PREDICTIVE QUERYING IN SPATIO-TEMPORAL ENVIRONMENT

Sam Makki, Ho Ling Hsing, Demetrios Kazakos
Department of Electrical Engineering and Computer Science
University of Toledo, Toledo, Ohio, USA

Keywords: Spatial, temporal, tree, query, index, moving objects

Abstract: Moving objects are spatial objects in which their positions change over the time. The process of storing the location information and processing queries efficiently on these moving objects are challenging problems in spatio-temporal databases. Many researches have been conducted to address the storing and querying problems related to moving objects. The majority of these researches concentrated on modifying and optimizing the indexing techniques for querying moving objects. These indexing approaches update and retrieve the locations of moving objects by traversing the nodes and inserting and deleting the nodes in the index structures. These insertion and deletion operations eventually lead to the rebuilding of the index structures in order to maintain query performance. However, periodic rebuilding of index structures can be expensive and it should be avoided if possible. To tackle this problem, we propose alternate method to query the positions of moving objects. The proposed method uses a double-grid structure that eliminates the need for insertion and deletion operations during updates and retrieval. The advantages of using a double-grid structure instead of indexing are the significant improvement in time for querying moving objects, and the elimination of the need to rebuild the grid structure.

1 INTRODUCTION

The process of storing and processing queries on the location of dynamic objects produced a number of new and challenging issues, as the traditional databases can only store and allow querying of static data. The static data has either spatial or temporal characteristics. However many objects in real world are not static but rather they are dynamic (or moving). Therefore neither the spatial nor temporal database is able to store these objects. To store these objects in a database, such database must have the capabilities of both spatial and temporal databases. If these capabilities are integrated, then moving objects can be represented in a database. This type of database is termed spatio-temporal database.

Spatio-temporal databases are the same as spatial databases except the objects are moving. Spatio-temporal databases usually involve the representation and querying of physical objects. Such a database can be used to provide location based services because it keeps track of the locations of different objects (Wolfson).

In traditional databases, the attributes of an object are static, which means that these attributes change only when users make explicit updates to them. On the contrary, in spatio-temporal databases, the attributes of a moving object are called dynamic attributes which are attributes that change constantly as a function of time (Sistla, et al, 1997). An example of a dynamic attribute are the coordinates of a moving object (x,y). Whenever a dynamic attribute is queried, the returned answer represents the set of possible values of the dynamic attribute at the time the query is submitted. In other words, the spatio-temporal model that uses dynamic attribute is different from traditional database systems, since traditional database systems return the same answer from the attribute regardless of the time at which the query is submitted. The answer may be different at 1:00pm and 1:30pm even though the database has not been explicitly updated during this 30-minute period. Another important difference between traditional databases and spatio-temporal databases is that the answer to future queries is tentative. This means that the query results returned by spatio-temporal databases are accurate based on what are currently known about the moving objects. However, since this current information can change, the answers to the queries can change as well. Indexing dynamic attributes enables answering queries regarding moving objects without examining all the objects.

Spatio-temporal databases usually represent objects' movements as motion functions. The most common function represents linear motion because it requires the minimum number of parameters. Moreover, it can be used to describe more complex movements using interpolation.

To store moving objects in a database, their motion information can be stored using the motion parameters and location functions. The location functions are used to compute the positions of any moving objects at any time (Porkarw et al, 2001). Depending on the type of applications, initial values of motion parameters such as starting location and velocity are required. The location function can then predict the position of a moving object at any time.

After the motion information of a moving object is stored in a database, the next step is to determine how often the motion information can be updated since the location of the moving object is constantly changing. If the database is updated every time when the object is moved, then tremendous amounts of resources will have to be used and yet the query result will still not be precise. So a different approach has to be taken to handle the updating of the motion information.

There are a number of approaches that can be used to update the location of a stored moving object. One of the approaches is that, updates are triggered whenever there is a change in the motion parameter (Wolfson). The second approach is that an update is necessary only when the velocity changes. The other approach is the use of a deviation threshold (Meng et al, 2003). The deviation threshold of a moving object can be set first and the database is updated only when the deviation exceeds this threshold value. The actual update can include the current location, current velocity, and current direction.

The majority of papers (Porkarw et al 2001, Theodoridis et al, 1998) proposed the use of the tree structure for indexing moving objects stored in spatio-temporal databases. An alternative method to query the positions of moving objects is proposed in this paper. The proposed method involves the use of a double-grid index structure that can represent moving objects' data in spatio-temporal databases.

The topics discussed and the order that they are presented in this paper are as follows: Section 2 introduces the types of Spatio-temporal Queries. Section 3 describes the query languages for spatio-temporal databases and gives examples of the query languages. Section 4, 5 describe the different types of access methods and indexing queries and their shortcomings. Section 6 describes our proposed method in representing moving objects. Section 7 gives conclusions and directions for future work.

2 TYPES OF SPATIO-TEMPORAL QUERIES

There are various types of spatio-temporal queries that can query a moving object based on the type of application. They are range query, k-nearest neighbor query, and spatio-temporal join query.

A range query retrieves the objects that are within distance x from a region R within a query interval T (between times t1 and t2) (Wolfson). The following sentence is an example of a range query. "Retrieve all the trucks that are within 10 miles from the University of Toledo between 2pm and 3pm".

A k-nearest neighbor query specifies a query point and retrieves the k objects that will come closest to the query point during the query interval T (Kollios 2003). An example of a k-nearest neighbor query is "Retrieve all the taxi cabs that are closest to 1234 Douglas road at around 10am".

A spatio-temporal join query will return all pairs of objects from two sets of data that will come within distance d from each other during the query interval T (Kollios 2003). For example, "Retrieve the pair of taxi cabs that will be within 2 miles from each other".

These three types of queries (i.e. range query, k-nearest neighbor query, and join query) can be sub-categorized based on the time interval of the query. These sub-categories are past queries, present queries (these are the type of queries that traditional databases can handle), future queries, and triggers (Wolfson).

The followings are some examples:

1- Past Query: How many airplanes in LAX were delayed by more than an hour in 2003?
2- Present Query: How many restaurants are there within 2 miles of my current location?
3- Future Query: Retrieve the trucks that will reach a certain destination within 20 minutes.
4- Trigger: Send me a message when the mailman is within 1 mile of my home address.

3 QUERY LANGUAGES FOR SPATIO-TEMPORAL DATABASES

Structured Query Language (SQL) works well for static data but it will be cumbersome if it is used for querying the locations of moving objects. The reason is that SQL do not have temporal operators that can be used to describe the temporal aspects of moving objects (Sistla, et al, 1997). Therefore a query language that is capable of querying moving objects, but simpler to use is needed. One of the proposed query languages is called Future Temporal Logic (FTL) language (Wolfson). FTL has SQL/OQL type syntax. As an example consider the following query.

Retrieve all cars that will enter highway H in the next 10 minutes and stay on H for 20 minutes.

```
RETRIEVE C.type
WHERE Not Inside(C,H)
AND Eventually-within-10 (Always-for-20 (Inside (C,H))
AND H.type = highway
```

4 SPATIO-TEMPORAL DATABASES ACCESS METHODS

The location of any moving objects at any particular time has to be modelled or predicted before it can be queried. Over the years, many spatio-temporal access methods have been developed to support spatio-temporal queries. Most of the proposed spatio-temporal access methods use a multidimensional index structure for storing and querying the positions of moving objects. The goal of multidimensional index structures is to index a large number of 3-D trajectories in order to perform efficient querying and updating of trajectories. Most of these access methods are tree structures (Porkarw et al 2001, Theodoridis et al, 1998) and the various tree structures from different papers are basically derived from R-tree (Guttman, 1984), which is a spatial access method. Besides indexing the locations of moving objects, indexing the queries of moving objects has also been proposed by Kalashnikov et al, 2002. Query indexing is especially useful when there are numerous concurrent, continuous queries over large numbers of moving objects.

The spatio-temporal access methods can be grouped into three categories based on the type and time of the queries they can support (i.e. the past, current, and future queries) (Mokbel et al, 2003). The three categories include methods that are:

1- indexing the past (i.e. index historical spatio-temporal data)
2- indexing the current (i.e. keep track of the current status of spatio-temporal data)
3- index the future (i.e. help answer/predict queries related to the future)

However according to Kalashnikov et al, using multidimensional indexes requires constant updating if the moving objects keep moving and this leads to poor query processing performance. To counter this problem, many indexing methods proposed to limit the updating frequency of the locations of moving objects by using motion functions. However, this is also a drawback because as the updating frequency is decreased, the uncertainty of the positions of moving objects increases.

Most of the indexing methods employ the idea of MBRs to represent moving objects. MBRs are a good idea for representing static objects or objects with low mobility (e.g. multimedia objects) but they can introduce a lot of unused space (Nievergelt et al, 1984) in the index structure when used to represent objects with high mobility and this leads to inefficient indexing (Theodoridis et al, 1998).

Furthermore in the case of moving objects, since it is expected that there are many insert and delete operations in the tree structure, the query performance can deteriorate over time. This is attributed to the fact that the tree structure may eventually become unbalanced. When this happens, the whole tree structure will have to be rebuilt periodically to ensure it is working under optimum performance level. However, rebuilding an index is an expensive operation and it should be avoided if possible.

5 INDEXING THE QUERIES

The indexing methods that have been mentioned so far indexes the moving objects with tree structures according to their spatial proximity. However, if the locations of moving objects are continuously queried, these indexing methods will suffer from the problems of constant updating which will result in poor performance. To tackle this problem, an indexing method with a different approach was proposed by Kalashnikov et al, 2002. This particular indexing method attempts to index the possible set of queries for moving objects instead of their locations. In particular, these possible queries are continuous range queries.

In contrast to regular queries that are evaluated once, continuous queries stay active over a period of time and have to be continuously evaluated during this period of time. If there are a considerable number of queries and moving objects, it is impossible to keep evaluating each query whenever an object moves. For this reason, Kalashnikov et al, stated that the traditional approach is not a practical approach for indexing moving objects. They believed that evaluating all queries periodically with consideration of the most recent positions of moving objects is more preferable. In order for the results of the continuous query to be useful, the goal of evaluating continuous queries should be, to keep the evaluation period as short as possible.

As the name implies, query indexing is about indexing queries, not about indexing the locations of moving objects. In comparison, the set of queries change less frequently than the set of locations of moving objects. Therefore, there is no need to change the query index unless the set of queries change. Furthermore, this approach does not impose any constraints on the velocities or routes taken by the moving objects. In other words, the moving objects can move in any way they desire without affecting the query performance.

Query indexing uses the simple one-level grid index approach and the grid index is a 2-D array of cells. The grid index is partitioned uniformly in cells and each cell represents a space region. This is a better approach than other traditional indexing approaches such as Quad-tree, especially when the data is skewed. This is because the grid structure will not become lopsided like tree structure.

Although this approach does not make any assumptions about the velocity or route taken of moving objects, it does assume that the information about updated locations of these moving objects are already available. How often the location information of moving objects is updated has a significant impact on the query performance and precision. Without considering this factor, the actual performance of the query indexing approach cannot be accurately determined.

5.1 Update and Retrieval

The indexing methods mentioned in (Porkarw et al 2001, Theodoridis et al, 1998) use the tree structure to index moving objects. Updates sometimes require inserting or deleting nodes when there are overflows or underflows respectively. This is an efficient method for static objects but not for moving objects. There can easily be overflows or underflows in the tree structure because of the nature of moving objects. Numerous insertions and deletions of nodes in a tree can have significant impacts on the query performance. Additionally, the whole tree structure might have to be traversed before a query result can be returned. However, this is not the case for the grid structure. Retrieval using a grid structure can be done in a shorter time than tree structure because there are no nodes to traverse. The location of a certain moving object can just be retrieved at the appropriate cell in the grid. Likewise, when the location of a certain moving object needs to be updated, it can be updated at the cell where the previous location of the moving object is stored.

5.2 Index Rebuilding

When there are numerous insertions or deletions of nodes in a tree structure for a certain period of time, it is very likely that the tree structure becomes unbalanced, which leads to deterioration in query performance. This will inevitably require rebuilding, which reorganizes the structure of the index to eliminate fragmentation, to maintain optimal query performance. However, rebuilding is an expensive operation in terms of time and I/O resources. Since a grid index is always a balanced structure, it does not require rebuilding no matter how many updates there are to the index. In other words, the grid index can maintain its optimum query performance even without rebuilding.

6 PROPOSED METHOD

As established in previous sections, grid structures are preferable over tree structures in terms of the time required to update and retrieve the locations of moving object and the need to rebuilding the index structure. According to (Prabhakar et al., 2002), a no-index strategy yields better performance when querying moving objects. Therefore, instead of developing an indexing approach which is based on R-tree or its derivations. We propose to query moving objects using a double-grid index structure.

The propose method is based on following two criteria for improving query operation of moving objects: firstly it can update and retrieve in shorter time than the conventional tree structure approach. Secondly it does not require periodic rebuilding.

The double-grid index structure as depicted in Figure 1 uses two grid structures, the first grid is called the 'actual location grid' and the second grid is called the 'storage grid'.

The actual location grid encloses the actual moving objects (along with their IDs and pointers) which can move in any one of the cells in the grid structure.

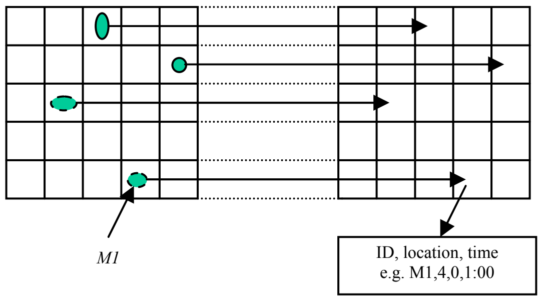

Figure 1: The actual location grid and storage grid.

The IDs and pointers are linked to the records of the moving objects' location in the storage grid. The actual locations of various moving objects are assumed to be received via Global Positioning System (GPS) which is a widely used technology, (GPS is used to track the locations of moving objects). The 'storage grid' stores the moving objects' IDs, locations, and the time at which they move to the location. Since there can be more than one moving object in a cell, each cell in the storage grid can store multiple records for multiple moving objects.

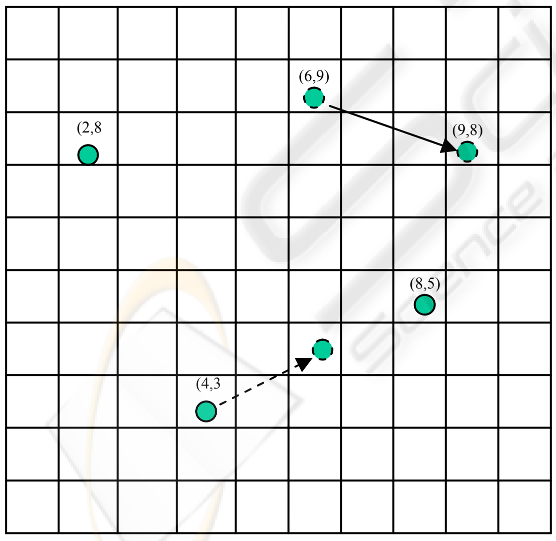

Figure 2: There are four moving objects in the actual location grid here.

The coordinates for thefour moving objects are M1(2,8), M2(4,3), M3(6,9), M4(8,5). M2 and M3 move to different grids.

As an example as shown in Figure 2, suppose that the IDs of the moving objects are M1, M2, M3, and M4 and their respective initial locations are (2,8), (4,3), (6,9), and (8,5) in the actual location grid. Assume M3 moves from (6,9) to (9,8). Therefore the corresponding records in the storage grid for these objects are shown in Figure 3.

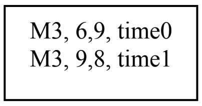
M3, 6,9, time0
M3, 9,8, time1

Figure 3: Storage grid stores records for each moving object.

If M3 is being queried about its location, the coordinates will be retrieved depending on the nature of the query. If it is a past query, then the query result is (6,9). If it is a present query, then the query result is (9,8). If the query asks about the future location of M3, then the future location will be estimated by a motion function. The motion function is a function of time which is used to estimate the future location of a moving object by considering its current and/or past locations, direction, and velocity. The concept of motion function is introduced by (Sistla et. al., 1997) and there is no known alternate method so far for modelling moving objects (Chon et. al., 2001).

Each moving object has different patterns, so they each will have a motion function. Suppose the motion function for moving object M3 predicts that it follows an almost straight route, then the cell (8,3) will be returned as the query result. Retrieving the moving object location directly from the appropriate cell is apparently faster than from a tree structure, since in a tree structure the path to the proper location has to be traversed. Also, updating the moving object location in a grid structure is faster than in a tree structure. Suppose M3 suddenly changes direction (the dotted arrow in Figure 4a). Whenever the predicted location from the motion function is different from the actual location, the motion function will initiate a location update. In the case of M3, the updated location is (7,5) and the motion function will provide a different predicted location. So whenever an update is necessary, it does only involve updating the required record in a cell. Therefore there is no need for traversing, inserting or deleting of the cells. This certainly will shorten the time required for the update operation. The number of cells in a grid structure and the structure itself remain the same at all times. So even after numerous updates, there is no time penalty imposed when retrieving a query result from a cell in a grid structure. Also the query performance stays the same after frequent updates because there is no need for rebuilding the structure. Figure 4b shows the past and current locations' records for object M1, while Figure 4a shows the result of motion function which is used to predict the future location of moving object from one location to the next.

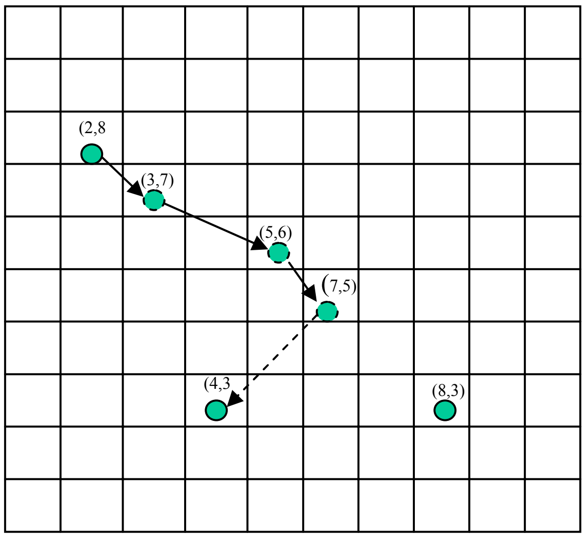

Figure 4a

M1, 2,8, time0
M1, 3,7, time1
M1, 5,6, time2
M1, 7,5, time3

Figure 4b

7 CONCLUSIONS

The process of storing and processing queries on the location of moving objects still remains a challenging problem in spatio-temporal databases. The majority of current researches on the subject proposed indexing methods that were based mostly on R-tree and its derivation. These indexing structures require rebuilding periodically to ensure optimal efficiencies especially because operations such as insertion and deletion can degenerate the index structure and degrade the query performance. The continuous need for periodic rebuilding to prevent degeneration and to maintain query performance is an expensive process. This paper proposed an alternative method that uses the grid structure to query the positions of moving objects. In contrast to indexing methods, grid structures do not require periodic rebuilding because the number of levels of a grid structure remains unchanged during the updates and retrievals, therefore query performance can be maintained at no extra cost.

REFERENCES

Chon H. D., Agrawal D., Abbadi A. E., 2001. Using Space-Time Grid for Efficient Management of Moving Objects. *In Proceedings of the 2nd ACM International Workshop on Data engineering for Wireless and Mobile Access, 59-65*.

Guttman A., 1984. R-trees: A Dynamic Index Structure for Spatial Searching. *In Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, 47-57*.

Kalashnikov D. V., Prabhakar S., Aref W. G., Hambrusch S., 2002. Efficient Evaluation of Continuous Range Queries on Moving Objects. *In Proceedings of the 13th International Conference on Database and Expert Systems Applications, DEXA*.

Meng X., Ding Z., DSTTMOD, 2003. A Future Trajectory Based Moving Objects Database. *DEXA, 444-453*.

Mokbel M. F., Ghanem T. M., Aref W. G., 2003. Spatio-temporal Access Methods. *In IEEE Data Engineering Bulletin, 26(2), 40-49*.

Nievergelt J., Hinterberger H., Sevcik K. C., 1984. The Grid File: An Adaptable, Symmetric Multikey File Structure. *ACM Transactions on Database Systems, 9(1), 38-71*.

Porkarw K., Lazaridis I., Mehrotra S., 2001. Querying Mobile Objects in Spatio-Temporal Databases.*http://infolab.usc.edu/csci599/Fall2001/paper/porkaew.pdf*

Prabhakar S., Xia Y., Kalashnikov D., Aref W. G., Hambrusch S., 2002. Query Indexing and Velocity Constrained Indexing: Scalable Techniques for Continuous Queries on Moving Objects. *IEEE Transactions on Computers, Volume 51, Issue 10, 1124-1140*.

Sistla A. P., Wolfson O., Chamberlain S., Dao S., 1997. Modeling and Querying Moving Objects. *In Proceedings of IEEE Data Engineering, 422-432*.

Theodoridis Y., Sellis T., Papadopoulos A. N., Manolopoulos Y., July 1998. Specifications for Efficient Indexing in Spatiotemporal Databases. *In Proceedings of SSDBM'98, Capri, Italy*.

Wolfson O., Location Management and Moving Objects Databases. *http://www.cs.uic.edu/~wolfson/presentation/Presentation.ppt*.

Kollios, G., 2003. Spatial-temporal Databases: Time Parameterized Queries. Lecture: CS 562: Advanced Database Applications.http://www.cs.bu.edu/faculty/gkollios/ada03/LectNotes/tpqueries.ppt.