UIL : A Novel Indexing Method for Spatial Objects and Moving Objects

Xuguang Huang* Sung-Ha Baek** Dong-Wook Lee**
Weon-Il Chung*** Hae-Young Bae****

Abstract Ubiquitous service based on Spatio-temporal dataspaces requires not only the moving objects data but also the spatial objects. However, existing methods can not handle the moving objects and spatial objects together. To overcome the limitation of existing methods, we propose a new index structure called UIL (Union Indexing Lists) which contains two parts: MOL (Moving Object List) and SOL (Spatial Object List) to index the moving objects and spatial objects together. In addition, it can suppose the flexible queries on these data. We present the results of a series of tests which indicate that the structure perform well.

Keywords : Index, Inverted List, Spatial Object, Moving Object, Ubiquitous Services

1. Introduction

A Spatio-temporal database supports not only efficient query processing on a large number of moving objects but also the spatial objects (e.g. maps, roads, mountains, etc.), and has numerous applications in practice. The existing studies mainly focus on retrieving the data of moving objects. However, nowadays, *ubiquitous service* based on spatio-temporal dataspaces requires not only the moving objects data but also the spatial objects and other unstructured data. In this paper, we consider maintaining an index structure for moving objects and spatial objects, where the goal is to suppose the flexible queries on spatio-temporal dataspaces.

There have been many research efforts to develop strategies for indexing and querying moving objects. However, the existing methods for moving objects suffer from certain disadvantages. The R-tree based index [1][2][3][4][5][6] may require high management cost to track moving objects, because moving objects require frequent updates which made to an R-tree index lead to deteriorated performance due to a significant amount of overhead to split and merge nodes. Although some of methods such as STRIPES [9] are efficient to update, it has high space consumption and low query performance. The B^x-tree [10], the state-of-the-art, involves several B^+-tree indexing the order of objects on a space filling cure, which do achieve satisfactory query efficiency due to the large number of false hits.

Nonetheless, all of the existing methods above do not consider the spatial objects data together with moving objects data. They can not process the queries which required both the moving objects data and spatial objects data, such as “Return the cars that passed by the red building whose name is ‘National’”.

In this paper, we present a new index structure called UIL(*Union Indexing Lists*). The UIL consists of two main structures: Spatial Objects-List (SOL) and Moving Objects-List(MOL). The SOL is an inverted list to support robust indexing of loosely-coupled collections of heterogeneity spatial objects data. The MOL is a series of Moving Objects Tables (MOT) to handle the spatio-temporal data of moving objects. Multiversion techniques used on these two structure to produce a time and space efficient index structure, and is particularly useful for processing various quer-

† This research was supported by a grant(07KLSGC05) from Cutting-edge Urban Development - Korean Land Spatialization Research Project funded by Ministry of Construction & Transportation of Korean government.

* Master Student, Inha University Dept. of Computer Science and Information Engineering, xuguanghuang@yahoo.cn
** Doctoral Student, Inha University Dept. of Computer Science and Information Engineering, {shbaek, dwlee}@dblab.inha.ac.kr
*** Instructor, Hoseo Universitry Dept. of Information Security Engineering, wnchung@hoseo.edu(corresponding author)
**** Professor, Inha University Dept. of Computer Science and Information Engineering, hybae@inha.ac.kr

Manuscrip received : 2009.05.12 Accepted : 2009.06.15

ies on spatio-temporal dataspaces.

The MOL contains a series of MOT which grows up along with the passage of time. A MOT in MOL is a table, where rows represent the spatial information and columns represent the moving objects. In each grid cell, there is a variable to record whether a moving object appears in that location at that time. The SOL is an inverted list which is a two-dimensional table, where the row represents the spatial information and the column represent the keywords of geography objects. In section 4, we will introduce how the combination of the two structures can answer the flexible queries easily.

The rest of the work is organized as follows: Section 2 discusses related works. Section 3 describes our indexing structure in details. Section 4 gives the algorithms for loading and supporting flexible queries. Section 5 presents experiments evaluation and Section 6 concludes.

2. Related work

The two bodies of work most close to ours are R-tree based methods and inverted index methods. Next, we will discuss each category in turn.

2.1 R-tree based methods

There have been many indexing methods to store and retrieve the past, current and future locations of moving objects. Our work focuses on the past and current data of moving objects, so we mainly introduce the works on this area.

The RT-tree[1] which adopts ideas from the R-tree[2] and TSB-tree[3] couples time intervals with spatial ranges in each node of the tree. It is inefficient that it would require a complete scan of the index, when queries are processed based on the temporal domain as searching and splitting a node of RT-tree guided by spatial information not the temporal information.

The 3DR-tree[4] is very space-efficient and can process time interval query very efficiently, as it stores only the different object versions without redundant copies and the temporal attributes and the spatial attributes are tightly integrated. However, it is poor to process the query at a specify time, since all objects are indexed in a single tree and thus the query processing time depends on the total number of entries.

The HR-tree[5] construct a set of R-tree for each timestamp, and it can handle the queries of a specify time very efficiently. For time interval queries and range queries, however, it can be very slow, as these queries require searching as many trees as the number of distinct timestamps in the interval.

MV3R-tree[6] proposed by Tao and Papadias combines the concepts of multi-version B-trees and 3DR-tree. Although it is very efficient to deal with time interval queries and range queries, it requires more space and needs high management cost due to the combination of the MVR-tree and 3DR-tree.

2.2 Inverted List

An inverted list is a record level inverted index which contains a list of references to documents for each word. An inverted index is an index data structure storing a mapping from content, such as words or numbers, to its locations in a database file, or in a document.

Recently, Xin Dong[7] proposed a new indexing structure KIL extended the inverted list to index heterogeneous data in dataspaces. The extended inverted list KIL is a two-dimensional table, where the i-th row represents indexed keyword and the j-th column represent instance. The cell (i-th, j-th) records the number of occurrences. It is very efficiently to process keywords query and neighborhood queries. However, it only cannot handle the spatial data which contains the location information.

3. Design of Union Indexing List

In this section, we first describe the model of moving objects data and spatial objects. Then the structure of UIL and the details of MOL and SOL are introduced.

3.1 Various data

Our goal is to support efficient queries over moving objects data and spatial objects which are not necessarily semantically and ghited by existing Spatio-temporal database or Spatio-temporal data integrated system.

An object's position at some time t is given by oi=$(p_1,p_2,\cdots,p_d)$,which is a point in a d-dimensional space.In this paper, we consider a moving object data as a set of discrete manner point objects which grows along with time. We assume that a point moves in a two-dimensional space and our approach can be easily expanded into a higher dimensional space. Unlike some R-tree based method, we do not use MBR, but instead it stores the exact sample locations of each moving object.

Considerable work has been done on indexing static spatial objects. Probably the most popular index methods are two types: one is the cell methods, such as Quad tree[15] and K-D-B tree[16] which use the grid file to handle the point spatial data; another is R-tree, a balanced structure that clusters objects by their spatial proximity. However, all of these methods only consider the location information of spatial objects. Beside this information, there is also other unstructured information of spatial object. For example, a gray building located in No. 10 of First road and its name is "*Hana*" bank. To describe a building, we can not only use its geography coordinates but also use its name "*Hana bank*", address <"*no.10*","*first road*"> and color "*gray*" information. How to index the spatial object using these predicate keywords and geography coordinates is a big challenge. In this paper, we model the spatial object as a set of tetrameric elements (ID, location, attribute, keywords) which can adapt the variety data model. And SOL proposed by this work which is an extended inverted list is used to index the various spatial data.

3.2 Data model of UIL

Formally, in UIL index structure proposed in this paper, the data of moving objects and spatial objects are modeled within grid cells.

In a three-dimensional space (X,Y,T), a trajectory of a moving object M can be presented as (x,y,t), where $x=[x_0,x_{n-1}]$, $y=[y_0,y_{n-1}]$ and $t=[t_0,t_{n-1}]$. And they can be divided into cells according to the sample interval such that: $x=[x_0,x_1)U[x_1,x_2)U\cdots U\ [x_{n-2},x_{n-1}]$, $y=[y_0,y_1)U[y_1,y_2)U\cdots U[y_{n-2},x_{n-1}]$ and $t=[t_0,t_1)U[t_1,t_2)U\cdots U[t_{n-2},t_{n-1}]$, where tn means current time.

At a certain time, a data of a moving object is a point data which can be presented in the grid file made by location (x,y). For example, at t time, two points data of moving objects oi and oj were located at (1,1) and (2,0), where as fig. 1(a) shows, the gray cell means there is a point data and the value in the cell present the location value. Different from this, in our method cell ap the two-dimensional (x,y) into a one-dimensional *L* which is taken as y-axis, and we take the pointers of moving objects as the x-axis. If the value of each cell is not zero, it means the moving object appeared at corresponding location at that time. The example above can be presented as the fig. 1(b) shows.

As the discussion in the first part of this section, the spatial objects can be described as its location in

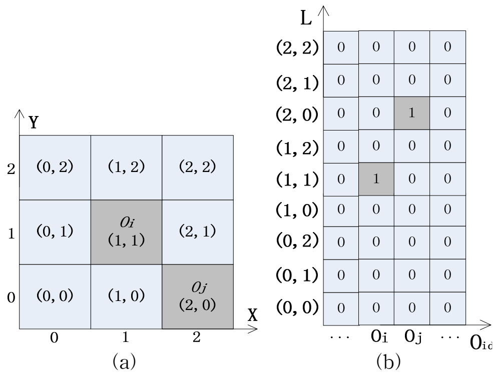

Fig. 1. The Grid Files of Storing Moving Objects

formation and predicate keywords. In this work, we use inverted list to represent these information. For example, there is a spatial object I0 named '*HANA bank*' and located at (1,1), (1,2), (2,1) and (2,2), where as fig. 2(a) shows. The same as moving objects, the location information of spatial objects also can be mapped into one-dimensional *L* as fig. 2(b) shows.

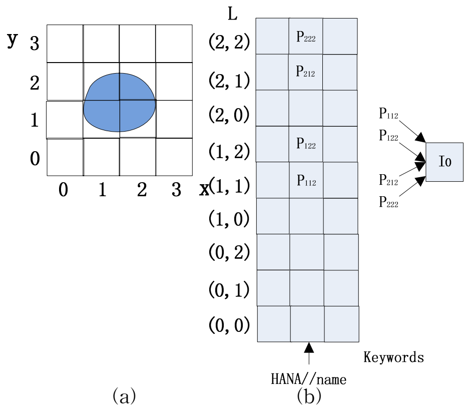

Fig. 2. The Grid Files of Storing Spatial Objects

We extract the predicate keyword {"*HANA*"} in the "name" attribute from the spatial object I_0, and then we can construct four keyword elements {"HANA// name",(1,1)|(1,2)|(2,1)|(2,2)} for I_0. In the fig. 2(b), an inverted list is built where its x-axis represents the predicate keyword elements, and its y-axis represents the location information. The cell (x,y) is a pointer to a list which stores a set of spatial objects. Here the cell p_{112}, p_{122}, p_{212} and p_{222} point the same list which contains the spatial object I_0 to save the space. If one of set of pointers point only I_0 but also other spatial object, then a new list that contains I_0 and other spatial objects will be created.

3.3 Moving Object List

The MOL is a collection of MO-Tables, each of which serves as an index for a set of moving objects at a specific time interval. The MOL can grow up along with the passage of time. As at a specific time interval, the number of moving objects is unfixed, the width of each of MO-Tables is difference. Fig. 3. illustrates the structure of the MOL. The structure of each of MO-Tables is the same as that of fig. 1(b) shows. When new records of moving objects are sampled, the entries of each moving objects are added to its corresponding cell.

Initially, a MO-Table is partitioned to C cells. The choice of C value may have nontrivial impact to the performance of the UIL index ionstorage utilization and query processing. Bentley et al.[8] show that a nearly optimue size for the cells is the same as those of the query window sizeg. Bemost applications, howehat, the queries will vary ionsize as well as inowehat, t,nso there is little i queries w available for riking a good choice of king sizeg.Thus we propose ntleyollowing alterniesves to choose C:on1) one based on the population of moving objectsw size(p the other based on the mostmue distance that ano choosecan move during a gsven time period.

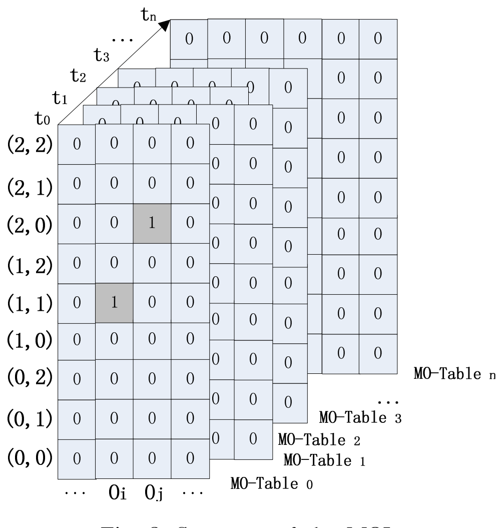

Fig. 3. Structure of the MOL

3.4 Spatial Object List

Although R-tree based method are known for their effectiveness in indexing for a wide spectrum of spatial database applications, they are not particularly well supposed the flexible queries, such as keywords query. To avoid such drawback of R-tree based methods, we take an entirely different approach called Spatial Object List (SOL) as shown in Fig. 2 (b).

The SOL is an inverted list which is a two-dimensional table, where the x-th ro lispresents predicate keywords K and the y-columnlispresents the location information L. The cell at x-th ro land y-th column, denoted (K_x,L_y),list whica pointer toca list which storeica set of spatial objects. All of the attributes of this set of spatial objects contain the keywords K_x.

Note that the inverted list, as described in fig. 2 (b), set the combination keyword element *keyword//attribute* as the values of rows, and these keyword elements are ordered in alphabetic order. In that example, we know the building's name is "*HANA*", and the four cells corresponded to the keyword element "*HANA//name*" are all pointing to the spatial object I_0.

4. Algorithms for loading and querying

4.1 Inserting moving objects and spatial objects to UIL

In this section, first we describe how to insert a spatial object into UIL, and then we describe the processing of inserting a moving object.

As we introduced above, spatial objects do not change in a long time. So we always treat spatial objects as static data. Because of this, there is only one SOL in UIL, and the SOL is built at the beginning of initializing UIL. Here we take an example to present how to insert a spatial object into SOL.

EXAMPLE 1 Inserting a spatial object I_1 "the National museum which is a red building and located at (1,0),(1,1), and (1,2)" into the SOL showed in fig. 2(b).

From this example, we can extract a set of predicate keyword elements (National//name, (1,0)), (National//name, (1,1)), (National//name, (1,2)), (red//color, (1,0)), (red//color, (1,1)), and (red//color, (1,2)). Then we insert these elements into a SOL and fig. 4 shows the finally SOL.

Note that in our method, to save the storage space, the pointers of the same column point to the same spatial objects list if the predicate keyword element only owned by a spatial object. If there are two or above two spatial objects own the same predicate keyword element, the corresponding pointer will point to a new list which contains these spatial objects. For example, if there is another spatial object I_2 contains the element "red//color", and I_2's location is (1,1), the pointer P114 will point to a new list which contain I_1 and I_2.

The method of inserting moving objects contains 3 steps. At a specify time t_1, an moving object Ok is sampled and its location is (x_k,y_k). First step we find the MO-Table at time t_1, or build a new one if it is not existed. The second step we add a new column in this MO-Table for Ok and set its x-axis is Ok. Finally, we find the grid cell $(O_k,(x_k,y_k))$ and set the value of this cell $(O_k,(x_k,y_k))$ from *null* to 1.

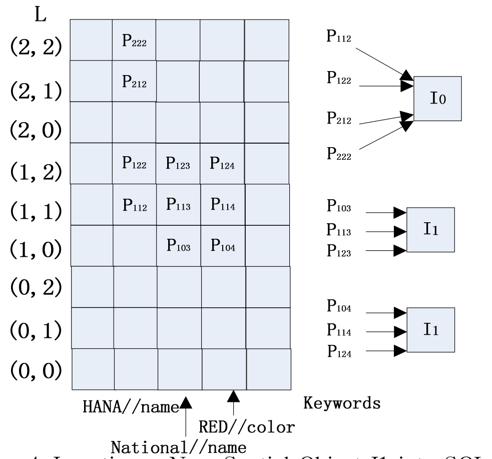

Fig. 4. Inserting a New Spatial Object I1 into SOL

4.2 Query processing

UIL contains two parts: MOL and SOL which can index both spatial objects and moving objects data. And it must suppose the flexible queries, such as range queries, time interval queries, keywords queries and NN queries.

Here we take an example to describe how to execute these queries.

EXAMPLE 2 Fig. 5 shows a UIL instance, and the query "find the moving objects which passed by a red building whose name is national from 9:00 to 9:05, and return the similar spatial objects" can get the results by it.

From this query, we can extract the predicate keywords "national//name" and "red//color". According to keywords "national//name", we can find a set of spatial objects $\{I_0, I_1\}$ whose name is "national", and similarly, spatial objects $\{I_1, I_2\}$ whose color is red can be returned. The intersection of the two sets of spatial objects $\{I_1\}$ and its location information {(1,1),(1,2)} can be returned.

During the time from 9:00 to 9:05, there are two corresponding MO-Tables. From MO-Table 1, we can find there are two moving objects $\{O_1, O_2\}$ which are at (1,1) and (1,2). In the MO-Table 2, two new moving objects $\{O_3, O_4\}$ are at (1,1) and (1,2). The four moving objects $\{O_1, O_2, O_3, O_4\}$ are the results of the query in example 2.

From the SOL of this UIL, we can find that the names of $\{I_0, I_5\}$ also contain "national" and the color of $\{I_2\}$ is red too. So the result of this NN query "the similar spatial objects" is $\{I_0, I_5, I_2\}$.

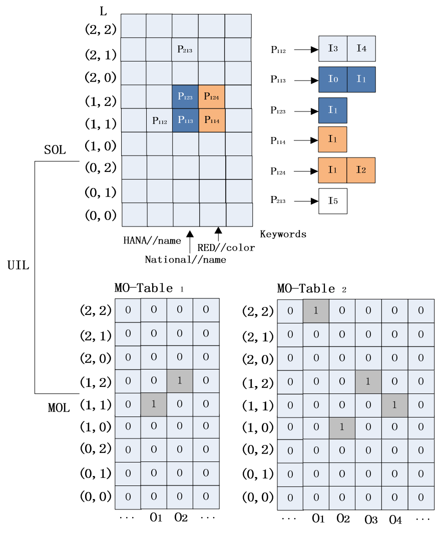

Fig. 5. An Instance of UIL

5. Experiments Evaluation

We now describe a set of experiments that validate the efficiency of our indexing methods and compare them against several alternatives. As there is no existing method to deal with the spatial objects and moving objects together, we split our method into two parts. One part is indexing spatial objects and compared with corresponding R-tree based methods. Another part is indexing moving objects and compared with STR-Tree, HR-Tree, 3DR-Tree and MV3R-Tree.

The main data set we use is the GSTD spatio-temporal data[11] and 2007 TIGER/Line Shapefiles[14]. GSTD has been widely employed[12] [13] as a benchmarking environment for access methods dealing with moving points. Each of the following datasets contains 10,000 regions with density 0.5 and is generated as follows. Objects' initial positions are determined following a Gaussian distribution. Timestamps are modeled as float numbers ranging from 0 to 1with granularity 0.01. The performance of different access methods is measured by running workloads. Each workload contains 500 queries with the same area and interval length. The 2007 TIGER/Line Shapefiles are extracts of selected geographic and cartographic information from U.S. MAF/TIGER database. It contains geographic linear features, boundaries and point features, area features and attributes information. In this paper, we use part of the information of LosAngeles. The primary parameters used in the experiments are outlined in Table 1, where the default values are given in bold fonts.

Table 1. Parameters used in the Experiments

PARAMETERS	VALUES
Number of Moving Objects	10K
Moving Objects Tables	512
Number of Spatial Objects	3K
Keywords of Attributes	9k
Number of Queries	1000

Fig. 6. shows the performance of inserting different number of new moving objects and spatial objects. We compared the UIL with some R-tree basedl othoIL wFrom fig. 6, we can find that UIL is ILite better than otherl othoIL if the number of new objects is not large. If there can a largew ot of objects he be inserted, the performance of UIL is almosture 6, we cs that of HR-tree, MV3R-tree and 3DR-tree, as it costs much time to extract the keywords from the spatial objects.

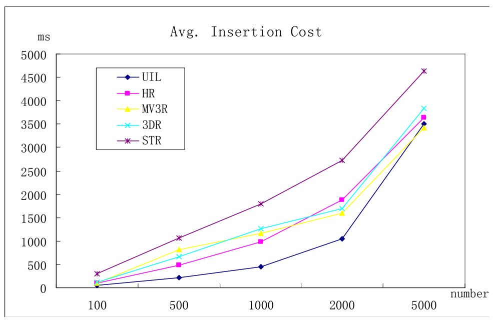

Fig. 6. Average Insertion Cost of Each Methods

Fig. 7 shows the performance of updating different number of moving objects and spatial objects. We can find that UIL consistently outperformed R-tree based methods whatever the number of updated objects is small or large. The reason is that updating operation of R-tree based methods are always includes three steps: searching, deletion and insertion, whereas the updating of UIL only has two steps: searching and recovery.

In these experiments, we use 1000 queries which include range queries (40%), time interval queries (40%), NN queries (10%) and predicate keywords queries (10%). Fig. 8 shows the average response time of per query.

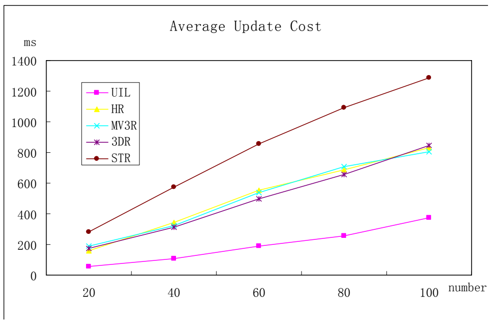

Fig. 7. Average Updating Cost of Each Methods

In fig. 8, we can find that the performance of UIL is quite better than the R-tree based methods in processing range query, time interval query and neighborhood query. We do not list the performance of processing keywords query of R-tree based methods, as they can not suppose the keywords query. And it is on average a little time to response per keywords query using UIL.

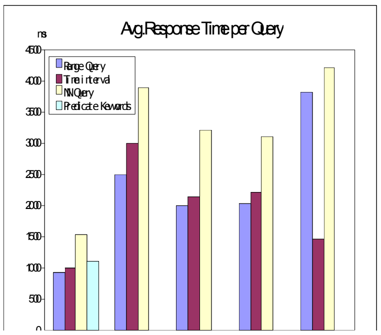

Fig. 8. Average Response Time per Query

6. Conclusion

This is the first study that adopts a union index structure for the spatial objects and the current and historical data of moving objects. Instead of the existing R-tree based methods, we proposed an union list based approach－UIL, which consists of two main structures: SOL and MOL. The SOL utilizes an inverted list to handle both the location information and the set of predicate keywords. The MOL is a collection of MO-Tables that can index the sample data of moving objects at specify time.

Through a set of experiments, the UIL has proven to be an effective access method well suited for handling the spatial data and moving objects. In addition, the UIL outperformed R-tree approaches consistently and considerably for flexibly queries such as range queries, time interval queries, NN queries and predicate keywords queries.

Reference

[1] X. Xu, J. Han, W. Lu, RT-tree: an improved R-tree index structure for spatiotemporal databases, Proceedings of the 4th Intl. Symposium on Spatial Data Handling, SDH'90, Zurich, Switzerland,1990, pp. 1040-1049.

[2] Antonin Guttman, R-trees: a dynamic index structure for spatial searching, Proceedings of the 1984 ACM SIGMOD international conference on Management of data, June 18-21, 1984, Boston, Massachusetts.

[3] David B. Lomet, Betty Salzberg, Transaction-time database, Temporal databases, 1993, pp. 388-417.

[4] Theodoridis, Y., Vazirgiannis, M., and Sellis, T.: Spatio-Temporal Indexing for Large Multimedia Applications. In Proc. of the 3rd IEEE Int'l Conference on Multimedia Computing and Systems, 1996, pp. 441-448.

[5] Mario A. Nascimento, Jefferson R.O. Silva, Towards historical R-trees, Proceeding of the 1998 ACM Symposium on Applied Computing, Atlanta, GA, February 1998, pp. 235-240.

[6] Yufei Tao, Dimitris Papadias, MV3R-tree: a spatio-temporal access method for timestamp and interval queries, Proceedings of 27th International Conference on Very Large Data Bases, Roma,Italy,September2001.

[7] Dong Xin, Halevy Alon. Indexing dataspaces. In: Proc. of the 27th Int'l Conf. on Management of Data (SIGMOD 2007). NewYork:ACMPress, 2007. 43-54.

[8] Jon Louis Bentley, Donald F. Stanat, E. Hollings Williams Jr., The complexity of finding fixed-radius near neighbors, Information Processing Letters 6 (6), 1977 209-212.

[9] Patel, J.M., Chen, Y., Chakka, V.P., STRIPES: an efficient index for predicated trajectories. In: Proceedings of SIGMOD, 2004, pp. 637-646.

[10] Jensen, C.S., Lin, D., Ooi, B.C., Query and update efficient B+-tree based indexing of moving objects. VLDB 2004, pp. 768-779.

[11] Theodoridis, Y., Silva, J. Nascimento, M. On the Generation of Spatiotemporal Datasets. SSD, 1999

[12] Nascimento, M., Silva, J., Thedoridis, Y. Evaluation of Access Structures ofor Discretely Moving Points. International Workshop on Spatio-Temporal Database Management, 1999.

[13] Pfoser, D., Jensen, C., Theodoridis, Y. Novel Approaches to the Indexing of Moving Object Trajectories. VLDB, 2000.

[14] Miller, Catherine L. TIGER/Line Files Technical Documentation. UA 2000. U.S. Department of Commerce, Geography Division,U.S. Census Bureau. http://www.census.gov/geo/www/TIGER/TIGERua/ua2ktgr.pdf

[15] J.R. Smith and S.-F. Chang. Quad-Tree Segmentation for Texture-Based Image Query, ACM 2nd Multimedia Conference Proceedings, 1994.

[16] J.T. Robinson. The k-D-B-Tree: A Search Structure for Large Multidimensional Dynamic Indexes, Proc. ACM SIGMOD, 1981, pp. 10-18.

Xuguang Huang

2004 Dept. of Computer Sciences, Chongqing University of Posts and Telecommunications(B.S.)

Present Master Student, Dept. of Computer Science and Information Engineering, Inha University

Research Interests : Spatial-temporal Data, Dataspace.

Sung Ha Baek

2005 Dept. of Computer Science and Information Engineering, Inha University (B.S.)

2007 Dept. of Computer Science and Information Engineering, Inha University (M.S.)

Present Ph.D student, Dept. of Computer Science and Information Engineering, Inha University

Research Interests : Data Stream, Cluster, Location based Services

Dong Wook Lee
2003 Dept. of Computer Science and Engineering, Sang Ji University (B.S.)
2005 Dept. of Computer Science and Information Engineering, Inha University(M.S.)
2005~Present Ph.D course, Dept. of Computer Science and Information Engineering, Inha University.
Research Interests : Spatial Data Warehouse, Spatial Information Management, Spatial DBMS in Ubiquitous Environment

Warn Il Chung
1998 Dept. of Computer Science and Engineering, Inha University(B.S.)
2004 Dept. of Computer Science and Information Engineering, Inha University (Ph. D.)
2004~2006 Senior Researcher, ETRI
2007~Present Instructor, Dept. of Information Security Engineering, Hoseo University
Research Interests : Database, Data Stream, Moving Object

Hae Young Bae
1974 Dept. of Applied Physics, Inha University(B.S.)
1978 Dept. of Computer Science and Engineering, Yonsei Univercity(M.S.)
1989 Dept. of Computer Science and Engineering, Soongsil University(Ph.D.)
1985 Guest Professor, Univ. of Houston
1992~1994 Director, Electronic Calculations Office of Inha University
1982~Present Professor, Dept. of Computer Science and Information Engineering, Inha University
1999~Present Director, Intelligent GIS Research Center
2000~Present Emeritus Professor, Chongqing University of Posts and Telecommunication, China
2004~2006 Director, Graduate School of Information and Communication, Inha University
2006~2009 Director, Graduate School, Inha University
Research Interests : Distributed Database, Spatial Database, Geometric Information System, Multimedia Databases