International Journal of Electrical and Computer Engineering (IJECE)
Vol. 9, No. 4, August 2019, pp. 2556~2563
ISSN: 2088-8708, DOI: 10.11591/ijece.v9i4.pp2556-2563

Mobile location indexing based on synthetic moving objects

Thu Thu Zan, Sabai Phyu
University of Computer Studies, Myanmar

Article Info

Article history:

Received Jun 20, 2018
Revised Dec 20, 2018
Accepted Mar 4, 2019

Keywords:

Indexing
Mobile locations
Moving object
Nearest neighbor
Presorting
Range queries

ABSTRACT

Today, the number of researches based on the data they move known as mobile objects indexing came out from the traditional static one. There are some indexing approaches to handle the complicated moving positions. One of the suitable ideas is pre-ordering these objects before building index structure. In this paper, a structure, a presorted-nearest index tree algorithm is proposed that allowed maintaining, updating, and range querying mobile objects within the desired period. Besides, it gives the advantage of an index structure to easy data access and fast query along with the retrieving nearest locations from a location point in the index structure. A synthetic mobile position dataset is also proposed for performance evaluation so that it is free from location privacy and confidentiality. The detail experimental results are discussed together with the performance evaluation of KDtree-based index structure. Both approaches are similarly efficient in range searching. However, the proposed approach is especially much more save time for the nearest neighbor search within a range than KD tree-based calculation.

Corresponding Author:

Thu Thu Zan,
University of Computer Studies,
No.129, Zewaka Hostel, Hlaing Compus, Yangon, Myanmar.
Email: thuthuzan@ucsy.edu.mm

1. INTRODUCTION

With advances in location-based services (LBSs), mobile devices and telecommunication systems are the vital support for the positioning and tracking of moving mobile objects [1]. Besides, the increasing numbers of mobile users are already emerged along with the improvement of new technologies. So, using the mobile phone is the main function on a daily requirement with the user's search queries. Most of the required queries or information is usually based on current or anticipated locations. For example, searching nearest restaurants, viewing the route and inquiry required information are the user required queries which are supported by pop technology [2]. This technology normally based on static data or locations. In addition, the services such as receiving weather information, emergency alerts, advertisement getting access to what's known as automatic notifications are aided by push technology [3]. This service is usually based on moving positions or current mobile positions. In such environments, mobile devices regularly send their current locations to a server. The server receives the current locations and processes desired queries such as "which mobiles are currently located within the desired area". To reply such queries appropriately, the application server has to search all of the current mobile locations that are in the desired area. Therefore, an appropriate structure and searching method for the nearest are required.

Generally, the nearest searching technique can be divided into two parts: structureless technique and structure-based technique. For example, a well-known technique such as k the nearest neighbor is a structureless technique and it is very easy to implement. The general work of structureless technique is that distance is calculated from all nodes to the service node of a query and the node with the closest distance is regarded as the nearest neighbor [4-5]. These techniques are very simple but the value of k affects the result. To provide the speed of query and memory requirement, a variety of tree-based index structures called structure-based techniques are applied in many areas.

In structure-based techniques, most of the indexing structures and nearest neighbor searches are separately taken by two parts: building index structure and searching nearest neighbor of the desired query point p. In the building of a tree also includes relevant structure along with collecting and storing data points. Then, an appropriate procedure or algorithm is used to search the nearest points from the desired query point [6]. It is totally good for static or non-moving objects structure but it may issue in moving locations or objects indexing. Moving object index structure with the regular update will also need to cooperate updating of the nearest points searching algorithm. It may think consistency between index structure and searching algorithm for discretionary data points query and search will be required concurrency control.

Recent advances in moving objects based research and applications, the positioning technologies and indexing based structures have collaborated together thus it leads to a perfect technique for maintaining and updating mobile position in the dynamic environment. Normally, the purpose of the traditional index structure is generating queries faster and easier with the required information. However, a good mobile objects' indexing needs not the only faster query but also ability to update regularly. In this paper, the presorted-nearest index tree structure is proposed that combines the concept of the nearest neighbor algorithm and tree-based index structure. Thus, it supports generating nearest locations by index structure from the desired query point. In this structure, all of the location nodes are placed by level order thus nodes at any distance can easily find without traveling the whole tree and the searching may reduce greatly. It is harmonized to solve nearest neighbor location queries since the locations of the data points are based only on their relative distances from each other.

In reality, the millions of mobile positioning data are unavailable for performance evaluation of index structure. If the mobile location dataset can create synthetically, it would be free from location privacy and confidentiality. It can be used to get location data for performance evaluation of proposed index structure. Thus, the requirement of generating a synthetic dataset for dynamic mobile locations that seem realistic is a challenge. Therefore, proposing a novel procedure to produce the synthetic dataset which is based on dynamic two-dimensional points is included in this paper. According to environmental needs, many datasets are generated synthetically such as animal, mobile user, bus, Hurricane dataset, and so on. A synthetic dataset usually comes from application-dependent generation. Therefore, a dataset generator is required about their domain of interest before making any dataset synthetically.

This paper explains how to process dynamic location attributes in the proposed index structure and an appropriate process flow of mobile object generator is built to deal with the overall system. As mobile positions are basically used in this system so that it requires continuous evaluation as the location result needs to valid after a suitable period of time. When the use of indexing based on location or position, the three basic positions are normally analyzed on historical, current and anticipated future position. This system takes the second one which is based on the current position thus it has to get its location with the appropriate update will be there. It combines Spring scheduler that automatically takes in all of the updating cases thus it leads to reduce both server update and traffic cost [7]. The necessary performances are tested by using a JUnit framework, which can apply to fetch and test repeatable automated tests. The results are used for execution time, updating time and CPU usage time with the different number of mobile datasets.

This paper is described into five sections: indexing and its applied areas are introduced in section 2. A proposed index structure, presorted-nearest index tree structure is explained in section 3. A virtual mobile dataset generation is presented in section 4 with architecture and a process flow of mobile objects generator. Finally, the experimental results of performance evaluations and discussions are collected in section 5.

2. INDEXING AND ITS APPLIED AREAS

Indexing is a special data structure that allows having quick access to data or objects systematically. Indexing is usually used for users to display results that match the desired user criteria. In recent years, there has been a survey on indexing algorithms together with the activities of moving object databases to fast and efficient processes [8]. Normally, indexing has its own relevant structure that needs time to build it. It is saved by building the structure only for points that need to receive information and it can be used in the multicast notification. Nowadays, indexing has various types and fieldworks to apply in the real world. Some types are good in querying and some are in updating. Based on the types of moving objects, such as geographically distributed moving objects, there is a special index structure that allows generating range monitoring queries over content-match and group-aware queries [9].

Some of the indexes preserve only for a specific type such as the trajectory of moving objects [10]. The common index for different types of query special is grid structure. With the rapid distribution of spatial moving objects, a Distributed Grid Index (DGI) was proposed [11] that accurately interrogate the objects' location by unique object's ID. Sometimes, the spatial index structure such as R tree used the partition of grid index that became less overlap and gained information [12]. Like the grid structure, a quadtree index

offered [13] better performance regardless of the amount of moving objects. However, [14] using a simple, uniform grid index was basically supported to query processing and it is weak in update performance. In order to better help mobile objects' query was proposed [15]. Some survey paper thought uncertain data or object index structures were not applicable to all types of query processing [16]. As [17] discussed that some tree-based algorithms such as R trees are not suitable for high-dimensional data in real-life applications. Instead, they proposed Grid-index algorithm (GIR) that offers reverse rank queries with a little memory cost. On the other hand, [18] discussed that it is hard to get an optimal resolution of index based on grid and also R tree with the skewed mobile objects. As a result, a qualified index structure depended on the structure of index, types of moving objects, and queries.

Some index for moving objects are based on queries regardless of moving positions, speed or movement nature [19]. A suitable way of indexing and querying to moving objects is having a good design, such as a spatio-temporal indexing using key-value store [20]. Due to the variation of moving objects, the concurrency problem may occur in their indexing so that distributed index structure that exploits in multiple machines [21]. Actually, an indexing based structure is very suitable for continuous queries within a specified time by sending multicast messages, reports or notifications. It is more suitable as location points are received within a range by only once the query. Without indexing, the range query is available based on either a search according to the distance or other measures. But, it has to search along with a sequential or order match pattern by each object in the dataset. Therefore, two issues are found in this way.

a. Such a scanning of the large dataset can be very high cost.
b. The longer dataset, the more execution time is required by sequential searching.

Especially for repeating a single range, indexing based range searching will be faster than each of the single match of range searching. As [22] proposed an index structure called TM-RTree by planning moving objects with different modes such as transportation, temporal and spatial. They claimed that their proposed index structure well supported to answer queries related to the desired ranges.

3. RESEARCH METHOD

3.1. Building index structure with presorting

If the data is added by sorted list, the creation of an index is faster than the usual because it does not have to search for the correct space to store the new value, and it saves both I/O and CPU costs. The new value that will always be joined adjacent to the last value that was stored. The index tree would be fast as it is easily built sequentially. If an index structure that is built by unsorted data, it might be bigger than the index structure of the sorted data. Especially it is going to be hard to build and store in a huge amount of data size. Ordering or sorting the data may take additional extra time, but it will be faster than any other unsorted structure and thus it will bring to be a fast and compact index.

In this paper, a presorted-nearest index tree is proposed and it can be used for any types of two-dimensional points. All of the incoming objects or points are ordered before building index structure. The following algorithm is about the detailed structure of the proposed index tree.

3.2. Presorted-nearest index tree

```
Input c: center of location query,
        R[m]: mobile locations in range lists that are sorted by latitude and longitude of each location,
        m: mobile locations
Output Nearest locations index structure from center points by level order
Algorithm: 2D-Nearest Index Tree(c, R (m))
        if m != null then Tree->root := c;
        min_Index = 0;
        Tree->leftKey: =FindNearest(c);
        Tree->rightKey: =FindNearest(c);
        GenerateTree(c);
FindNearest(c):
        nearest := Node[0]; mindist=0;
        for i=1 to R(m)
        d= distance(c, nearest);
        if(d[i]<d[mindist])
        nearest=d[i];
        return nearest;
```

```
GenerateTree(c):
        for d = 1 to height(Tree)
        PrintGivenLevel (Tree, d);
        if level is 1 then
        print (Tree->c)
        else if level>1 then
        PrintGivenLevel (Tree->leftKey, level-1);
        PrintGivenLevel (Tree->rightKey, level-1);
```

Firstly, the procedure takes mobile lists (rangeMobiles) which are in the maximum and minimum range boundaries. The range mobiles are input as two sorted arrays. Both of these arrays are arrays of locations which have the same location points but one is ordered by latitude and another one is ordered by longitude. Then, the nearest location tree is built according to the CenterPoint and service range. In this location tree, the CenterPoint take place as the root of other nodes in the tree. Then, nearest location points are recursively located to the right and left of each root. The position of location points will be placed and retrieved according to their level order in the tree for easy and quick to access nearest neighbor locations.

For range queries, the first thing is determining whether moving objects are in the circular range area or not. This can easily calculate by bounding coordinates between the center and desired service distance: (latitude, longitude, distance) in the following formula.

- minLongitude = longitude - Math.toDegrees (distance/R/Math.cos (Math.toRadians (lat)));
- maxLongitude = longitude + Math.toDegrees(distance/R/Math.cos (Math.toRadians (lat)));
- maxLatitude = latitude + Math.toDegrees(distance/R);
- minLatitude = latitude - Math.toDegrees(distance/R);

4. VIRTUAL MOBILE DATASET GENERATION

4.1. Moving object generator

In reality, different datasets have different applications and usage. Therefore, all of the researchers should think about their domain of interest before making any dataset synthetically. It has to compare the behaviors and functions of real data so that it seems realistic.

The required phases and detail explanations of this process bring a good idea for making any other moving objects generator. In this architecture, synthetic mobile users' dataset is generated along with their behaviors and activities. Practically, mobile users are generally located at stationary or no moving type, sometimes they drive, and some are walking. Especially for updating moving data, one of the suitable forms is updating their process based on their motions. As an example, a user who is walking and a user who is driving will not be the same motion as the required update. The more dataset varies, the more result different, and all of these outcomes will be used in part for incoming research. In order to better findings, virtual mobile users are created with different forms and versions and tested by using indexing in this paper.

A suitable way that can generate synthetic dataset is deriving a model or architecture with valid properties of real data. An appropriate and valid architecture, a moving objects generator is proposed in this paper. The flow of the proposed moving objects generator is shown in Figure 1.

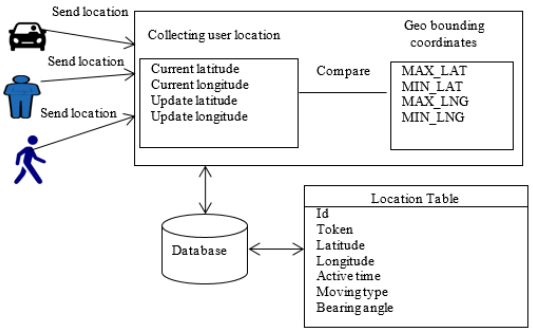

Figure 1. The architecture of mobile object generator

In each of the mobile user, different action and characteristic take different location update. Two main steps are included in implementing this architecture. Mobile users' locations are generated randomly within the maximum and minimum bounding coordinates that are in -90 to 90 and 0 to 180.

Thus, the first step is generating mobile users according to the random location formula that is separately calculated with latitude x and longitude y of each user location.

X=minX + (random ()*(maxX-minX) + 1);

Y= minY + (random ()*(maxY-minY) + 1);

Since the mobile user location has its own latitude and longitude, each of them is calculated separately. After getting random locations, all of these are divided by three types of mobile users who are walking, driving and stationary according to the time and distance threshold values along with their updates are done for each. At the second step, a location update policy is drawn as mobile users' behaviors and actions are quite different. In this step, time and distance based location update strategies are used to separate mobile users' types. The output will be a synthetic dataset that includes finding locations and updating them appropriately.

4.2. Process flow of generating mobile dataset

The process flow of the proposed architecture on moving objects generator is clearly explained in Figure 2.

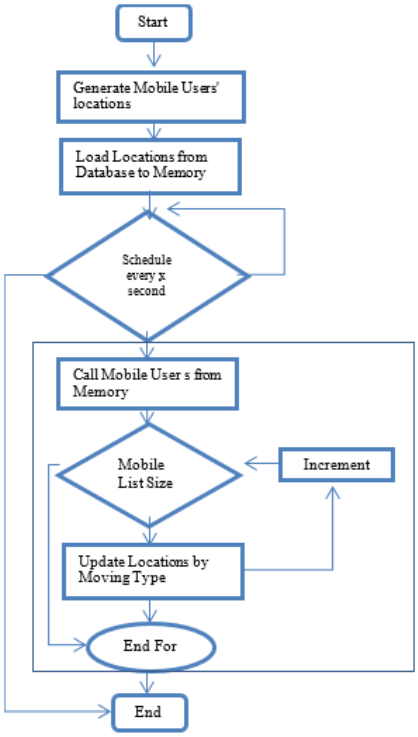

Figure 2. Process flow of mobile objects generator

The first phase is to find the location of mobile users synthetically because it is hard to get the millions of mobile locations in reality and it is also a way of privacy protection for real world. After getting these locations of mobile users, all of these locations are loaded from database to memory. Then, the following two steps are done by Spring scheduler that runs with the help of Spring Boot at the specific time. The required update locations are done by using the following formula.

Update Lat =.asin (sin (latitude)*cos (UpdateDistance/R) + cos (latitude)*sin (UpdateDistance/R)* cos(BearingAngle));

Update Lon = longitude+atan2 (sin(BearingAngle)*sin(UpdateDistance/R)*cos(latitude),cos(Update Distance /R)-sin(latitude)* sin(updateLat));

Firstly, mobile users are taken out of memory before doing any process. Second, mobile users' locations are updated by moving types after checking mobile user lists which are available for processing.

5. EXPERIMENTAL RESULTS

We managed an experimental setting on a computer with an Intel Core i7-4590U CPU, 8G RAM, and 1-TB hard disk storage. We applied the proposed synthetic dataset generator for mobile locations to simulate moving objects that basically came from the random function of three different behaviors such as the car, walking and stationary. All of the activations of this generator are done by spring boot [17]. It is designed chastely for all applications based on spring development. Its important concepts are automatic configuration, starter dependences, and the command line interpreter. Spring boot provides fast and wide accesses for all development and provides non-functional features.

In order to get the performance evaluations, we tested about proposed index tree construction, its range queries and nearest neighbor queries with the comparison of using KD tree. Moreover, CPU time is calculated and noted with the increasing number of mobiles. To improve performance evaluations, we clearly showed the tested results by different range values and number of mobiles. The required parameters and their values for performance evaluations are described in Table 1.

Table 1. Terms and values

Parameters	Values
Synthetic Mobile Generation Range	Min Latitude 9.6, Max Latitude 28.5 Min Longitude 92.2, Max Longitude 101.17
Number of Objects	1000 – 100,000
Object Types	Car, Walking, Stationary
Bearing Angles	$0^{\cdot}$ – $360^{\cdot}$
Update Distance	0.00832km,0.01112km,0.0102km
Schedule Fixed Rate (for Random Mobile Users)	2000msec
Schedule Fixed Rate (for Building Tree)	10000msec
Initial Delay	5000msec

5.1. Tree construction time comparsion

The Figure 3 describes the results of the experiment on presorted-nearest index tree and KD tree over tree construction time. For this comparison, the results are confirmed after testing an average of 15 times with the dataset is from 1000 to 10,000 mobile locations. In this experiment, the number of mobiles is generated by proposed generator which is initialized by adding the same number of moving object types. It can be concluded that the proposed tree construction is one third faster than KD tree. This is because both comparative trees are unbalancing two-dimensional trees and the proposed tree input is being included the order queries that provide to be fast in tree construction. In addition, the form of the proposed tree is already out of their nearest neighbor by level order since tree construction.

5.2. Range searching Time Comparsion

The Figure 4 shows the test for the duration of searching time within a distance of 100 kilometers to 600 kilometers range search queries. This experiment includes a dataset 10000 and a center latitude and longitude that take as University of Computer Studies, Yangon GPS coordinate. It was not described the tested results within a distance of less than 100 km range searching time because it was taken nearly the same in (1 millisecond). According to the figure, both KD tree and presorted-nearest index tree are adequately supportive to range searching.

5.3. Comparison of Nearest Neighbor Search within a Range

The Figure 5 shows the comparison results of the presorted-nearest index tree and KD tree that acts within a range of mobile locations from the nearest are marked and calculated.. In this experiment, the circular range is used for finding range search. To undertake the duration of the experiment, the mobile locations are firstly checked within the range or not, which has marked the time. Then, the displaying the mobile user lists by nearest called nearest neighbor search is tested and also marked the time. Such a query can be used for not only static or stationary locations but also moving the location query that repeatedly undertakes as a continuous range query. According to the experimental results, the presorted-nearest index tree execution takes a half time in the execution of KD tree. This is because although the searching time over two comparison trees are nearly the same, the structure of presorted-nearest index tree is already supported to nearest neighbor within the tree construction thus it can search for nearest neighbor within one millisecond.

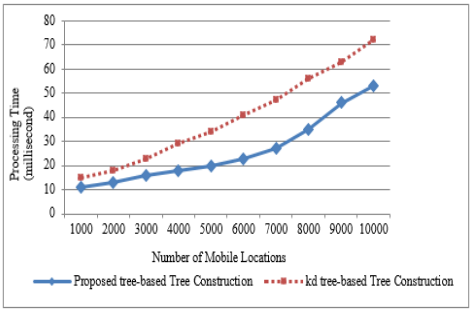

Figure 3. Processing and comparison of constructing time

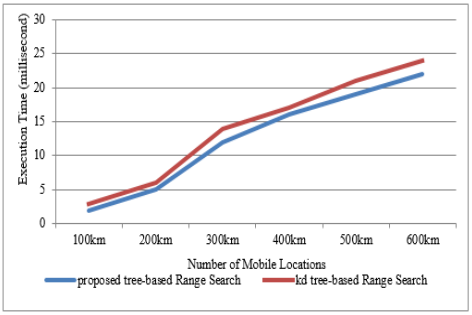

Figure 4. Evaluation of range searching time

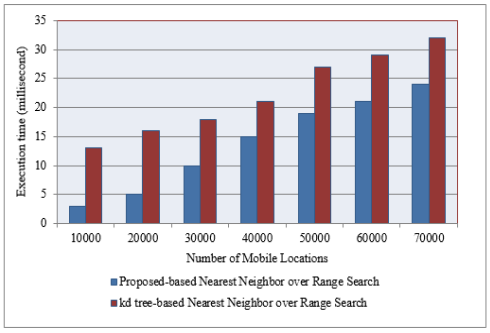

Figure 5. Execution time between proposed tree and KD tree

6. DISCUSSION

The calculation of the performance evaluation over two non-balanced trees (presorted-nearest index tree and KD tree) is based on the tree construction time, execution time during range searches and nearest neighbor searches. In this calculation, the generating of the latitude and longitude which is the source of the synthetic mobile dataset is taken by Myanmar country boundaries within the maximum and minimum latitudes and longitudes along with the proposed formula to retrieve and update. One prominent characteristic within tests found that the proposed tree (presorted-nearest index tree) is faster in tree building due to ordered data by queries as the input data. It can be found that both comparative trees (presorted-nearest index tree and KD tree) are conveniently and fast in the range search without delay. Besides, the proposed tree provided the nearest neighbor by level order since the tree building so that it does not need to give private time in the nearest neighbor searching.

7. CONCLUSION

This paper emphasized an efficient indexing and querying structure of mobile objects on a synthetic dataset. According to the difficulties of getting millions of mobile locations, a synthetic mobile dataset model is proposed along with the required framework and its process flowchart. The presorted-nearest index tree is proposed that systematically addressed and updated the locations of mobile and supported to desired range queries over mobile objects. Experimental results indicate the efficiency of using proposed index tree is more than using the KD tree structure for moving objects. The variation of mobile objects is handled by presorting in the proposed tree structure. However, the proposed structure which is presorted by input queries resist the skewed distributions of mobile objects. Moreover, the proposed structure performed the desired range queries and nearest neighbor search within the short execution time, especially in the large mobile dataset.

ACKNOWLEDGEMENTS

I would like to express very special thanks to my supervisor Dr. Sabai Phyu, Professor, the University of Computer Studies, Yangon. I would not be able to continue without her supervision and valuable guidance during period of the research study.

REFERENCES

[1] A. Dhumal, *et al.*, "Vehicle Tracking System using GPS and Android OS," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol/issue: 4(4), 2015.

[2] Brata K. C., *et al*., "Location-based Augmented Reality Information for Bus Route Planning System," *International Journal of Electrical and Computer Engineering (IJECE)*, vol/issue: 5(1), pp. 142-149, 2015.

[3] Syafie N., *et al.*, "The Development of Online Disaster Information System Using Location Based Service LBS Technology," *International Journal of Informatics and Communication Technology (IJ-ICT)*, vol/issue: 3(1), pp. 47-58, 2014.

[4] S. Dhanabal and S. Chandramathi, "A Review of various k-Nearest Neighbor Query Processing Techniques," *International Journal of Computer Applications (0975 –8887)*, vol/issue: 31(7), 2011.

[5] S. Ghorbani, *et al.*, "Continuous Mutual Nearest Neighbour Processing on Moving Objects in Spatiotemporal Datasets," *International Journal of Information and Education Technology*, vol/issue: 7(5), 2017.

[6] H. J. Cho, "Efficient Shared Execution Processing of k-Nearest Neighbor Joins in Road Networks," *Mobile Information Systems*, Article ID 1243289, 2018.

[7] Jovanovic Z., *et al.*, "Java Spring Boot Rest WEB Service Integration with Java Artificial Intelligence Weka Framework," *International Scientific Conference. Gabrovo*, pp. 270-274, 2017.

[8] Ayabakan I. and Kilimci P., "Moving Object Databases-Indexing Algorithms," *International Journal of Computer Theory and Engineering*, vol/issue: 6(6), pp. 455-459, 2014.

[9] Jung H. R., *et al.*, "Evaluation of Content-Matched Range-Monitoring Queries over Moving Objects in Mobile Computing Environments," *Sensors*, vol/issue: 15(9), pp. 24143-24177, 2015.

[10] Pfoser D., *et al.*, "Novel Approaches to the Indexing of Moving Object Trajectorie," *Proceedings of the 26th International Conference on Very Large Databases (VLDB'00)*, Cairo, Egypt, pp. 359-406, 2000.

[11] Park K., "Location-based grid-index for spatial query processing," *Expert Systems with Applications, Elsevier*, vol/issue: 41(4), pp. 1294-1300, 2014.

[12] Rslan E., *et al.*, "Spatial R-Tree Index Based on Grid Division for Query Processing," *International Journal of Database Management Systems (IJDMS)*, vol/issue: 9(6), pp. 25-36, 2017.

[13] Zhang F., *et al.*, "Real-Time Spatial Queries for Moving Objects Using Storm Topology," *International Journal of Geo-Information*, vol/issue: 5(10), 2016.

[14] Sidlauskas D., *et al.*, "Trees or Grids? Indexing Moving Objects in Main Memory," *The ACM GIS '09. Seattle, WA, USA*, pp. 236-245, 2009.

[15] Ziwei Z., *et al.*, "A Vor-KVGQ Index Structure of Mobile Objects Based on CoveRed Area," *International Journal of Hybrid Information Technology*, vol/issue: 8(5), pp. 177-186, 2015.

[16] John A., *et al*., "Indexing and Query Processing Techniques in Spatio-Temporal Data," *ICTACT Journal on Soft Computing*, vol/issue: 6(3), pp. 1198-1271, 2016.

[17] Dong Y., *et al.*, "Grid-Index Algorithm for Reverse Rank Queries," *The 20th International Conference on Extending Database Technology (EDBT)*, Venice, Italy, 2017.

[18] Park Y., *et al.*, "A Fast and Compact Indexing Technique for Moving Objects," *14th International Conference on Information Reuse & Integration (IRI) IEEE*, San Francisco, CA, USA, 2013.

[19] Kalashnikov D. V., *et al.*, "Efficient Evaluation of Continuous Range Queries on Moving Objects," *The 13th International Conference on Database and Expert Systems Applications*, pp. 731-740, 2002.

[20] Le H. V., "Distributed Moving Objects Database Based on Key-Value Stores," *Proceedings of the VLDB 2016 Ph.D. Workshop*, New Delhi, India, 2016.

[21] Lee Y. and Song S., "Distributed Indexing Methods for Moving Objects based on Spark Stream," *International Journal of Contents*, vol/issue: 11(1), pp. 69-72, 2015.

[22] Xu J., *et al*., "The TM-RTree: an index on generic moving objects for range queries," *An International Journal on Advances of Computer Science for Geographic Information Systems*, vol/issue: 19(3), pp. 487-524, 2015.

BIOGRAPHIES OF AUTHORS

Thu Thu Zan is a research candidate at Cloud Computing Lab, University of Computer Studies Yangon. Her current research is based on generating synthetic dataset for moving objects, wireless network operation and cloud computing. She is also a member of IAENG (International Association of Engineer).

Sabai Phyu is a dean of Cloud Computing Lab at University of Computer Studies, Yangon (UCSY) Myanmar. She is interested in virtualization and cloud computing. She currently works at the Software Department, UCSY Myanmar.