yadhav and Rajalakshmi [26] proposed two k-NN query processing methods, the Explosion (EXP) and Spiral (SPI) methods to reduce congestion and to make efficient query results.

2.5 Indexing of Spatial Data

Spatial indexes are data structures that give the users efficient access to spatial objects. This is a common technique used in spatial databases. Without indexing, all records in the database would have to be "sequentially scanned" when searching for features, which would significantly increase processing time. In the spatial index building process, the smallest bounding rectangle acts as an object approximation. Different types of spatial indexes across commercial and open-source databases make measurable performance differences. Spatial index technology plays a central role in the manipulation of time-critical applications and spatial big data.

2.5.1 Spatial Data Access of Location-based Services

LBSs are one of the applied services in mobile devices. Mobile client users can access LBS applications efficiently provided by service providers. Range queries (eg. window query and radius query) and nearest neighbor queries are mostly used in LBS queries. Frentzos et al. [21] proposed searching and pruning strategies to perform NNs stationary query points or trajectories and generalize them to find the k-nearest neighbors. These algorithms are applied to variations of the R-tree of orbital data, in which both the 3DR tree and the TB tree are used for performance research. A grid index file is organized into a two-dimensional structure. The geographically related data are stored in the same data block. The grid index cannot represent objects, it can only present points. The only kind of index that can handle where I am queries: is given by a location (i.e., coordinate), and then find the objects that contain the location. Therefore, the grid index for range queries is to find objects that are located within a certain range. The purpose of the grid index is to find the nearest neighbor of

the data point [19]. The grid covers the part of the space where the object's position is within the boundaries of the grid cells and this object belongs to this cell. Šidlauskasetal [20] proposed a grid index, which is one of the main framework index structures for moving object databases. The performance of grid index is commonly used in multi-dimensional queries. The grid index file can be stored entirely in memory. The storage structure of a grid index file stores the size parameters m and n which is the block pointer of the grid. Then index files store the buckets of the grid. A grid is a two-dimensional array in which all cells in the array match square cells of grid cell size length. Every grid cell in the array has a list of buckets that contain data objects [20]. Each bucket has a bucket size and a metadata field that contains the number of entries, and a pointer to the next bucket.

Abd Elwahab et al [27] implemented two well-known indexes used in spatial databases. Web applications have been implemented in the cloud to provide location-based services where users search for specific locations. The system is used aRB-Tree which is called Rectangle-Tree (R-tree) and the Balanced-Tree (B-tree) to search the created medical institution database containing the addresses of doctors, hospitals, clinics, etc. Users use search queries to request specific medical information, and the system uses R-Tree indexes to retrieve results. However, if the user requests the details of these results, the system will use the B-tree index. The system has been implemented and is currently being validated. The results approved that aRB-Tree is more efficient than R-Tree.

Boucetta et al. [28] proposed mRTree, which can handle complex queries and large amounts of data that are obstacles to mobile GIS development. This technique used the hybrid method which combined R-tree and quad-tree indexes. This choice saves time in communication between the server and the mobile client. This technique can reduce the processing delay time compared to the case without indexing. Therefore, Broadband wireless networks contribute to speed up query processing, even in difficult situations, but spatial data indexing algorithms like mRTree have made a significant contribution in this case as well.

Distributed Spatial Index (DSI) is popular for supporting LBS in wireless broadcast systems that mix multiple search paths into a linear index structure in broadcast cycles. DSI is highly resilient in error-prone wireless communication

environments. Two classic location-based queries based on DSI are window queries and k-NN queries.

Wang-Chien and Baihua [29] proposed a fully distributed spatial index named DSI. DSI naturally mixes multiple search paths into a linear index structure that is completely distributed throughout the broadcast cycle. As a result, you can start the search immediately and recover the interrupted query processing immediately when the packet is corrupted or lost. This research has contributed significantly to the development of location-based wireless data broadcasting services.

The DSI method creates an exponential index on a spatial object whose Hilbert curve also imposes a linear order. Index indexes consist of index tables, each of which contains redundant index information. In the DSI method, the server broadcasts an interleaved spatial data object with the index table. Sungwon and Hyunho [30] proposed that a new spatial index called HMI (Hilbert curve-based MBR filtering index) is efficient for windows query and k-NN queries on a single wireless broadcast channel. HMI is a tree-structured index built on the Hilbert curve and MBR (Minimum Bounding Rectangle). By combining the MBR structure with the Hilbert curve order, HMI gains the benefits of both structures. The system uses Hilbert curves to create a linear broadcast order of spatial objects, and the MBR is used to group spatial objects to build an HMI. In this process, the HMI's internal and leaf nodes represent MBRs and spatial objects, respectively. HMI can provide high filtering capabilities. Based on the HMI, this system can provide efficient windows and kNN query processing.

The push method uses an air index, where the server sequentially transfers broadcast program information and actual broadcast content over a wireless broadcast channel, and the index data received by the client is used to selectively tune to the required broadcast. Kwangjin [31] presented an indexed structure and search algorithms need to be devised to overcome the problems of the wireless broadcast environment. The system proposed a new hierarchical distributed spatial index called DGI (Distributed Grid Index). This index divides the local area into grid cells and assigns each grid cell a unique ID number. DGI can pinpoint the location of an object using only the unique ID of the object. Unique IDs are sent from top to bottom,

specifying the location of known objects on the map, from the highest point object to the lowest point object.

The necessity for spatial query processing over smart-aware devices includes fast and accurate search and low power consumption. Most location-based services (LBS) are provided using an on-demand method. This is suitable for light load systems where there is less serious competition between wireless channels and server processing. As the number of LBS users increases, the server's query processing capabilities are limited, resulting in a rapid decline in performance. In addition, query response time can increase significantly at the same time as user queries are concentrated on the server. The server needs to determine the location of the user and potential objects for the final result before sending the response individually to the client over the point-to-point channel. At this time, the inefficient structure of spatial indexes and search algorithms can result in very large access delays.

To address this issue, Kwangjin and Valduriez [32] proposed the Hierarchical Grid Index (HGI), which provides a simplified location-based index structure for LBS. HGI minimizes index size by using hierarchical location-based identities. It also supports the process in broadcast environments through sequential data transfer and retrieval based on the location of objects. They also proposed top-to-bottom search and reduction counter search algorithms for query processing. Since HGI eliminates replication pointers, it is suitable for broadcasting environments and enables accurate spatial search by reducing redundant data.

In recent years, users have become more interested in location-dependent information services. Unlike traditional environments, mobile users are concerned about both access latency and power savings. Data broadcasts provide elegant scalability for unlimited mobile users. Therefore, data broadcast is also an important technique to reduce limitations in the mobile environment. It disseminates data to mobile users.

Two types of indexing are solution-based indexing and object-based indexing. Both of them have some disadvantages. Baihua et al. [33] proposed a new index structure, grid-partition index to address nearest neighbour search, which is important in LBS, in both mobile computing demand access and regular broadcast modes. This

partitioning method is a hybrid of a solution-based approach and an object-based approach. The result is a much more compact approach than a pure solution-based approach, avoiding the backtrack traversal required by the object-based approach.

Slimani et al. [34] focused on handling location-dependent queries based on the LBS contextual index structure. Traditional indexes (B-trees, KD-trees, etc.) employ either an overlapping partitioning scheme or a back track search algorithm. These features increase the response time of in-memory indexes in on-demand data access mode. Also, these indexes are not practical in broadcast data access mode. As a solution, they proposed a new index called the VNR tree (Voronoi-Neighbouring Regions tree) based on the Delaunay triangulation. The VNR tree employs a unique partitioning scheme, supports search algorithms without backtracking, and produces special clustering of Voronoi adjacencies. This clustering is suitable for processing some types of location-dependent queries based on the mobile client's search near Voronoi. Baihua et al. [35] studied energy-conserving air indexes for nearest-neighbour (NN) search which is an important class of queries in location-based services in a wireless broadcast environment.

2.5.2 Indexing on Continuous Moving Objects

In traditional spatial databases, objects are usually assumed to be constant unless explicitly updated, so indexes are primarily designed to speed up retrieval. Therefore, to capture continuously moving objects, traditional indexes need to continuously update the location of moving objects (for example, once per timestamp). Kollios et al. [36] presented an indexing technique for moving object databases. They suggested a way to index moving objects to return the result of distance search about the specified locations. This issue arises in real-world applications, such as predicting future congested areas of highway systems and allocating where mobile phone concentration is imminent. Their approach transforms the problem into an easy-to-index dual space. A major indexing technique for static spatial data that is less dominant in R-trees when faced with such a large amount of sampling conditions that are streamed to the database [28].

R-tree and its variants are implicitly or explicitly designed to support querying spatial objects effectively. However, applications that involve indexing moving objects show workloads that feature heavy updates in addition to frequent queries. It considered not only the process of the query but also the process of updating when the increasing number of moving objects is issuing updates. They compared with the TPR tree, which is to index moving objects based on time-parameterized index nodes. This approach uses equivalent query time processing, but at a much lower update cost.

Frentzos et al. [21] investigated the mechanism by which an NN search is performed in an R-tree-like structure that stores historical information about the trajectory of moving objects. The proposed (depth-first and best-first) algorithms depend on the type of query object (quiet or moving point) and the type of query result (whether history is contiguous).

Therefore, they have proposed new indicators to support search ordering and pruning strategies. Based on the proposed new metrics that support search and pruning strategies, we have presented algorithms that respond to NN query for resting query points or trajectories, generalizing them to find the k-nearest neighbours. Pfoser et al. [37] Focused on the space-time subdomain, the orbit of a moving point object. They presented a new type of spatiotemporal query and an algorithm for processing them. In addition, two access methods have been introduced that are spatiotemporal R-tree (STR tree) and orbital bundle tree (TB tree). The former is an R-tree-based access method that also considers the identity of the trajectories in the index. The latter is a hybrid structure that preserves the trajectory and allows for a general range search of the R-tree in the data.

Yon-Gui et al. [38] focused on continuous k-nearest neighbor (CkNN for short) queries using on R-Tree and Quadtree (QR-Tree) that supports continuous k-nearest neighbor queries for moving objects. The idea is to use a QR tree to divide the static space of a moving object. The QRTree and hash table uses as indexes to store the moving objects, calculate the distance between the query point and the moving objects, and get the result. This paper has to calculate the number of objects by using the proposed index, the maximum number of records in each node of the R-tree, and estimate the amount of maximum memory space. Pre-allocate object pools and use

unused objects for recycling. If a new object needs to insert in the allocated space, initialize it and select an unused object to use from the object pool. If the program does not need the object, it returns the object to the object pool. To improve process robustness, in some cases when the object pool is empty, the system can reclaim new dynamic memory space by avoiding the failure of new objects to the application. This avoids memory fragmentation, allows you to store more objects in less memory, and reduces the overall memory requirement for R-tree indexes.

2.6 Summary

This chapter presents the literature review that is related to database systems in mobile computing. Firstly, it starts with location and queries. Besides, location-based query processing, nearest neighbour queries in the mobile environment, continuous k-nearest neighbour search are presented. It also describes indexing techniques that are applied in LBS and wireless broadcasting.

CHAPTER 3
THEORETICAL BACKGROUND OF NEAREST NEIGHBOUR SEARCH AND INDEXING METHODS

This chapter describes the details of the spatial index structures. Moreover, it has explanations of how spatial queries are used in spatial indices.

3.1 Spatial Access Method

Spatial access method is to support the selection of objects based on spatial properties efficiently. For example, a range query selects objects that are within the specified coordinate range. Nearest neighbour query finds the object closest to the specified object. Spatial access methods are also used to efficiently implement spatial analysis such as map overlays and other types of spatial joins. Two characteristics of spatial datasets are that the dataset grows frequently and that the data is very often randomly distributed. Spatial access methods should consider both spatial indexing and clustering techniques. If there are not used a spatial index, it needs to check all the objects in the database and fully scan all tables of a relational database. Spatial datasets are usually very large, so such checks are not acceptable for interactive use or most other applications. If the entire spatial dataset resides in the main memory, it is sufficient to know the address of the requested object, as main memory storage allows random access and does not cause significant delays.

3.2 Nearest Neighbour Search

The k-Nearest neighbour (kNN) query retrieves the top k nearest neighbour based on the user's location. It is also a method to classify objects based on the closest matching entries obtained from training data. Training data has several attributes that represent its characteristics. This is modeled as a many-dimensional space representation. The space is divided into sections based on the classification of training data. The distance between neighbour is commonly calculated using Euclidean distance and Manhattan distance.

In the training phase, the algorithm stores unique features represented as vectors and the class of each data. In the classification phase, the whole distance of training data is calculated against an object to be classified. Once the result is obtained, the distances are then sorted ascending to obtain the closest to the farthest

similarity. Then, it is selected in several k to obtain the k number of stores entries with the closest distances from the sorted data.

The higher value of k will reduce the noise on the classification. It causes the boundaries between each classification to become more blurred. An example of an unknown object class is shown in Figure 3.1. An object is about to be classified using kNN algorithm. If we set k = 3, the object is classified into triangular class because 2 of 3 objects in the k=3 boundary are triangles. However, if the number of k is increased to be k = 5, the objects are classified into class star shape because there are more stars than a triangle in the k=5 boundary.

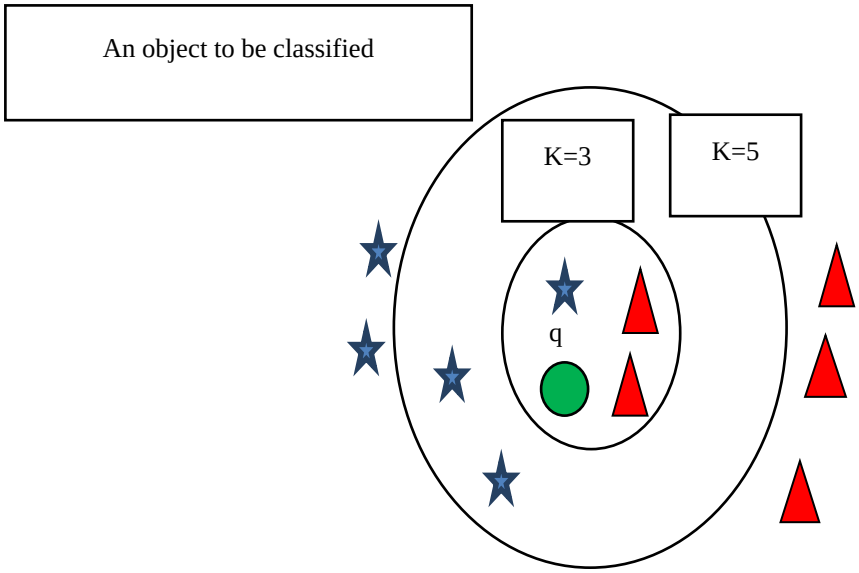


Figure 3.1 Illustration of k-Nearest Neighbours

Figure 3.2 shows the classification process using the k-nearest neighbour to be used in the point of the interest classification system. There, points of interest locations are separated into classes defined by systems at the number of k. The input of the classification process is the value of x, y, and the context of the task as a result of data obtained from the GPS. This process is called data testing. The classification process begins by calculating the distance between (x, y) with each data training (xt, yt) using Haversine formula.

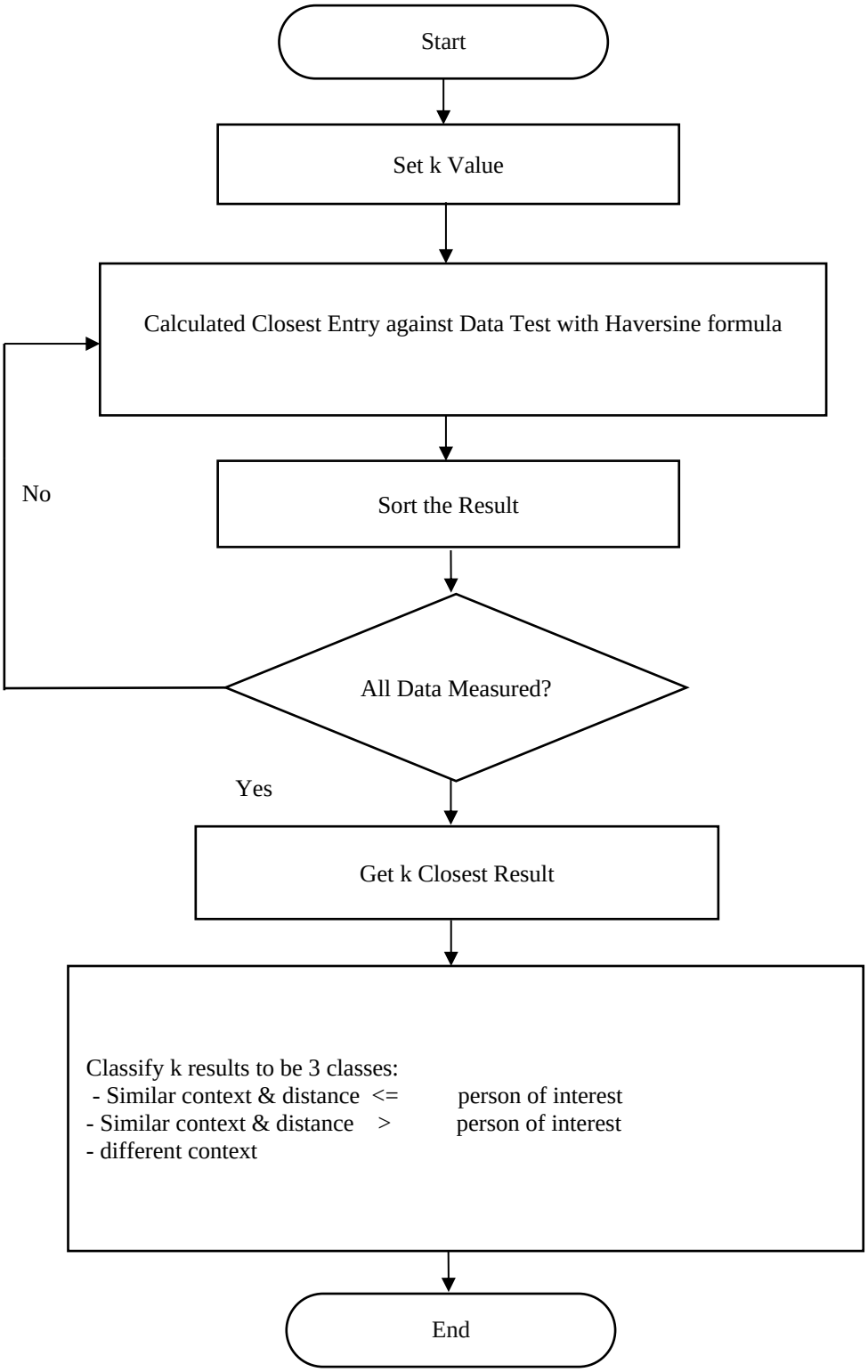


Figure 3.2 Workflow of k-Nearest Neighbours Algorithm

Several data training k with the nearest distance with data testing will be stored. Recall that there will be two classifications results, namely (main recommendations, and not recommended) for each data training, the determination of the point of interest is based on the two classifications results. An example of the

result of the k-nearest neighbour classification using k=3 would result in three category names, are: 1= main recommendations, 2=main recommendations location, 3= not recommended. By considering the result, the users are suggested to choose the first two results [39].

3.3 Indexing Techniques

Indexing techniques support improving the efficiency of queries in the data processing. This is a very important aspect of relational databases, and it is also important for spatial database extended query syntax and data. During the past years, indexing approaches for spatial and spatiotemporal queries have been one attractive and unique field for research groups. Figure 3.3 shows the spatial access method.

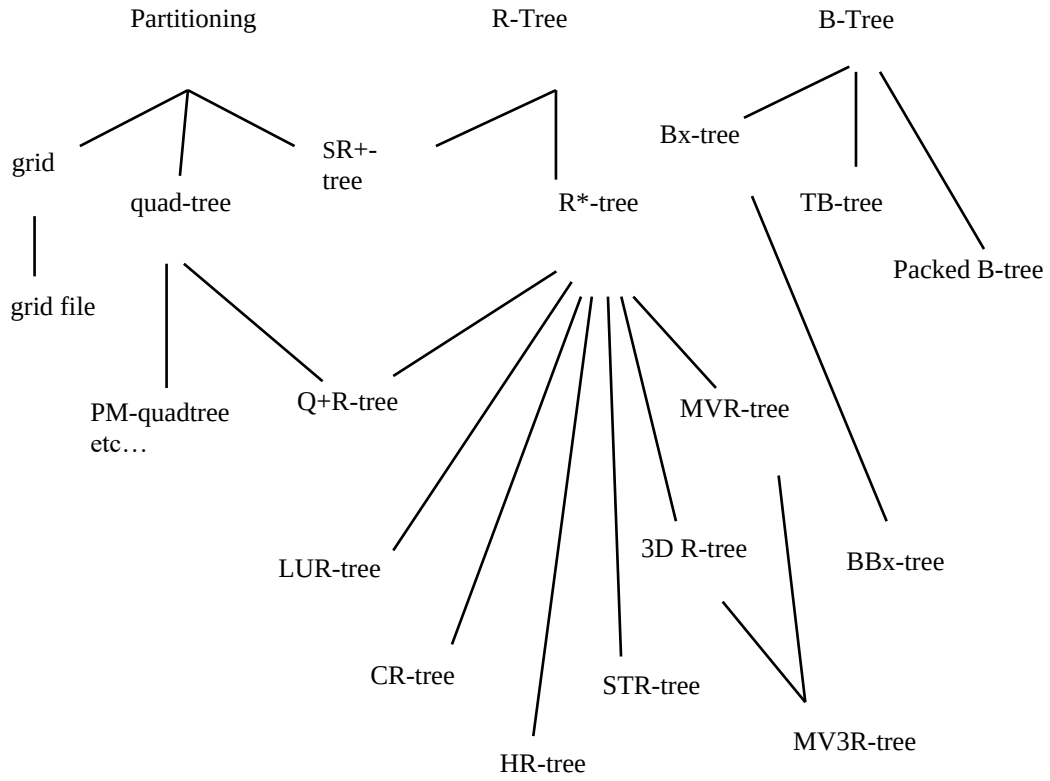


Figure 3.3 Spatial Access Method

3.3.1 B-Tree

B-tree is a data structure for storing data that has a myriad of insert and delete execution times. The B-tree structure is shown in Figure 3.4. B-trees are a special type of tree whose properties help store and retrieve information.

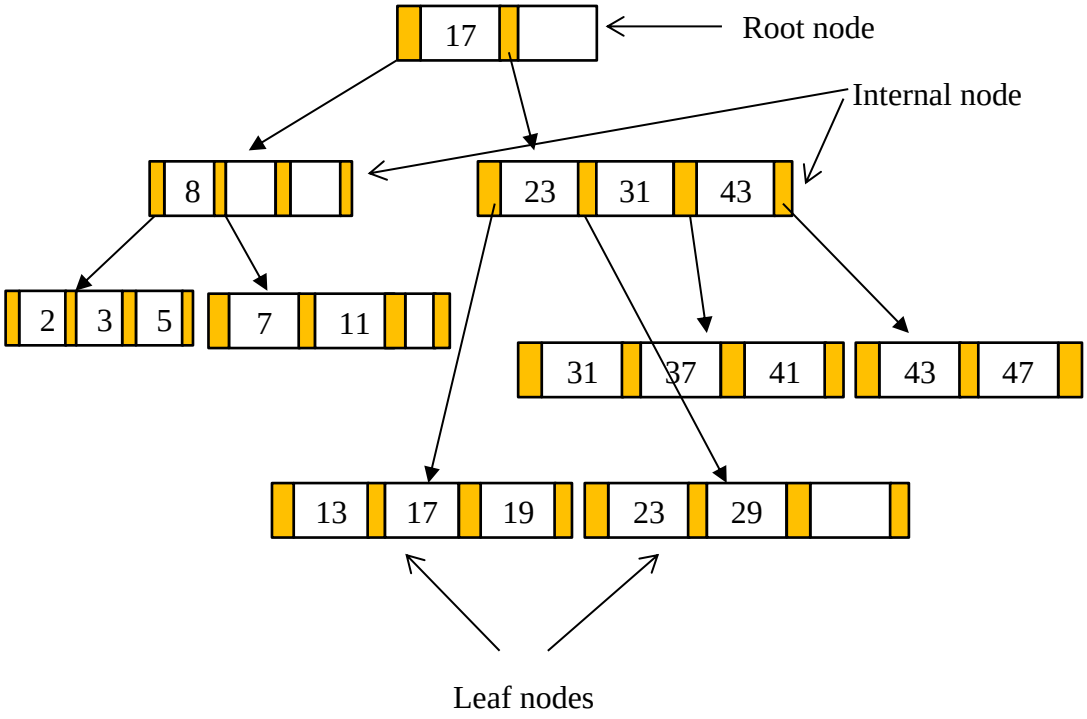


Figure 3.4 B-tree Structure

B-tree is the multi-level index that is organized as a balanced tree structure. The structure of B-tree includes the internal nodes (including root node) and leaf nodes. Each node has search keys and pointers. Each internal node occupies one disk block in the database. Search keys in a node have the search values that can use to speed up the search operation. The pointers in the node define the database address that points to another node. The root node is the top level of the tree structure and internal nodes may be one or more child nodes. The leaf node is located at the same level as the root node [40]. The search process of B-tree is shown in Figure 3.5. It needs to use the search key to search for the record. The index file is stored in the leaf nodes. Figure 3.6 describes how to insert a new record into the allocate node. It needs to search allocate node that contains the search key and object (record) pointer of the leaf node in B-tree.

```
BtreeSearch(x, k)
 i = 1
 while i ≤ n[x] and k ≥ keyi[x]
   do i = i + 1
if i  n[x] and k = keyi[x]
   then return (x, i)
if leaf [x]
   then return NIL
else
   return BtreeSearch(ci[x], k)
```

Figure 3.5 Search Algorithm in B-tree

```
Insert(T,k)

 r ← root[T]

 if
        n[r] = 2t - 1

    then s ← ALLOCATE-NODE()

        root[T] ←  s

        leaf[s] ← FALSE

        n[s] ← 0
        c1[s] ←  r

       B-TREE-SPLIT-CHILD(s,1,r)

       B-TREE-INSERT-NONFULL(s,k)
 else B-TREE-INSERT-NONFULL(r,k)
```

Figure 3.6 Insert Algorithm in B-tree

The structure of an internal node (including the root node) has one more pointer than search keys in a node as shown in Figure 3.7. If there are *k* search keys then there are *k+1* pointer. There must be used at least half of the pointer in the internal node. The root note must have at least 2 pointers and one search key. The number of pointers in each node must be counted. The structure of the leaf node also

has search keys and location pointers. The index files are stored in the leaf nodes in the corresponding disk block. The search values of each search operate to make speed up the search operations. The last pointer in the leaf node will point to the next leaf node of the tree that is shown in Figure 3.8.

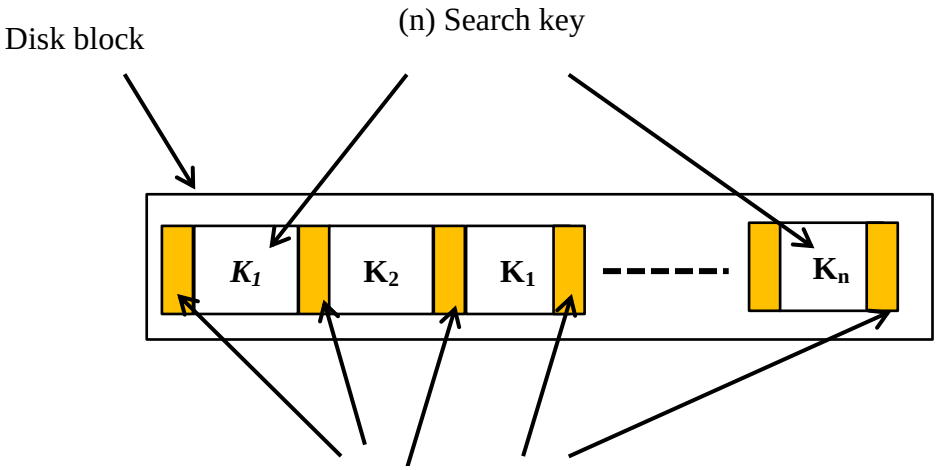


Figure 3.7 Internal Node in B-tree Structure

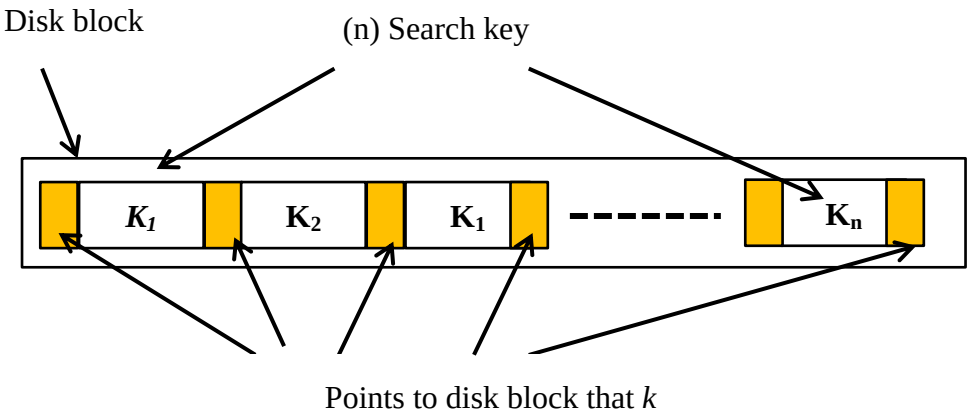


Figure 3.8 Leaf Nodes in the B-tree Structure

3.3.2 KD-Tree

The KD-tree is a tree structure that is the generalization of the binary search tree for multiple dimensions. The KD-tree is a well-known space partitioning data structure for storing points of multi-dimensional space. The two-dimensional points are stored in the internal nodes. Considering the example, the set of points is on the x coordinate and y coordinate of the 2D plane. Each node in the KD-tree contains a data point and at most two children. Each level in KD-tree has a cutting dimension. Figure 3.9 shows an example of a two-dimensional KD-tree.

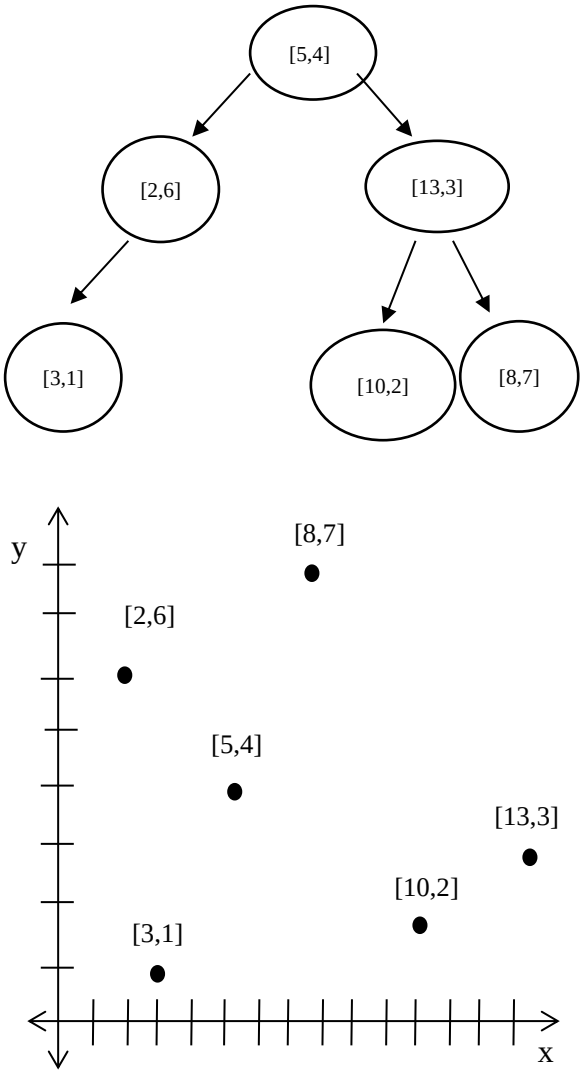


Figure 3.9 Example of KD-Tree Structure

The space is partitioned horizontally or vertically. It splits the point set alternatingly by x-coordinate and by y-coordinate. If it splits by x-coordinate, it splits point set by a vertical line that has half the points left and half right. Otherwise, if it

splits by y-coordinate, it also splits points by a horizontal line that has half the points below and half above. Each internal node stores the spitting node along x (or y).

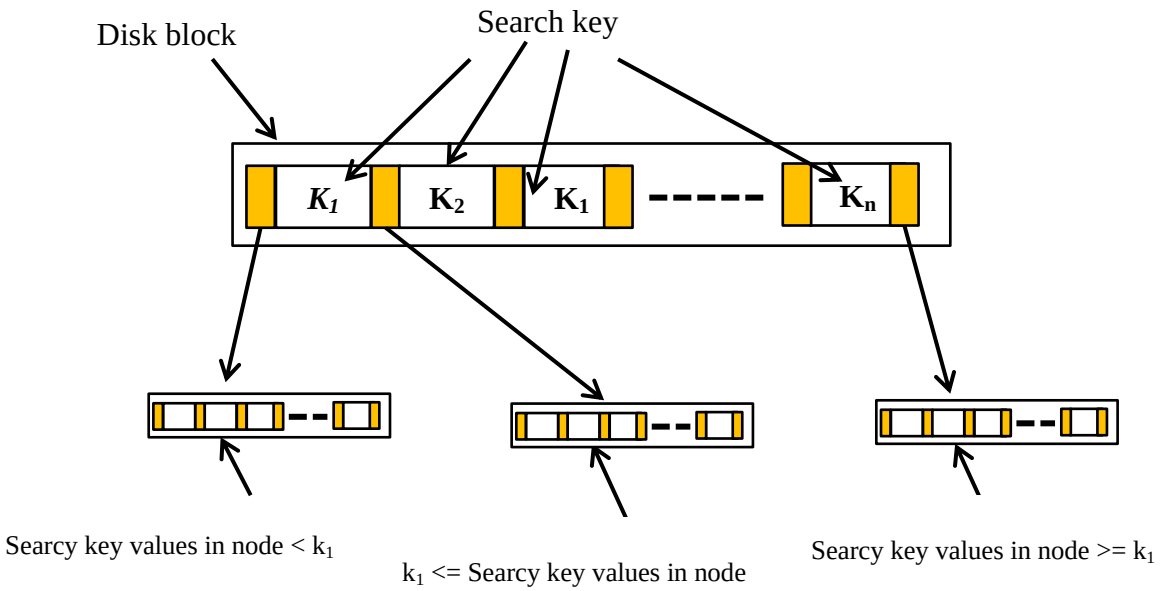


Figure 3.10 Diagram of KD-Tree Structure

The KD-tree can be used to speed up k-nearest neighbor queries using the ball-rectangle intersection test. Given the query points p and k-nearest neighbors, the true k-nearest neighbour is inside the ball entered on p and passing through the current k-th closest candidate as shown in Figure 3.10. When searching for better candidates, nodes that do not intersect this ball can be skipped without considering their children [40]. The search process in KD-tree is to find the located objects which are partially within a given distance. The root node is the initial step for searching in KD-tree and then it traverses the internal node until to get the leaf of the actual record. Figure 3.11 shows the step by step searching algorithm in KD-tree. In the insertion process, a search key is required to store the record in the space. The search key finds the allocation space that is each dimension of the internal node. Then the key get the space at the leaf node to insert the record as shown in Figure 3.12.

```
Record search( x )
  {       n = root node;
    while ( n ≠ leaf node )
    {
      n is indexed using component i;
      if ( x.compi < n.key )
           n = n.left;
        else
           n = n.right;
    }
    if ( x ∈ n (leaf node) )
    {
      return Record for x
    }
    else
    {
       return No Found;
    }
  }
```

Figure 3.11 Search Algorithm in KD-tree

```
insert ( Record x )
  {
    Use the search key in x to find:
      the leaf node L that will hold x

    if ( L has space store store x )
    {
      insert record x in (block) L;
    }
    else
    {
      Let i = search dimension in parent node of L

      Let j = the next dimension after i
      K = the median of search key values in dimension j;
         Move all records with key <  K into block L1;
         Move all records with key >= K into block L2;

         Replace L by an internal node with key K;
      Set:
              K.left  = L1;
              K.right = L2;
    }
  }
```

Figure 3.12 Insert Algorithm in KD-tree

3.3.3 D-tree

The D-tree is a binary tree that divides regions with polylines to perform accessing the data structure. A set of data areas has two main subspaces which consist of the same number of regions. Each subspace is partitioned by polylines. The partition of each subspace is defined by vertical dimensional and horizontal dimensional. Each internal node in D-tree may have a partition that divides two subspaces. A leaf node also has a partition that has two data regions. The two complementary sub-spaces in the internal node have the pointer that stores the address of the data regions. The regions exists their representative subspace [41]. The example of the D-tree structure is shown in Figure 3.13. In table 3.1, the attributes of the node are shown.

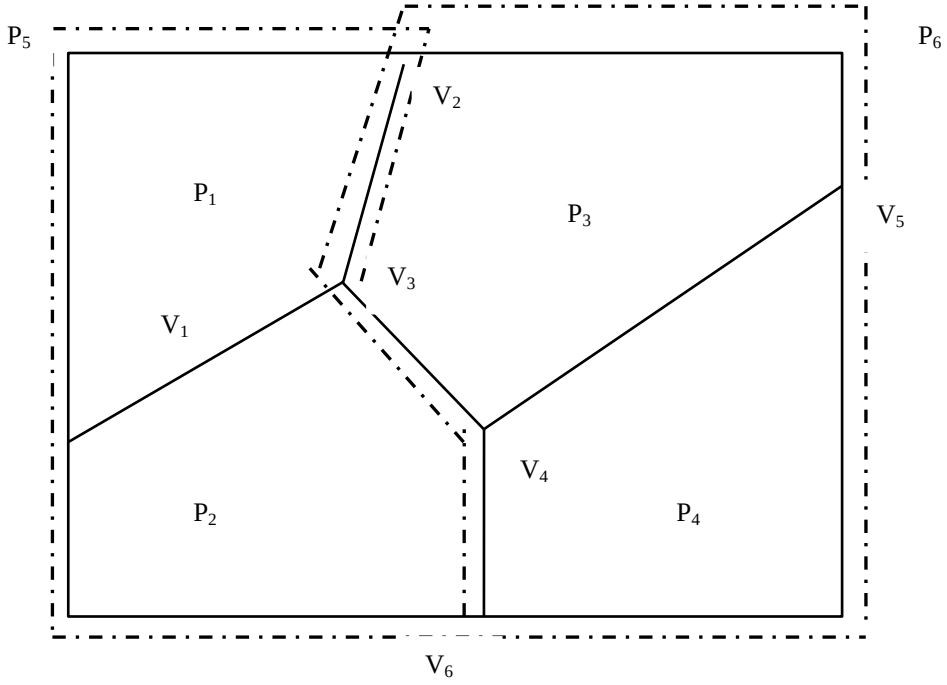


Figure 3.13 Example of D-tree Structure

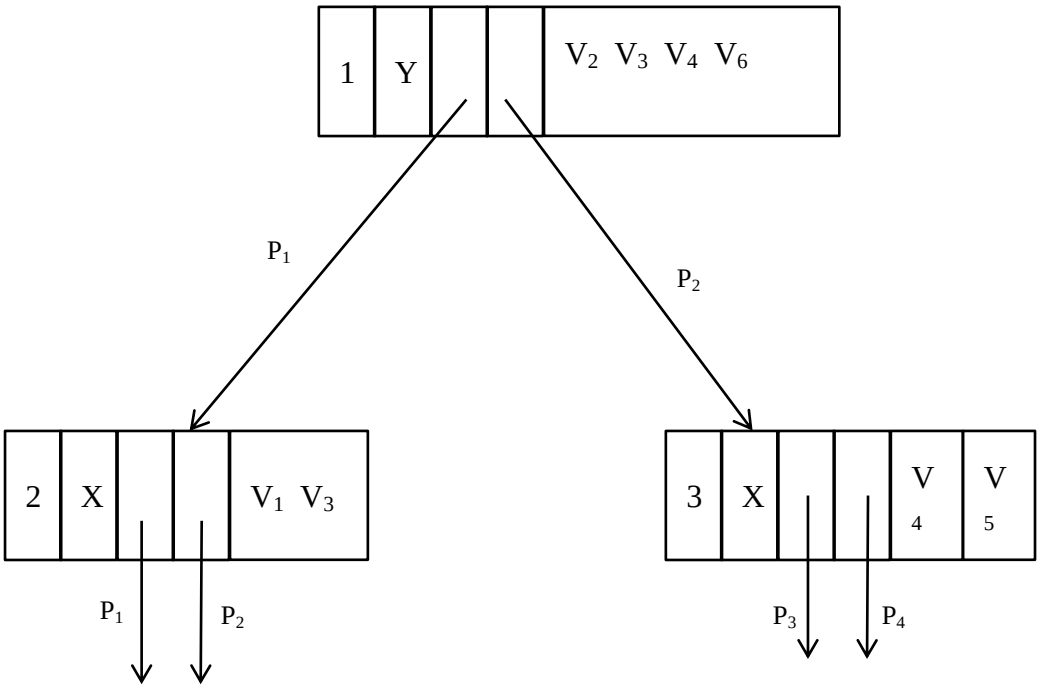


Figure 3.14 Example of D-tree Structure

bid	header	left-ptr	right-ptr	Partition

Figure 3.15 D-tree Node (Index Bucket)

Table 3.1 Description of the Attributes in Index Bucket

Attribute	Description
bid	Unit id of the tree structure
header	Includes a flag that indicates the partition style and size, whether the bucket is occupying multiple packets
left-ptr	Type (data or node pointer) and offset to the beginning of the left child
right-ptr	Type (data or node pointer) and offset to the beginning of the right child
partition	A set of coordinates representing the left and right child partitions

The internal node contains a partition that divides the current space into two complementary subspaces. The pointer on the left points to the node contains the data area in the subspace on the left, and the pointer on the right points to the node contains a data area in the right subspace and some control parameters including bids and headers [42].

A leaf node contains partitions for two data areas, including pointers to the data buckets that correspond to the areas, and control parameters. Therefore, the spatial data area is inferred by the partition as it follows the path from the root to the leaf node. The binary D tree meets four properties:

1. Every node has at least two children.

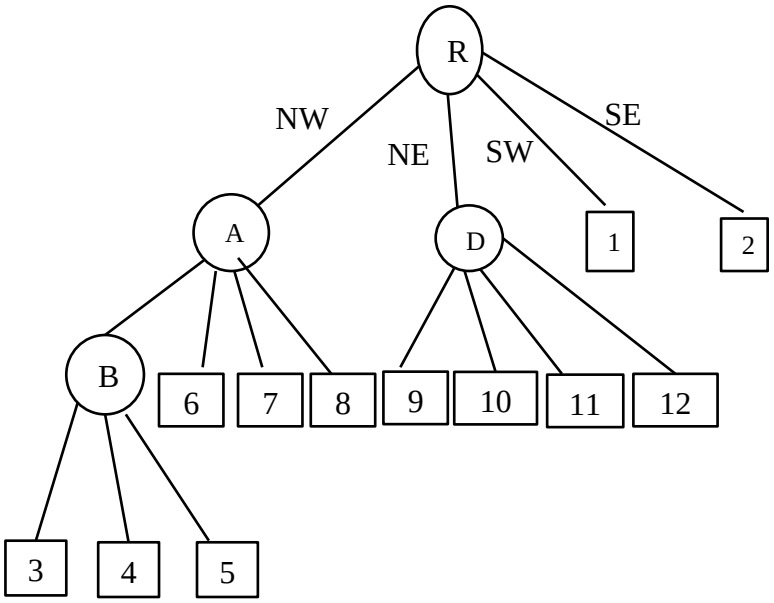
2. All objects in the left subtree of the node are in the left subspace of the partition, and all objects in the right subtree are in the right subspace.

3. The height of the tree is balanced. That is, all leaves are displayed at about the same level, up to one level different.

4. The search time for point queries is O (log N) for the visited node. Where N is the number of data areas in the original space.

3.3.4 Quadtree

Quadtree is an index structure that divides a search space in every dimension. A quadtree node contains search key values for each dimension and child node pointers for each node. Except for the root node, there must have one parent node pointer. The child node pointer will point to every possible relationship with the search key values. In figure 3.16, the quadtree will divide the search space as the following quadrants subdivision form.



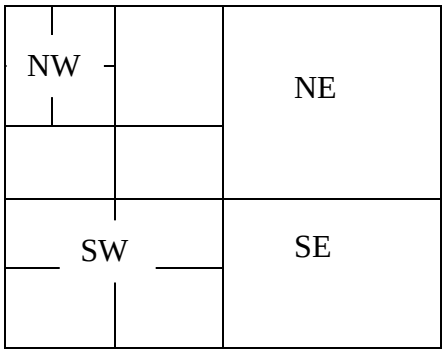


Figure 3.16 Region Quadtree

The children of the root node represents as rectangles which are commonly labelled as the northwest (NW), northeast (NE), southwest (SW), and southeast (SE) quadrants. Quadtrees are not always balanced. Subtrees that correspond to densely populated areas may be deeper than other areas. Searching in the quadtree is similar to searching in the regular binary search tree [40]. To find the record at each internal node, select the subtree based on the relationship of the search key and the key value stored in the internal node. Figure 3.17 gives finding the record using the search key of a quadtree that is stored in the internal node. For point queries, only one subtree is eligible, but for range queries, there are often multiple subtrees. This search step recursively repeats until it reaches the leaves of the tree as shown in Figure 3.18.

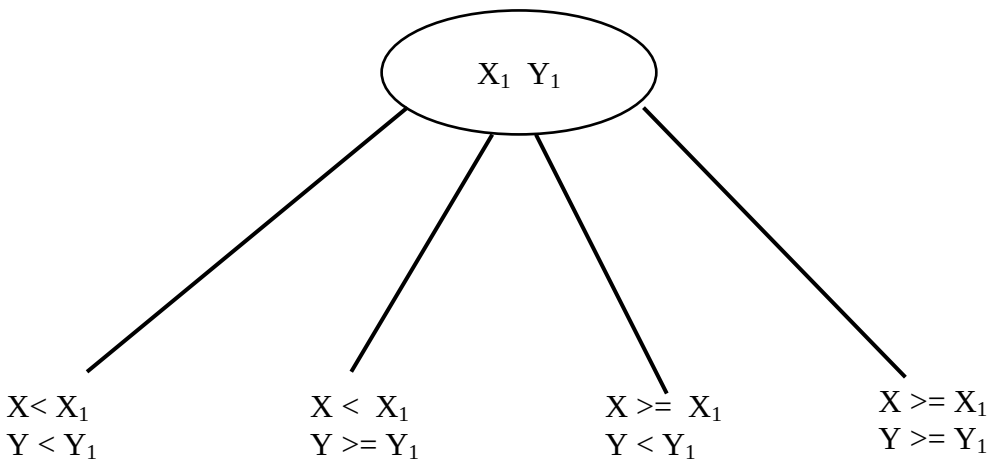


Figure 3.17 Search Records of Quadtree

If no point is found in the tree, it will be inserted into the leaf node where the search is finished. Figure 3.18 shows the insertion process of the Quadtree algorithm how to insert the data step by step. The corresponding partition is divided into 2D subspaces cantered on the new point. To remove the point, there is needed to rebuild the subtree under the corresponding Quadtree node. Another variant is the area quadtree. It is based on the normal decomposition of the area. That is, the 2D subspaces that result from partitions are always the same size. This makes searching much easier.

```
Insert( x1, x2, ..., xn )
  {
    Find the leaf node L;

    if ( leaf node L has enough space )
    {
      insert (x1, x2, ..., xn) in L;

          return;
    }
    else
    {
      Let m1  = the middle of the x1 range of values in L;
      Let m2  = the middle of the x2 range of values in L;

      Let mn  = the middle of the xn range of values in L;


      Split the records in L into 2n blocks according m1, m2, ..., mn

          Replace L by an internal node I with seacrh key (m1, m2, ..., mn);

          Make the 2n blocks as child nodes of I;
    }
  }
```

Figure 3.18 Insert Algorithm in Quadtree

```
Search(Point p)
{

   if (!inBoundary(p))
      return NULL;

   if (n != NULL)
      return n;

   if ((topLeft.x + botRight.x) / 2 >= p.x)
   {
      if ((topLeft.y + botRight.y) / 2 >= p.y)
      {
         if (topLeftTree == NULL)
            return NULL;
         return topLeftTree->search(p);
      }

      else
      {
         if (botLeftTree == NULL)
            return NULL;
         return botLeftTree->search(p);
      }
   }
   else
   {
      if ((topLeft.y + botRight.y) / 2 >= p.y)
      {
         if (topRightTree == NULL)
            return NULL;
         return topRightTree->search(p);
      }
        else
      {
         if (botRightTree == NULL)
            return NULL;
         return botRightTree->search(p);
      }
   }
};
 bool Quad::inBoundary(Point p)
{
   return (p.x >= topLeft.x &&   p.x <= botRight.x &&
           p.y >= topLeft.y &&         p.y <= botRight.y);
}
```

Figure 3.19 Search Algorithm in Quadtree

3.3.5 R-tree

The R-trees indexing method was presented by Guttman [43]. R-Tree is an index structure that uses a bounding box (BB) as search keys. The tree is height-balanced. R-trees are very similar to B-trees. The following Figure 3.20 illustrates the workflow of R-tree that is this situation in a simple two-dimensional structure. All objects in R-tree lie inside BB. BB is a rectangle that contains a group of objects. Each object has a unique identifier which is comprised by the minimum bounding rectangle (MBR). The MBR is the smallest rectangle that contains a group of objects as shown in Figure 3.21. The structure of an internal node (including root node) of the R-tree is represented with BB and child node pointer. The leaf nodes contain the MBRs of the referenced all objects and their pointer.

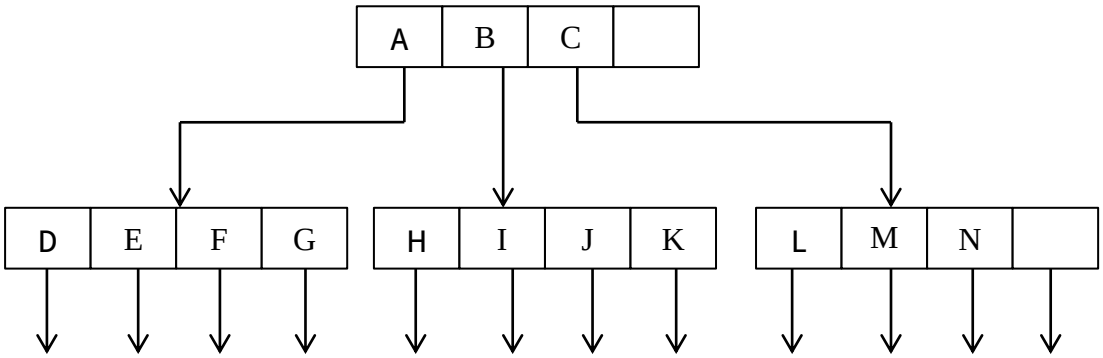


Figure 3.20 Workflow of R-tree Structure

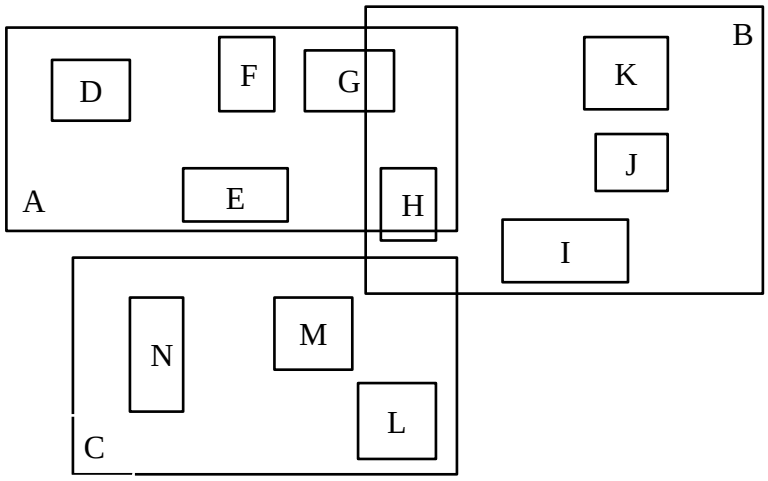


Figure 3.21 R-tree Structure

R-tree has the following properties:

1. Every node contains between “m” and “M” entries.
2. Every non-leaf node has between “m” and “M” children unless it is the root. For each entry) pointer - child, (“I” in a non-leaf node, “I” is the smallest rectangle that spatially contains the rectangles in the child node.
3. The root node has at least two entries unless it is a leaf.
4. All leaves appear on the same level [43].

The structure of a leaf node in R-tree contains index record entries of the form (*I,tuple-identifier*) where *I* is an n-dimensional rectangle which is the bounding box of the spatial object indexed and *tuple-identifier* is the object that lies in the data block of the database[48]. Note that $I = (I_0, I_{1,\ldots\ldots,} I_{n-1})$ and I_i is the closed bounded interval [1,b] describing the extent of the object along dimension *i*. The figure 3.22 shows *I* as BB which contains two objects.

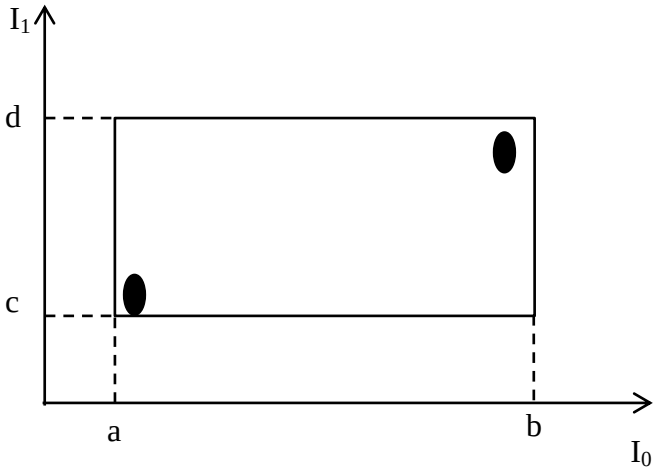


Figure 3.22 Representation of *I* with *n = 2*

The structure of the non-leaf node contains entries of the form (*I,child-identifier*) where child-pointer is the address of a lower level node in R-tree and *I* covers all rectangles in the child node’s entries. Figure 3.23 shows a rectangle that covers the child node’s entries D, E, F, G.

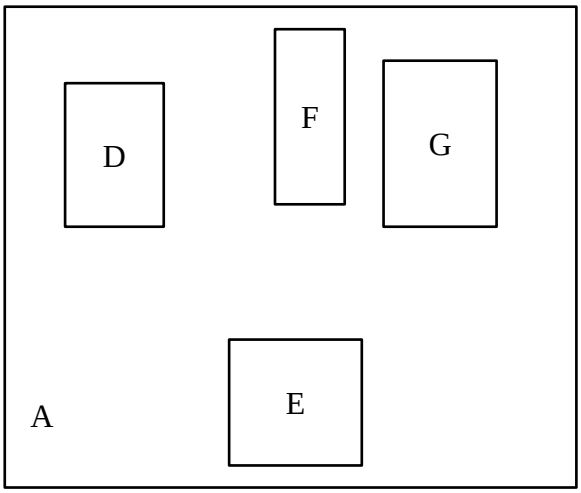


Figure 3.23 A rectangle Covers Child Node's Entries D,E,F,G.

R-tree considers partitioning node to avoid the overflow condition in the search process. It must be minimized the total areas of the bounding boxes of the split and must be minimized the overlapping areas between bounding boxes. Figure 3.24 shows a bad split because the coverage area of the node will increase the search space. Figure 3.25 shows a superior split because the area of the BB will decrease the search space.

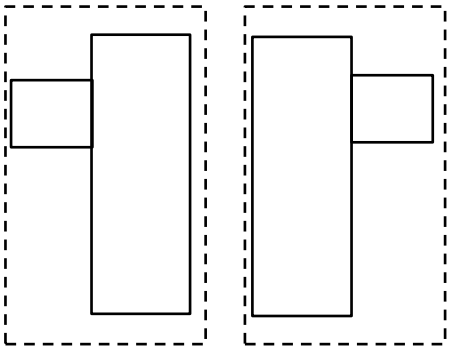


Figure 3.24 Bad Split

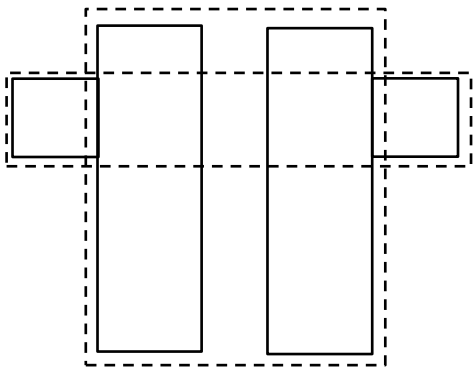


Figure 3.25 Good Split

3.3.6 Voronoi Diagram

The Voronoi diagram is one of the basic structures defined by the grid. It is applied in geometric structures and many other fields. It is defined as a line that bisects the line between the center point and the points around it.

The bisector and the connecting line are perpendicular to each other. With this rule, the area is completely covered by adjacent polygons. It is also defined as a collection of geometric objects that divide a plane into cells. Each cell consists of points that are closer to one particular object than the other objects [40].

Another definition of a VD for a set of sites (points) is a collection of areas that divide the plane, where each area corresponds to one of the sites and every point in one area corresponds to more than any site. The boundary between two adjacent areas is a line segment that contains a line segment and a vertical bisector of the line segment connecting the two sites. This is considered an important characteristic of VD. If three sites determine the Voronoi region where they meet at a Voronoi point, the circle passing through those sites is centered on that point and there are no other sites within the circle. Figure 3.26 shows VD built for a set of points.

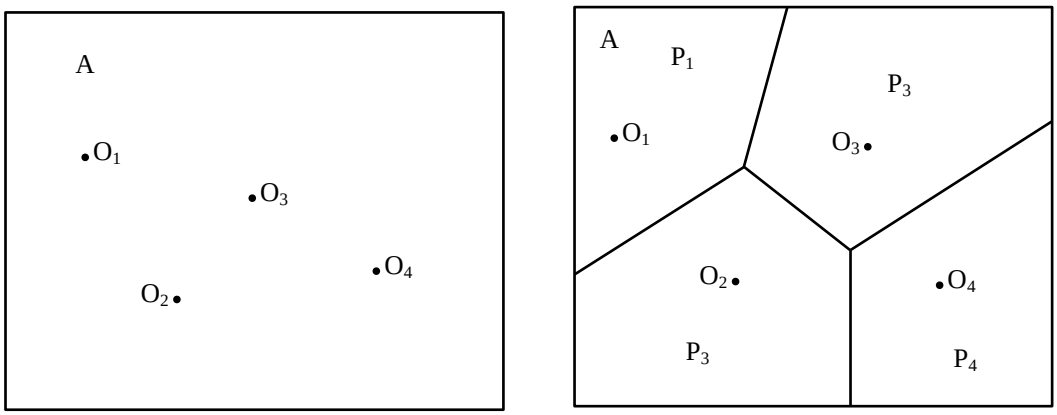


Figure 3.26 Voronoi Diagram

3.3.7 Grid Index

Grid index file is an index that is organized into a two-dimensional structure. The spatial objects which are geographically related data are stored in the same data block. GIS usually employs a grid index structure in the implementation process. Grid index is a table-based access method in a multi-dimensional index structure. Firstly, the grid index file stores the size parameters *m* and *n* of the grid line, and then the index file stores the bucket of the grid. Note that $v=(v_{1,} v_{2,---,} v_m)$ is the key-value *m*

parameter and $x = (x_1, x_2, \ldots, x_m)$ is the the key value of v parameter of the grid. Finally, the index file contains m and n block pointers. Figure 3.27 shows the example of the grid index structure.

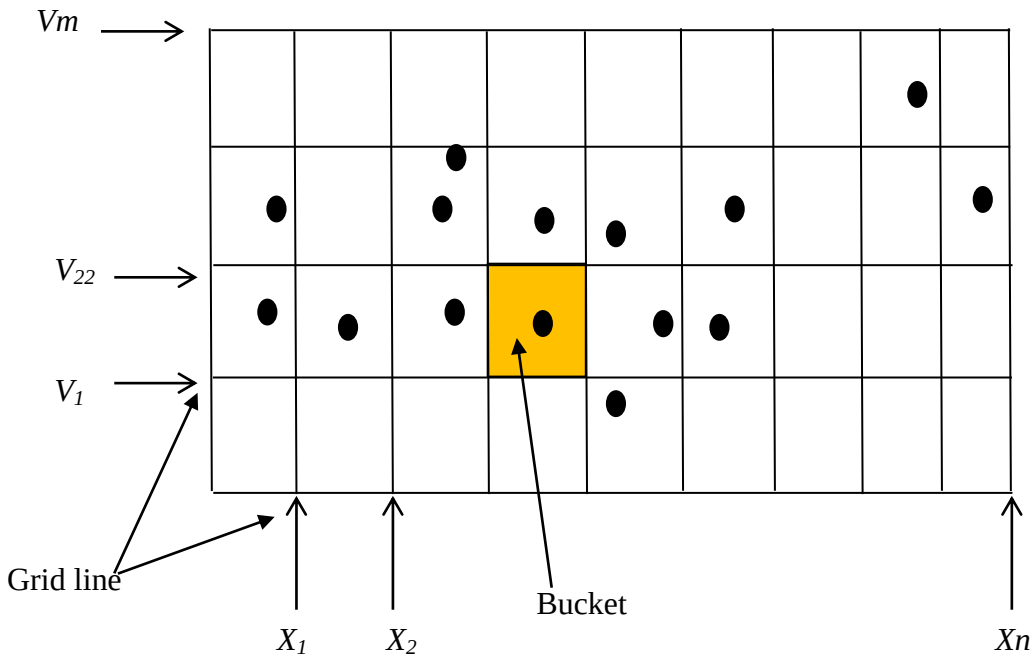


Figure 3.27 Example of Grid Index

Commonly used grids in GIS are regular grids and can be applied to both points and face objects. When a user performs a spatial query, it calculates the grid in which the first queried object resides and uses it to quickly query the selected geographic object. Grids often inherit code into the grid according to Morton code to establish a linear form of spatial index, establishing relationships between Morton code and spatial objects. Therefore, indexing algorithms are simple and easy for computer programming. It can be assigned directly to the corresponding spatial entity via the bucket number, which speeds up indexing. From the above analysis, it can be seen that the idea of the grid is simple and easy to understand and implement [45]. Figure 3.28 shows a new record insert algorithm in the grid index structure.

```
insert( recordSearchKey, recordPtr )
    {
      B = Lookup( recordSearchKey );

         Let b = the last bucket block in the chain B;

         if ( b has room for record )
         {
           Insert recordSearchKey, recordPtr in block b;
      }
         else
         {
           Allocate an overflow block for bucket;

           Link overflow block to b

           Insert recordSearchKey, recordPtr in overflow bucket block;
      }
    }
```

Figure 3.28 Insertion Algorithm in Grid Index

3.4 Summary

This chapter describes the spatial database system and indexing approach. It also represents application areas, mobile database query processing, and location categories. This chapter also mentions the spatial access method and the k-nearest neighbour method in general. Moreover, many indexing approaches used to improve the efficiency of queries to data processing are shown.

CHAPTER 4
SYSTEM DESIGN AND IMPLEMENTATION OF THE PROPOSED SYSTEM

In this chapter, the proposed system is explained in detail. This system offers nearest neighbor results to the user based on location position quickly by using R-tree based grid indexing technique. The proposed system develops the client-server model. The implementation of the proposed system used Google API for the map while developing for both server application and client application. Google Maps API features are supported to perform geo-related computations such as geocoding and also provide places and routes.

4.1 System Framework of The Proposed System

Location-based services (LBS) are services that combine location information (with coordinates latitude and longitude) with textual description to give valuable information for mobile users. The location-based applications types are emergency services, car navigation systems, tourist guide planning, or information delivery. The basic requirements of LBS are efficient computing and are easy-to-use friendly. It is necessary to work on LBS functionalities which are mobile device, communication network, positioning system, and service provider. Location-based services (LBS) are a general concept of services that denote geographic location (that is, spatial coordinates). The system framework of the system is shown in Figure 4.1 which includes the server components and the geolocation content database.

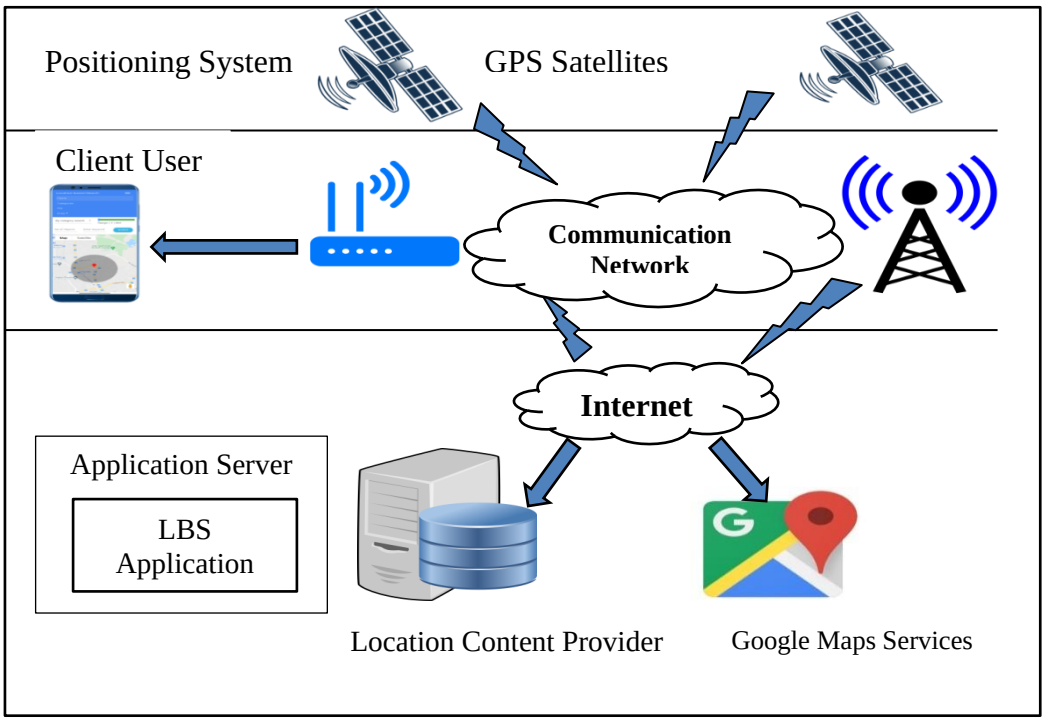


Figure 4.1 System Framework of the Proposed System

4.2 Proposed System Architecture of Research Work

This section shows the system architecture of the proposed system. Figure 4.2 shows the system architecture. The system has two main parts. The first one is the proposed methods R-tree with based grid index structure. Data with capable geolocation are stored in the spatial database. While storing the data, the proposed methods are calculated to be correct and efficiently inserting and retrieving data. This first part is implemented for the server-side with the location content provider. The second one is query processing of query processors from the server system when clients request the geo-location objects. The server will respond to clients according to the proposed methods. The client is responsible for sending the user's request and the geographic location of the mobile device to the server, and the server is responsible for providing the service based on the geographic location of the mobile device user. The server puts the collected information from the fields into the database and serves all clients based on the spatial database.

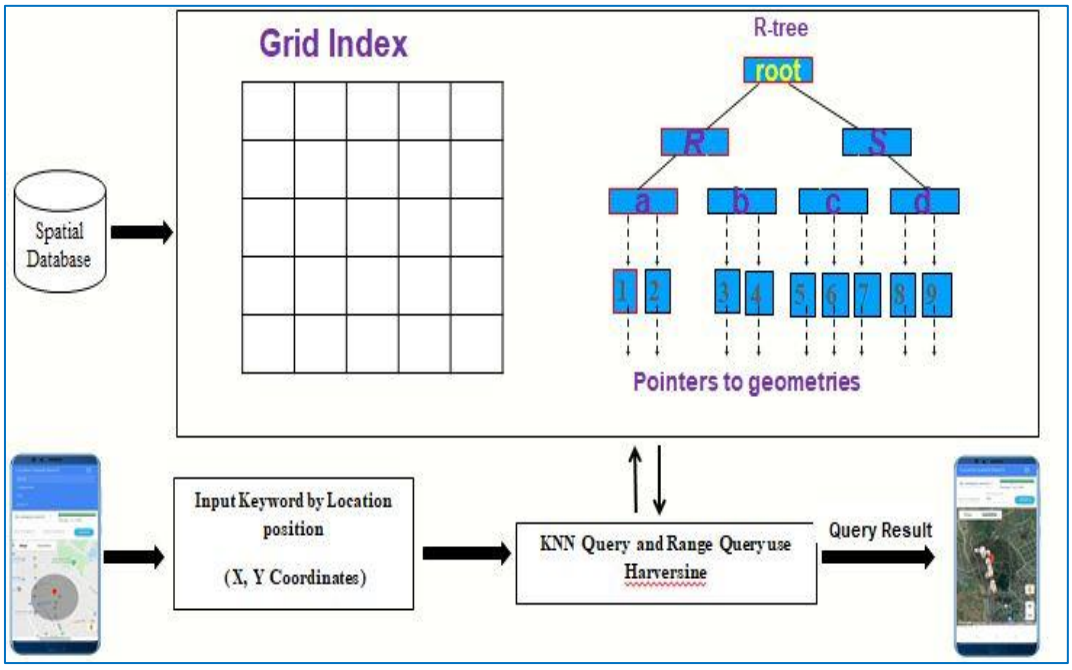


Figure 4.2 System Architecture

4.2.1 Flowchart of The Proposed System

Firstly, objects with geo-location (Lat/Long) are stored in the spatial database by constructing the proposed method to respond to the requested devices effectively and efficiently. The mobile device users need to get the specific position or current position. Mobile users input keywords to search objects or users can also choose the category. The next step is to input range (km) or the number of objects that need to be input by the mobile users to show nearby objects.

The proposed system starts performing structure of spatial index methods which use grid-based R tree method. Then proposed indexing method is computed based on the mobile user's specific location and retrieves the user preference services and gets k-nearest neighbour objects by calculating distance. Thus nearest indices around the index of current location can be acquired. The grid index extracts only locations from the nearest indices and sends these indices to R-tree. Therefore, the usage of large memory space in the R-tree can be reduced.

In R-tree, the nearest indices that are sent by grid index are constructed as a tree. R-tree is composed of the root node, intermediate node, and leaf node. R-tree sort active branch list (ABL) by ordering MINDIST. ABL is a list that calculates the distance of objects. kNN works by finding the distance between the query and all the objects in the spatial database, selecting the specified number of examples (k) that are closest to the query, and (k) displaying the output of the object. The flowchart of the proposed system is shown in Figure 4.3.

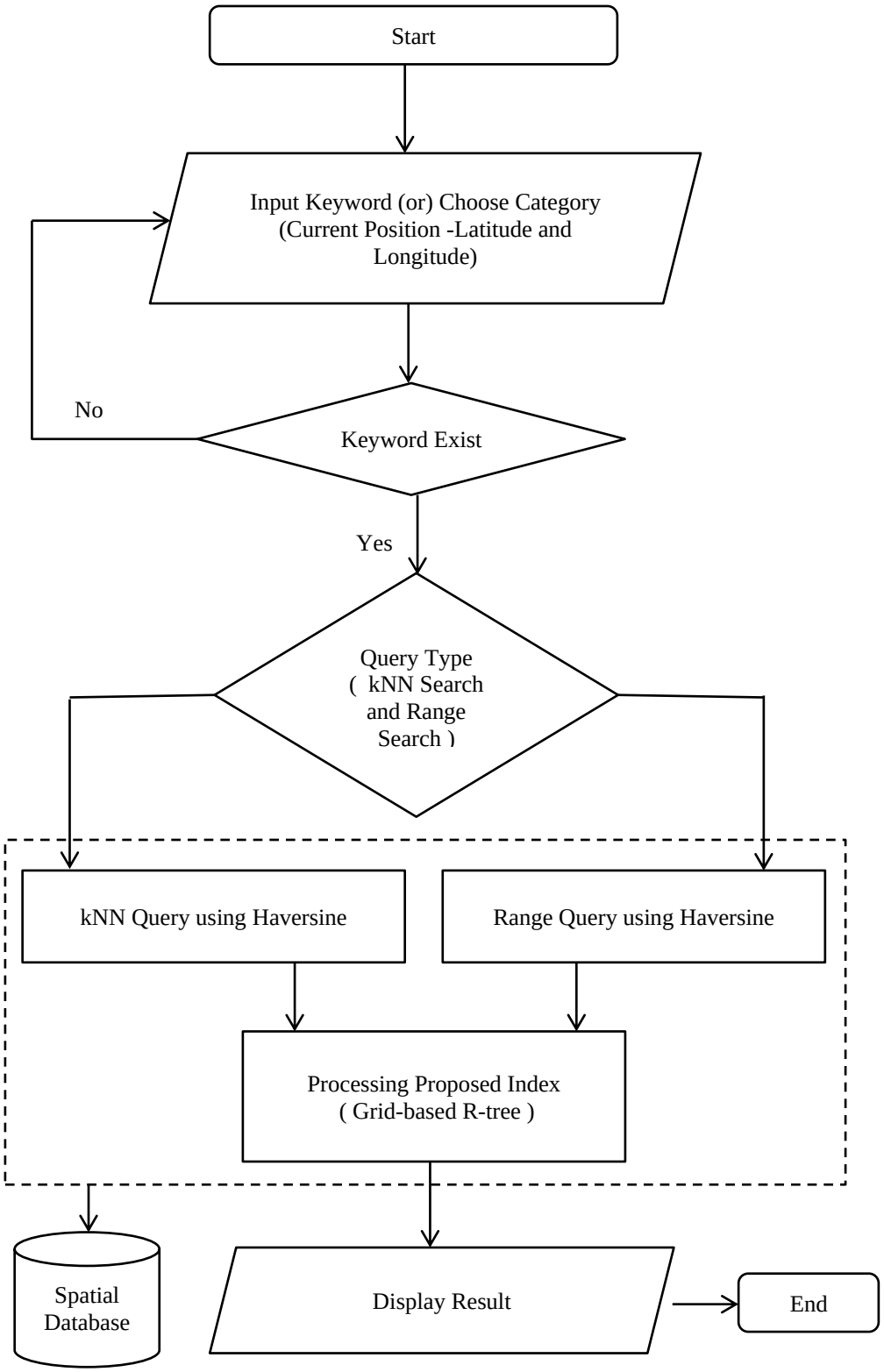


Figure 4.3 Flowchart of the Proposed System Detail Information

4.2.2 Computing Structure of Grid Index

The grid index is an index that is organized into a two-dimensional structure. The grid index structure is stored geographically related data in the same data block. Firstly, a grid index file stores the size parameters *m* and *n* of the grid. Then the index file stores the bucket of the grid. Finally, the index file contains *m* and *n* block pointers.

The grid index is composed of grid cells. Each cell represents an area of space created by dividing the domain using a uniform grid, which can be assigned unique identifiers and used for spatial indexing purposes. It uses coordinates of objects and sorts them into grids, where grids have their identifier index for faster querying. This is a very simple and efficient way of spatial indexing. Grid-based spatial indexes have the advantages that the proposed system can create the index structure first and then add data continuously without changing the index structure. If different data collection and indexing activities use a common grid, such indexes can be easily merged from different sources. Figure 4.4 shows the nearest indices of the current location in the grid index.

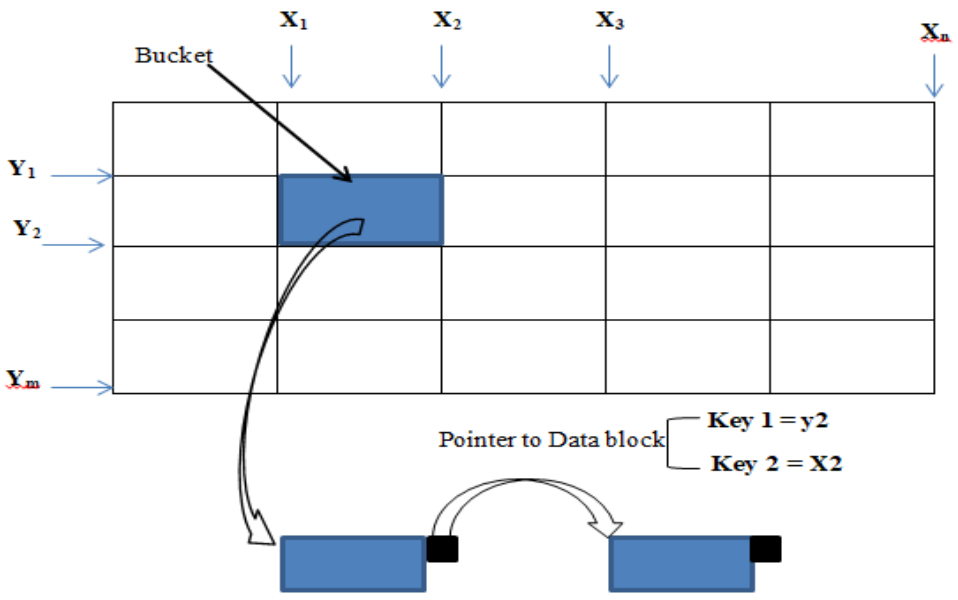


Figure 4.4 Grid Index Structure

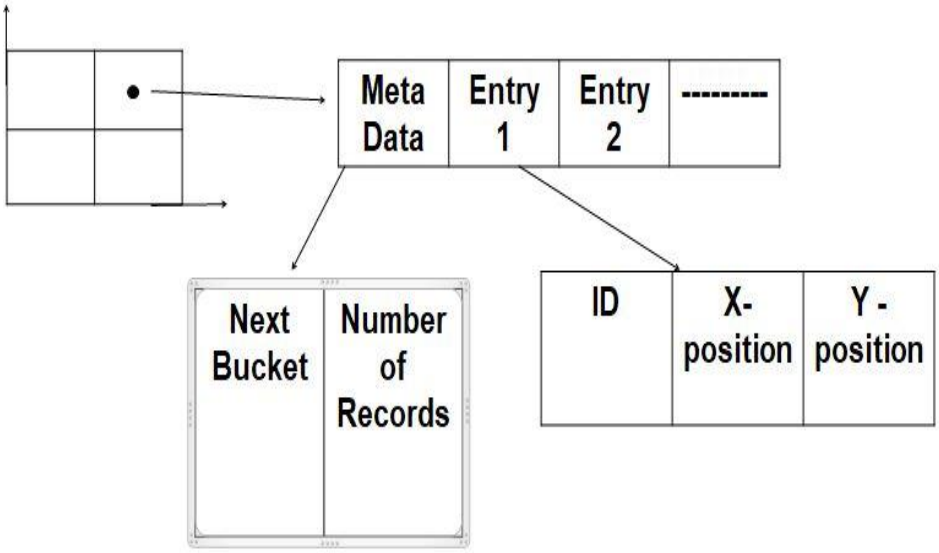


Figure 4.5 Two Dimensional Grid Index Structure

The grid index file is organized in a two-dimensional structure. The geographically related data are stored in the same data block shown in Figure 4.5. The grid index cannot represent objects, it can only present points. The only kind of index that can handle where I am queries: is given by a location (i.e., coordinate), and then find the objects that contain the location. Therefore, the performance of grid index for range queries is to find objects that are located within a certain range and such performance of grid index for nearest neighbour queries is to retrieve the nearest neighbour of a data point. Figure 4.6 shows the computing grid-index algorithm before constructing the R-tree structure.

```
Input latitude, longitude
Output grididx
Begin
        Latlng ygntl = (17.136160, 96.009705);
        LatLng ygnbr = (16.7851054, 96.1791551000);
        latdiff = (ygntl.latitude- latitude);
        londiff = (ytntl.longitude – longitude);
        w = (latdiff / factor);
        h = (londiff / factor);
        grididx = (h * maxwidth ) + w;
End
```

Figure 4.6 Grid Index Algorithm

4.2.3 Nearest Neighbour Query Using Grid Index

The data point of each object is stored in the grid cell. Each grid line is represented by two-dimensional coordinates. Geolocation objects are represented latitude value and longitude value to the grid cell. To perform a nearest neighbor search in Grid index, the query point has latitude and longitude values and a set of keywords. A query point determined the position in the grid for each dimension. Query point visits all possible blocks of the grid. A query point must find the minimum distance between any object which covers a grid cell. First of all, A query point finds the data point in the bucket nearby. Then the grid cells are visited by ordering the minimum distance from the query point. The objects have to sort in ascending order to the query point nearby. To limit the search space, it needs to define the search distance from the query point. The query point searches the data point in the grid that interests the disk blocks. Figure 4.7 shows how to search for nearest neighbour object using grid cell.

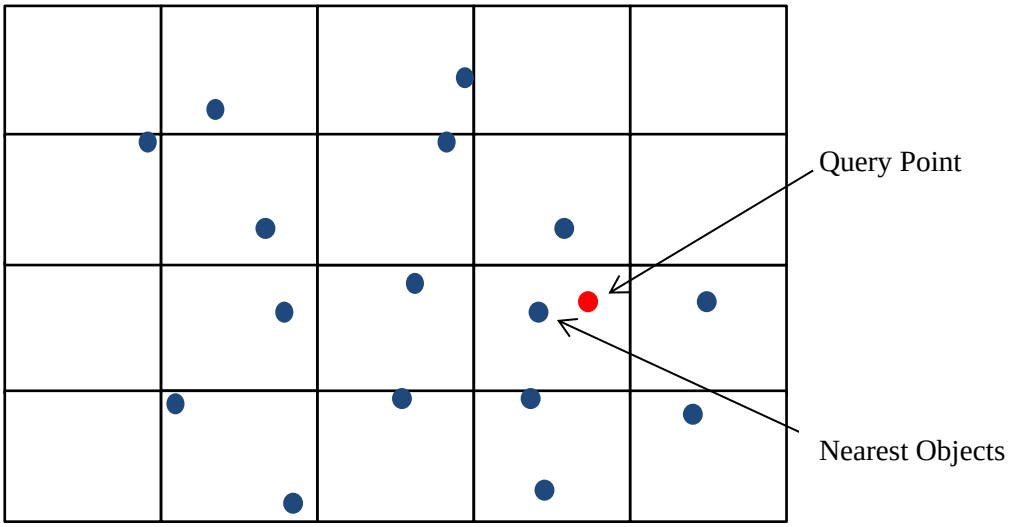


Figure 4.7 Nearest Neighbour Query in Grid Index

4.2.4 Range Query in R-tree

The R-tree is an index tree structure derived from the B-tree that uses multidimensional indexes. An R-tree is a dynamic index structure used for spatial searches that organize areas into the minimum bounding area (MBR). R-tree is used to implement both root and leaf nodes, leveraging the MBR to divide the area, and child nodes using the grid to implement the MBR in cells of the same size. The update cost required for the grid is small. Therefore, the framework does not readjust when the object changes its location.

In R-tree, the nearest indices that are sent by grid index are constructed as a tree. The R-tree consists of root nodes, intermediate nodes, and leaf nodes. The processing step of the R-tree construction for nearest neighbour search is shown in Figure 4.8.

In this algorithm, p is the query point and points in a node are objects. MBR means the Minimum Bounding Rectangle of each leaf node. The two ordering metrics in the R-trees are MinDist and MinDist. MinDist is the distance of object O from query point P. MinMaxDist is the minimum possible maximum distance to the face or vertex of the MBR containing O. MinDist and MinMaxDist provide the lower and upper bounds of the actual distance of object O and query point P respectively.

```
begin
    if  node is a leaf node
        Compute distance from p to all points contained in current node.
        Update the current best NNs and Current best distance if the
        distances between p and all points are less than the maximum
        distance of nearest NNs list.
    else
        Sort the MBRs contained in current node using MINDIST in
        ascending order.
        This sorted list is denoted the Active Branch List (ABL).
        Compute the minimum of the MINMAXDIST of all MBRs.
        Apply pruning strategy to the ABL to remove unnecessary
        branches.
        Iterate on this ABL until it is empty. For each iteration step, apply
        pruning strategy recursively to its children node.
        When the ABL is empty,
        Return nearest neighbor.
end
```

Figure 4.8 Nearest Neighbour Search on R-tree

There are two pruning strategies. They are downward pruning and upward pruning. In downward pruning, it allows pruning an entry from Active Branch List (ABL). If MINDIST of MBR [i] (length of ABL) is greater than MINMAXDIST, discard MBR[i] and all other nodes with greater MINDIST from ABL. In upward pruning, discard MBR[i] and all other nodes with greater MINDIST from ABL if MINDIST of MBR[i] is greater than the current best distance. MBR is a d-dimensional rectangle, which is the minimal rectangle that fully encloses (bounds) an object (or a set of objects). Each rectangle refers to the coordinate of the lower-left corner and the coordinate of the upper right corner. Every face of any MBR contains at least one point of an actual spatial object. Figure 4.9 shows the query point retrieving the nearest objects which contain within MBR.

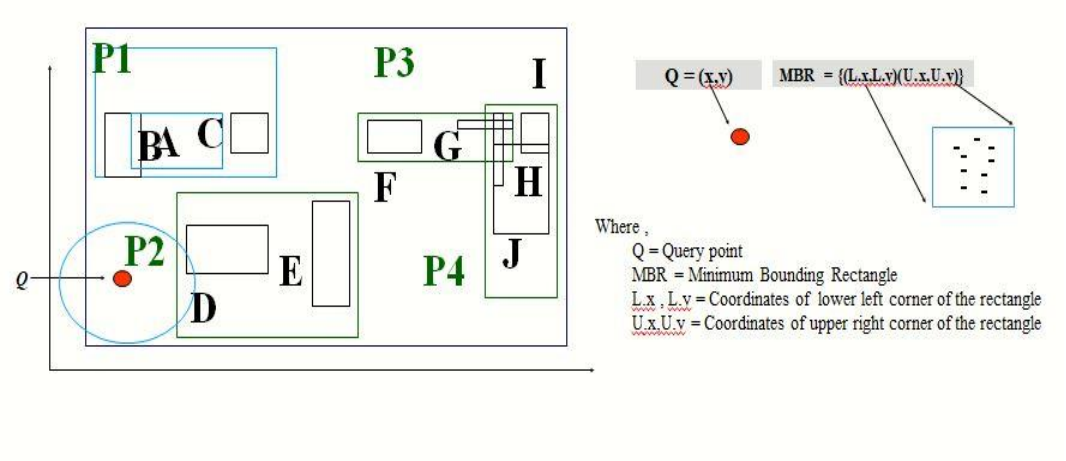


Figure 4.9 Distance Computing with MBR

$$mindist\ (P, R) = \sqrt{\sum_{i=1}^{d} (p_i - r_i)^2} \qquad 4.1$$

where,

R = the minimum rectangle
P = the point is inside R
r = the radius value between lower left corner and upper right corner
d = the minimum of all the (d) distances

4.2.5 Nearest Neighbour Query using Haversine

The nearest neighbor query retrieves the top k nearest neighbours based on the user's position. It is also a method to classify objects based on the closest matching entries obtained from training data. Training data has several attributes that represent its characteristics. This is modeled as a many-dimensional space representation. The k-NN finds the distance between the query and all the objects in the data, selects the specified number of examples (K) closest to the query, and votes for the most frequent label (for classification) or averaging the labels (in the case of regression). The most common type of nearest neighbour search is the point k nearest objects of the current location is taken based on the value of k. To implement the k-nearest neighbours query, this study has used coordinates of the top left corner and right bottom corner within a specified area. The following process of kNN query is shown in Figure 4.10 with flowchart diagram and step by step procedure to find kNN objects by using Haversine;

1. Load the data
2. Initialize *K* value that user-chosen the numbers of objects
1. Calculate the distance between user position and the position of the object from the spatial database
2. Sort the ordered collection of distances and indices of nearest points
3. Retrieve the first K entries from the sorted collection
4. Get the labels of the selected K entries
5. If regression, return the mean of the K labels
6. If classification, return the mode of the K labels

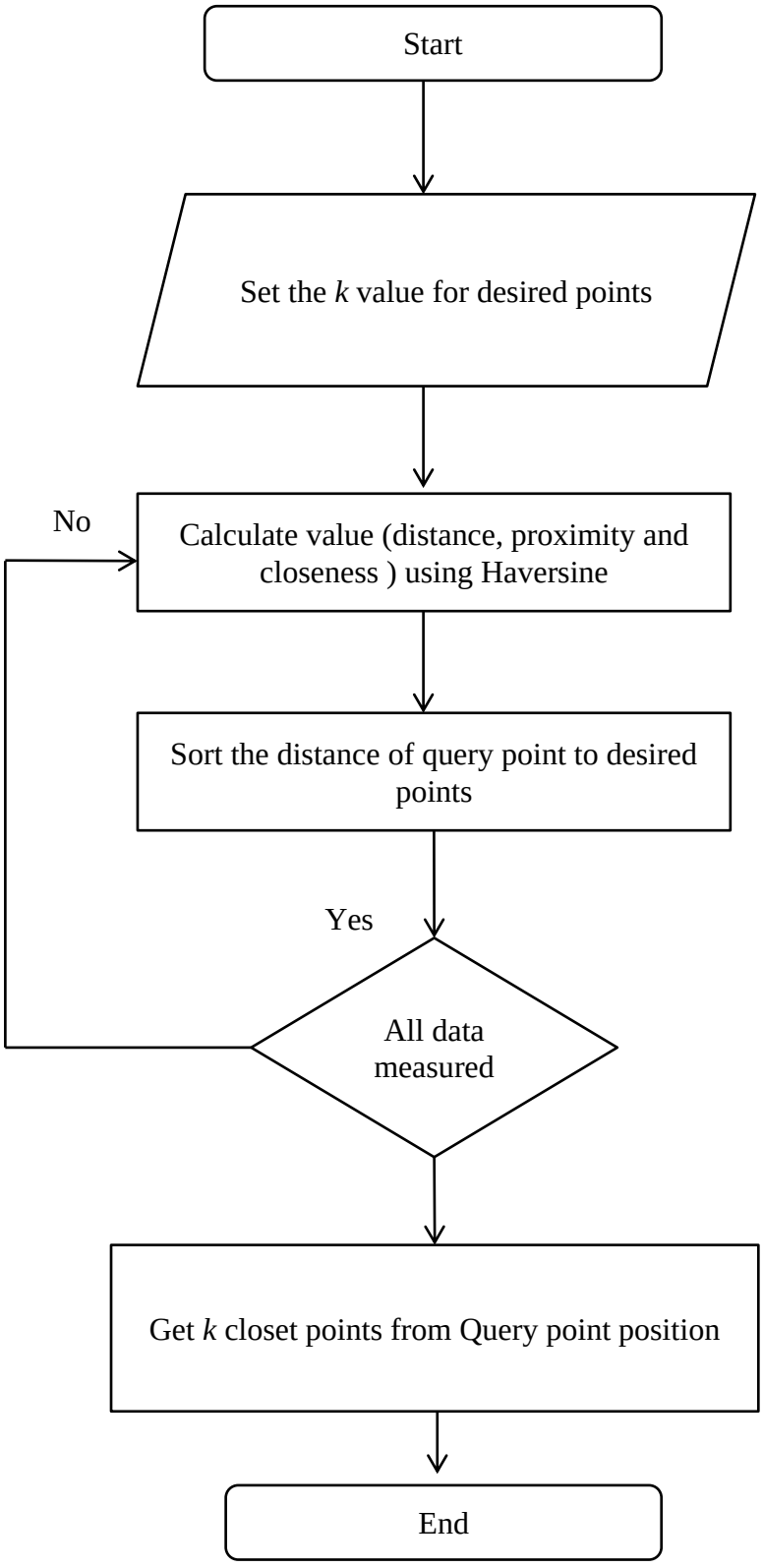


Figure 4.10 Flowchart of kNN Query with Haversine

The distance is calculated by using Haversine formula. This formula is used as follows. Let long1, lat1 be longitude and latitude of current location and lon2, lat2 be longitude and latitude of next location respectively.

dLong = long2-long1 4.2

dLat = lat2-lat1 4.3

a = (sin (dLat/2))^2 + cos(lat1)× cos(lat2)×(sin (dLong/2)) ^2 4.4

C = 2× atan2 (sqrt (a), sqrt(1-a)) 4.5

D =R×C 4.6

where, a=the square of half of the straight-line distance between the two points

c=the great circle distance in radians

D=the distance between current point and query point

R=radius of the earth (R=6371.01km)

Haversine formula is suitable for calculating distance for the spatial object. After calculating distance, R-tree keeps the sorted buffer of the nearest neighbour.

4.2.6 Inserting Process in R-tree

The inserting process in R-tree is similar to the B-tree. The new object is always inserted into a leaf node which contains the search key and object (record) pointer. To insert a new object into a leaf node, the R-tree needs to update with the new object. Before Insertion, the minimum bounding box (MBB) finds the bounding box (BB) in the root node that is enlarging BB to contain the new object. Then, it will occupy at the least amount of area to the BB. If the space is not available on the leaf node for the new object, there is a need to split the leaf node into two pages. To split the leaf page, the parent node has to split into two internal nodes. Then, a new object is inserted into a new leaf node according to the procedure of inserting process in R-tree. The flowchart of the inserting process in the R-tree shows in Figure 4.11. Inserting a new object in R-tree algorithm is shown in Figure 4.12.

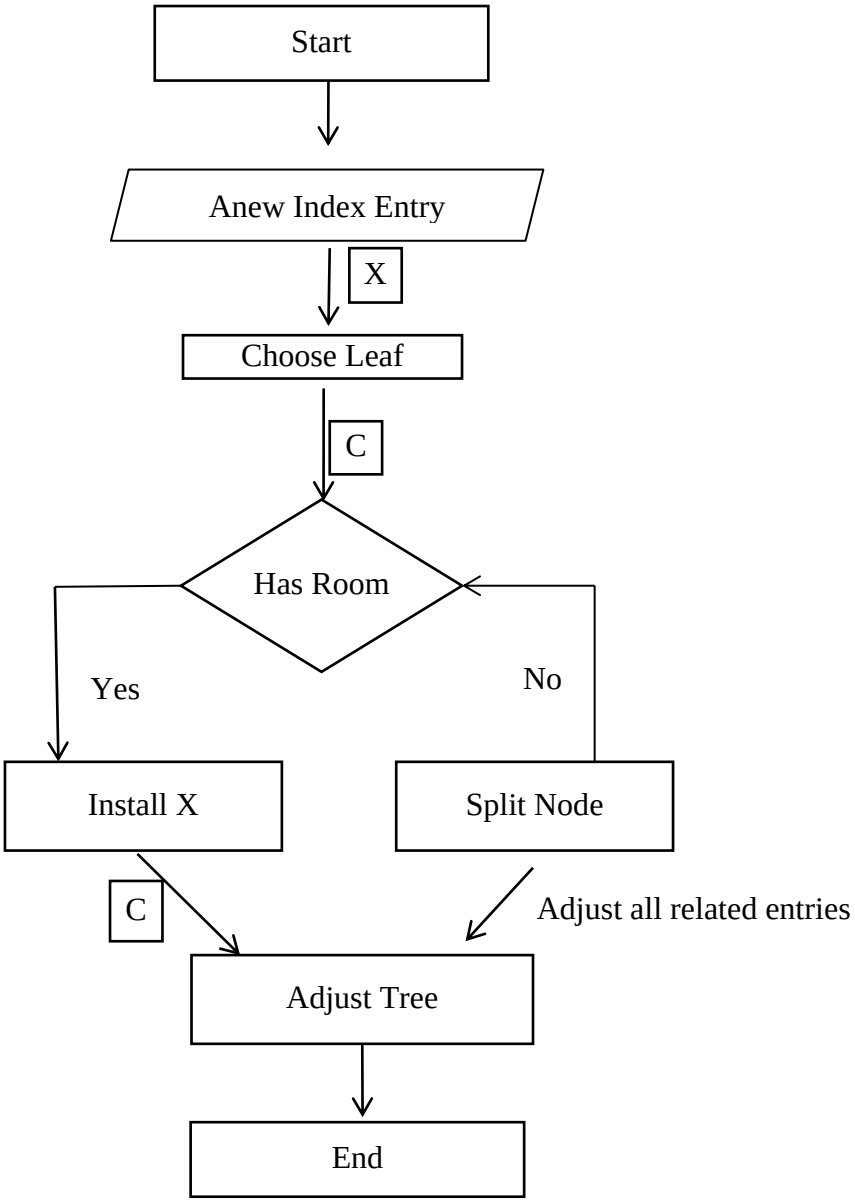


Figure 4.11 Insertion Process in R-tree

```
Algorithm: Insert ( (mbb,ObjID), n)

Insert ( (mbb, ObjID), n )
  {
      if ( n == internal node )
      {
        for ( each entry ( BB,  childptr ) in node n ) do
        {

          if ( mbb of inserted object ⊆ BB )
          {

            Insert( (mbb,ObjID),  childptr;

              return;
        }
      }

        Insert( (mbb,ObjID),  childptr);

      return;
      }

      else

      {
        if ( leaf node has space to hold object )
        {
          Insert (mbb, ObjID) in the leaf node;
        }
        else
        {
          Split the objects in the leaf node into set1 and set2;

          Find the bounding box BB1 for set1 of the objects;
          Find the bounding box BB2 for set2 of the objects;

          Replace the parent's (BB, ptr) by (BB1, set1) and (BB2, set2);

        }
    }
  }
```

Figure 4.12 Insertion Algorithm in R-tree

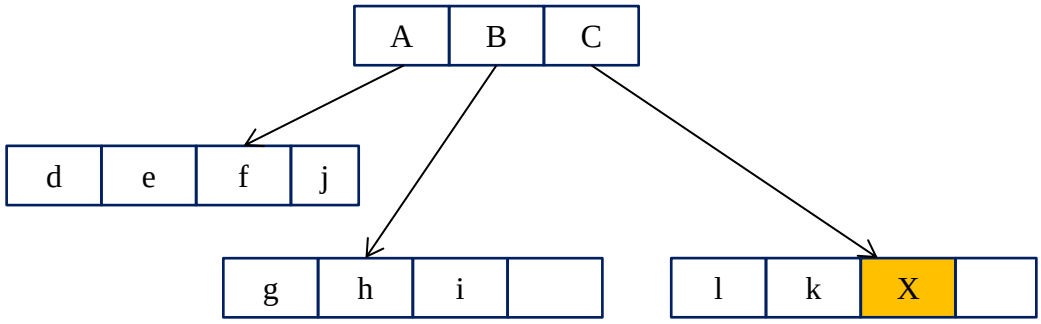


Figure 4.13 Example of Insertion in R-tree

4.2.7 Searching Process in R-tree

The search process in R-tree will find the objects that are located within the corresponding bounding box. The objects in the R-tree contain the two-dimensional point (with x and y coordinates). The search algorithm will check in the first BB. If the object does not exist in the BB, it will skip the step and then check the second BB. The search algorithm will search the second subtree and the object contains within the MBB in the leaf node. The search algorithm in R-tree is shown in Figure 4.14. In the algorithm, n is the internal node (including root node) of the R-tree. The object is a set of coordinate points with x and y. BB is the child node of the R-tree. Each node has a search key and node pointer. The search process of the flow diagram in R-tree is shown in Figure 4.15. The example of searching the object in the R-tree is shown in Figure 4.16.

```
Algoritm: Search((x,y),n,output)
Search ( ( x,y ) , n, output)
{
        if ( n == internal node )
        {
          for ( each entry ( BB,  childptr ) in internal node n ) do
          {
            if ( (x,y) ∈ BB )
            {
              Search( (x,y),  childptr, output);
          }
        }
        }
        else
        {

          for ( each object Ob in node n ) do
          {
            if ( (x,y) ∈ MBB(Ob) )
            {
              Add Ob to output;
          }
    }
  }
```

Figure 4.14 Search Algorithm in R-tree

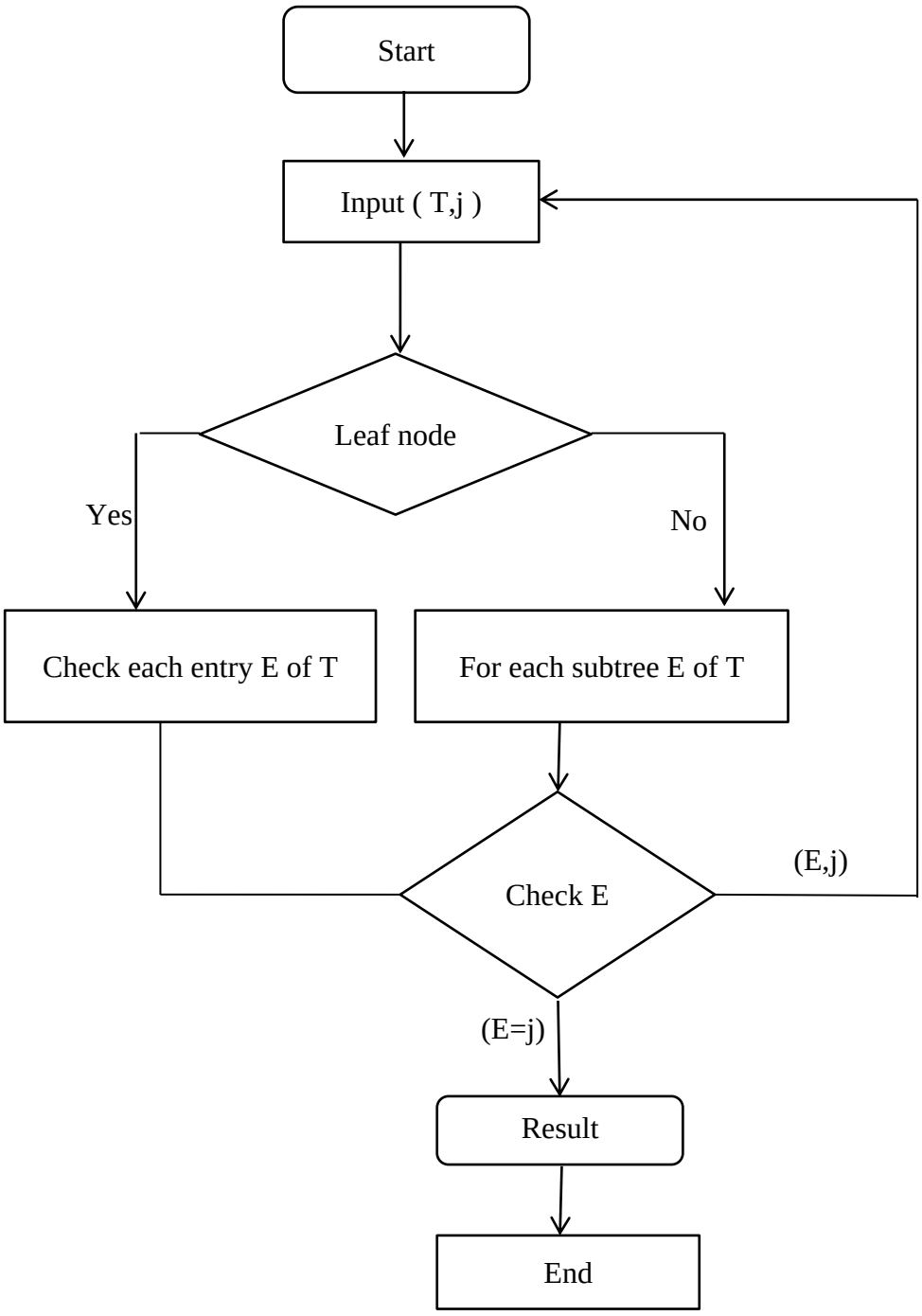


Figure 4.15 Flowchart of Searching in R-tree

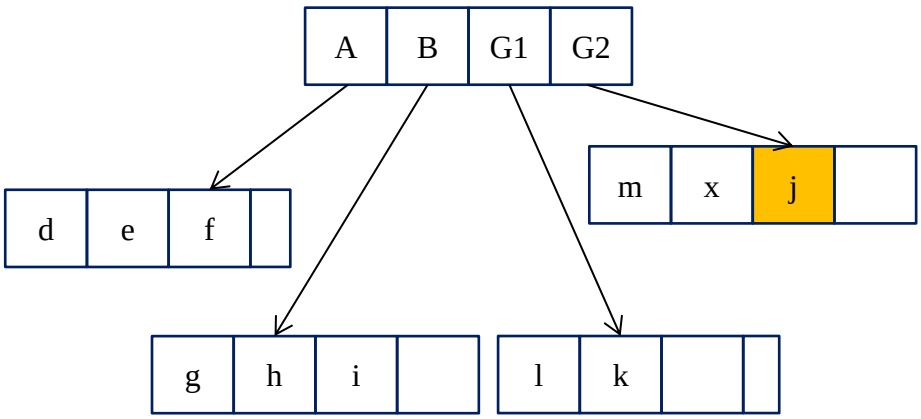


Figure 4.16 Searching in R-tree Structure

4.2.8 Database Implementation

The proposed system is designed for the client-server architecture. In the server-side implementation, PHP and JavaScript are easily used to develop web applications and mobile applications. PHP is also a server scripting language and a powerful language to make dynamic and interactive Web apps. MySQL is a widely used relational database system (RDBMS). The table structure of the database is with some basic fields in MySQL database. Figure 4.17 shows creating tables of the proposed system.

```
--
-- Database: `location`
--

-- --------------------------------------------------------
--
-- Table structure for table `category`
--
CREATE TABLE `category` (
  `category_id` int(11) NOT NULL,
  `name` varchar(40) COLLATE utf8_unicode_ci NOT NULL,
  `icon` varchar(50) COLLATE utf8_unicode_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Dumping data for table `category`
--

INSERT INTO `category` (`category_id`, `name`, `icon`) VALUES
(1, 'clinic', 'clinic.png'),
(2, 'mini-mart', 'mini-mart.png'),
(4, 'restaurant', 'restaurant.png'),
(6, 'shop', 'restaurant.png'),
(7, 'store', 'restaurant.png'),
(8, 'university', 'university.png');

-- --------------------------------------------------------
--
-- Table structure for table `markers`
--

CREATE TABLE `markers` (
  `id` int(255) NOT NULL,
  `name` varchar(60) COLLATE utf8_unicode_ci NOT NULL,
  `address` varchar(80) COLLATE utf8_unicode_ci NOT NULL,
  `phone` varchar(20) COLLATE utf8_unicode_ci NOT NULL,
  `lat` float(10,6) NOT NULL,
  `lng` float(10,6) NOT NULL,
  `type` varchar(30) COLLATE utf8_unicode_ci NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

Figure 4.17 Table Structure of Data

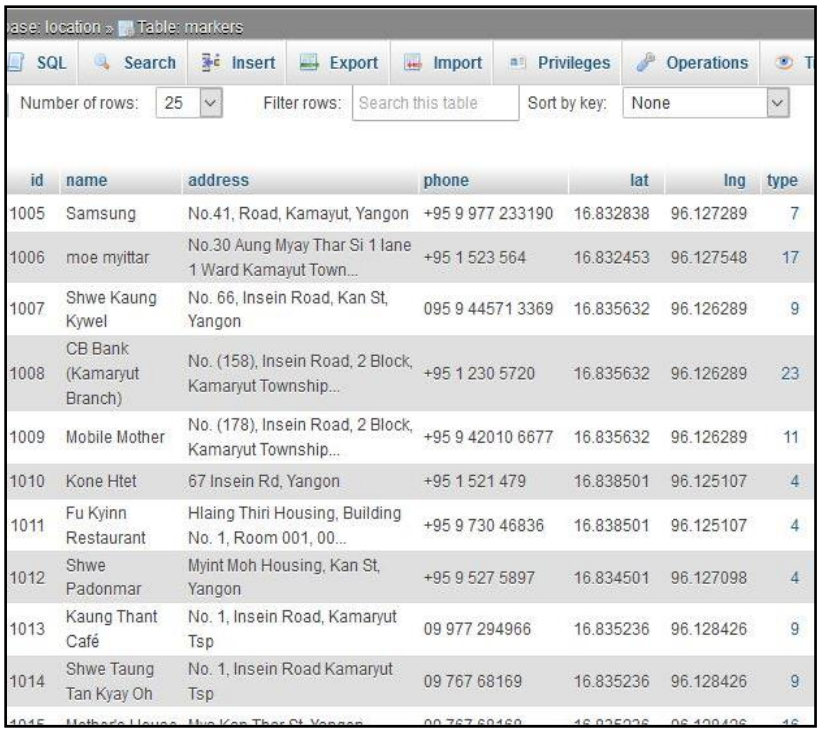
PHP Data Objects (PDO) will work to connect MySQL database. Before accessing data in the database, that will need to be able to connect to the server. It is required to write a script open connection to MySQL.The benefit of PDO is an exception class to handle the problems in database queries. The script shows to access data from the database is shown in Figure 4.18.

```php
<?php
@session_start();
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "location";
try{
    $con=new PDO("mysql:host=$servername;dbname=$dbname;
    charset=utf8",$username,$password);
    $con->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
}
catch(PDOException $e){
    echo "Error::". $e->getMessage();
}

function return_success($msg,$data = array()){
    $return_obj['status'] = true;
    $return_obj['msg'] = $msg;
    $return_obj['data'] = $data;
    //return $return_obj;
    echo json_encode($return_obj);
    exit;
}
function return_fail($msg,$data=array()){
    $return_obj['status'] = false;
    $return_obj['msg'] = $msg;
    $return_obj['data'] = $data;
    //return $return_obj;
    echo json_encode($return_obj);
    exit;
}
?>
```

Figure 4.18 PDO Script for MySQL Opens the Connection

The geo-data are stored in MySQL database. In the database, geo-location (Latitude and Longitude) objects with attributes are categorized by service names such as cinema, restaurants, and hotels. The sample data are stored in MySQL database as shown in Figure 4.19.



ase: location » Table: markers

SQL | Search | Insert | Export | Import | Privileges | Operations

Number of rows: 25 Filter rows: Search this table Sort by key: None

id	name	address	phone	lat	lng	type
1005	Samsung	No.41, Road, Kamayut, Yangon	+95 9 977 233190	16.832838	96.127289	7
1006	moe myittar	No.30 Aung Myay Thar Si 1 lane 1 Ward Kamayut Town...	+95 1 523 564	16.832453	96.127548	17
1007	Shwe Kaung Kywel	No. 66, Insein Road, Kan St, Yangon	095 9 44571 3369	16.835632	96.126289	9
1008	CB Bank (Kamaryut Branch)	No. (158), Insein Road, 2 Block, Kamaryut Township...	+95 1 230 5720	16.835632	96.126289	23
1009	Mobile Mother	No. (178), Insein Road, 2 Block, Kamaryut Township...	+95 9 42010 6677	16.835632	96.126289	11
1010	Kone Htet	67 Insein Rd, Yangon	+95 1 521 479	16.838501	96.125107	4
1011	Fu Kyinn Restaurant	Hlaing Thiri Housing, Building No. 1, Room 001, 00...	+95 9 730 46836	16.838501	96.125107	4
1012	Shwe Padonmar	Myint Moh Housing, Kan St, Yangon	+95 9 527 5897	16.834501	96.127098	4
1013	Kaung Thant Café	No. 1, Insein Road, Kamaryut Tsp	09 977 294966	16.835236	96.128426	9
1014	Shwe Taung Tan Kyay Oh	No. 1, Insein Road Kamaryut Tsp	09 767 68169	16.835236	96.128426	9

Figure 4.19 Location Data with Related Attributes

In the database, the entity-relationship(ER) diagram uses essential for modeling the data stored. ER diagrams specify entities, relationships, and attributes. The database designs table with attributes and lines that represent the relationships between the table is shown in Figure 4.20.

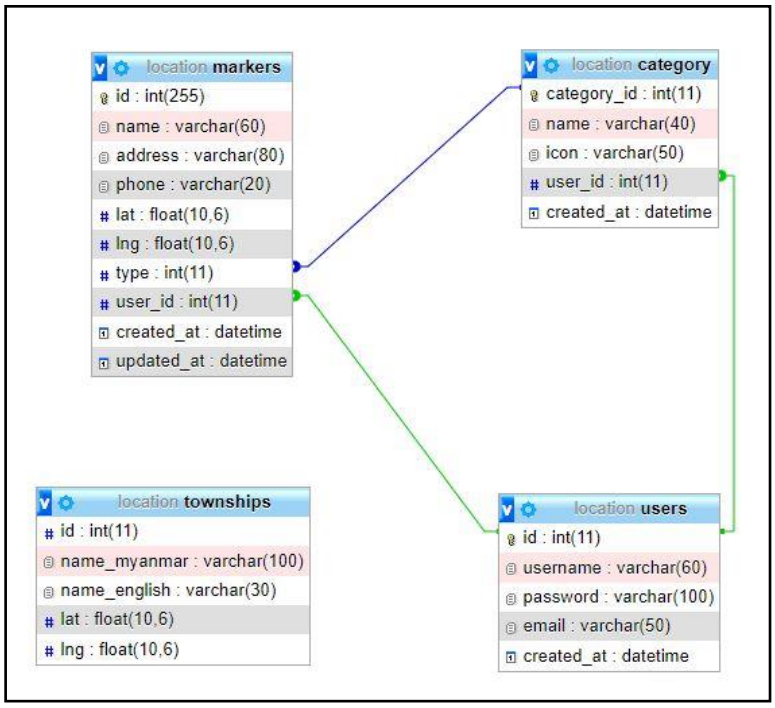


Figure 4.20 Entity-Relationship(ER) Diagram of the Proposed System

4.2.9 Structure of Proposed Index Structure

To be retrieved data effectively from the database, the proposed system is built index structure for querying geo-data efficiently. The location objects are stored in the database by using the proposed index structure. In this index structure, R-tree is used for querying spatial objects which are two-dimensional points with the value of latitude and longitude, and its attribute data is shown in Figure 4.21. To store an object into the database by using the proposed index structure, the object is covered by the bounding box rectangle that is assigned by geolocation coordinates (i.e. latitude and longitude). The bounding box of the leaf node is the minimum sized rectangle that contains all the objects. Each rectangle of the leaf node has a pointer to the object.

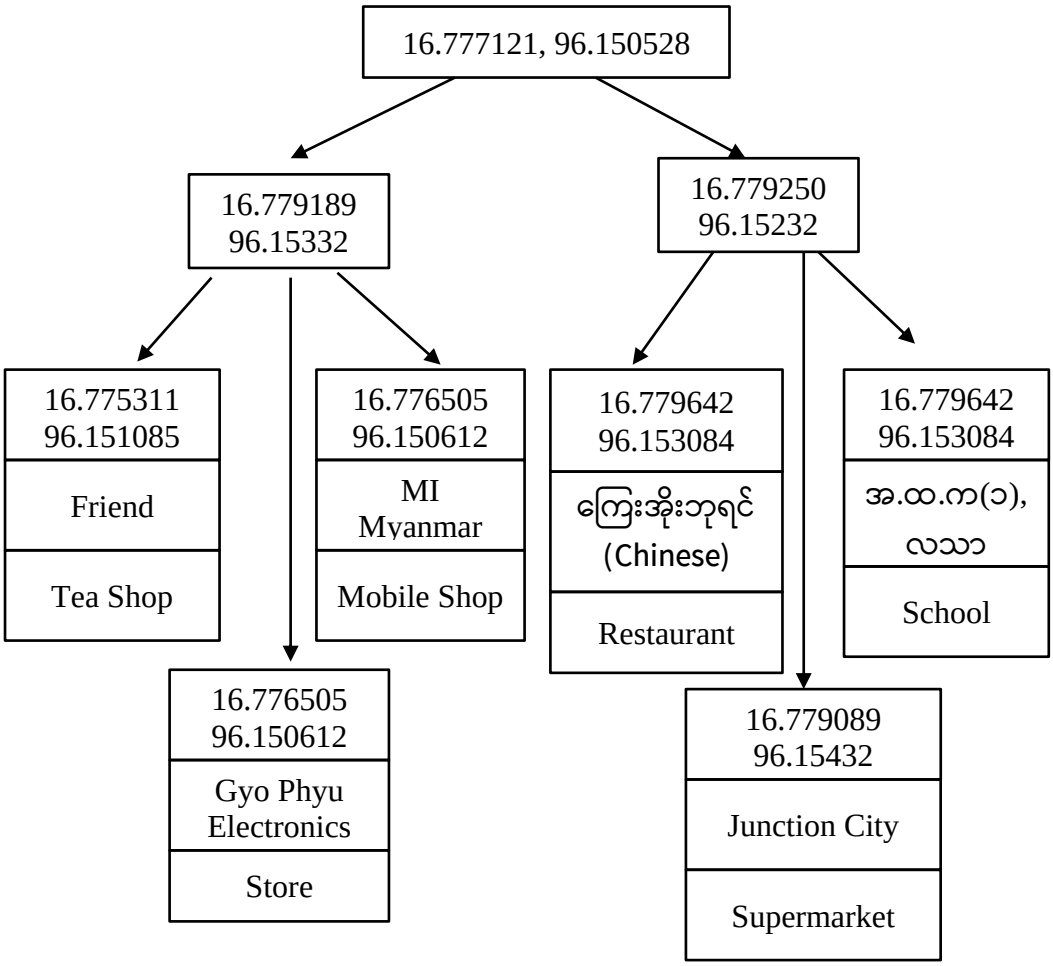


Figure 4.21 Structure of the Proposed Index Structure

4.2.10 Summary

This chapter presents the implementation of the proposed system architecture. At first, it explains the system overview and workflow of spatial indexing techniques. The proposed system implemented the client-server model. In the implementation of the proposed system, the process of grid index structure is explained in detail. Then, it describes the computing process of R-tree works in nearest neighbor search for the range query. In addition, it presents the process of kNN query applied with Haversine that can access nearest neighbour objects efficiently. It describes the structure of database design and how the data retrieve to be quickly accessed. Therefore, the grid-based R-tree index technique can reduce search space and can speed up the performance of spatial queries.

CHAPTER 5
EXPERIMENTAL RESULTS AND EVALUATION RESULTS

This chapter describes the experimental results and evaluation results of the research work. The main objective of experimental results is to show the proposed system how to implement nearest neighbor queries which are range query and k-Nearest Neighbour (kNN) query of retrieving spatial objects. The system supports the client user to find nearby places of interest from the specified user's geo-location. Moreover, this chapter presents the evaluation of the proposed system's computing performance and response time. The comparison of evaluation results describes the proposed indexing schemes.

5.1 Prerequisite of System Experimentation

The system develops the client-server model. On the server side, it uses Amazon Elastic Compute Cloud (Amazon EC2) that offers the compute platform with the choice of processor, storage, networking, operating system. On the client-side, it uses Apache Cordova that is an open-source mobile development framework. It allows the developers to use standard web technologies - HTML5, CSS3, and JavaScript for cross-platform development. The following requirements need to be the system implementation;

1. Amazon EC2 : virtual server in the AWS Cloud.
2. XAMPP : Open-source cross-platform web server solution for localhost.
   a. Apache
   b. MySQL
3. CentOS 7 (Linux OS for web server)
   a. PHP 7.3
   b. MariaDB 10.3
4. Apache Cordova 10.0 (Open source mobile development framework).
   a. Android SDK (Android API 8 or higher)
5. Google Map API v.3

5.1.1 Server-side Implementation

On the server side, the proposed system is implemented with Amazon Elastic Compute Cloud (Amazon EC2). Amazon EC2 provides a web-based user interface and the Amazon EC2 console. In the console, the proposed system uses a Linux instance. Instances are virtual servers in the AWS cloud. Using Amazon EC2, the proposed systems set up and configure the Linux operating system (CentOS) and Application running on the instance. The proposed system uses two core processors, 4048 RAM, and 8GB storage. Figure 5.1 shows the console of AWS cloud instance.

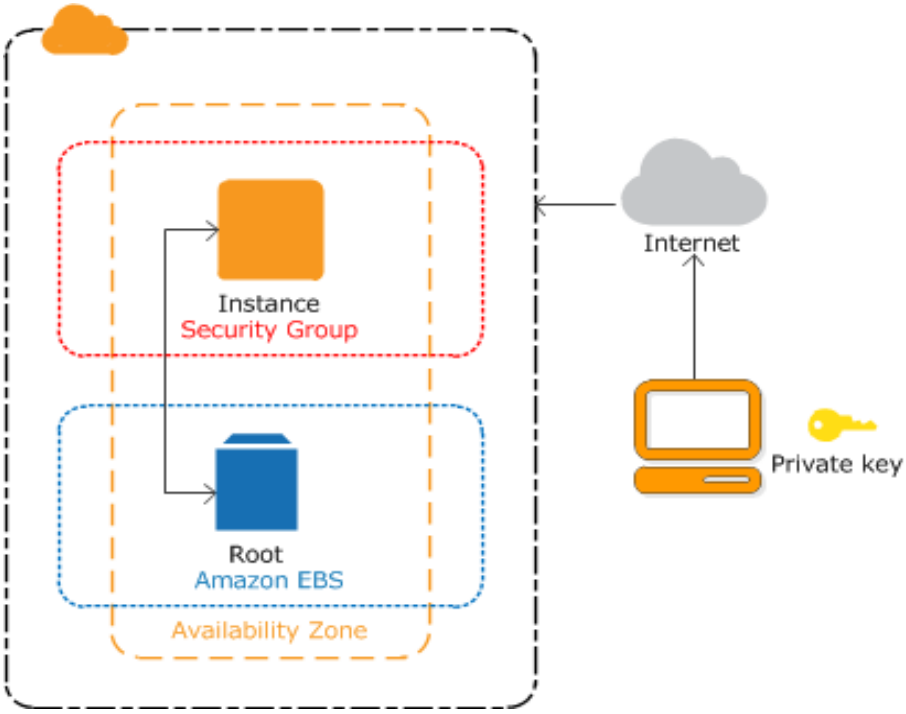


Figure 5.1 Amazon EC2 Instance Console

In the EC2 console, Apache web server with PHP and MariaDB is installed in CentOS Linux. The solution package of the proposed system replaces the server environment. Figure 5.2 shows the configuration of the web-server installation.

```
[root@www ~]# vi /etc/httpd/conf/httpd.conf

# line 86: change to admin's email address
ServerAdmin root@srv.world

# line 95: change to your server's name
ServerName www.srv.world:80

# line 151: change
AllowOverride All

# line 164: add file name that it can access only with directory's name
DirectoryIndex index.html index.cgi index.php

# add follows to the end
# server's response header
ServerTokens Prod

[root@www ~]# systemctl start httpd
[root@www ~]# systemctl enable httpd
```

Figure 5.2 Web-server Installation

The following Figure 5.3 shows the installation process of the database in MariaDB. Tables and data of MySQL database will be stored in the storage server.

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| location           |
| m0vie              |
| mysql              |
| performance_schema |
+--------------------+
5 rows in set (0.000 sec)

MariaDB [(none)]> use location;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [location]> show tables;
+--------------------+
| Tables_in_location |
+--------------------+
| category           |
| markers            |
| users              |
+--------------------+
3 rows in set (0.000 sec)
```

Figure 5.3 Database Creation in MariaDB

5.1.2 Dataset for Spatial Database

The system created the dataset of Yangon Region for implementation and evaluation because Yangon is the most crowded city in Myanmar and all of the best facilities are available can get in the Yangon region. There are four districts and forty-five townships in the Yangon Region. The dataset is created for thirty-three townships which are more business townships except for other townships in Yangon. Table 5.1 describes the townships which are the created dataset.

Table 5.1 Available Townships Including Dataset

No.	Township	Latitude	Longitude
1	ရွှေပြည်သာ(Shwepyitha)	16.95504	96.08329
2	မင်္ဂလာဒုံ(Mingaladon)	16.94914	96.12795
3	အင်းစိန် (Insein)	16.90177	96.09596
4	မြောက်ဥက္ကလာပ(North Okkalapa)	16.91878	96.16303
5	ဒဂုံမြို့သစ်အရှေ့ပိုင်း(East Dagon)	16.91281	96.21323
6	လှိုင်သာယာ(Hlaingthaya)	16.85402	96.06893
7	မရမ်းကုန်း(Mayangon)	16.86619	96.14261
8	ဒဂုံမြို့သစ်(မြောက်ပိုင်း)(North Dagon)	16.87774	96.12523
9	လှိုင်(Hlaing)	16.84793	96.12523
10	တောင်ဥက္ကလာပ(South Okkalapa)	16.84625	96.17986
11	ဒဂုံမြို့သစ်(တောင်ပိုင်း)(South Dagon)	16.84016	96.22582
12	ဒဂုံမြို့သစ်ဆိပ်ကမ်း(Dagon Seikkan)	16.84049	96.27256
13	ကမာရွတ်(Kamayut)	16.82768	96.13244
14	ရန်ကင်း(Yankin)	16.83601	96.16246
15	သဃ်ိန်းကျွန်း(Thingangyun)	16.83101	96.19323
16	ကြည့်မြင်တိုင်(Kyimyindaing)	16.81457	96.12188
17	ဗဟန်း(Bahan)	16.81543	96.15611

18	စမ်းချောင်း(Sanchaung)	16.80381	96.13729
19	တာမွေ(Tarmwe)	16.81018	96.17646
20	ဒဂုံ(Dagon)	16.79495	96.14693
21	အလုံ(Ahlon)	16.7824	96.12786
22	မင်္ဂလာတောင်ညွန့် (Mingala Taungnyunt)	16.78891	96.16786
23	သာကေတ(Thaketa)	16.79301	96.20296
24	လမ်းမတော်(Lanmadaw)	16.77991	96.14234
25	လသာ(Latha)	16.77712	96.15053
26	ပန်းပဲတန်း(Pabedan)	16.77698	96.15601
27	ပုဇွန်တောင်(Pazundaung)	16.77965	96.17439
28	ဒေါပုံ(Dawbon)	16.78192	96.18435
29	ကျောက်တံတား(Kyauktada)	16.77426	96.16164
30	ဗိုလ်တထောင်(Botataung)	16.77197	96.16971
31	ဆိပ်ကြီးခနောင်တို(Seikkyi Kanaungto)	16.75779	96.11637
32	ဒလ(Dala)	16.75855	96.14354
33	သန်လျင်(Thanlyin)	16.76378	96.25184

The collected dataset is nearly 10000 datasets including geo-location and its attributes respectively. The sample of the collected dataset is shown in Table 5.2. The dataset with geolocation is collected in two ways using Google places services and is manually checked in the places in townships. There are thirty-one categories of services of geolocation data in the database as shown in Table 5.3. Each service has the name of the service, address, phone number, and geo-position of service. The places of services are created by using MySQL database and saving them into it. In addition, the system offers the client to create its dataset by choosing the category of service. The system provides user-generated content according to the user's choice of specified location and services.

Table 5.2 Sample Dataset of Location Objects

ID	Name	Latitude	Longitude	Service
B1	AYA Bank	16.82485	96.14223	Bank
B5	AGD Bank	16.82476	96.14245	Bank
B20	MAB Bank	16.82478	96.14318	Bank
B34	CB Bank	16.82517	96.14372	Bank
R5	Khunna Mile (7-Miles)	16.85611	96.14112	Restaurant
R14	Cafe Del Seoul	16.85656	96.14108	Restaurant
R233	Royal Thai Restaurant	16.85326	96.14253	Restaurant
C23	Kaung Kaung	16.90192	96.09898	Clinic
C29	Wint War	16.79042	96.20398	Clinic
C78	Dr. Wai Wai	16.78355	96.20081	Clinic
C99	Kaing Clinic	16.78012	96.18991	Clinic
A23	MAB ATM	16.84269	96.11278	ATM
A56	CB Bank ATM	16.83415	96.13498	ATM
A78	AYA ATM	16.82487	96.14037	ATM
CS125	ABC convenience Store	16.85229	96.12408	Convenience Store
CS129	G&G Convenience Store	16.85128	96.12383	Convenience Store
CS134	Joy Convenience Store	16.8518	96.12061	Convenience Store
P3	အာယုဆေးဆိုင်	16.81393	96.12545	Pharmacy
P12	Thiddi Win Pharmacy	16.78225	96.18201	Pharmacy
P30	Health Assistant Zone	16.78203	96.18275	Pharmacy
E5	Embassy of France	16.7903	96.14233	Embassy
E8	Embassy of Israel	16.7857	96.14217	Embassy
E9	Embassy of Russia	16.78582	96.14368	Embassy
E11	Embassy of Indonesia	16.790203	96.142014	Embassy
F8	ဒေလီယာ	16.88904	96.10805	Fashion
F45	Shu Tine Yin	16.88698	96.11106	Fashion
F156	Ju Ju	16.88669	96.11124	Fashion
MP15	MI Myanmar	16.776505	96.150612	Mobile Shop
MP45	JMart Mobile	16.779606	96.152176	Mobile Shop

Table 5.3 Available Categories of Services

No.	Type of Services	No. of Dataset
1	Clinic	906
2	Mini-mart	466
3	Restaurant	1102
4	Shop	644
5	Store	1374
6	University	14
7	Tea Shop	824
8	ATM	170
9	Mobile Shop	632
10	Supermarket	169
11	Convenient Store	486
12	Dentist	218
13	Pharmacy	236
14	Car Sale & Service	210
15	Hospital	182
16	Fast-food	248
17	Embassy	28
18	School	30
19	Beer Station	166
20	Cafe	160
21	Bank	70
22	Services	28

23	Hostel	150
24	Fashion	758
25	Bus Stop	200
26	Hotel	330
27	Police Station	60
28	Library	14
29	Fuel	140
30	Computer Services	110
31	Cinema	10

5.1.3 Implementation of Spatial Database

Datasets are stored in the MySQL database. The database contains the tables that are service type and related attributed data. Geo-location (Latitude/Longitude) and its related attributed data are being stored by dividing into grid index computing. Then, the data are constructed with R-tree indexing respectively. The datasets of places of interest (PoI) are taken from the spatial database. The data can be acquired from the specified area. It contains the service type and related attributes. Figure 5.4 shows the form of the dataset in the database.

1124	Aung Kaung Convenience Store	01-519531	16.853020	96.123901	13	0
				Thiri Myine St		
1125	Myat Myittar clinic	9.59420076755E11	16.852816	96.123962	1	0
				Thiri Myine St		
1126	Win Win Store	+95 1 230 5720	16.852757	96.123947	1	0
				Thiri Myine St		
1127	La Pyae Won (Glass, Aluminum and Steel)	+95 1 230 5720	16.852558	96.123978	7	0
				Thiri Myine St		
1128	999 air composer	095 9 44571 3369	16.852507	96.123970	24	0
				Thiri Myine St		
1129	Aung Thuka Electronic Store	+95 9455182146	16.852448	96.123985	7	0
				Thiri Myine St		
1130	Nokia Sales & Services	+95 1 230 5720	16.852329	96.124382	11	0
				Thiri Myine St		
1131	Buy Now IT & Mobile accessaries	+95 1 230 5720	16.852205	96.124420	11	0
				Thiri Myine St		
1132	Ko Lwin Ko Fashion	+95 1 230 5720	16.851761	96.124474	28	0
				Thiri Myine St		
1133	Saw Thaung Motor and Punk Sales	095 9 44571 3369	16.852320	96.124054	7	0
				Thiri Myine St		
1134	ABC convenience Store	+95 9 730 46836	16.852200	96.124084	13	0
				Thiri Myine St		
1135	Phoo Thit Fashion	+95 1 521 479	16.852148	96.124077	28	0
				Thiri Myine St		
1136	Tun Tuck Kyal Store	01-519532	16.852119	96.124084	28	0
				Thiri Myine St		
1137	Easy Mobile Sales & Services	01-519531	16.852087	96.124100	7	0
				Thiri Myine St		
1138	Bartar Bus stop	+95 9420076772	16.852062	96.124153	11	0
				Thiri Myine St		
1139	Swan Pharmacy	+95 1 230 5720	16.852339	96.123245	15	0
				Thiri Myine St		
1140	Ko Zayar Store	095 9 44571 3369	16.851992	96.123314	7	0
				Thiri Myine St		
1141	Phyo Htet Electronic	+95 1 230 5720	16.851700	96.123474	7	0
				Thiri Myine St		
1142	SCS Mobile	+95 9455182146	16.851625	96.124512	11	0
				Thiri Myine St		
1143	Yangon BBQ	+95 9 730 46836	16.851583	96.124207	4	0

Figure 5.4 Dataset in Database

5.1.4 Using Google Map API in Server-Side

To be added places with geolocation, Google Maps APIs need to use in the proposed system. After creating Google Map API on the server-side, the collected geolocation objects are stored in the database. Google Maps JavaScript API is used and generated as the following Figure 5.5.

```
    //SEARCH OBJECT
//google.maps.event.addDomListener(window, 'load', myMap);
</script>
<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCTLZ4SmUhAYOITG1UxT58GhJi9RnAUWhQ&callback=myMap"></script>
<script>
    var slider = document.getElementById("km");
    var output = document.getElementById("demo");
    output.innerHTML = slider.value;
</script>
```

Figure 5.5 Google Map API key

After assigning Google place API, the collected geolocation objects are placed on the map. The Places API is a service that returns information about places using HTTP requests. The mobile client can search places of interest with geographic locations. Server responses the results of places include with summary information about each place to the mobile clients. The mobile client's current location is shown in Figure 5.6.

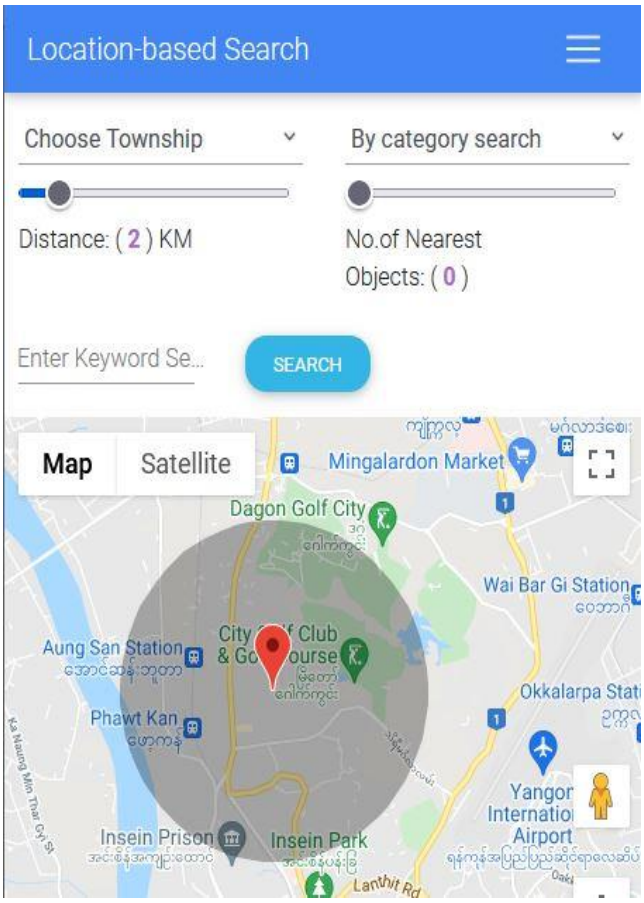


Figure 5.6 Location Marker at the Current Position

On the server-side, the collected geolocation objects are computed with the grid index algorithm for each object. After computing the grid index, R-tree will be retrieved the nearest objects from the user's current locations. Then, the server sends these nearest results to the client.

5.2 User Interface Design

The user input section and the query results area can see in the user interface design. In the user input section, the client users can choose township and type of category. For the range query, the user needs to choose the slide bar for the desired distance search. For the kNN query, the user needs to choose the slide bar for the number of nearest objects. For the keyword search, the user can use the keyword input box to keyword query. In the query result area, the user can see both street view and satellite view on the map. The located objects are shown on the map according to the user's choice of querying. In this system, townships in Yangon Region are selected to provide services that make it easy for users to find location information near those townships. By selecting Hlaing township as shown in the Figure 5.7, the location marker is located in the center of Hlaing township and the user can search for nearby location objects around that township. This system is designed to search for location information by service type and is shown in the Figure 5.8. The user interface design for keyword search that supports for both range search and nearest neighbour search for client users. Figure 5.9 is shown all the location objects that are displayed by using keyword search within the specified range (for example two kilometer distance from current location). The mobile users can view detail information of each objects that show on the map. In Figure 5.10, the top most nearest object with its detail information is selected by the client user.

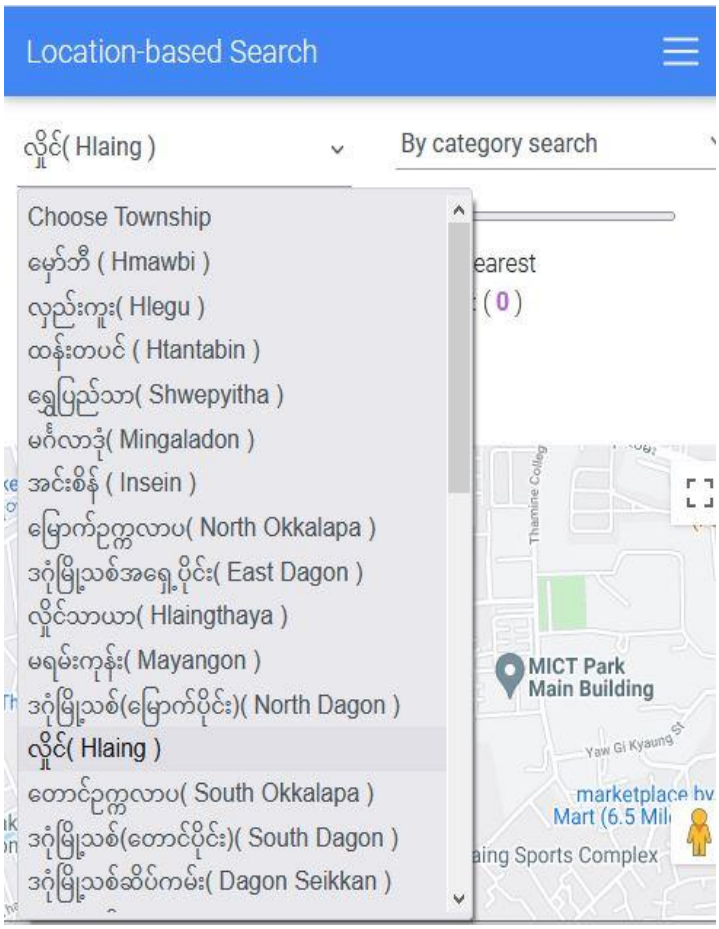


Figure 5.7 Choose Townships in Yangon Region

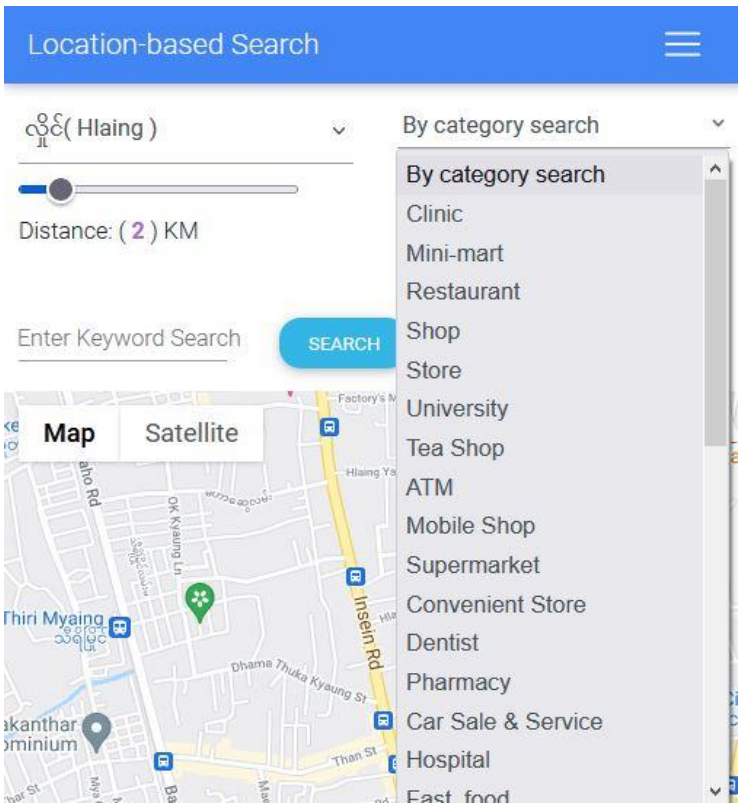


Figure 5.8 Choose Category of Service

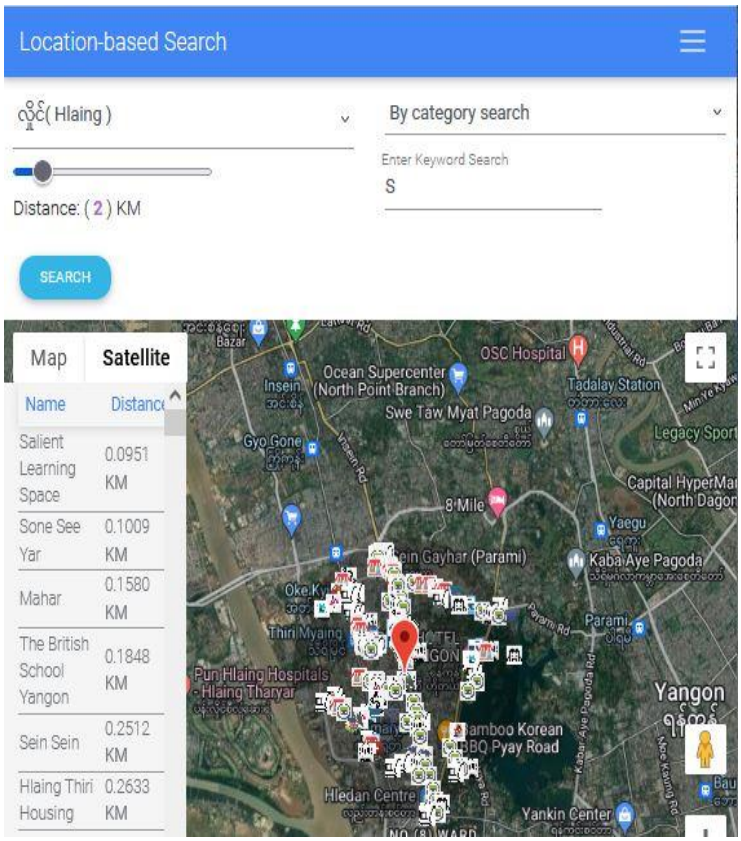


Figure 5.9 Query Results

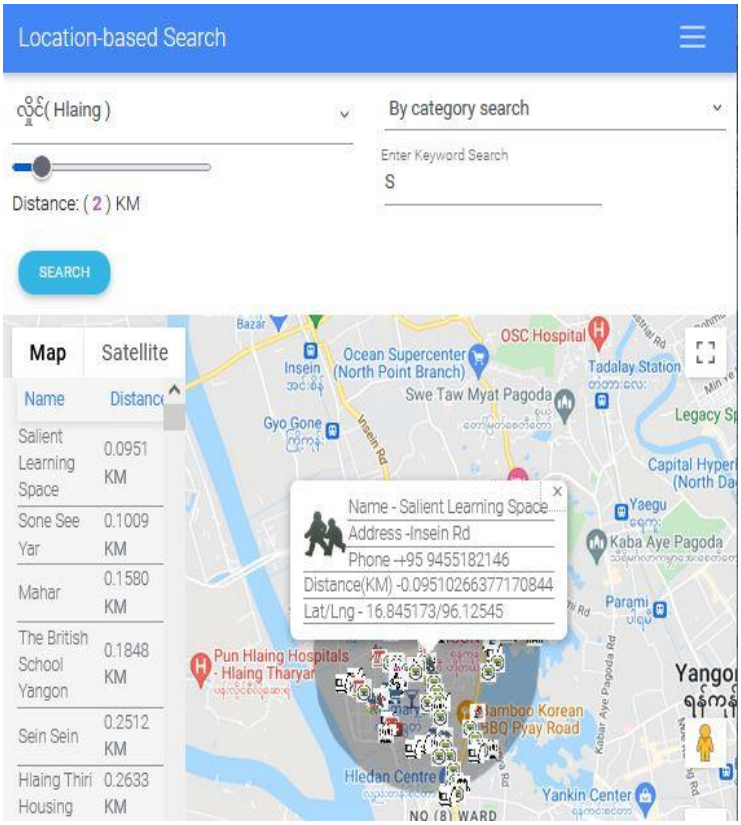


Figure 5.10 Detail result of the Located Object

5.3 Experimental Results of Client Users

The proposed system is designed very conveniently for mobile client users. The mobile users can search not only keyword search but also category search for places of interest. The main idea of the system is focused to approve efficiency and effectiveness while retrieving nearest neighbor queries such as kNN query and range query. The user interface is designed user-friendly for client users. The client system has developed a responsive view for laptop and mobile devices.

5.3.1 K-Nearest Neighbor Query (kNN) Results of The Client User

The user interface design of the mobile application is easy to use. The mobile user can point location marker easily on the map. Then, the system gets the user's current location. The mobile user can choose query types which are kNN query or range query. Then, the user can input keyword search or choose the service type. If the user chooses kNN query, the proposed system responds to the user with the requested objects. In Figure 5.11, the mobile user chooses Hlaing township and service type for the bank to get top 10 nearest objects from the current location. The system responds to the client user according to the user input types. The results of location objects are ordered by the nearest distance and displayed to the user. The location objects are ordered by nearest distance as shown in Table 5.. The detail result of nearest object from the query point is shown in Figure 5.12.

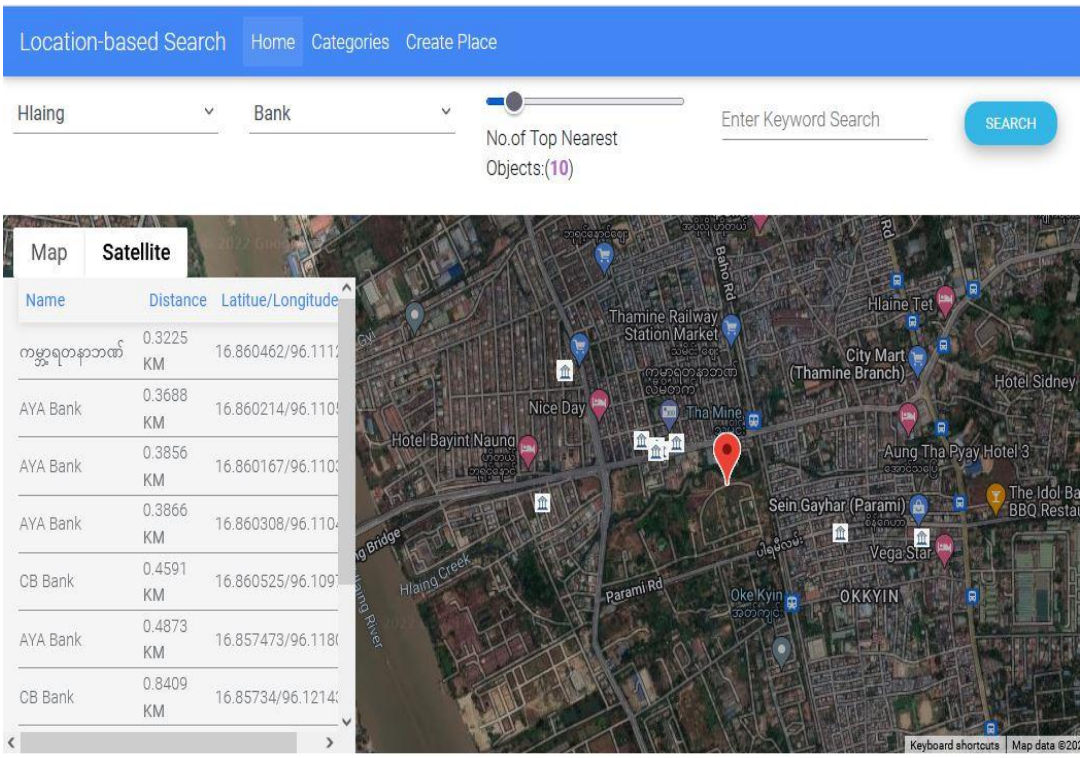


Figure 5.11 kNN Search Results of Objects

Table 5.4 Sorting the Objects by Nearest Distance

Name	Distance (Kilometer)	Latitue/Longitude
ကမ္ဘာ့ရတနာဘဏ်	0.3338	16.860462/96.111244
AYA Bank	0.3814	16.860214/96.110573
AYA Bank	0.3983	16.860167/96.110374
AYA Bank	0.3991	16.860308/96.110435
CB Bank	0.4717	16.860525/96.109787
AYA Bank	0.4739	16.857473/96.118057
CB Bank	0.8273	16.85734/96.121437
KBZ Bank	0.8690	16.858458/96.105637
UAB Bank	0.8913	16.862776/96.106575
မြန်မာ့နိုင်ငံသားဘဏ်	0.8981	16.862885/96.1065

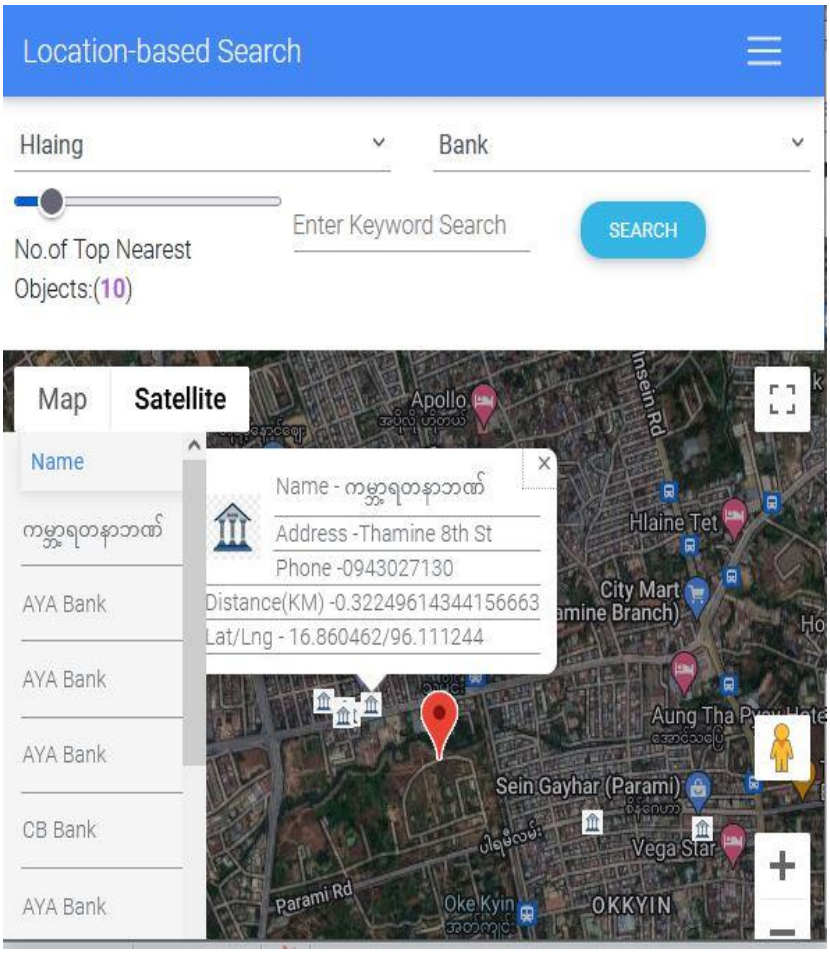


Figure 5.12 Detail Result for kNN Query

5.3.2 Sorting The Results for kNN Query

In the kNN query, the system responds query results to the clients by sorting the nearest objects from the query point. The results are the top nearest objects from the client user's position. Each object has the distance value from the query point's location. Figure 5.13 shows the results which are sorted by calculating distance values from the query point to the location objects respectively. The objects are ranked by ascending nearby distance. There are 15 location objects which are different type of services as shown in Table 5.5.

Table 5.5 Ranking Location Objects by The Nearest Distance

No.	Name	Distance (km)	Latitude/Longitude	Service Type
1	Embassy of Israel	0.062	16.785723, 96.142174	Embassy
2	Embassy of Russia	0.130	16.785822, 96.143677	Embassy
3	Ni Ni	0.350	16.782154, 96.142838	Fashion
4	Phyu Sin	0.354	16.782158, 96.141975	Dentist
5	Shin Shin	0.417	16.782293, 96.140221	Computer Services
6	Naing Myanmar	0.451	16.782654, 96.139359	Car Sale & Service
7	Embassy of Malaysia	0.503	16.789263, 96.140305	Embassy
8	Dr. Lin Lin	0.508	16.780855, 96.143738	Clinic
9	Embassy of Indonesia	0.549	16.790203, 96.142014	Embassy
10	Embassy of France	0.556	16.790295, 96.142334	Embassy
11	Min Min	0.562	16.78281, 96.137978	Beer Station
12	Zwe Naing	0.579	16.780083, 96.142799	Mini-mart
13	Zaw Naing	0.582	16.780123, 96.141678	Mobile Shop
14	Myo Taw Mobile	0.607	16.780094, 96.144333	Mobile Shop
15	Feel Myanmar Restaurant	0.609	16.790665, 96.143707	Restaurant

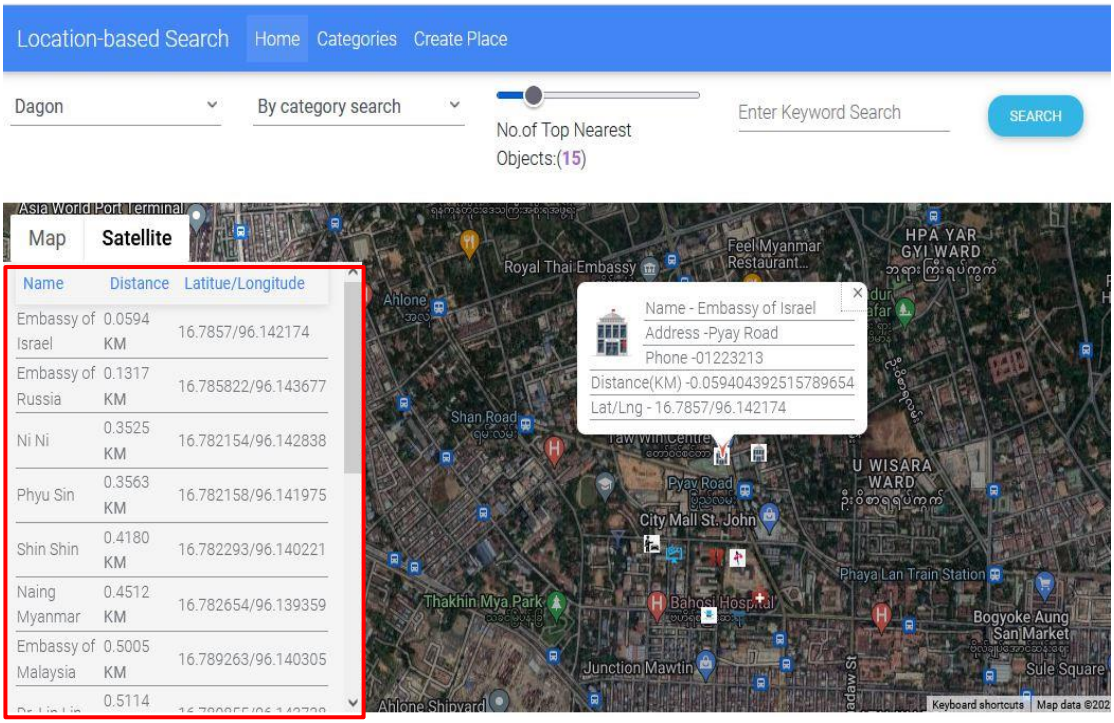


Figure 5.13 Sorting the Nearest Objects with Different Service Types

5.3.3 Range Query Result of The Client User

In range query, the client user needs to define the search radius value (for example 2 kilometers) from the current position as shown initial point (red marker). The systems compute the distance value of the given distance value from the current position. Then, the client user can choose the service type and can search the location objects. This system responds to the user the location objects according to the specified distance value. Figure 5.14 shows the rage query results that are user defined by service type (i.e clinic) and distance (i.e 2km). As the distance search, the location objects are ordered by distance values that show in Table 5.6. The located a short distance away from query point is shown in Figure 5.15 with details.

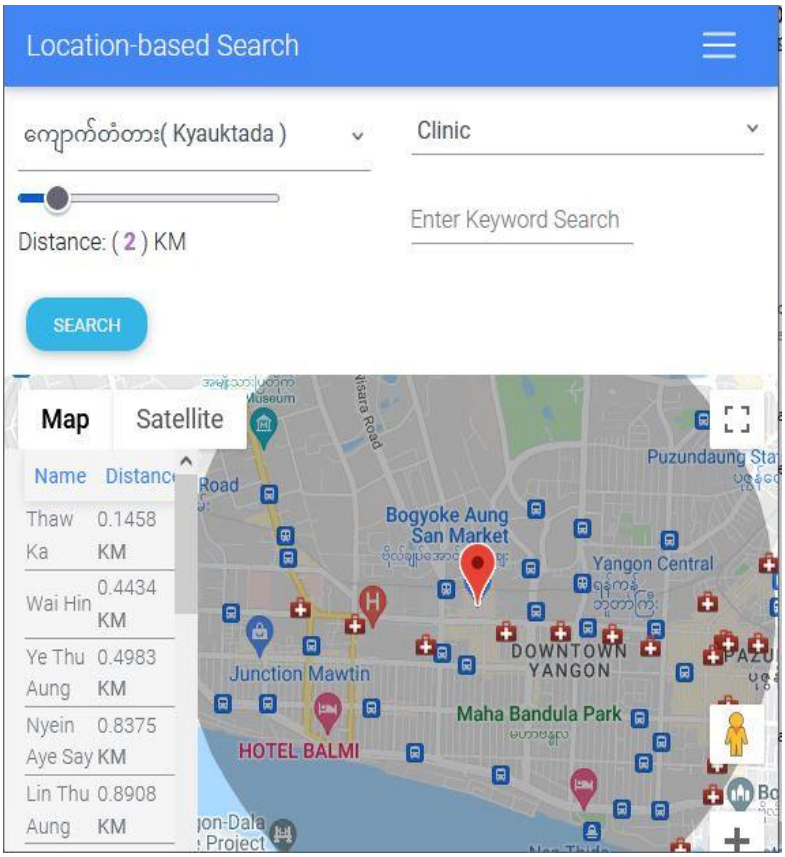


Figure 5.14 Range Search Results for the Objects

Table 5.6 Ranking Location Objects by Distance Values

Name	Distance (KM)	Latitude/Longitude
Thaw Ka	0.3744	16.779572/96.1567
Wai Hin	0.3811	16.778858/96.15168
Ye Thu Aung	0.7078	16.779564/96.160179
Lin Thu Aung	0.7315	16.779982/96.147202
Nyein Aye Say	1.0399	16.779423/96.163376
Dr. Lin Lin	1.0815	16.780855/96.143738
Kaung Zay	1.3146	16.778727/96.165855
Win Clinic	1.6560	16.78117/96.169418
Ye Naing Clinic	1.7321	16.778267/96.169769
Moe Kaung	1.7854	16.779057/96.170433
Yaung Ni Oo	1.8167	16.779007/96.170723
Si Thu	1.9209	16.775421/96.170731

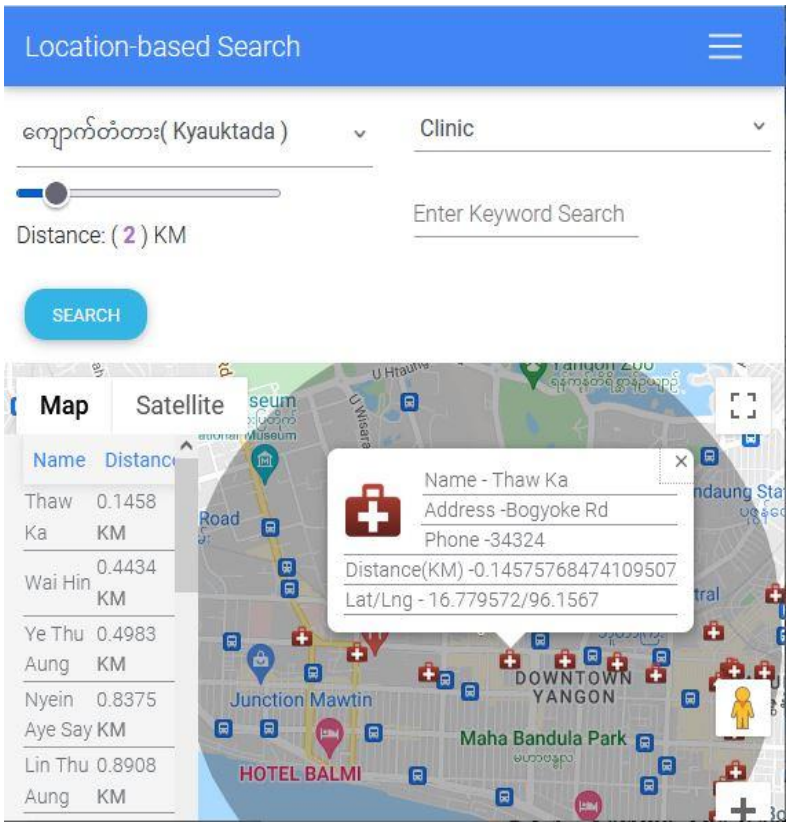


Figure 5.15 Detail Result for Range Query

5.3.4 Sorting The Results for Range Query

In the range query, the system responds to clients the query results by sorting distance for each object from the query point. Each object has the distance value from the query point's location that is shown in Table 5.7. Figure 5.16 shows the range query results which are sorted by calculating distance values from the query point to the target point of each location object. The objects are ranked by ascending nearby objects. In the figure, there are nine location objects as the results of the range search which are the distance from the current location position. These nine location objects are restaurants which are the user choice of services.

Table 5.7 Range Query Result by Ordering Distance

No.	Name	Distance	Latitude/ Longitude	Service Type
1	ကြေးအိုးဘုရင်(Chinese)	0.285	16.77825, 96.155151	Restaurant
2	နယူးဒေလီစားတော်ဆက်	0.293	16.777117, 96.155624	Restaurant
3	She Let Yar	0.324	16.778702, 96.155266	Restaurant
4	Thin Thin Aung	0.330	16.777805, 96.155830	Restaurant
5	Shoda Sushi	0.442	16.773624, 96.150505	Restaurant
6	Wuhan duck neck	0.446	16.774181, 96.14978	Restaurant
7	Pyone Pan	0.561	16.778416, 96.157906	Restaurant
8	Kaung Mon	0.746	16.778572, 96.159668	Restaurant
9	Way Way	0.860	16.77841, 96.160805	Restaurant

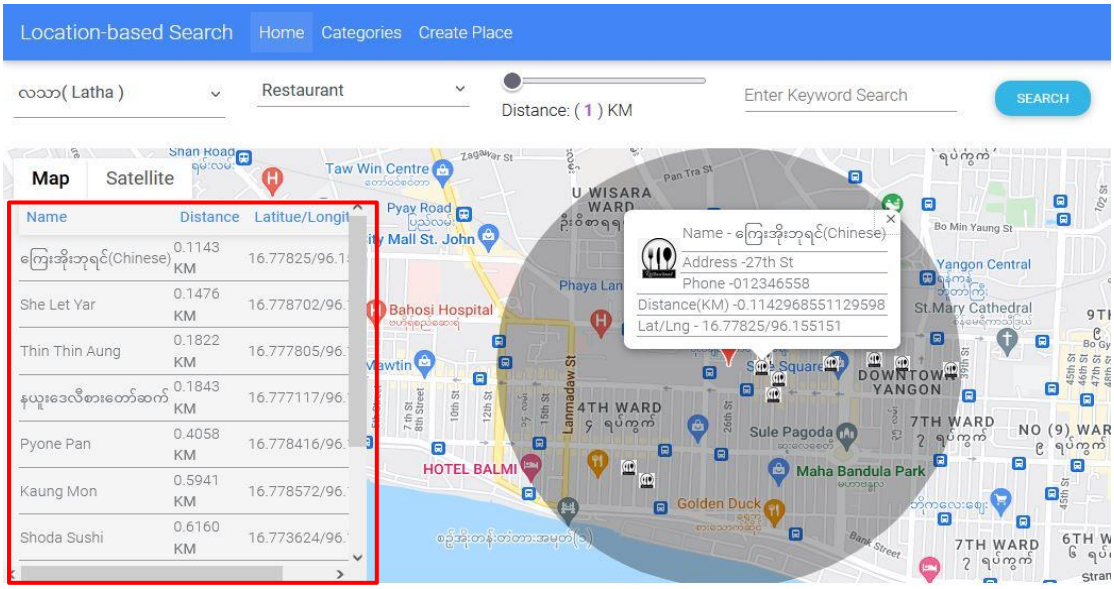


Figure 5.16 Sorting the Objects in Distance Calculation

5.3.5 Creating Places for Client Users

The users can create the places of services by using this system. This system provides user-generated content for the client users. Most of the users desired to advertise their business services on the map. In addition, mobile users desire to check in to places where they are visited. To perform creating a new place, at first, the user needs to choose township and type of category and then click on the map where the place to publish. The latitude and the longitude values are automatically generated in the lat/long value box respectively. Then name, address, and phone number are required to fill in the input text boxes. Finally, the user can create the geolocation place. Figure 5.17 shows a new place creation by using the system.

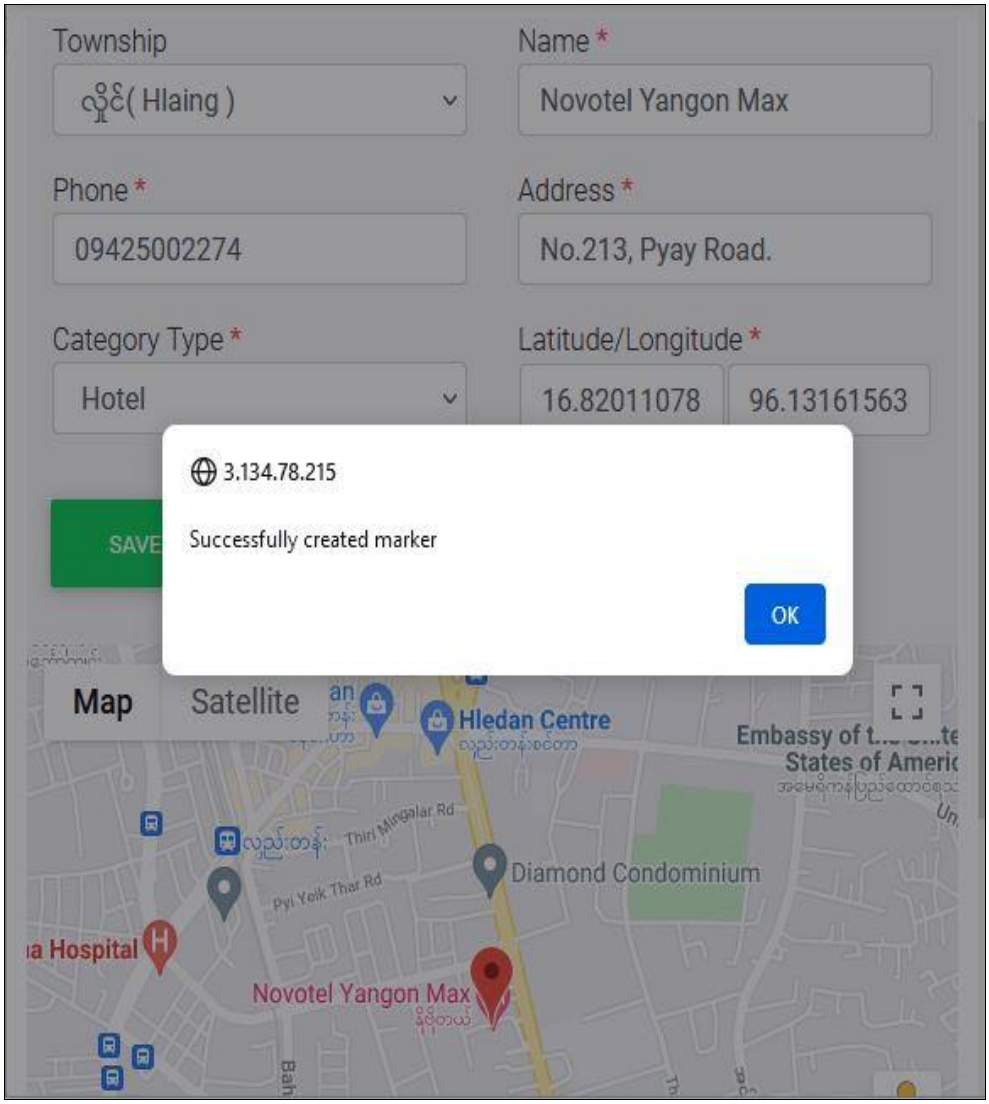


Figure 5.17 Create A Place of Service

In addition, the users can create the category of service in the system. The user requires giving a new type of service name and choosing the photo of the service. Then the user can successfully create a new category in the system. After creating a new category of service, the other users can advise the services by choosing this new service in the category column. Figure 5.18 shows how to create a new category of service. Moreover, the created category can be edited by the users. The users can edit the name of the category and can change the picture of the service by browsing the picture box as shown in Figure 5.19.

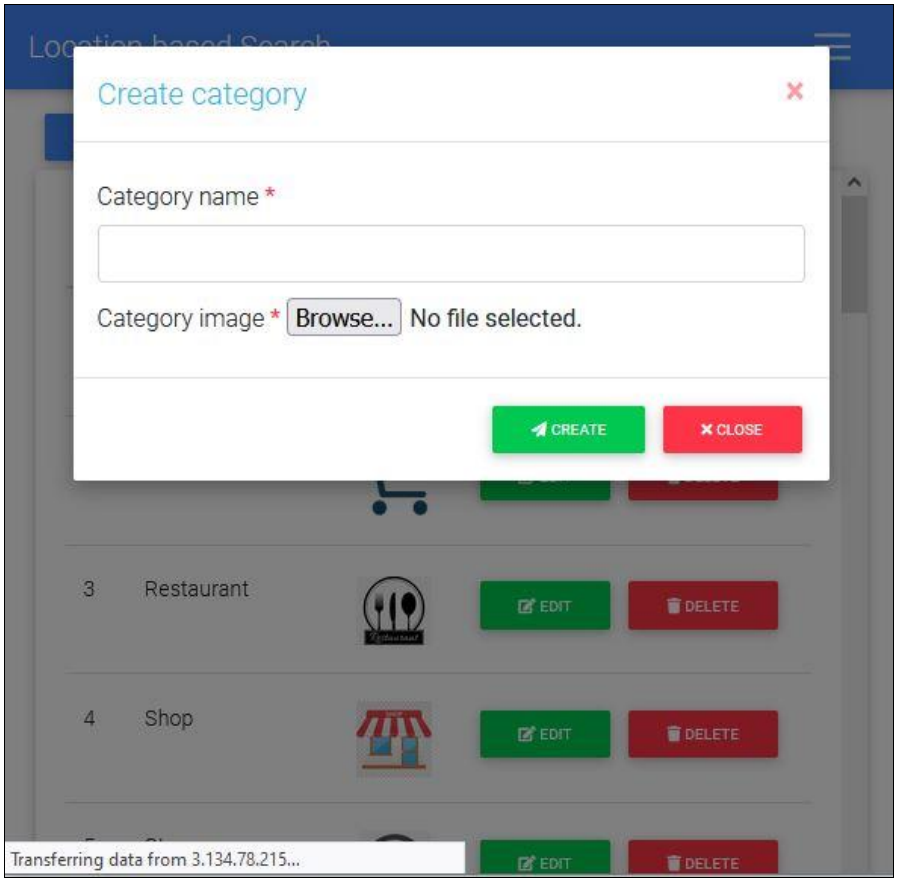


Figure 5.18 Create New Category of Service

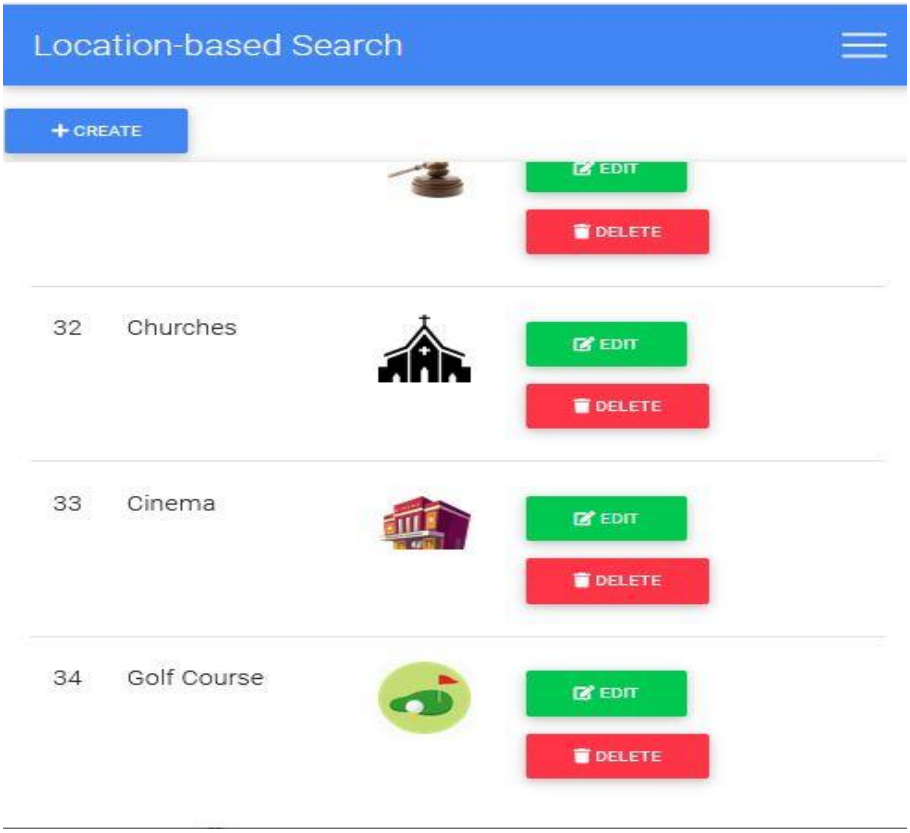


Figure 5.19 Edit the Category

5.4 Evaluation Results

This section presents the evaluation results of the proposed system. Mainly, the proposed system used combining index structures which are grid indexing and R-tree indexing. In the system implementation, the proposed system scheme is approved for the computing time compared with the original R-tree structure. The accuracy measuring on the nearest neighbour search is also shown in the section.

5.4.1 Evaluation of The Processing Time

The evaluation of the proposed system is considered with the server processing time. The proposed index technique is the grid-based R-tree index structure. Grid index is easy to perform multi-dimensional queries. Range query with the grid index can search all objects that are geographically located in a certain area. Furthermore, R-tree can also organize any dimensional data by representing the bounding box. However, R-trees can generate a lot of overlap and coverage between the minimum bounding rectangles. Therefore, the proposed system is designed with indexing techniques that are R-tree and grid index.

In Table 5.7, the processing time is shown as the particular object of computing time. On the top, it will be shown the whole execution time for retrieving objects. The processing time is starting to compute until the whole process including (the result is sent back to the user) is done.

$$Time_{total} = time_{end} - time_{start} \quad 5.1$$

where,

$Time_{total}$: processing time

$time_{start}$: starting time

$time_{end}$: end time in which the whole process is completed

The processing time is better than the traditional R-tree indexing scheming. Figure 5.20 shows the chart of processing time.

Table 5.7 Processing Time Comparison of Indexing Schemes

Service Type	R-Tree Indexing	Proposed Indexing
Clinic	65ms	28ms
Store	87ms	21ms
Shop	77ms	20ms
Mini-mart	84ms	10ms
Restaurant	125ms	24ms

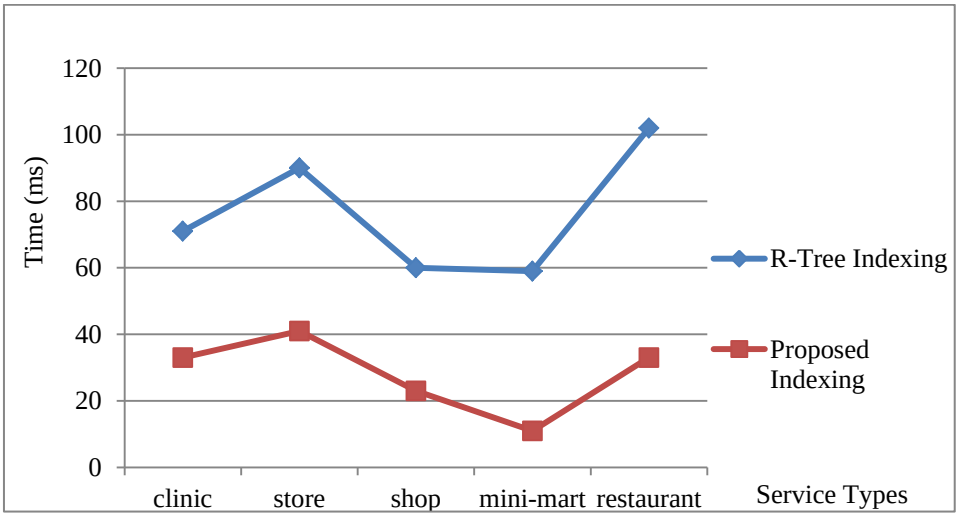


Figure 5.20 Processing Time of Different Indexing Schemes

5.4.2 Accuracy Measurement of kNN Value

The accuracy results are tested based on the number of k values to approve kNN query. The accuracy rate of the proposed system is stable in these tested times. The measurement of accuracy is calculated the total number of k values in the classification. The k value is used to calculate the nearest objects that are the predicated values. The performance of kNN depends on the measurement of determining the value of distance calculation. Figure 5.21 shows the accuracy testing of indexing schemes.

$$T_{acuracy} = (TP + TN) / N \quad 5.2$$

where,

$T_{acuracy}$ = Total number of correctness in the classification process

TP = Positive data detected correctly

TN = Negative data detected correctly

N = Number of k values

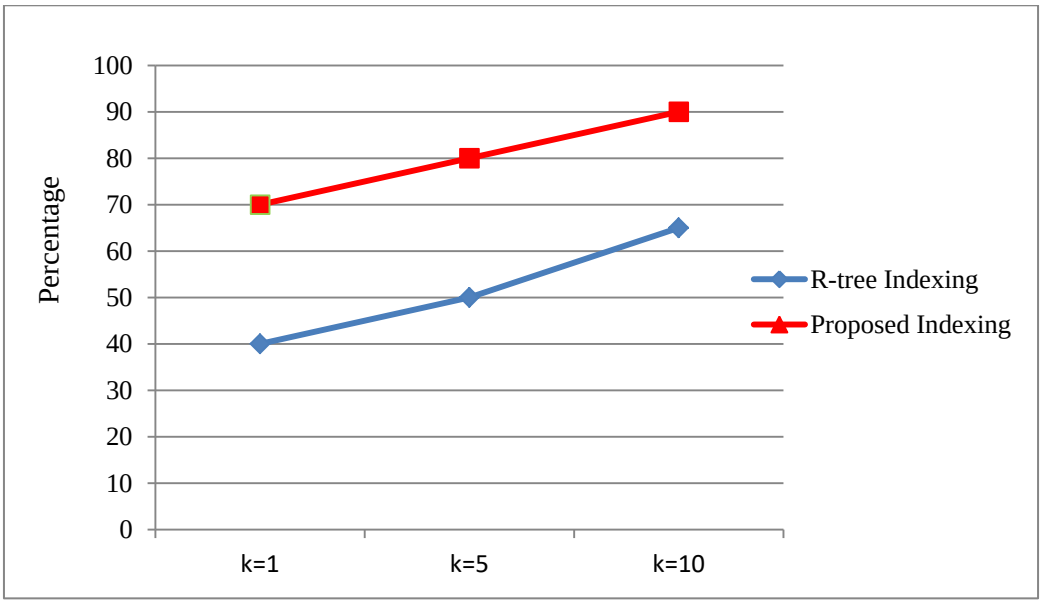


Figure 5.21 Accuracy Testing of kNN Query

5.5 Summary

To evaluate the performance of the proposed indexing scheme, the processing time is computed. The performance evaluations of the proposed system are cover processing time and inaccuracy measurement efficiently. Moreover, the comparisons on evaluation results show the proposed indexing scheme with the traditional R-tree indexing scheme.

CHAPTER 6
CONCLUSION AND FURTHER EXTENSION

This research focuses primarily on range queries and k-nearest neighbor (kNN) queries for searching objects that are geographically located data. To fully query the data, the system implements a spatial index structure, which is a grid-based R-tree indexing method. The grid index structure and the R-tree index structure are multidimensional indexes that can support to retrieval of spatial data efficiently. R-tree can search overlapping and coverage between minimum bounding rectangles (MBR). However, Updating and creating the index might be a slowdown. Therefore, the combination of R-tree and grid index is used. The grid index is easy in implementations of updating and creating the index. The user interface design of the proposed system is designed for the client's users easy to use. The mobile users can search location objects from the specified location.

6.1 Conclusion

The system provides spatial keyword search for range query and kNN query for location objects. K-nearest neighbor queries get the object closest to the location of the query point. This proposed system can be a good performance in spatial keyword search. The proposed system can reduce server processing time and the computing process.

The system provides keyword search and category-based search for spatial objects. The users can choose search type based on location position and then users need to choose kNN query or range query. The system responds to the results according to the user input. kNN query will retrieve *k* objects which are the nearest objects from the query point location. Range query will retrieve objects that are chosen by distance from the query point. The system developed a hybrid platform for android and windows. The system created a responsive view for any device. Therefore, users can use the system ease to user friendly.

The system uses geo-location data of townships in the Yangon Region. These data are the places of interest that users often search for entertainment, health care, education, transportation, emergency, and other services. The proposed index structure effectively stores information so that users can access it quickly and

accurately. Users can search for not only nearby services but also range search by selecting their preferred location. This system uses google maps API to capture the user's desired location. The proposed index structure has been constructed to accurately answer the queries from this location, and the kNN query and range query have been implemented to answer the nearest location data.

This system is built for mobile applications and web applications for client users to make it easy to use. Users need an internet connection to use this system because Google's map API service and cloud database system are built into the system. The system provides nearly 10,000 data points for 33 townships in Yangon Region. To search for location information services, users can search for nearby services and surrounding areas of services by selecting townships or marking a preferred location. For the kNN search, the system responds to the user by locating objects that are the closest to the farthest from the user's location. For the range search, the system responds to the user by displaying information within the desired distance from the user's location.

6.2 Advantages and Limitations

The system can provide the users with location information about what kind of places of services exist nearby and can support getting the places of users' interest within the particular distance. The system is designed to be convenient for the client users. The benefits of using index structure can reduce the processing time while querying to the server. The system applied kNN query and range query which are the most common type of searches of the client users. Moreover, the system provides the users to generate the geo-location contents at any location position in Yangon Region. The system also enables the client user to create the category of services.

The limitation of the system is that the proposed system cannot use the offline map. The system has only used the Google Maps API service. The spatial database also uses cloud storage. For this reason, the client devices need an internet connection. Mobile devices may have the capacity for memory usage while using the system. R-tree limits depend on how objects are inserted and deleted from the tree. This may need to go through some rectangles to find the object.

6.3 Further Work

As the further work in the same field, an extension would provide the optimal route displaying on the road network to the nearest spatial objects. In the extension, the system needs to consider the road network and to support optimal route finding to the destination. The spatial database is required to create the road network database by using appropriate algorithms. As a further extension, the system can develop for offline map. In addition, the research work would consider continuous nearest neighbor query retrieving the nearest neighbour (NN) of every point on the line segment. It is intended for moving forward vehicles on the road network.

AUTHOR'S PUBLICATIONS

[P1] Aung Zaw Myint and Khin Mo Mo Tun, University of Computer Studies, Yangon Myanmar, "**Keyword Search in Web-based Geographic Informational Retrieval System**" in Proceedings of the 15th International Conference on Computer Applications (ICCA2017), Yangon, Myanmar, pp. 262-265, February 16^{th}-17^{th} , 2017.

[P2] Aung Zaw Myint and Khin Mo Mo Tun, University of Computer Studies, Yangon Myanmar, "**An Index Structure for Nearest Neighbor Search of User Preference Services on Location-based Service "** in Proceedings of the 1st International Conference Science and Technology Development (CSTD2017), Pyin Oo Lwin, Myanmar, pp. 227-230, November 1-2, 2017.

[P3] Aung Zaw Myint and Khin Mo Mo Tun, University of Computer Studies, Yangon Myanmar, "**Location-based Spatial Keyword Search on KNN Query and Range Query For User's Preference Services "** in Proceedings of the 33rd International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC 2018), Bangkok, Thailand, pp. 365-368, July 4-7, 2018.

[P4] Aung Zaw Myint and Khin Mo Mo Tun, University of Computer Studies, Yangon Myanmar, "**Grid-based Spatial Index Method for Location-based Nearest Neighbour Search"** in Proceedings of the 11th International Conference on Future Computer and Communication (ICFCC 2019), Yangon, Myanmar, pp. 178-183, Feb 27- March 1, 2019.

[P5] Aung Zaw Myint and Khin Mo Mo Tun, University of Computer Studies, Yangon Myanmar, "**Grid-based Spatial Index Method for Location-based Nearest Neighbour Search"** in the International Journal of Future Computer and Communication (IJFCC), IJFCC 2020, Volume 9(2), pp. 40-45, Jun 2020.

BIBLIOGRAPHY

[1] A.Rifaat. " Mobile GIS and Location-Based Services (LBS)" In Introduction to Geospatial Information and Communication Technology (GeoICT). Springer, Cham. 2016. https:// doi.org / 10.1007/978-3-319-33603-9_5

[2] Steiniger, S., Neun, M., and Edwardes, A. "Foundations of Location-Based Services". <http:// www.e-cartouche.ch>, October 2006.

[3] C. Pontikakos, M. Sambrakos, T. Glezako,T. Tsiligiridis, "Location-based services: A framework for an architecture design" Computer Science, 2010. https://www.semanticscholar.or.

[4] V.Subodh and J.B.Christopher. "Spatially-Aware Information Retrieval on the Internet". In Conference Proceedings Fu2003 Spatially AwareR, 2003.

[5] Seydim, A.Y., Dunham, M.H., and Kumar, V. . "Location Dependent Query Processing". Proceedings of the 2nd ACM International Workshop on Data Engineering for wireless and Mobile Access, Vol.1: pp. 47-53, 2001.

[6] Ilarri, S., Mena, E., and Illarramendi, A. . "Location-dependent Query Processing: Where We Are and Where We Are Heading". ACM Computing Survey (CSUR), Vol.42, No.12, 2010.

[7] Rajachandrasekar, R., Ali, Z., Hegde, S., Meshram, V., and Dandapantula, N. "Location-Based Query processing: Sensing Our Surroundings". Department of Computer Science and Engineering, The Ohio State University, 2011.

[8] R.Philippe, S.Michel Scholl, and V.Agnes. . "Spatial Databases: With Application to GIS". The Morgan Kaufmann Series in Data Management Systems, ISSN 1046-1698, 2002.

[9] Roussopoulos, N., Kelley, S., and Vincent, F.. "Nearest Neighbor Queries". Proceedings of the ACM SIGMOD international conference on Management of data, June 1995, pp.71–79.

https://doi.org/10.1145/223784.223794

[10] King Lun, C., and Ada Wai-Chee, F . “Enhanced Nearest Neighbour Search on the R-tree”. SIGMOD Record, Vol. 27, No. 3: 16-21.

[11] Hjaltason, R., and Samet, H. 1999. “Distance Browsing in Spatial Databases”. ACM Transactions in Database Systems, Vol. 2, pp. 265-318, 1998.

[12] I. F. Ilyas, G. Beskales, and M. A. Soliman. “A survey of top-k query processing techniques in relational database systems”. ACM Computing Surveys (CSUR), Vol. 40(4), pp 1-58, October 2008. https://doi.org/10.1145/1391729.1391730

[13] I. De Felipe, V. Hristidis and N. Rishe, "Keyword Search on Spatial Databases," 2008 IEEE 24th International Conference on Data Engineering, 2008, pp. 656-665, doi: 10.1109/ICDE.2008.4497474.

[14] G. Cong, C. S. Jensen, and D. Wu. “Efficient Retrieval of the Top k Most Relevant Spatial Web Objects”. Proceedings of the VLDB, Vol. 2, pp 337–348, August 2009.

[15] X. Cao, G. Cong, and C. S. Jensen. “Retrieving top-k prestige-based relevant spatial web objects”. Proceedings of the VLDB, Vol. 3, pp 373–384, 2010.

[16] L. Chen, G. Cong, C. S. Jensen, and D. Wu. “Spatial keyword query processing: an experimental evaluation”. Proceedings of the VLDB, Vol.6, pp 217–228, 2013.

[17] D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung and M. Kitsuregawa, "Keyword Search in Spatial Databases: Towards Searching by Document," IEEE 25th International Conference on Data Engineering, pp. 688-699, 2009.

[18] Kai Zheng, Bolong Zheng∗, Jiajie Xu, Guanfeng Liu, An Liu, Zhixu Li. “Popularity-aware Spatial Keyword Search on Activity Trajectories.” World Wide Web Journal (WWWJ), Vol. 20, pp 749-773, July 2017. https://doi.org/10.1007/s11280-016-0414-0

[19] K. Park. “Location-based grid index for spatial query processing.” Expert System with Applications, Vol. 41, pp 1294-1300, 2016. https://doi.org/10.1016/j.eswa.2013.08.027

[20] D. Šidlauskas, S. Šaltenis, W. Christiansen and M. Johansen. “ Trees or grids?: indexing moving objects in main memory”. Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 236–245, 2009. https://doi.org/10.1145/1653771.1653805

[21] Frentzos, E., Gratsias, K., Pelekis, N., and Theodoridis, Y. “Algorithms for Nearest Neighbor Search on Moving Object Trajectories”.University of Piraeus, Piraeus, Greece, Geoinformatica , Vol. 11, pp. 159–193, 2007. https://doi.org/10.1007/s10707-006-0007-7

[22] W. Wu and K. Tan, "iSEE: Efficient Continuous K-Nearest-Neighbor Monitoring over Moving Objects" 19th International Conference on Scientific and Statistical Database Management (SSDBM 2007), pp. 36-46, 2007. doi: 10.1109/SSDBM.2007.37.

[23] Wei, Z., Jianzhong, L., and Haiwei, P . “Processing Continuous k-Nearest Neighbor Queries in Location-Dependent Application”. International Journal of Computer Science and Network Security, Vol.6, No.3, pp.1-9, 2006.

[24] Huang, X., S. Jensen, C., and Saltenis, S. . “Multiple k Nearest Neighbor Query Processing in Spatial Network Databases”. Journal of ADBIS, Vol.2, pp. 266-281, 2006.

[25] Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.. “k Nearest Neighbor Search for Location-Dependent Sensor Data in MANETs”. IEEE Journal of Industrial Sensor Networks with Advanced Data Management: Design and Security, Vol. 3, pp. 942-954, 2015.

[26] Yadhav, A., and Rajalakshmi, R.. “Nearest Neighbor Query in Location-Aware Mobile Ad-Hoc Network”. International Journal of Computer Science and Mobile Computing, Vol. 4, No. 3: pp. 51-55, 2015.

[27] M. A. M. Abd Elwahab, K. M. Mahar, H. Abdelkader and H. A. Khater, "Combining R-Tree and B-Tree to Enhance Spatial Queries Processing" 23rd International Conference on Computer Theory and Applications (ICCTA), pp.184-190, 2013. doi: 10.1109/ ICCTA

32607.2013.9529800

[28] Boucetta, S., Daman, D., and Shaik, S.. “Intelligent Selection Technique for Database Indexing To Augment The Speed Performance of Query Processing on Mobile Device”. Life Science Journal, Vol.11, No.4, pp. 239-245, 2014.

[29] W. Lee and B. Zheng, "DSI: A Fully Distributed Spatial Index for Location-Based Wireless Broadcast Services," 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05), pp. 349-358, 2005. doi: 10.1109/ICDCS.2005.26.

[30] Sungwon, J., and Hyunho, M.“A Spatial Indexing Scheme for Location-Based Service Queries in a Single Wireless Broadcast Channel”. Journal of Information Science and Engineering, Vol. 30, pp. 1945-1963, 2014.

[31] Kwangjin, P. “Location-Based Grid-Index for Spatial Query Processing”, Journal of Elsevier, Vol. 41: pp. 1294–1300, 2014.

[32] Kwangjin, P., and Valduriez, P. “A Hierarchical Grid Index (HGI), Spatial Queries in Wireless Data Broadcasting”. Distributed and Parallel Databases, vol.31 pp. 413–446, 2013. https://doi.org/ 10.1007/s10619-013-7121-y

[33] Baihua, Z., Jianliang, X., Wang-Chien, L., and Dik Lun, L . “Grid-Partition Index: A Hybrid Method for Nearest-Neighbor Queries in Wireless Location-Based Services”. The VLDB Journal, Vol.15, No.1, pp. 21-39, 2006.

[34] Slimani, H., Najjar, F., and Slimani, Y. “Voronoi-Neighboring Regions Tree for Efficient Processing of Location Dependent Queries”. International Journal of Advanced Science and Technology, Vol. 33, pp. 101-119, 2011.

[35] Baihua, Z., Jianliang, X., Wang-Chien, L., and Dik Lun, L. “Energy-Conserving Air Indexes for Nearest Neighbor Search”. Lecture Notes in Computer Science 2992, pp.48-66, 2004.

[36] Kollios, G., Papadopoulos, D., Gunopulos, D., and Tsotras, J. “Indexing Mobile Objects Using Dual Transformations”. The journal of VLDB, Vol.14, pp. 238-256, 2005.

[37] Pfoser, D., S. Jensen, C., and Theodoridis, Y. “Novel Approaches to the Indexing of Moving Object Trajectories”. In proceedings of the 26th VLDB Conference, Cairo, Egypt, pp. 395-406, 2000.

[38] Yon-Gui, Z., Song, Q., and Fu-Ping, Y. “ An Query Processing for Continuous K-Nearest Neighbor Based on R-Tree and Quad Tree”. Chongqing University of Posts and Telecommunications, Chongqing, China, pp. 35-40, doi: 10.1109/ICCSIT.2010.5563984, 2010.

[39] Mazharuddin, A., Ijtihadie, R., Giovanni, I., Studiawan, H., Wibisono, W., and Tohari A. No Date. “Location-Based Agenda using GPS with K-Nearest Neighbor Algorithm Context Matching”. Department of Informatics, Faculty of Information Technology, Jl. Arief Rahman Hakim, Surabaya, 2013.

[40] Volker, G., and Gunther, O. “Multidimensional Access Method”. ACM Computing Surveys, Vol.30, No.2, pp. 170–231, 1998. https://doi.org/10.1145/280277.280279

[41] Jianliang, X., Baihau, Z., Wang-Chien, L., and Dik Lun, L. “ Energy Efficient Index for Querying Location-Dependent Data in Mobile Broadcast Environments ”. International Conference on Data Engineering, pp. 239-250, March 2003. DOI: 10.1109/ ICDE. 2003.1260796

[42] Jianliang, X., Baihau, Z., Wang-Chien, L., and Dik Lun, L. “The D-tree: An Index Structure for Planar Point Queries in Location-based Wireless Services”. In IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 12, pp. 1526-1542, Dec. 2004.

[43] Guttman, A. “ R-Trees: A Dynamic Index Structure for Spatial Searching”. University of California, Vol. 4, pp. 47-57, 1984. https://doi.org/10.1145/971697.602266

[44] Sebastian.K. , Thomas B. “Algorithms and Data structures for Database Systems” University of Passau,Germany, 2003.

[45] Guobin, L., and Jin’e, T. No Date. “A New K-nearest Neighbor Query Algorithm based on Grid Hierarchical Division”. International Conference on Computer Science and Information

Technology, Vol. 6, pp. 255-257, 2010.

ACRONYMS

2D	Two-dimensional
ABL	Active Branch List
API	Application Programming Interface
ATM	Automatic Teller Machine
AWS	Amazon Web Services
BB	Bounding-Box
CKNN	Continuous kNN Nearest Neighbour
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DBMS	Database Management System
DSI	Distributed Spatial Index
EC2	Elastic Compute Cloud
ER	Entity-Relationship
GIS	Geographical Information System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HGI	Hierarchical Grid Index
HMI	Hilbert curve-based MBR filtering Index
HTML	Hyper Text Makeup Language
ID	Identification
IR2	Information Retrieval R
KD	K-Dimensional
KNN	k-Nearest Neighbour
LBS	Location-Based Services
LkPT	Location-aware top-k Prestige-based text
MBR	Minimum Bounding Rectangle
mCK	m-closet keyword
NE	Northeast
NW	Northwest

PDO	PHP Data Objects
POI	Point Of Interest
RAM	Random Access Memory
RDBMS	Relational Database Management System
RFID	Radio Frequency Identification
SE	Southeast
SDK	Software Development Kit
STR	Spatio-Temporal R-tree
SW	Southwest
SQL	Structured Query Language
UGC	User-Generated Content
VD	Voronoi Diagram
VNR	Voronoi-Neighbouring Regions