Efficient Indexing of Moving Objects Using Time-Based Partitioning with R-Tree*

Youn Chul Jung, Hee Yong Youn, and Ungmo Kim

School of Information and Communications Engineering,
Sungkyunkwan University,
Suwon, Korea
{Jimmy4u, youn, umkim}@ece.skku.ac.kr

Abstract. A number of spatiotemporal access methods such as 3DR-Tree and MV3R-tree have been proposed for timestamp and interval query. These access methods consist of a single index structure covering the entire time span, and thus the performances quickly degrade since the indexed time region gets large as time progresses. To overcome the problem, we propose to employ time-based partitioning on the R-tree called Time-Based Partitioning R-tree (TPR-Tree). Since the structure of the TPR-Tree efficiently fits various queries on two dimensional data, it significantly outperforms other access methods for queries of various timestamps and time intervals. Extensive simulation validates the performance of the proposed scheme.

Keywords: Indexing moving object, location based service, R-Tree, spatiotemporal database, time stamp and interval query.

1 Introduction

Recently, due to the progress of wireless networking with GPS (Global Positioning System) and wide spread of personal communication devices such as mobile phone, concerns on contents services based on user location are increasing. The contents services are called LBS (Location Based Services). The LBS is defined as the service which tracks the location of mobile user and provides useful information associated with the location such as map service, traffic information service, traveling guide service, vehicle navigation service, public service, and so on. In order to quickly search for the requested locations for a variety of services, DBMS system needs an access method capable of efficiently searching a large amount of objects.

Most recently developed spatiotemporal access methods such as HR-Tree [1], MV3R-Tree [3], 3D R-tree [4], TB-Tree, and STR-Tree [4] are for indexing timestamp and interval queries of moving objects. The spatiotemporal access methods cover the entire time range of the network managed using a single index structure.

* This research was supported by the Ubiquitous Autonomic Computing and Network Project, 21st Century Frontier R&D Program in Korea and the Brain Korea 21 Project in 2004. Corresponding author: Hee Yong Youn.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 568 – 575, 2005.

Therefore, the performance may decrease as time passes since the size of indexed time gets very large. Also, they can neither efficiently support a variety of queries nor reflect the change of various queries along with time intervals.

To solve the problems, we focus on reducing the cost of various queries, i.e. timestamp and interval query. We propose a multiple-indexing method using time-based partitioning called time-based partitioning R-tree (TPR-Tree). It efficiently fits various queries (i.e. timestamp query and interval query) from the given spatial region of a fixed network. Compute simulation reveals that the proposed scheme outperforms other access methods for various queries, while the improvement gets substantial for timestamp query.

The rest of the paper is organized as follows. Section 2 reviews the existing access methods for timestamp and interval query. Section 3 describes the structure of the proposed TPR-Tree, and insert and search operation with the proposed structure are demonstrated. Section 4 presents the results of performance comparisons. Finally, we conclude the paper in Section 5.

2 Related Work

In this section we introduce three typical access methods – 3D R-Tree, MV3R-Tree, and HR-Tree.

2.1 3D R-Tree

The 3D R-tree [4] simply considers time as another dimension of the R-tree. Whenever an object moves to another position or changes its shape, a new minimum bounding region (MBR) is created to represent the change of the object. The MBR containing the spatial extent and lifespan is inserted in the 3D R-tree.

Figure 1 shows an example of the 3D R-tree. In this figure, R_1, R_2, and R_3 represent the MBRs describing the movements of the objects, and R_0 contains R_1, R_2, and R_3. As time passes, thus, the time region enlarges accordingly. This causes the performance of the 3D R-tree to degrade because a single R-tree keeps the whole time region in the 3D R-tree. Also, long-lived records result in a huge dead space not covering any record but being a part of the MBR. Therefore, they degrade the query performance of the 3D R-tree. However, the 3D R-tree takes smallest space since it has no duplicate data.

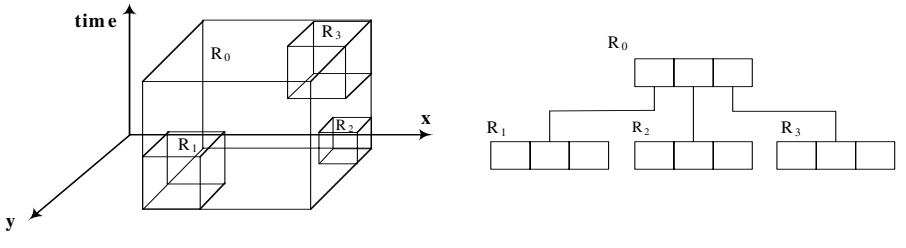


Fig. 1. The stucture of the 3D R-Tree

2.2 MV3R-Tree

The MV3R-tree [3] combines the Multi-Version R-tree (MVR-tree) and a small auxiliary 3D R-tree built at the leaf nodes of the MVR-tree. The former is used to answer timestamp and short interval queries and the latter is to answer long interval queries. Figure 2 illustrates the structure of the MV3R-tree.

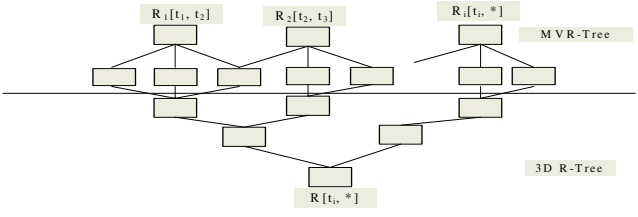


Fig. 2. The stucture of the MV3R-Tree

As shown in the figure, the MVR-tree consists of multiple R-trees that have their own jurisdiction intervals. For example, $R_1[t_1,t_2]$ deals with the records whose time intervals belong to $[t_1,t_2]$, while $R_2[t_2,t_3]$ manipulates the records whose time intervals belong to $[t_2,t_3]$. Although the size of the auxiliary 3D R-tree is very small since it shares the leaf nodes of the MVR-tree, it improves the performance of interval queries and provides flexibility to the algorithm in processing other spatial queries such as join and k-nearest neighbor.

2.3 HR-Tree

The Historical R-tree (HR-tree) [1,2] creates an R-tree whenever the objects in the previous R-tree change their positions or shapes, but common branches of consecutive R-trees are stored only once in order to save the space.

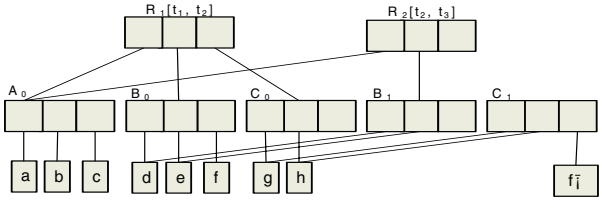


Fig. 3. The stucture of the HR-Tree

The timestamp query is directed to the corresponding R-tree and the search is performed only inside the tree. Thus, the timestamp query becomes an ordinary window query and is handled very efficiently. The interval query should search the corresponding R-trees of all the timestamps involved. Even when only one object has changed its position, the HR-tree may update the nodes contained in the path between the leaf node corresponding to the object and the root node. Therefore, the size of the HR-tree is several times larger than that of the 3D R-tree. Figure 3 shows an example

of the HR-tree. $R_1[t_1,t_2]$ deals with the records whose time intervals are contained in time interval $[t_1,t_2]$, and $R_2[t_2, t_3]$ manipulates the records whose time intervals are contained in $[t_2,t_3]$. Suppose that object f changes its position to f′ from B_0 to C_1 at $[t_2,t_3]$. The three nodes A_0, B_0, C_0 which are associated with f and f′ are copied and updated to nodes C_1.

3 The Proposed Scheme

Most spatiotemporal access methods consist of a single index structure covering the entire time span of the network, which degrades the performance as time progresses. To solve the problem, we propose the time-based partitioning R-Tree (TPR-Tree) consisting of 2-dimensional networks. We first introduce the structure of the TPR-Tree. We then describe the insertion and search algorithm based on the TPR-Tree.

3.1 The Structure of the TPR-Tree

As shown in Figure 4, the TPR-Tree consists of a single 2D R-Tree and multiple 1D R-Trees. The 2D R-Tree is used to index the spatial data of the spatial region. It is similar to the original R-Tree [9,10] and each leaf node of the 2D R-Tree contains pointers pointing the root of each 1D R-Tree on the given spatial data of the region. Consequently, each leaf node of the 2D R-tree corresponds to a 1D R-tree. Each of the 1D R-Tree is responsible for a fixed time interval derived from various queries. Also, each 1D R-Tree is generated according to the time intervals and used to index various queries on the given spatial region. Accordingly, the TPR-Tree can be considered as multiple 1D R-Trees on top of a single 2D R-Tree.

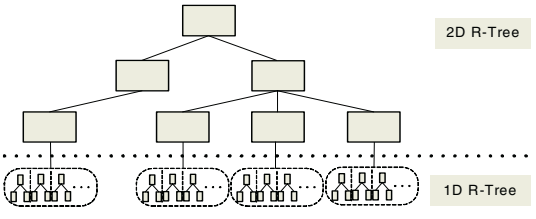


Fig. 4. The stucture of the TPR-Tree

Figure 5 illustrates the structure of the 1D R-tree. Each leaf node entry of the 1D R-Tree is of the form <ID, MBR, (t_{start}, t_{end})>. Here ID is the object identifier, MBR covers the boundaries of the children nodes, and (t_{start}, t_{end}) are the times when the record is inserted and deleted, respectively. If an entry has not yet been logically deleted, t_{end} is marked as "*". The time region boundary designates the lifespan of the record of each 1D R-Tree, and L_1, L_2, and L_i designate the interval of R_1, R_2, and R_i.

We apply time-based partitioning to the 1D R-Tree, which splits the 1D R-tree according to time interval. As a result, the TPR-Tree contains multiple 1D R-Trees in a given MBRs on a 2D R-Tree. The 1D R-Tree can dynamically support various queries and efficiently reflect the change of queries along with time interval. The

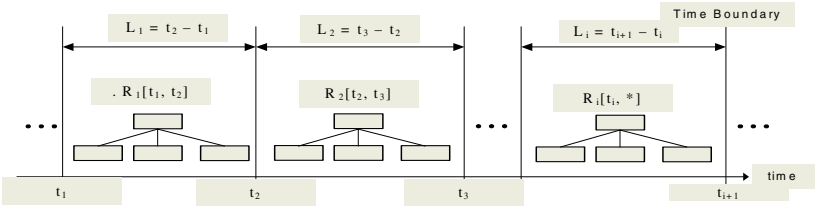


Fig. 5. The stucture of the 1D R-Tree

structure of the 1D R-tree is similar to the MV3R-tree, but the TPR-Tree does not have a shared node and employ the version spilt policy.

The processing of timestamp and interval query in the 1D R-Tree is similar to that of the 3D R-Tree [4]. However, since the 3DR-Tree is built using a single tree structure, it should search a large space for timestamp query. Note that the structure of 1D R-Tree dynamically fits various queries of time interval and timestamp. This reduces the search space and also dead space with time interval partition.

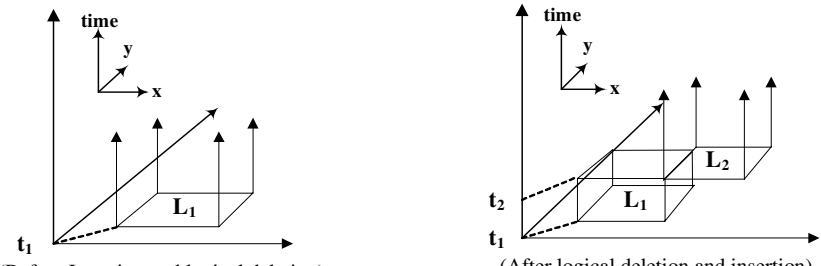


Fig. 6. The insertion and logical deletion

Figure 6 shows how insertion and deletion process work. Record R_1 is logically deleted by time boundary at timestamp t_2, where L_1 and L_2 indicate the spatial MBR of R_1 and R_2, respectively. $[t_1,t_2]$ and $[t_1,*]$ indicate the lifespan of R_1 and R_2, respectively.

3.2 Insertion and Search with the 1D R-Tree

When an insertion occurs at a specific spatial region with the TPR-Tree, executed the following steps are executed. In the 1st step, the R-Tree insertion algorithm is executed in the 2D R-Tree in order to insert the MBR covering the specific region of the 2D R-Tree leaf node. It then follows the pointer to the 1D R-Tree and finds the 1D R-trees with time interval $[T_{start},T_{end}]$. In the 2D R-Tree, we use the original R-Tree algorithm [9, 10] to index a spatial region. The next step is to execute the 1D R-tree algorithm.

Figure 7 shows the insertion algorithm with the 1D R-Tree. The algorithm of the 1D R-Tree employs the concept of time boundary. Therefore, if the lifespan of a record does not include the boundary, the record is inserted in the 1D R-Tree covering the suitable interval that contains the lifespan of the record. However, if the records inter-

sect between time intervals, each of the records split by data fragmentation is inserted in the 1D R-tree of suitable time interval containing the lifespan of the record.

```
Procedure insertion
Begin
/*Find Ri which contains the record and insert it
  into Ri*/
for Ri(ti, ti+1) /* i=1 … k */
    if (ti, ti+1) contains current time interval
        insert record to Ri;
        break;  /* exit for loop */
    endif
/* the logical deletion */
if (current record=(ID, MBR,[ti+1, *])
    previous record=[ti, *] is changed into
    previous record=[ti, ti+1];
endif
/*record intersect between time intervals*/
  for Ri(ti, ti+1) /* i=1 … K */
    if(ti, ti+1) intersect the lifespan of the record
        insert record to suitable time interval's Ri;
    endif
End
```

Fig. 7. Insertion algorithm with the 1D R-Tree

4 Performance Evaluation

This section describes the simulation environment and provides the results of simulation.

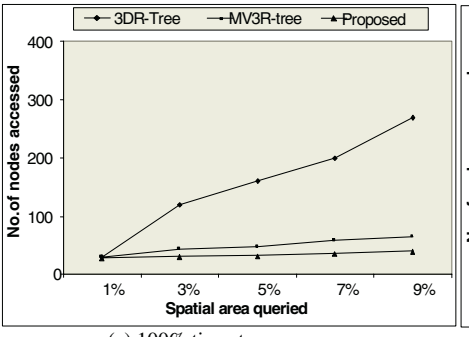
4.1 Simulation Environment

In all our experiments we use a 2.4GHz Pentium IV machine with 1GB of main memory. Due to unavailability of movement data of actual object, we use a synthetic dataset generated by GSTD methods [8] which has been widely employed as a benchmarking environment for studying the access methods handling moving points and region. It was designed to generate realistic data for the evaluation of database algorithms with indexing and storing dynamic location data.

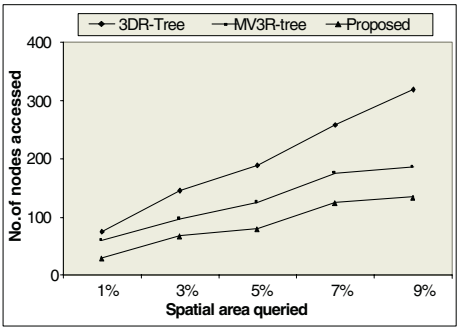
The number of moving objects is 1,000. Spatial areas queried are 1%, 3%, 5%, 7%, and 9% of the whole space. The simulation evaluates the cost of queries for combined timestamp and interval time.

4.2 Simulation Results

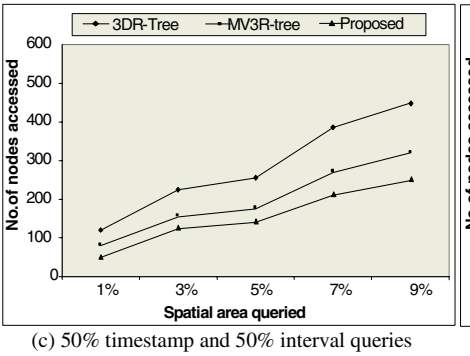
Figure 8 shows the performance of each access method for various queries of time interval and timestamp in terms of the number of nodes accessed. We compare MV3R-



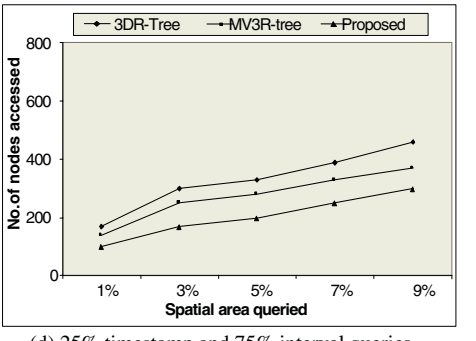
(a) 100% timestamp query



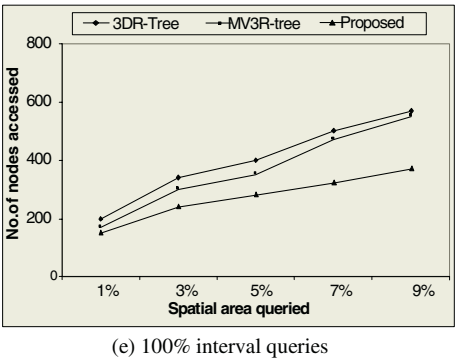
(b) 75% timestamp and 25% interval queries



(c) 50% timestamp and 50% interval queries



(d) 25% timestamp and 75% interval queries



(e) 100% interval queries

Fig. 8. The comparisons of the cost of query of the three methods

tree, 3DR-Tree, and the proposed TPR-Tree. As shown in Figure 8, the TPR-Tree outperforms the other methods for various queries of timestamp and interval time.

Notice that performance of the 3D R-tree declines with timestamp query. This is because a single R-Tree keeps the entire time region in the 3DR-Tree while the time region gets large as time passes. The proposed TPR-Tree consists of multiple indexing trees and 1D R-Tree with the time-based partitioning method. It can thus dynamically fit various queries on the given interval time.

5 Conclusion

Most existing spatiotemporal access methods consist of a single index structure covering the entire time span of the network. The performance would decrease since the indexed time of a specified region enlarges as time progresses. Also, they can neither support a variety of queries efficiently nor reflect the change of the queries with time-intervals. We have proposed a method consisting of multiple trees of two dimensions with time-based partitioning to the indexing. It can efficiently handle various queries (i.e. timestamp query, interval query) from the given spatial region. The interval length of the TPR-Tree well fits time interval of the query. Computer simulation reveals that the proposed TPR-Tree outperforms other methods for various queries of timestamp and interval time. We will further investigate the performance of the proposed approach for other operational environment.

References

[1] Nascimento, M.A., Silva, J.R.O.: Towards historical R-Tree. In Proceesings of the 13th ACM Symposium on Applied Computing (ACM-SAC'98), 235-240, 1998.
[2] Nascimento, M.A., Silva, J.R.O, and Theodoridis, Y.: Evaluation for access Structures for discretely moving points. In Proceesings of the International Workshop on Spatio-Temporal Database Management (STDBM'99), Edinburgh, Scotland, 171-188, 1999.
[3] Tao, Y., and Papadias, D.: Mv3R-tree: a spatiotemporal access method for timestamp and intervalqueries. In Proceedings of the 27th International Conference on Very Large Databases, 431-440, 2001.
[4] Theodoridis, Y., Vazirgiannis, M., and Sellis, T.: Spatio-temporal Indexing for Large Multimedia Applications. In Proceedings of the 3rd IEEE Conference on Multimedia Computing and Systems, Hiroshima, Japan, 441-448, 1996.
[5] Pfoser D., Jensen C.S., and Theodoridis, Y.: Novel Approaches to the Indexing of Moving Object Trajectories. In Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt, 395-406, 2000.
[6] Elias frentzos.: Indexing object moving on fixed networks. Advances in Spatial and Temporal Databases: 8th International Symposium, SSTD 2003 Santorini Island, Greece, 289-305, July 24-27, 2003 Proceedings.
[7] Brinkhoff, T.: Generation Network-Based Moving Objects. In Proceedings of the 12th Int'l Conference on Scientific and Statistical Database Management, SSDBM'00, Berlin, Germany, 253-255, 2000.
[8] Y.Theodoridis, J. R. O. silva, and M.A Nascimento: On the Generation of Spatiotemporal Datasets. SSD, 147-164, 1999.
[9] A.Guttman: R-Tree: A Dynamic Index Structure for Spatial Searching. ACM SIGMOD, 47-57, 1984.
[10] N.Beckmann, H. Kriegel, R.Schmeider, and B. Seeger: The R*-Tree: An Efficient and Robust Access Method for Point and Rectangles. ACM SIGMOD, 322-331, 1990.