4th WSEAS Int. Conf. on COMPUTATIONAL INTELLIGENCE, MAN-MACHINE SYSTEMS and CYBERNETICS Miami, Florida, USA, November 17-19, 2005 (pp99-104)

Moving Objects Management System Supporting Location Data Stream

KYOUNGHWAN AN, JUWAN KIM
Telematics•USN Research Division
Electronics and Telecommunications Research Institute
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350
Republic of Korea

Abstract: - Recently, the need for LBS (Location Based Services) is increasing due to the widespread of mobile computing devices (e.g. PDA, cellular phone, and notebook computer) and positioning technologies (e.g. GPS and RFID). In LBS, there are many applications that need to manage moving objects (e.g. taxies, persons). The MODB (Moving Objects DataBase) usually are used for the purpose of managing the moving objects. However, MODB only supports pull-based queries, although push-based queries are necessary for the monitoring application which is one of the important applications of LBS. In this paper, we extend functionalities of MODB to support push based queries. To achieve this goal, we suggest new data model, query language, and architecture.

Key-Words: - LBS, location based services, moving objects database, data stream

1 Introduction

Location Based Service (LBS) is any product, service, or application that uses locations of things or people to provide value added information. The services are enabled by mobile devices (e.g. PDA, cellular phone, notebook computer, etc.), wireless communication, and positioning technologies based on mobile network or GPS. Since the infrastructures previously mentioned are wide spread already, LBS will be more prevalent in the future. The examples of LBS are a buddy finder service to find a location of a friend, a navigation service to provide routing information to a driver, L-commerce to advertise goods based on customer's location, and "E911" service for emergency calls.

To implement those applications, DBMS should be able to support the following functions. First, it should provide a function to retrieve the locations of "moving objects". The locations include past trajectories or the locations of current/near future of the moving objects. Second, it should support triggering function. The triggering means that predefined action is performed when given condition becomes satisfied. The typical condition is "Enter", "Leave", and "mobile station available". The major applications of the triggering are L-commerce and traffic monitoring application. Third, input monitoring function should be applied. Since the reports of the locations can be enormous, several methods such as input rate control or filtering or approximation are necessary in the input of the system.

To satisfy the previously mentioned requirements, we can consider several approaches. The first approach is to use the traditional disk resident relational DBMS by extending additional functionalities. However, it has several problems. In general, there are enormous numbers of "moving objects" to be managed and queried. Since the moving objects may report their locations frequently, a database system should be able to handle a huge number of updates and queries quickly. However, traditional disk resident relation DBMS cannot handle the updates and queries efficiently. Even more it does not support data model, query language, and spatio-temporal operator specific to handling moving objects. It means application developer should concern all the things to update and retrieve the locations.

The second approach is to use the moving objects database systems (hereafter MODB) that are studied in [1, 2, 3, 4, 5, 6, 8]. They provide data model, query languages related to managing moving objects. They concentrate on the pull-based queries, which mean query results are generated when they are requested. They, however, do not provide push-based queries, which mean query results are asynchronously generated after a query is registered. Since the push-based queries are used when implementing the triggering function, MODB can not support full functionalities that are necessary for LBS.

The third approach is to extend DSMS (Data Stream Management System) [7, 9, 10, 11]. DSMS concentrates on the push-based queries, called

continuous query. However, since the previous studies do not address spatio-temporal related problems, it is difficult to apply DSMS without modification.

To cope with the problems of the previous approaches, it is reasonable to add the functionalities of DSMS to MODB. MODB can support pull-based queries while DSMS can provide push-based queries. In this paper, we designed new query language (DDL and DML), and suggest the architecture of MODB supporting location data stream. In this paper, we call the report of a moving object as location data when the data is used for pull-based queries. We call the report as location data stream when the data is used for push-based queries. The property of data is determined when table is created by using DDL. Hybrid architecture of two types of system can satisfy the requirements we suggested.

The remainder of this paper is organized as follows. In section 2, we explain related works. In section 3, we compare MODB and DSMS in more detail. In section 4, we suggest moving objects management system supporting location data stream and finally we conclude in section 5.

2 Related Works

In this section, we examine related studies of MODB and DSMS. Although DSMS does not support spatio-temporal functions, we explain main characteristics relevant to our goal.

2.1 MODB (Moving Objects Management System)

Moving objects database concerns storing and processing the locations of continuously moving objects. The extensive works were done in DOMINO[4, 5, 6] and CHOROCHRONOS project[8].

The challenge of MODB is to provide data model and query language that can handle the locations of moving objects efficiently. There are two types of approaches. The first is to model past trajectories of the moving object, which store coordinates and timestamp of the moving objects[2]. In this model, spatio-temporal operator and trajectory related operators are provided[1]. To process queries efficiently, R-tree variants such as STR-tree or TB-tree are used[2]. The second model is to find the current location or to predict the future location of the moving object, which stores start point and velocity vector[3]. In this model, access methods such as TPR-tree or TPR*-tree are used[3]. The previous study[4] suggested FTL query language to retrieve locations of moving objects.

2.2 DSMS (Data Stream Management System)

In the case of DSMS, there are several data stream prototypes. STREAM prototype[9] developed CQL (Continuous Query Language) supporting management of data stream. TelegraphCQ prototype[10] also introduces StreaQuel, which extends semantics of relational operators to streams. The query language also support window that specifies part of the stream to approximate or clarify semantic of the result. AURORA prototype[11] follows the data flow paradigm that enable users to define the flow of data stream. Currently, there are no data stream prototypes that support spatio-temporal related functions. The following figure shows typical structure of DSMS[7].

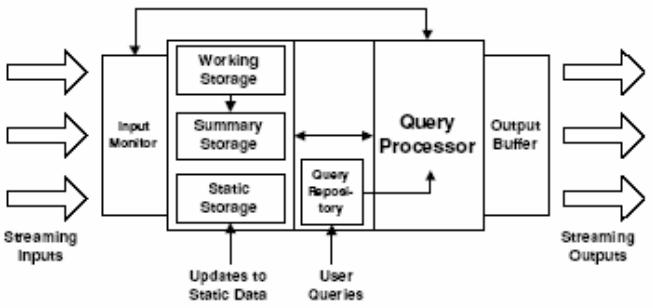

Fig.1. Structure of DSMS

An input monitor controls the input rates or support filtering functions to keep up. Data stream are stored in working storage for window queries, and stream synopses are stored in summary storage. Meta data are stored in static storage. Continuous queries are registered in query repository, and processed by query processor. Since results can be streamed to users, output buffer is used.

3 MODB vs. DSMS

In this section, we compare MODB with DSMS. Table 1 summarizes functionalities of each system.

Table 1. Supported Functionalities

Required Functionalities	MODB	DSMS
Spatio-temporal	O	

data model		
Nearest-neighbor query	O	
Range query (Spatio-temporal)	O	
Past query (Trajectory)	O	
Future query (Future Location)	O	
Spatio-temporal operator	O	
Trajectory operator	O	
Window		O
Continuous query		O
Streamed Input/Output		O

Since MODB focuses on the spatio-temporal properties of the moving objects, it provides plenty of constructs to process locations of the moving objects. To the contrary, DSMS focuses on processing data stream which is typically assumed unbounded. In this paper we combine the functionalities of MODB and DSMS to achieve our suggested requirements.

4 MODB Supporting Location Data Stream

4.1 Data Model

In the real world, moving objects like car show continuous movement. However, the continuous movement can not be represented in the database. Thus the movement should be sampled discretely and interpolated between sampled locations.

To track the past movement of the moving objects, we suggest several data types shown in table 1. In table 1, x and y means 2-dimensional coordinates of moving objects. The type MPoint is the most frequently used data types for modeling the moving objects. Cars can be modeled as a point at a specific time. As the time progresses, the time value and geographic coordinates are accumulated in MPoint. The semantic of the other data types can be easily interpreted without confusion. To use these data types, the moving objects should report the time value and the geometry simultaneously.

Table 2. Data Types for the Past Locations

Type	Well Known Text
MPoint	MPOINT (t, x, y, t, x, y, … …)
MLineString	MLINESTRING ((t, x, y, x, y, …), (t, x, y, x, y, …) …)
MPolygon	MPOLYGON ((t, (x, y, x, y, …),(x, y, x, y, …)), (t, (x, y, x, y, …), (x, y, x, y, …) …) …)
MultiMPoint	MULTIMPOINT ((t, x, y, t, x, y, …), (t, x, y, t, x, y), …)
MultiMLineString	MULTIMLINESTRING (((t, x, y, x, y …), (t, x, y, x, y …)), ((t, x, y, …),…))
MultiMPolygon	MULTIMPOLYGON ((((t, (x, y, x, y …), (x, y, x, y …)), (t, (x, y, x, y …), (x, y, x, y …))…), (((t, (x, y, x, y …), (x, y, x, y …)), (t, (x, y, x, y …), (x, y, x, y …))…))
MGeometryCollection	MGeometryCollection ((t,(Geometry)), (t,(Geometry)), …)

To query the current and future locations of the moving objects, the different data modeling is necessary. The future location means the location should be anticipated and calculated. To model the future location, we use a vector approach. For example, x or y coordinate can be computed by the following formula.

$$x = a + vt$$

The a means the current x coordinate, v means the speed of the x direction, and t means the time to be input to the query system. To make it possible to query the current and future location of moving objects, the moving objects should report their current location, speed, and direction. Also they report the values whenever they change their speed or direction over the specified threshold.

4.2 Query Language

We suggest query language for DDL and DML. In DDL, we can specify properties of location data stream when creating tables, and register continuous queries. In DML, we can provide data stream related syntactical constructs besides constructs of MODB.

❒ DDL (Data Definition Language)

We provide two kinds of CREATE statement. To enable continuous queries, the CREATE statement

should have additional constructs. Since internal structure is different from location data stream and normal location data.

The first syntax supports pull based queries. The overall syntax is similar to the traditional syntax. However, the developer should use the data types specific to the moving objects. If the purpose of the creation is to store the past trajectories of the moving objects, new constructs like "MIGRATE" or "PURGE" can be used. "MIGRATE" is applied to the so-called main memory DBMS. It means the trajectories migrate from memory to disk according to the given condition. "PURGE" is similar to "MIGRATE". It means the part of the trajectories should be removed when the given condition is met. The following shows the syntax of DDL for the creation of the table.

CREATE TABLE <table_name>
({ColumnDefinition}[,……]
[,TableConstraints]
[, **MIGRATE** (**TIME**|**SPACE**) (MigrateConstraint)]
[, **PURGE** (**TIME**|**SPACE**) (PurgeConstraint)]);

Example1 shows the creation of the table using "MIGRATE". The table truck has the field called position, which is the MPOINT type. Also it maintains trajectories only for 30 days. After the migration, query processor should be able to process queries on the trajectories both in main memory and disk.

Example1)
CREATE TABLE truck
(id INT, position MPOINT,
MIGRATE TIME 30 days);

Example2 shows the creation of the table using "PURGE". The trajectories in the table car are automatically deleted when it exceeds 100MB. When the migration occurs, meta data should be maintained to notify users of the fact that there were trajectories but they were deleted.

Example2)
CREATE TABLE car
(id INT, position MPOINT,
PURGE SPACE 100MB);

The second syntax supports push based queries. The syntax includes new syntactical constructs such as STREAM, ARCHIVE, and SOURCE. The STREAM means this table supports push-based queries. If a table is defined using this construct, the table becomes append-only. The ARCHIVE construct determines whether to delete processed data or not. If DISCARD construct is selected, queries referencing past data are not supported. If STORE construct is selected, MIGRATE or PURGE condition can be specified. Data stream can be added by using INSERT statement or by defining user-defined functions. The SOURCE construct is used to map user-defined function. The INIT construct means the registered function will be called before the stream begins. The NEXT construct means the function will be called whenever data are available. The DONE construct means the function will be called when data stream end. The user-define functions can be created using "CREATE FUNCTION" statement.

CREATE STREAM <stream_name>
({ColumnDefinition}[,……]
[,TableConstraints]
[, **ARCHIVE** (**STORE**|**DISCARD**) [(Migrate, Purge Constraint)]]
[, **SOURCE INIT** <function_name>]
[, **SOURCE NEXT** <function_name>]
[, **SOURCE DONE** <function_name>]);

Example 3 shows the creation of the stream with DISCARD specified. The specified functions are called by the system according to the arrival of the data stream.

Example3)
CREATE STREAM loc_stream
(id INT, position MPOINT
, **ARCHIVE DISCARD**
, **SOURCE INIT** loc_stream_init
, **SOURCE NEXT** loc_stream_next
, **SOURCE DONE** loc_stream_done);

Example4 shows the creation of the stream with STORE specified. Since STORE is specified, MIGRATE is specified. The query migrates data to disk after 30 days.

Example4)
CREATE STREAM trace_stream
(id, INT, position MPOINT
, **ARCHIVE STORE**, **MIGRATE TIME** 30 days
, **SOURCE INIT** loc_stream_init
, **SOURCE NEXT** loc_stream_next
, **SOURCE DONE** loc_stream_done);

The definition of the stream can be altered or dropped by the following syntax.

ALTER STREAM <stream_name> ({Stream Definition});

DROP STREAM <stream_name>;

❒ DML (Data Manipulation Language)

A. Update Statement

The update is the most characteristic feature of the moving objects database. The following example shows the new construct "add" to update a trajectory of a moving object.

Example5)
INSERT INTO truck **VALUES** (1, importfromwkt ('MPOINT(2005/04/01 12:00:01, 1, 1'));
UPDATE truck **APPEND** position = importfromwkt('MPOINT(2005/04/01 12:05:00, 4, 4)') **WHERE** id = 1;

First, a record representing a moving object should be inserted. After the insertion, the reported locations should be reflected in the database. The traditional SQL syntax of the update statement was to use a pair of update and set. However, we used "append" instead of "set" because the newly reported location does not replace whole trajectory but just appended at the end of the trajectory. In case of not maintaining the past trajectories, "set" construct can be used.

B. Retrieval Statement

The following syntax is used for pull-based queries or ad-hoc queries. The ad-hoc query means the query is posed to the system after the source stream begins. The following syntax has new constructs such as WINDOW and EXECUTE. The WINDOW is used for the STREAM. The previous studies only supported time interval. However, we suggest additional WINDOW option: spatial range and spatio-temporal range. The EXECUTE constructs can be used both type of table. It specifies duration and interval for execution of the SQL statement.

SELECT <select_list>
FROM <relation | stream>
WHERE <predicate>
[**GROUP BY** <group_by_expression>]
[**ORDER BY** <order_by_expression>]
[**WINDOW** <time interval | spatial range | spatio temporal range>]
[**EXECUTE INTERVAL** <time> **FOR** <time>];

Example6 shows the example of the previous syntax. It retrieves id and position of the table car every five seconds for thirty hours.

Example6)
SELECT id, position
FROM car
EXECUTE INTERVAL '5s' **FOR** '30hr';

To register a continuous query (push-based query) before stream begins, the following syntax can be used.

REGISTER CONTINUOUS QUERY <cq_name> (<SQL-statements>);

Example7 shows the example of REGISTER syntax. It uses WINDOW to narrow query results.

Example7)
REGISTER CONTINUOUS QUERY cq_loc
(**SELECT** id, position
FROM loc_stream AS loc
WHERE within(loc, MPOLYGON(…))
WINDOW loc ['10 min']
);

The continuous query continues until it is explicitly cancelled. The following syntax controls the execution of the continuous query. The DEACTIVATE/ACTIVATE pauses or restarts the continuous query, while the DROP cancels the registered query.

DEACTIVATE CONTINUOUS QUERY <cq_name>;

ACTIVATE CONTINUOUS QUERY <cq_name>;

DROP CONTINUOUS QUERY <cq_name>;

4.3 Architecture

The architecture of the proposed system is as follows.

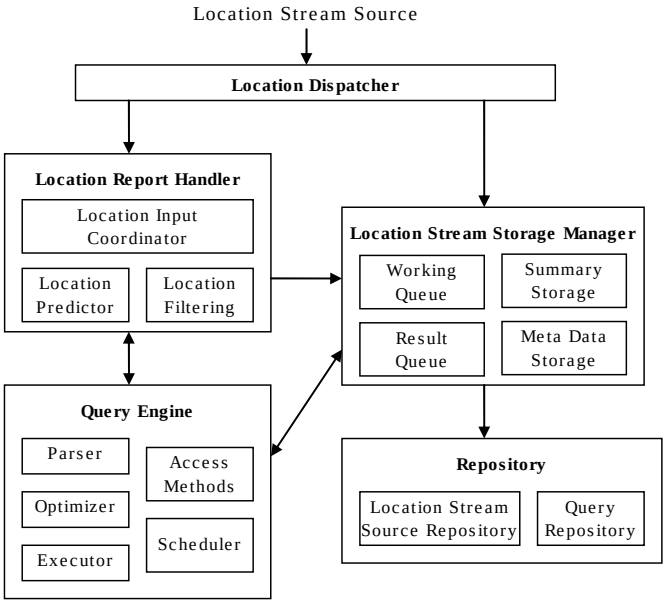

Fig.2.Architecture of the Proposed System

- **Location Dispatcher:** this module receives location reports from the moving objects. It supports both pull-based source and push-based source. In the case of push-based source the functions defined in the CREATE STREAM statement are called. It transfers the location to location report handler or location stream storage manager according to the configuration of the source.

-**Location Report Handler:** this module handles the location reports to reduce the overhead of processing location data. It controls the input rate of the location reports or reduces the number of the location data while maintaining quality or performs filtering.

-**Location Stream Storage Manager:** this module stores and manages the location data. It consists of working queue for query processing, summary storage for synopsis information, result queue for streaming results, and meta data queue for meta information of data source and QoS.

-**Query Engine:** this module processes registered queries or real-time queries (pull-based queries and ad-hoc queries). It can optimize multiple queries using shared global queue.

-**Repository:** This module manages and registers location data stream and continuous queries.

5 Conclusion

In this paper, we proposed moving objects management system supporting location data stream. To support all functionalities required by LBS applications, we added the functionalities of DSMS to MODB. To achieve the goals, we introduced data model, query language, and system architecture. The proposed system can support both the pull-based queries and push based queries. In the future, the system can be used as database system for ubiquitous sensor network.

References:

[1]R. H. Guting, M. H. Bohlen, M. Erwig, C. S. Jensen, N. A. L. M. Schneider, and M. Vazirgiannis, “A Foundation for Representing and Querying Moving Objects,” In ACM-Transactions on Database Systems Journal, pp.1-42, 2000

[2] D. Pfoser, C. S. Jensen, and Y. Theodoridis, “Novel approaches in query processing for moving objects,” Proc. of Int’l Conf. on Very Large Data Bases, pp. 395-406, 2000

[3] S. Saltenis, C. S. Jensen, S.T. Leutenegger, and M. A. Lopez, “Indexing the positions of continuously moving objects,” Proc. of the ACM SIGMOD Int’l Conf. on Management of Data, pp. 331-342, 2000

[4] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, “Modeling and Querying Moving Objects,” In Proc. of the International Conference on Data Engineering, pp.422-432, 1997

[5] G. Trajcevski1, O. Wolfson, F. Zhang, and S. Chamberlain, “The Geometry of Uncertainty in Moving Objects Databases,” in EDBT 2002, LNCS 2287, pp. 233-250, 2002

[6] O. Wolfson, S. Chamberlain, K. Kalpakis, and Y. Yesha, “Modeling Moving Objects for Location Based Services,” in IMWS 2001, LNCS 2538, pp. 46-58, 2002

[7] L. Golab and M. T. Ozsu, “Issues in Data Stream Management,” in SIGMOD Record, Vol.32, No. 2, pp. 5-14, 2003

[8]CHOROCHRONOS: http://www.dbnet.ece.ntua.gr/~choros/

[9] STREAM: http://www-db.stanford.edu/stream

[10] TelegraphCQ: http://telegraph.cs.berkeley.edu

[11]Aurora: http://www.cs.brown.edu/research/aurora