Adapting Relational Database Engine to Accommodate Moving Objects in SpADE *

Beng Chin Ooi Zhiyong Huang Dan Lin Hua Lu Linhao Xu
School of Computing, National University of Singapore, Singapore
{ooibc, huangzy, lindan, luhua, xulh}@comp.nus.edu.sg

1 Introduction

To cope with the increasingly growing mobile data, a real system supporting moving objects storage and retrieval is of importance. There are three approaches to implementing an MOD (Moving Objects Databases) system. One, a specialized system could be developed from scratch [1]. Such a system is lean and efficient. However, it may offer less functionalities that are required in conventional business processing, and such approach is time consuming and costly. Two, an existing DBMS could be modified to provide supports for moving objects inside the DBMS [3]. This approach requires good understanding of the DBMS and careful modification as any alteration to it may cause its component to be affected. Three, the required capability for handling spatio-temporal data and processing are built on top of an existing DBMS [4]. Such an approach is less costly and its existing core database engine can continue to serve the conventional business processing, and complement the MOD applications. This is in line with the approach being taken in the commercial DBMS products where different cartridges are built on top of the data server for different applications. Our work falls into this category.

In this work, we present our implementation for managing moving objects on top of a popular relational database system MySQL, namely SpADE (*Sp*atio-temporal *A*utonomic *D*atabase *E*ngine for managing moving objects). In our SpADE system, non-static entities like vehicles and pedestrians are abstracted as moving objects. They obtain positioning information with GPS (Global Positioning System) receivers installed, and are able to communicate via wireless network with the server, sending queries to and receiving results from it. The server is responsible for managing moving object information and processing queries from mobile users. By employing the industry standard JDBC for the data access, our server can also support providing services for other application interfaces such as the Web.

The crucial part of our SpADE system is the implementation of the B^x-tree [2] on top of MySQL, as the index service that helps to manage and query moving objects. In our implementation, moving object data is elaborately transformed and stored directly in MySQL, and queries are transformed into standard SQL statements which are efficiently processed in the relational engine. Most importantly, all these are achieved neatly and independently without infiltrating into the MySQL core.

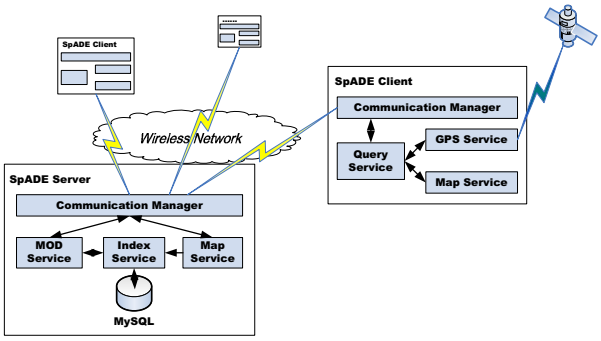

Figure 1. System Architecture

Figure 1 illustrates the SpADE system architecture. A client includes four components. *GPS Service Module* is responsible for receiving and parsing the GPS information. *Map Service Module* is responsible for displaying underlying map and moving objects if any, and listening to interaction events from the user. *Query Service Module* constructs spatial-temporal queries according to user triggered events on the map. *Communication Manager* sends requests to and receives data from the server.

The SpADE server comprises five modules. *Communication Manager* receives query messages or location update requests from the clients, dispatches them to other modules, and returns query results to the client user. *Map Service Module* maintains the map data displayed at the client side and notifies all clients to update their map if a new map is provided. *MOD Service Module* processes position updates and query requests from mobile clients. *Index Service Module* constructs B^x-tree index to facilitate processing spatial-temporal queries. *MySQL Database* stores the moving objects information in relational table and provides B^+-tree as

*This work was supported by A*STAR under grant no. 032 101 0026.

the underlying basis for B^x-tree.

2 System Implementation

2.1 Index Service Module

A moving object in our SpADE system is modelled by linear motion. By applying an attractive space-filling curve like Peano curve or Z-curve, a 1-dimensional value is generated for each moving object with known position and velocity. That value is called *B^x value* [2]. The index service module implements B^x-tree functionalities including the B^x value computation.

2.2 Relational Table Definition

We define for moving objects a proper relation scheme that can be indexed by the B$^+$-tree based B^x-tree. For this purpose, we add a field B^x_Value into the table of moving object and make it as the primary key indexed by the B$^+$-tree. A simple relation scheme is demonstrated in Table 1. Extra fields can be added according to specific practical needs.

Field	Type	Note
id	integer	Object identifier
$B^x\_Value^*$	long	Primary key indexed by B$^+$-tree
pos_x	double	Longitude of last update
pos_y	double	Latitude of last update
vel_x	double	Longitude speed of last update
vel_y	double	Latitude speed of last update
$time$	long	Update time

Table 1. Moving Object Relation Scheme

In our system, there are two such tables. One is to store the latest update information of the moving objects, called the *current information table*. Note that this table is indexed by the B^x_Value obtained according to the B^x-tree rationale. The other table simply keeps all the historical information, called the *historical information table*.

2.3 Processing Spatial-Temporal Queries

The SpADE system supports various types of queries such as historical (current/predictive) range and k-NN queries. The typical process of executing such a spatial-temporal query takes several stages, as shown in Figure 2. On the client side, a user selects the query type and specify query parameters by dialog based inputs or map based stylus actions. Then, a corresponding query request is constructed and sent out to the server. Upon receiving a query request, the server side communication part extracts query parameters from it and passes them to the MOD service module. Then SQL statements are composed according to the query parameters and the B^x-tree rationale, and passed to the MySQL query engine via the JDBC connection. Each of such SQL statements specifies a range query on the B^x value. Inside the MySQL, SQL statements are executed

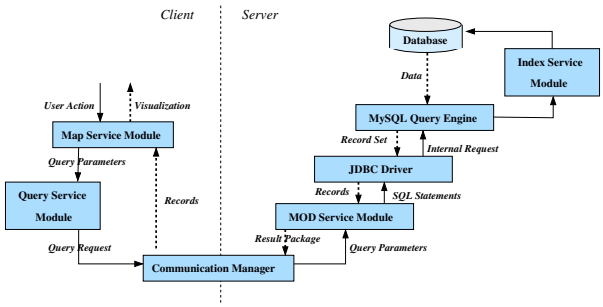

Figure 2. Execution of a Query

with the aid of the B$^+$-tree index to retrieve the data requested. The data then will be returned back along the reversed direction till the client side.

2.4 Updating Spatial-Temporal Data

To query moving objects correctly, the velocity and position of all moving objects must be kept fresh. The B^x-tree rationale requires that each moving object updates its velocity and position within a maximum period of time to the server. This period of time can be determined based on the scenario of a real application of moving objects.

Generally, the SpADE system takes four steps to refresh the velocity and position information of all moving objects. In the first step, the client obtains the velocity and position information from the GPS service module and then transfers them to the server. In the second step, according to the identifier of the moving object, the old information is removed from the table and inserted into a history table. In the third step, the index service module calculates the index key (B^x value) of the B^x-tree in terms of the new velocity and position data. Finally, the new index key, velocity and position data are inserted into the MySQL database.

3 Demonstration Outline

An environment, with a laptop as the SpADE server and one or more PDAs as clients, will be established to illustrate SpADE's features. Synthetic data sets will be used to demonstrate the moving objects management and query processing.

References

[1] B. Huang, Z. Huang, D. Lin, H. Lu, Y. Song and H. Li. ITQS: An integrated transport query system. In *Proc. SIGMOD*, pp. 951–952, 2004.

[2] C. S. Jensen, D. Lin, and B. C. Ooi. Query and update efficient B$^+$-tree based indexing of moving objects. In *Proc. VLDB*, pp. 768–779, 2004.

[3] M. F. Mokbel, X. Xiong, W. G. Aref, S. E. Hambrusch, S. Prabhakar, and M. A. Hammad. PLACE: A query processor for handling real-time spatio-temporal data streams. In *Proc. VLDB*, pp. 1377–1380, 2004.

[4] O. Wolfson, A. P. Sistla, B. Xu, J. Zhou, S. Chamberlain, Y. Yesha, and N. Rishe. Tracking moving objects using database technology in DOMINO. In *Proc. NGITS*, pp. 112–119, 1999.