Design of Moving Objects Server for Location Based Services

Dae-Soo Cho, Kyoung-Wook Min, Jong-Hun Lee

Spatial Information Technology Center, ETRI

161 Gajeong-dong, Yuseong-gu, Daejeon, 305-600 Korea

TEL : +82-42-860-6676

FAX : +82-42-860-4844

E-mail : { junest, kwmin92, jong }@etri.re.kr

ABSTRACT

Recently, location based services, which make use of location information of moving objects, have obtained increasingly high attention. The moving objects are time-evolving spatial objects, that is, their locations are dynamically changed as time varies. Generally, GIS server stores and manages the spatial objects, of which locations are rarely changed. The traditional GIS server, however, has a difficulty to manage the moving objects, due to the fact of locations being frequently changed and the trajectory information (past locations of moving objects) being managed.

In this paper, we have designed a moving object server, which stores and manages the locations in order to support various location based services. The moving object server is composed of a location acquisition component, a location storage component, and a location query component. The contribution of this paper is that we integrate the each work for location acquisition, storage, and query into a moving objects server.

Keywords : Moving Objects, Location-based Services

1. INTRODUCTION

Recently, various types of location-based services have obtained increasingly high attention according to the extensive spread of mobile handset, which is capable of accessing wireless internet, and the development of location determination technology, that is represented by GPS (Global Positioning System). Location-based services are related the moving objects which change their locations through time. Therefore, to provide location-based services efficiently, it is required that an efficient system which could acquire, store, and query the large number of locations. The time-evolving locations of moving objects are not efficiently managed by existing commercial Database Management System

(DBMS) as well as Geographic Information System (GIS). The reason is that there is a critical set of capabilities that are needed by moving objects database applications[8], such as location-based services, and are lacking in existing DBMS and GIS. The needed capabilities are location data model for moving objects, query language for moving objects, location index for moving objects, and so on.

Previous works for moving objects can be classified into two main groups; works related to location data models and query languages[1,5,8,9] and works related to indexing locations[3,4,6,7,12]. These works, also, can be classified by works for current and future locations[1,4,8,12] and works for trajectories (past locations) of moving objects[3,5,6,7,9]. Other type of previous works to is related to generate synthetic data[2,11,13]. Location data generator, which is capable of simulating real-world moving objects, are needed because it is not possible to obtain real datasets, either they do not exist or they are not accessible.

The purpose of this paper is to design the overall architecture of a Location Information Management System (LIMS) which is applicable to the real-world applications. We have integrated various kinds of works related to moving objects into the LIMS. The rest of the paper is organized as follows: In Section 2 we discuss the overall architecture of LIMS. The systems consist of three sub-systems, and these are explained in Section 3 to 5 respectively. Finally, Section 6 concludes by giving directions for future work.

2. ARCHITECTURE OF LIMS

The Location Information Management System (LIMS) devised in this paper is depicted by Figure 1. It is composed of three sub-systems, Location Acquisition Subsystem (LAS), Location Store Subsystem (LSS), and Location Query Subsystem (LQS).

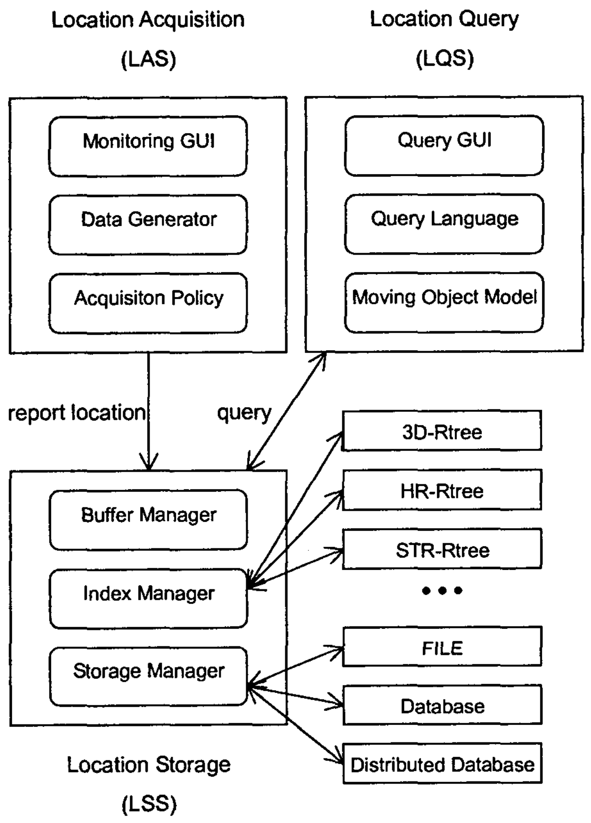


Figure 1 Architecture of Location Information Management System

- Location Acquisition Subsystem (LAS):
  According to the location acquisition policies, LAS acquire the current location of moving objects and report it into the LSS. The policies determine when LAS acquires the location of a moving object and how many threads LAS uses to acquire the locations of all objects.

- Location Store Subsystem (LSS):
  Location information that is reported from LAS is stored location indexes and location stores through the memory buffer.

- Location Query Subsystem (LQS):
  LQS executes the location query based on a moving object model newly defined in this paper. It defines data structure and operations to represent the moving objects.

3. LOCATION ACQUISITION SUBSYSTEM

LAS takes charge role that acquire the locations of moving objects and is composed as following.

- Monitoring GUI: It could monitor location information of each moving object.

- Data Generator: It could generate the synthetic data. Because it is very difficult to obtain the real datasets of moving objects, we use GSTD[2,11] algorithm and City Simulator[13] to generate synthetic datasets.

- Acquisition Policy: It could maintain the location acquisition policies.

The purpose of location acquisition policies is to minimize the communication cost to acquire locations of all of the moving objects. The basic concept of each policy is as follows. We could estimate the amount of future location changes approximately by analyzing past locations. Therefore, we could determine that the location acquisition interval is inversely proportional to the amount of future location changes. For example, if the amount of location change of a moving object gets larger, then the location acquisition interval is shortened. Consequently, if the locations of moving objects do not frequently changed, we could diminish the number of request to acquire the location.

4. LOCATION STORE SUBSYSTEM

LSS is composed of buffer manager, index manager, and storage manager. The location information reported from LAS is temporarily stored memory buffer which is maintained by buffer manager, and then they are permanently stored by index manager and storage manager.

4.1. BUFFER MANAGER

Memory buffer stores locations of a moving object into *MORow* object. *MORow* object depicted in Figure 2 is composed of *MOID* (Moving Object IDentifier), *Length* (the number of locations stored in it), *MBR* (Minimum Bounding Rectangle of the locations), *From* (time that first location in it is acquired) and *To* (time that last location in it is acquired). The *MaxLocation* means the maximum number of locations stored in a *MORow* object. If the *Length* of a *MORow* Object is equal to the *MaxLocation*, then all of the locations in the *MORow* object are transferred to storage manager in order to store permanently.

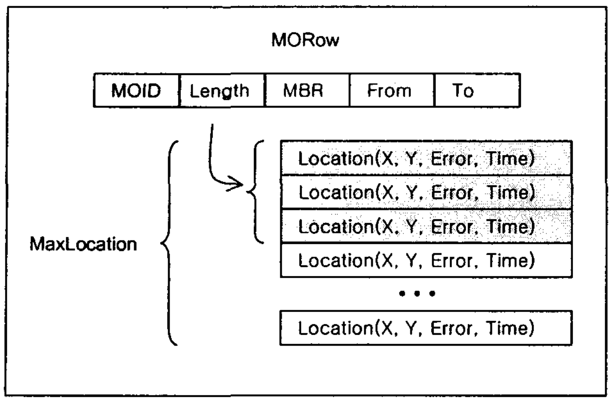

MORow
MOID
Length
MBR
From
To
MaxLocation
Location(X, Y, Error, Time)
Location(X, Y, Error, Time)
Location(X, Y, Error, Time)
Location(X, Y, Error, Time)
Location(X, Y, Error, Time)

Figure 2 Structure of MORow

Figure 3 shows the structure of memory buffer which is composed of a set of *MORow* objects. Each *MORow* object represents a trajectory of a moving object from *From* time to *To* time. As the figure indicates, there is a B-tree for indexing *MOID*s of *MORow* objects. The buffer manager, therefore, finds efficiently a corresponding *MORow* object by using *MOID*.

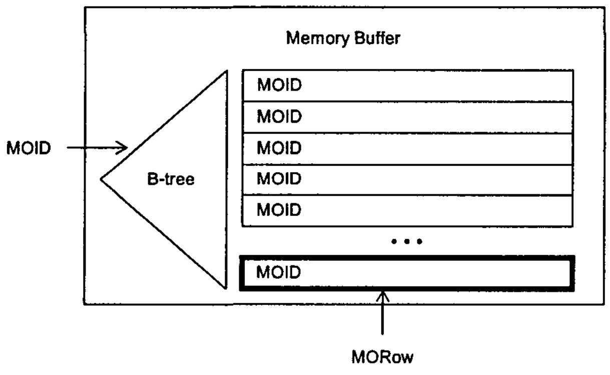


Figure 3 Structure of Memory Buffer

4.2. INDEX MANAGER

According to the previous works, there are three kinds of location indexes for moving objects. The index manger designed in this paper maintains two indexes at the same time. One belongs to current location indexes, and the other is to past location indexes.

- Current Location Indexes: The indexes[4,12] of this type take only current locations of continuously moving objects into consideration. And current locations are also used for anticipating future locations of moving objects. These indexes should have capabilities to process frequently updates of numerous moving objects.

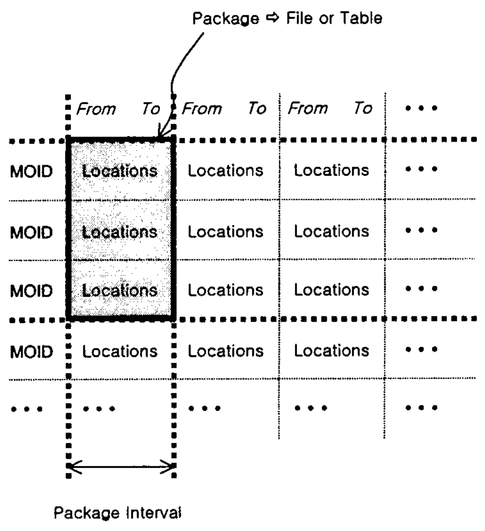
- Past Location Indexes for time interval (or time point) queries: The indexes, such as 3DR-tree[7] and HR-tree[6], have a special purpose of efficient processing of a time interval (or time point) queries for the current and past locations.

- Past Location Indexes for trajectory queries: The indexes, such as STR-tree[3] and TB-tree[3], have a special purpose of efficient processing of a trajectory queries for the past locations.

4.3. STORAGE MANAGER

Location information of moving objects is permanently stored into various types of storages by storage manager. There are three kinds of storages in this paper; files, databases, and distributed databases.

Each storage has the unit of storing locations. For example, a file is a storing unit of files storage, and a table is a unit of databases storage. The storage manager should determine amount of locations stored into a storing unit, because past locations of moving objects are infinitely increased through time. In this paper, we define a package, and store it into a storing unit. Figure 4 shows a package which is composed of a set of trajectory during a packing interval. Each package is identified by storage number which represents the list of MOIDs and *From* time of it.



Package ⇨ File or Table
From To From To From To • • •
MOID Locations Locations Locations • • •
MOID Locations Locations Locations • • •
MOID Locations Locations Locations • • •
MOID Locations Locations Locations • • •
MOID List for Storeage 001 (ID)
Package Interval

Figure 4 Package

5. LOCATION QUERY SUBSYSTEM

LQS should define location data model and location query language for the moving objects. The location data model means data structure and operations for the moving objects. In this paper, we revise the previous works[5,9], and newly define a data model, which is composed of a lot of classes such as MPoint, MLineSegment, and MPolygon.

We also revise the SQL syntax to support moving object. That is, we append the MOSELECT and MOWHERE clause to previous SQL syntax. Figure 5 shows the SQL syntax for moving objects. This extended SQL is processed as follows. SELECT, FROM, and WHERE clauses are executed by traditional databases which store an attribute data of all of the moving objects. And LOS executes MOSELECT and MOWHERE clauses and integrates the result of whole query.

```
SELECT          select_target_list
MOSELECT        moselect_target_list
FROM            table_list
WHERE           where_string
MOWHERE         mowhere_string

moselect_list   = function
                  | function , moselect_list
mowhere_string  = function
                  | function and mowhere_string
                  | not ( mowhere_string )
function        = function_name ( parameter_list )
function_name   = overlaps | snapshotbyvalidtime | slicebyvalidtime | ...
parameter_list  = parameter
                  | parameter, parameter_list
parameter       = function | object | const
object          = point() | line() | area() | period() | time() | mosmbr()
                  | string(geometry_column_name) | ...
```

Figure 5 Query Language for Moving Objects

6. CONCLUSIONS

In this paper, we have designed the location information management system for a large number of moving objects. We integrated various kinds of works related to moving objects as well as newly proposed a location data model, a location query language, and a method for storing moving objects. The system we proposed supports a diverse set of location acquisition policies, location indexes and location storages. Therefore, it is expected to be applied into various kinds of location based services. As future work, we should implement the proposed system, and develop algorithms to enhance the performance of location query processing.

REFERENCES

[1] A. Prasad Sistla, Ouri Wolfson, Sam Chamberlain, and Son Dao, "Modeling and Querying Moving Objects," ICDE, pp.422-432, 1997

[2] Dieter Pfoser and Yannis Theodoridis, "Generating Semantics-Based Trajectories of Moving Objects," International Workshop on Emerging Technologies for Geo-Based Applications, Ascona, Switzerland, 2000

[3] Dieter Pfoser, Christian S. Jensen, and Yannis Theodoridis, "Novel Approaches in Query Processing for Moving Object Trajectories," VLDB 2000, 395-406

[4] George Kollios, Dimitrios Gunopulos, and Vassilis J. Tsotras, "On Indexing Mobile Objects," ACM Symp. on Principles of Database Systems, pp261-272, 1999

[5] Luca Forlizzi, Ralf H. Güting, Enrico Nardelli, Markus Schneider, "A Data Model and Data Structures for Moving Objects Databases," Proc. ACM SIGMOD Conf. (Dallas, Texas), pp. 319-330, May 2000.

[6] M. A. Nascimento and J. R. O. Silva, "Towards historical R-trees," ACM SAC, 1998

[7] Michalis Vazirgiannis, Yannis Theodoridis, and Timos K. Sellis, "Spatio-Temporal Composition and Indexing for Large Multimedia Applications," Multimedia Systems 6(4), 284-298 (1998)

[8] Ouri Wolfson, Bo Xu, Sam Chamberlain, and Liqin Jiang, "Moving Objects Databases: Issues and Solutions," SSDBM 1998, 111-122

[9] R.H. Güting, M.H. Böhlen, M. Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider, and M. Vazirgiannis, "A Foundation for Representing and Querying Moving Objects," FernUniversität Hagen, Informatik-Report 238, September 1998, ACM Transactions on Database Systems, 25(1):1-42, 2000

[10] Y. Tao and D. Papadias, "MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries," VLDB, 2001

[11] Yannis Theodoridis, Jefferson R. O. Silva and Mario A. Naschimento, "On the Generation of Spatiotemporal Datasets," CHOROCHRONOS Technical Report CH-99-01, Proceedings of the 16th Int'l Symposium on Spatial Databases (SSD), 1999

[12] Zhexuan Song 1 and Nick Roussopoulos, "Hashing Moving Objects," MDM 2001, LNCS 1987, pp. 161-17, 2001

[13] http://www.alphaworks.ibm.com/tech/citysimulato r