aCN-RB-tree: Constrained Network-Based Index for Spatio-Temporal Aggregation of Moving Object Trajectory

Dong Wook Lee, Sung Ha Baek and Hae Young Bae
Department of Computer Science & Information Engineering, Inha University
253 Younghyun-dong, Nam-gu, Incheon – Korea Republic
[e-mail: {dwlee, shbaek}@dblab.inha.ac.kr, hybae@inha.ac.kr]
*Corresponding author: Hae Young Bae

Received July 19, 2009; revised August 24, 2009; accepted September 5, 2009; published October 30, 2009

Abstract

Moving object management is widely used in traffic, logistic and data mining applications in ubiquitous environments. It is required to analyze spatio-temporal data and trajectories for moving object management. In this paper, we proposed a novel index structure for spatio-temporal aggregation of trajectory in a constrained network, named aCN-RB-tree. It manages aggregation values of trajectories using a constraint network-based index and it also supports direction of trajectory. An aCN-RB-tree consists of an aR-tree in its center and an extended B-tree. In this structure, an aR-tree is similar to a Min/Max R-tree, which stores the child nodes' max aggregation value in the parent node. Also, the proposed index structure is based on a constrained network structure such as a FNR-tree, so that it can decrease the dead space of index nodes. Each leaf node of an aR-tree has an extended B-tree which can store timestamp-based aggregation values. As it considers the direction of trajectory, the extended B-tree has a structure with direction. So this kind of aCN-RB-tree index can support efficient search for trajectory and traffic zone. The aCN-RB-tree can find a moving object trajectory in a given time interval efficiently. It can support traffic management systems and mining systems in ubiquitous environments.

Keywords: Spatio-temporal index, moving object, trajectory aggregation, constraint network

This research was supported by Grant 07KLSGC05 from the Cutting-edge Urban Development - Korean Land Spatialization Research Project funded by the Ministry of Construction & Transportation of the Korean government. A preliminary version of this paper was presented in the 7th International Conference on Computational Science and Applications (ICCSA), Suwon, S. Korea, 2009.

DOI: 10.3837/tiis.2009.05.007

1. Introduction

Various moving objects are regarded as mobility sensors in ubiquitous environments. So it manages acquired data, such as trajectory, region and time from moving objects. And a spatial data warehouse is an integrated spatial data storage system that can support spatial data mining. The source data of a spatial data warehouse is from an SDBMS or DSMS service, such as ubiquitous GIS, and the data elements are managed after they have been filtered, modified and cleaned. A spatial data warehouse provides the base data for efficient data mining and for improving the performance of services using spatial data in ubiquitous environments [1][2].

We have researched an indexing method for efficient management of moving objects' trajectories, which raised the efficiency of storing and updating the aggregation value for trajectory analysis [1][3].

The trajectory of moving objects means their path of movement based on time [4][5]. It can be analyzed from the source data of an LBS system that stores the position data or filtered from a system which stores the trajectory. The present and past moving objects' trajectories are stored and managed in many systems for predicting moving objects' trajectories or analysis of historical paths of moving objects [6]. The trajectory aggregation value in this paper means the number of trajectory edges in a limited network-based index. In other words, it means the count of moving objects passing the same edge regardless of route [7]. Using the trajectory aggregation value, the shapes of moving objects' trajectories can be analyzed. It provides a trajectory aggregation value to analyse the diverse patterns of moving objects and also can be used to design new road paths or to allocate the time of a variable traffic lane for a traffic management system for spatial mining applications.

Many methods have been proposed in previous spatio-temporal research. Some indexes simply follow the spatial network model, but temporal relationships and aggregation are ignored [8][9][10]. Some indexes combine a spatial and temporal index and add an aggregation value, but they can't express the moving object direction [11][12]. An FNR-tree is proposed for moving objects' trajectories in a constrained network [13][14]. It can record the spatial, temporal and direction information, but there is no aggregation value. In summary, no previous index is able to cover all aspects.

In order to deal with the above problems, an index structure is proposed. Firstly, the network is divided into segments at the point of intersection. After each partition, it is connected with a cross point. The network index is constructed with a defined arrangement. The aggregation value is stored with the spatial object entry. There are two kinds of node in spatial parts (i.e. leaf node and non-leaf node). The leaf node contains the aggregation value of the moving object's trajectory and the pointer to the extended B-tree. An extended B-tree node contains the information of the timestamp and the aggregation values of the trajectories with two directions. Trajectory counter aggregation, timestamp and route sample are proposed for each segment in order to describe the trace of the moving object during a given time period. Query search, updating, insertion and deletion tasks can be operated using this index. Better granularity is available according to the timestamp and spatial index. And better performance can be achieved for storage management. A suitable location is chosen and an extended B-tree that fits the right timestamp is updated.

In this paper, we proposed aCN-RB-tree based on a constrained network to manage aggregation of moving object trajectory with direction. The rest of this paper is organized as

follows: Section 2 considers the background. Section 3 describes the aCN-RB-tree structure and algorithms. Then, Section 4 analyzes the results of experiments. Finally, conclusions and future works are reported in Section 5.

2. Related Work

There are many general spatio-temporal indexes such as MON-tree and IMORS [9][13]. They are mainly used in establishing network models without taking into account the time of movement. Since there is no limited interval for a stored time, it is quite bothersome in case that a long searching time cost is required. It is difficult to deal with a time query. So aggregation in spatio-temporal indexing is required. The aRB-tree and a3DR-tree have been studied [11][12]. Particularity in a typical object movement is a direction. In traffic, the number of moving objects is different because of the different directions, so a FNR-tree has been proposed [13]. But the problem is that an FNR-tree does not have an aggregation value. In this section we explain the problems of related works to support management of trajectory aggregation.

The aRB-tree (the full name is the aggregate R-B-tree) is used for moving objects inside a region [11]. The timestamp and aggregation value are included. This work is motivated by querying the summarized spatio-temporal data rather than the exact ID of every data element. For example, we only need to query the number of visitors and don't need to know the other characters, such as name or age. The regions that are only stored once comprise the spatial hierarchy and they are indexed by an R-tree. There is a pointer to connect a B-tree, which stores the temporal aggregation data about each R-tree's node [15][16].

An aRB-tree can deal with a region's aggregations from a cube. Those taken, for example, from region R1 and R2 form the boundary of node R3 in the aforementioned aRB-tree. **Fig. 1** shows the set of regions of 2-dimensions: regions and timestamps. Region R1 includes 20 objects during the first two timestamps. The sum of the aggregation values in a given region are stored in the total sum.

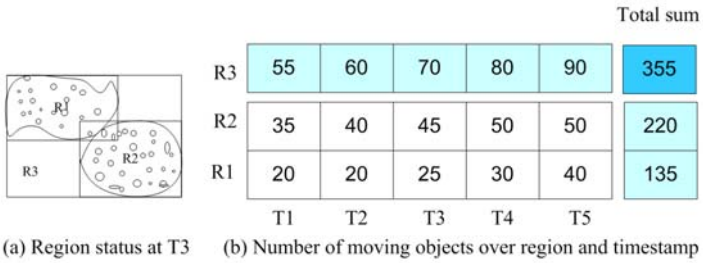


Fig. 1. Example of aggregation values in aRB-tree

As **Fig. 2** shows, the data of the cube in **Fig. 1**. is used. An aRB-tree combines the R-tree and B-tree and the timestamp and aggregation values are B tree's nodes [15][16]. For instance, the number 220 stored with R-tree entry R2 expresses that the total number of objects in R3 is 220 during the period from T1 to T5.

It can find the number of moving objects in a unique time or an interval time in some exact ranges and it can replace the data cube. If the aggregation value is not very dynamic, the storage space is smaller than the data cube. It is a mission impossible if data needs to be obtained with more accurate granularity in a data warehouse, for example, recording the road's position and the directions of moving objects. This tree is for the range query but if the query

object moves in the network, it can't achieve high efficiency because of the drawback of the R-tree. Using the R-tree to record the region, if there is an overlap of regions, the aggregation value will be incorrect. And the other drawback is that it simply describes the sum count aggregation but it can't express the direction, which is an important attribute of moving objects.

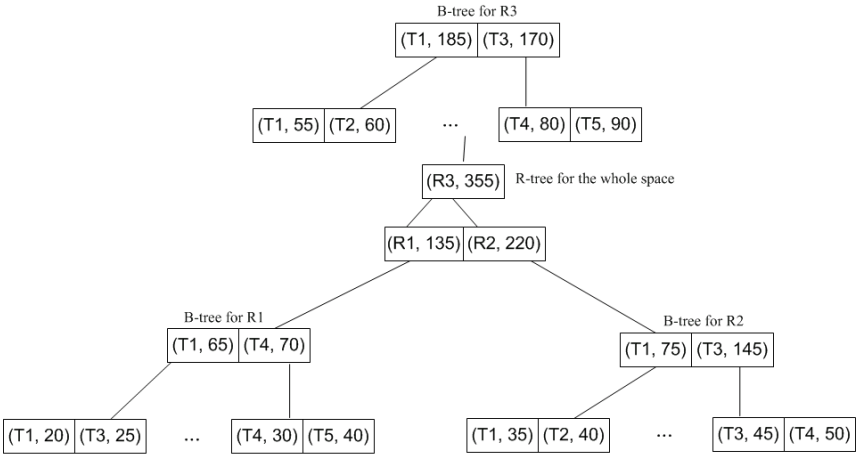


Fig. 2. Example of aRB-tree structure

A basic spatio-temporal index that covers both the spatial and temporal characteristics constitutes two indexes: a spatial index for objects with spatial characteristics and x, y coordinates and a temporal index for objects with temporal characteristics and a time interval, the start/stop time. For the spatial index, the R-tree is the classic example. For the temporal index, it adds at least one-dimension to the above data structure. In other words, a multi-dimension index can support spatio-temporal data indexing. The 2D R-tree is for the spatial part and the 1D R-tree is used in the temporal part. The indexing scheme is illustrated in **Fig. 3**. That is, the 3DR-tree simply adds the time as another dimension to the 2D-Rtree and transforms it to the 3DR-tree [6][14][16]. The structure is for a spatial and temporal query, which consists of spatial and temporal layout retrieval. There are three axes X, Y, T in a chart, represented by which the features of moving objects in the spatio-temporal aspect can be identified by six coordinates, the projections on the X, Y, T axes, which are x(points x1,x2), y(points y1,y2) and z(points z1,z2) in **Fig. 3**. As a height-balanced tree, the R-tree contains intermediate and leaf nodes. It is the one of the most efficient hierarchical multi-dimensional data structures. The MBR of the spatial area is stored in the leaf node of the R-tree. The intermediate nodes are used in grouping rectangles. We denote a branching factor of five, i.e.., each intermediate node contains at most five entries. The typical queries include spatial and temporal layout retrieval. This means that the spatial layout is based on queries such as "Find the objects and their positions at the timestamp of T1" The temporal layout is based on queries such as "Find the objects appear during the T2 to T3 time interval". The interval can be queried efficiently with the 3DR-tree. But if the query requires a long life span object, it can generate a lot of dead space and make the timestamp query inefficient.

The aggregate three dimensional R-tree is named an a3DR-tree. It adds the aggregate value to the 3DR-tree and a new entry is created. It can integrate spatial and temporal dimensions in the same structure. But it wastes space by storing the MBR each time an aggregate value changes.

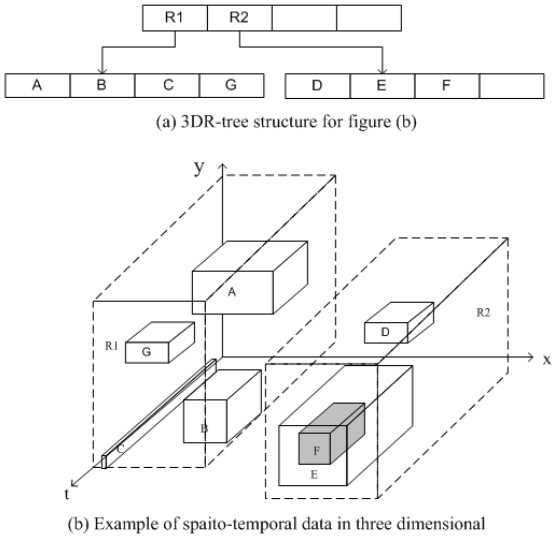


Fig. 3. Example of 3DR-tree structures

There are many kinds of network in the real world, consisting of roads, veins, lines etc. In this paper, the network only implies the road network, where there are many roads crossing each other or meeting at many points. The networks are classified as unconstrained, constrained, and transportation networks. The unconstrained network supports unconstrained movement, that is, the moving object can move arbitrarily and there is no fixed trajectory, e.g., sailors at sea. The constrained network is that the moving object can move on some roads arbitrarily and the trajectory is fixed, such as the movements of cars on roads. The transportation networks have the characteristic that they only allow a moving object to follow a fixed road without arbitrarily movement. Its start point and destination are fixed, e.g., trains [14].

In the constrained network, movement is regularized, for example, there are many cars on the roads during rush hour but few at night. In this case, we need to know the network's persistent condition, so aggregation value retrieval is proposed. The aggregation of trajectories of similar moving objects describes movement regularization. The trajectory involves recorded trace positioning and following an object moving through space. The proposed method involves dividing the map into spatial units [2][7][17]. These units record how many times the trajectories have passed the certain unit during a certain time period, by counting which the original trajectory information that cannot be stored entirely can be archived separately. However, the spatial relationship integrality will be destroyed.

FNR-tree is an index model for the moving object's positional management in the fixed network. The fixed network consists of conjoint roads. The 2DR-tree is used as the index of these roads, while the 1DR-trees are used to index the time interval of each object's movement inside a given network link. The 2DR-tree's leaf node contains <mbb, orientation>, where mbb is the segment's MBB (Minimal Bounding Box), and orientation is a flag that describes the exact geometry of the segment, taking values from the set{0,1}. Each leaf node of the 2DR-tree's form includes Line's ID, MBB and Orientation. Besides, each non-leaf node of the 2DR-tree contains pointer to child node using MBB. And each leaf node of the 1DR-tree consists of the Moving Object ID, Line segment ID, the time of entrance and exit and moving object direction. Each non-leaf node of the 1DR-tree contains pointer to child node and the time of entrance and exit. Each node of the 2DR-tree contains the information of each road

segment and connects the corresponding 1DR-tree. **Fig. 4** represents the 1DR-tree managing all records of moving objects passing the road segment of the corresponding 2DR-tree.

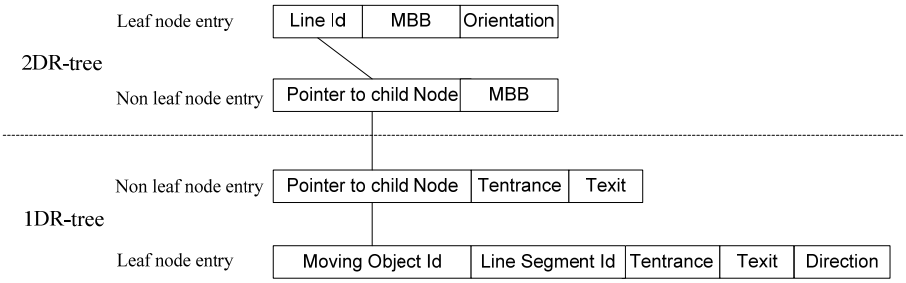


Fig. 4. Node connection between 2DR- and 3DR-tree in FNR-tree

The FNR-tree optimizes a query on moving objects in historical windows. But there are three drawbacks: (1) there are many 2DR-tree's leaf nodes, because each leaf node of the 2DR-tree includes only one road and this makes the system generate many leaf node entries. (2) The system can't deal with the halting of moving objects in networks, because the 1DR-tree only records the time interval of moving objects passing some segments but it can't describe the details of stoppages or changes in direction. (3) Because it is obligatory to record information when moving objects cross the segment's port, there are many insertions.

3. aCN-RB-tree: Constrained Network-based Indexing for Spatio-Temporal Trajectory Aggregation

Described in Section 2, we know that no indexing contains both aggregation and direction in previous studies. The index structure proposed in this paper issues trajectories' aggregations of objects moving along constrained networks during a time interval. So it needs to build the appropriate model, which not only describes the spatial relationship but also the time interval and aggregation.

The index structure consists of two blocks: spatial with aggregation and timestamp with aggregation. In the spatial part, it uses the constrained network R tree with aggregations, which we call the aCN-R-tree. In the temporal part, it uses the extended B-tree for timestamp with aggregation.

For the spatial part, we must find an index through which we can search for the router aggregation recording on spatial data. The spatial aggregation of the moving object is a trajectory during a certain time. The previous works divide the map into spatial units and then summarize the count of trajectories passing every spatial unit. This structure can't adapt the characteristics of the trajectory, and destroys the polyline's relationship, so some improvements are made by scanning some roads and dividing each road into sub-segments when they meet at the point of intersection. Moreover, the statistics about some spatial units' count are generated concurrently.

For roads, **Fig. 5** shows that the direction is represented by the object's movement and the aggregation of trajectory. The aggregation is not unique. There are two kinds of aggregations, because there are usually two directions in a closed road according to the traffic rule.

In this section, a new index structure to retrieve and store the aggregations efficiently is presented. The full name is aggregation of constrained network R and extended B tree and the short name is aCN-RB-tree.

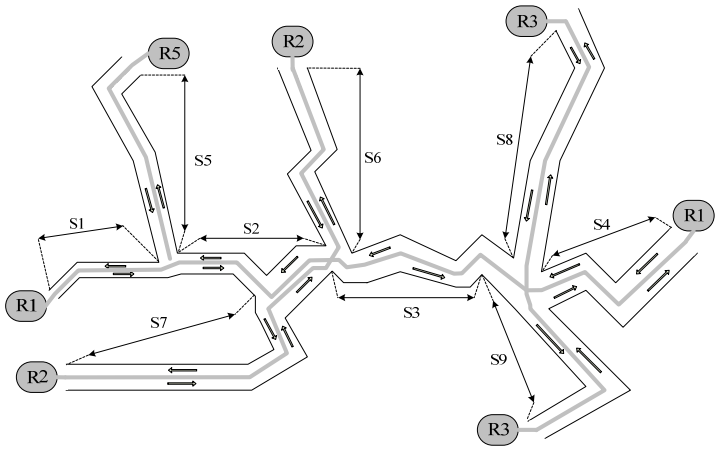


Fig. 5. Concept of Edge Segment with Direction on the Road Network

3.1 aCN-RB-tree Structure

The aCN-RB-tree proposed includes the main and sub-index. The main part is the index for the network, which uses the fixed network R-tree. And the sub-index is the extended B-tree, which includes timestamp and aggregations with two directions for each aCN-R-tree entry, as **Fig. 6** (c) shows.

The network is divided into several segments (or edges) according to the roads' intersecting points. **Fig. 6** shows (a) an example of a network which includes four roads <R1, R2, R3, R4> and (b) a situation where the roads are divided into many segments.

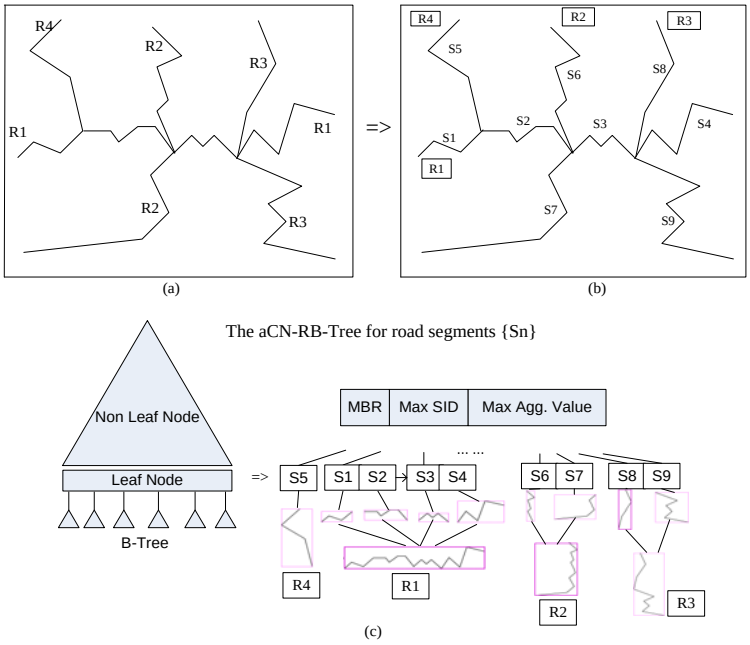
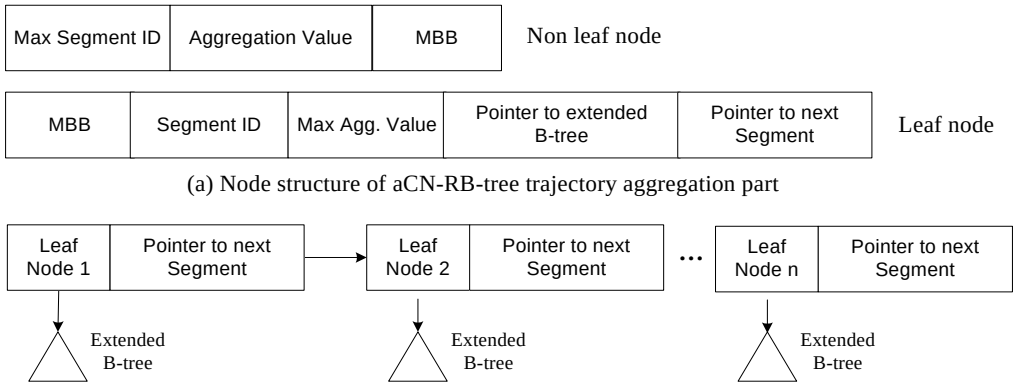


Fig. 6. Structure of aCN-RB-tree based on Constrained Network

Fig. 6 is separate from the integer network map. And the aCN-RB-tree is used to construct the index about these roads. At first the network is separated into unit segments. For example,

road R1 is divided into S1, S2, S3, and S4. And then the pointers are assumed to connect with the next segment until it is combined with integrated roads.

The aCN-RB-tree also consists of two parts: leaf node and non-leaf node. Each leaf node of the aCN-RB-tree for road segments includes the max aggregation value and the identification information of that segment among many different segments. Each non-leaf node expresses the basic information about the segment, such as the MBB and the pointer to the extended B-tree. The leaf node is used to store the spatial and temporal aggregation values and the non-leaf node is for leaf node connections between spatial and spatial, spatial and temporal relationships.



(a) Node structure of aCN-RB-tree trajectory aggregation part

(b) Connection between leaf node and extended B-tree in aCN-RB-tree

Fig. 7. Node Structure of aCN-RB-tree

Fig. 7 shows the segment node structure. There are leaf nodes and non-leaf nodes. The leaf node has the schema <Segment ID, Aggregation Value, MBB>. The aggregation value is the number of trajectories at the latest time, which is the sum aggregation of two directions. The leaf node is used to query the aggregation value in a certain map area represented by the nodes in the aCN-RB-tree. The non-leaf node is shown by the schema <MBB, Max SID, Max Agg.Value, Pointer to extended B-tree, Pointer to next Segment>. The MBB is used to ensure the integrity of the data. Especially, at the beginning of index building, the integrity of all the information can be fixed if the trajectories appear continuously among different roads. The Max SID identifies the segment by which most trajectories have passed**,** and the MaxAgg.Value is the max value of that segment. The segment units from S_n to S_m are connected to each other via pointers of non-leaf nodes.

Because a road segment has a unique time, the extended B-tree index is constructed for the timestamp. **Fig. 8** is an example of the extended B-tree. Every node contains three values: timestamp and the two aggregation values with two directions.

In the middle level, the node can store two timestamps and three aggregation groups at most; the left aggregation group is the average value of a given timestamp. For example, the aggregation value before T3 is the average value of T1 and T2. The middle value means the average of the timestamps. For instance, the value after T3 means the average of the timestamps between T2 and T5, namely T3 and T4. And the value on the right is the average of the later timestamps, such as T5 and T6.

The form of the extended B-tree's entries is expressed as <TID, L.agg., R.agg.> in **Fig. 9**. L.agg. and R.agg. mean the aggregation values in two directions. The leaf node of the segment contains the pointer link to the extended B-tree. It can query some aggregations in the time interval using the extended B-tree.

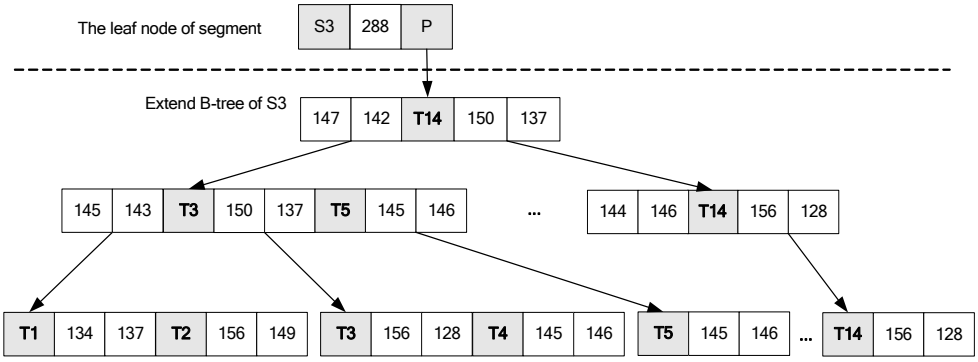


Fig. 8. Example of Extended B-tree in aCN-RB-tree

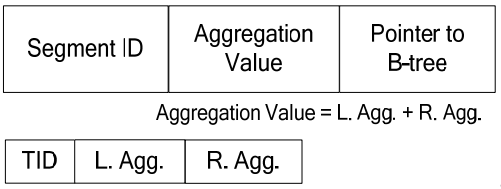


Fig. 9. Concept of Aggregation Value and Extended B-tree Node Structure

3.2 The Creation of aCN-RB-tree

The multi-dimension model is one of the most popular data models in data warehousing. In a spatial-temporal data warehouse, measurement is decided in two dimensions, which are the region and time. **Fig. 10** shows the number of tracks in region <S1,S2,… , S9> versus time <T1,T2,… , T5>. This is the L- and R-aggregation value of the segments from T1 to T5. And these groupings assume that the aggregation function is average.

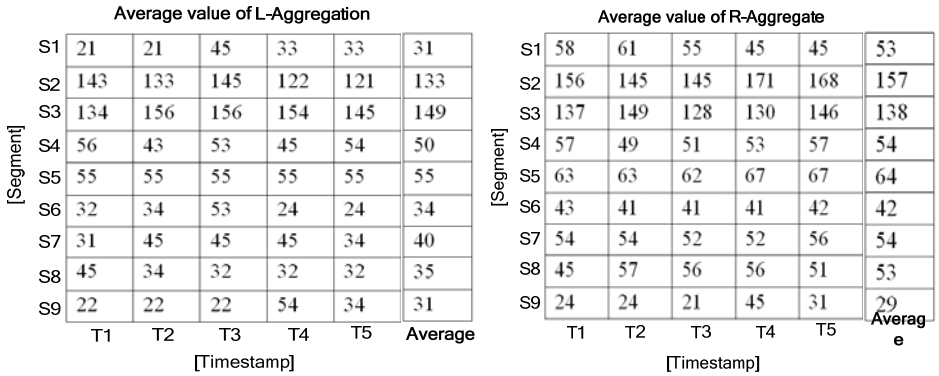


Fig. 10. Average Aggregation Value by Direction

The aCN-RB-tree is elicited to establish the index structure. The aggregate aCN-RB-tree is based on the following concept: The polylines of the network have a hierarchical spatial relationship. So we use the MBB to connect every polyline in order to ensure easy range query. And we depict the extended B-tree to index the temporal aggregation. In particular, each network entry has the form <Segment ID, MBB, L.agg, R.agg, Timestamp>. We use the segment MBB to fix the position and find the segment ID which is in accord with the MBB.

And we also insert the timestamp and aggregation into the extended-B-tree. The average aggregation values are also updated. We sum L.agg. and R.agg. and update the segment block. If the aggregation value is not changed, then the node is not updated.

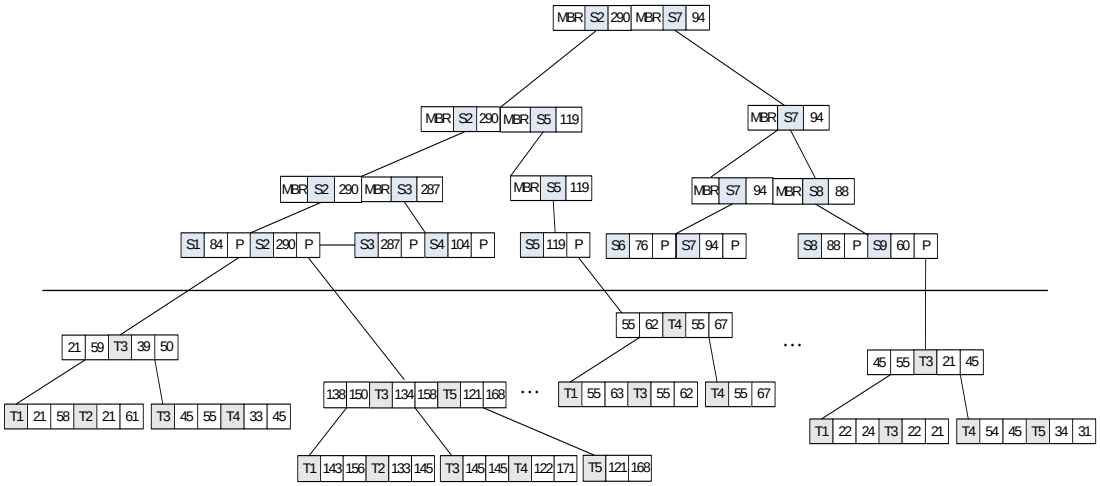


Fig. 11. Average Aggregation Value by Direction

Fig. 11 shows the aCN-RB-tree index about the network of **Fig. 10**. We can see that there are extended B-trees for the timestamp in the bottom level and the fixed network R-tree in the top level. We can query the spatio-temporal information efficiently by using it. There are extended-B trees in the bottom level and the middle level contains the segment ID and average aggregation value from the extended B-trees, which is the sum of L and R aggregation. P is the pointer to connect the extended B-trees and every node can store information of two segments at most. The value of the top level node is the merge of its left and right aggregations in the middle level. It contains the MBB of those segments and the segment ID and aggregation, using the aggregation value of the bigger MBB. For example, S2's value of 290 is bigger than S1's value of 84. So we choose S2 and its aggregation. Using this method we can query which segment is the busiest in an appointed region.

3.3 Insertion and Deletion Algorithm of aCN-RB-tree

The data insertion operation is the most important and basic operation in data warehousing and it is the key to constructing a data warehouse. We will introduce how to inset a new value.

The data insertion operation means that every road segment aggregation changes as time passes. So it is related mainly to the timestamp and aggregation value. The data series are inserted into the data warehouse. The data series contain many data blocks including segment ID, aggregation value and the timestamp but we only study the insertion of one data block, because the others are the same. At first we find whether the segment ID is in the aCN-RB-tree. If the aCN-RB-tree includes this segment ID, we find the timestamp in the extended B-tree depending on the pointer to the extended B-tree and then we insert some new value in the appropriate position. We calculate the average between the new value and the latest update up to the extended B-tree's root. Another problem which we have to consider is how to get the aggregated directions if the query window includes different directions. For example, the general collision exists when the road direction is either horizontal or vertical. Meeting this collisions, we can not define the direction. As a result, we make some average aggregations to handle it by obtaining the sum of L and R aggregations, and putting it into the segment node.

The Max aggregation needs to be updated when some new records are generated. Changes of storage space occur in the extended B-tree. Meanwhile, the value updates of selective nodes do not affect the storage space. The insertion of an aCN-RBtree is described in algorithm 1.

[Algorithm 1] Insertion algorithm of aCN-RB-tree

```
Input
Data: N data series inserted into data warehouse;
aCN-RB-tree: index tree from network;

Variable
Data.segmentID: identification of data segment;
TimeStamp: last refresh time of data;
R-tree.LeafNodeFlag: flag of updating;

Begin
01: Check the segment ID;
02: if(data.segmentID==aCN-RB-tree.segmentID) // if aCN-RB-tree includes input data segment
03:    Insert(data, extended_B_Tree, pointer_to_B_tree);
        // insert data into extended-B-tree depending on pointer to B-tree.
04: end if
05: if(insert(data, extended_B_Tree, pointer_to_B_tree)==true)
        //if there is data inserted into extended B-tree
06:    R-tree.LeafNodeFlag=true;
07:    Update R-tree.leafNodeAggregtionValue; // R-tree's leaf node flag= true, else false;
08: else  R-tree.LeafNodeFlag=false
09: end if
10: repeat  01-09 until N data input;
11: if(R-tree.LeafNodeFlag=true)
12:    Update R-tree.leafNodeAggregtionValue;
13: end if
14: if(R-tree.leaf_Node_Aggregation_Value> Parent_aggregation.Max_Aggregation_Value)
15:    Parent_Max_SID=R-tree.leaf_Node_Segment_ID;
16:    Parent_aggregation.Max_Aggregation_Value= R-tree.leaf_Node_Aggregation_Value;
17: end if
18: while(R-tree.leaf_Node_Aggregation_Value> Parent_aggregation.Max_Aggregation_Value)
19: repeat 11-18;
20: end while
End
```

If we want to delete an aggregation value with a given timestamp in a certain sector, we have to find that timestamp and its pointer pair in a leaf of the aCN-RB-tree.

As with data insertion, deletion is mainly related to the timestamp and aggregation value. The segment ID is found first. If the segment ID is in the aCN-RB tree, we find the timestamp in the extended B-tree. If the timestamp which we want to delete is the latest node, we simply delete the saved value of the last update up to the extended B-tree's root. However we follow the B-tree's delete operations and delete the aggregation value. If the segment ID which we want to delete is in MaxID, the last update value is compared with the MaxID value and updated.

[Algorithm 2] Deletion algorithm of aCN-RB-tree

Input
Deletion information: deletion information;
aCN-RB-tree: index tree from network;

Variable
Data.segmentID: identification of data segment;
TimeStamp: last refresh time of data;
R-tree.LeafNodeFlag: flag of updating;

Begin
01: Check the segment ID;
02: **if** (data.segmentID==aCN-RB-tree.segmentID) // if aCN-RB-tree includes input data segment
03: delete(data, extended_B_Tree, pointer_to_B_tree);
// delete data from extended-B-tree depending on pointer to B-tree.
04: **end if**
05: **if** (delete(data, extended_B_Tree, pointer_to_B_tree)==true)
//if data is deleted from extended B-tree
06: R-tree.LeafNodeFlag=true;
07: Update R-tree.leafNodeAggregtionValue;
08: **else** R-tree.LeafNodeFlag=false
09: **end if**
10: **if(**R-tree.LeafNodeFlag=true)
11: Update R-tree.leafNodeAggregtionValue;
12: **end if**
13: **if** (R-tree.leaf_Node_Aggregation_Value > Parent_aggregation.Max_Aggregation_Value)
14: Parent_Max_SID=R-tree.leaf_Node_Segment_ID;
15: Parent_aggregation.Max_Aggregation_Value= R-tree.leaf_Node_Aggregation_Value;
16: **end if**
17: **while**(R-tree.leaf_Node_Aggregation_Value> Parent_aggregation.Max_Aggregation_Value)
18: repeat 13-17;
20: **end while**
End

3.4 Search Algorithm of aCN-RB-tree

The search operation of the aCN-RB-tree can be divided into the case where it searches for an aggregation value in a fixed range and the case of a search at one or several points. In case of a range search, it finds a corresponding road segment in the centered R-tree for the request query and processes the search operation. In a point search, it also finds the corresponding road segments for a range search and returns the point value for the same point as the one requested in the road segment.

For example, suppose that a user wants to find all objects with some direction in some network that overlap the query window of **Fig. 12** during time interval [T2,T4]. S5 and S1 are fully inside the shaded window and we visit the corresponding extended B-tree to retrieve the temporal information. Some parts of S2 are also in the query window, because the aggregate function is average and S2 is also available. We can query the information depending on **Fig. 13**. There are two kinds of query in the aCN-RB-tree, region query and point query. For the region query, we find the window query's MBR in the R-tree and we can obtain the corresponding segment ID and Max aggregation. If the query doesn't need direction

information, we simply find the aggregation in the segment node without querying the extended B-tree. Otherwise, we can only find the aggregation with directions from the extended B-tree depending on the TID. For the point query, we need to find the point in a certain segment and the information about this segment. We use the same method as the one for the region query. **Fig. 13** shows a flow chart about the search algorithm of the aCN-RB-tree. Algorithm 3 and 4 expresses the search algorithm.

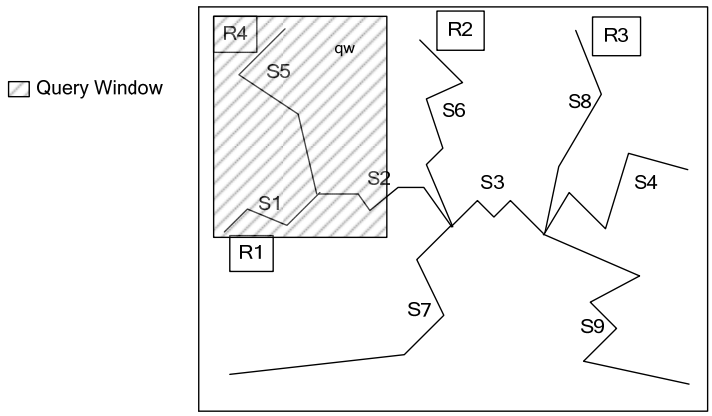


Fig. 12. Example of window query in aCN-RB-tree

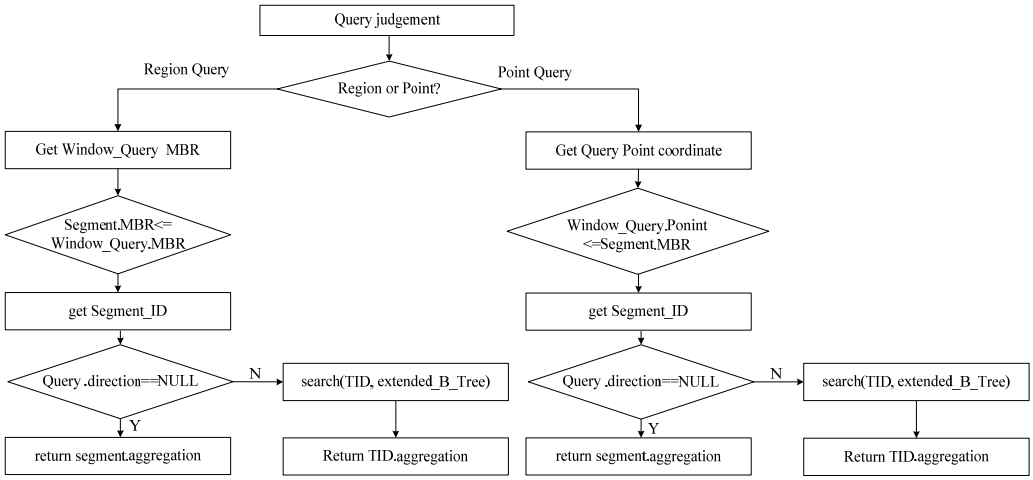


Fig. 13. Flowchart about Search Algorithm of aCN-RB-tree

[Algorithm 3] Region search algorithm of aCN-RB-tree

Input
Window_Query: query window;
query type: Region query;
aCN-RB-tree: Index tree from network;

Variable
Window_Query.MBR: MBR of query windows;
Segment.MBR: MBR of segment;
TID: identification of timestamp;
TID.aggregation: aggregation value of timestamp;
Segment.ID: identification of segment;

```
Query .direction: flag of query direction

Begin
01: Search (MBR,R-tree); // find corresponding MBR in R-tree, and then we can obtain Max
        aggregation and Segment ID
02: if(Segment.MBR<=Window_Query.MBR)
03:    get Segment_ID ;
04: end if
05: if (Query .direction==NULL)// Query doesn't need direction information
06: return Max.aggregation;
07: else search(TID, extended_B_Tree); // else query needs the direction information, so search TID
from extended B-tree
08: end if
09: return TID.aggregation;
End
```

[Algorithm 4] Point search algorithm of aCN-RB-tree

```
Input
Window_Query: query window;
query type: Point query
aCN-RB-tree: Index tree from network;

Variable
TID: identification of timestamp;
TID.aggregation: aggregation value of timestamp;
Segment.ID: identification of segment;
Segment.Point: point in the segment;
Query .direction: flag of query direction

Begin
01: if (point ==Segment.point)// if the point is in the segment
02:    Search Segment_ID;   //we search this segment ID;
03: end if
04: if (Query .direction==NULL)// Query doesn't need direction information
05: return segment.aggregation;
06: else search(TID, extended_B_Tree); // else query needs the direction information, so search TID
from extended B-tree
07: end if
End
```

Suppose our query involves finding how many cars move from left-to-right on road R2 during a certain time? In order to solve this problem, we can search for the association of the trajectories list and R-tree represented in the aCN-RB-tree. A trajectory list is created including the trajectories of moving objects. The attributes are: TID, Node ID, the aggregation with timestamp, and Node-link, as **Fig. 14** shows. TID denotes the trajectory ID, which is simply a number. The node ID expresses the segment ID with directions. In order to facilitate tree traversal, an item header list is built so that each item points to its occurrences in the tree via a Node-link chain. This data model is stored in a spatial data warehouse and can be accessed for query processing.

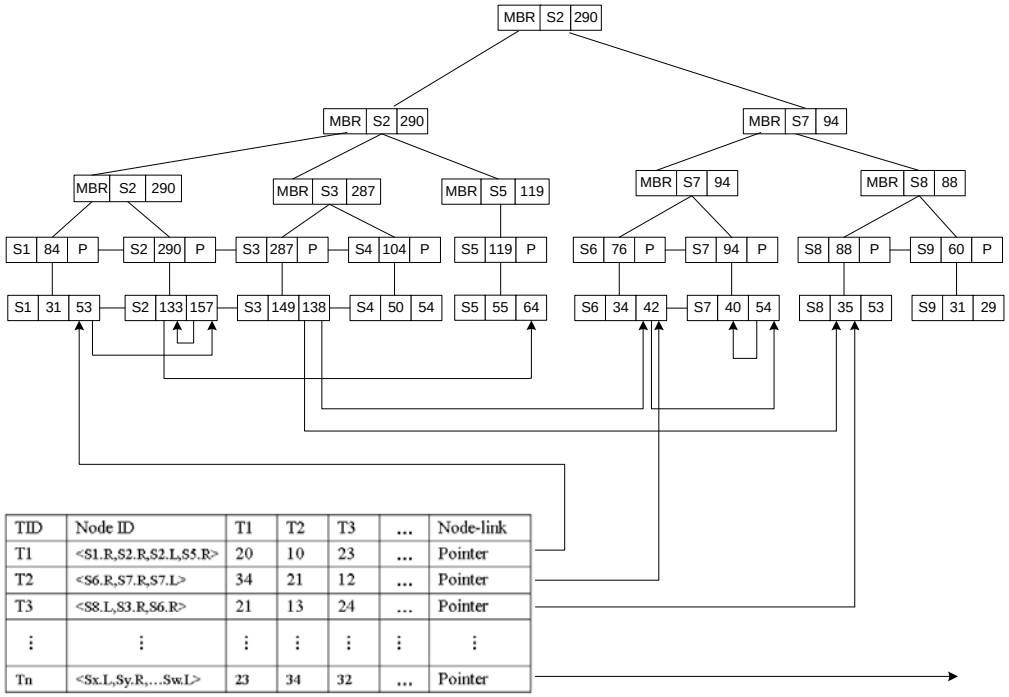


Fig. 14. Example of aCN-RB-tree associated with Trajectory List

3.5 Update Algorithm of aCN-RB-tree

There are two kinds of updating methods: polyline-related updating and temporal-related updating. If a polyline is changed, we need to construct a fixed network R-tree. The temporal-related method needs to update every sub-index of the extended B-tree. To summarize, the aggregation with direction is changed.

The polyline-related update is an algorithm for processing a moving object data set when it updates a spatial region such as an MBR or polyline, which can be seen as the key value. It is kind of traffic redesigning updates [Algorithm 5], including the operations of changing, adding or deleting some roads. Changing roads involves shortening, lengthening and altering shapes. However, the road names are not changed. So we need other pages to store the historical spatial data with a timestamp and attach a label to the road. Adding or deleting some roads needs R-tree reconstruction. First, we obtain this road's MBR and find it in the R-tree. And then we can find the appropriate node for making changes.

[Algorithm 4] Polyline update algorithm of aCN-RB-tree

Input

Update_Segment: segment information which will be updated;
Update_Type: type of updating;
aCN-RB-tree: index tree from network;

Variable

aCN_RB_Tree.Segment: aCN-RB-tree segment;
New_Segment.MBR: MBR of new segment;
aCN-RB-tree.Segment.MBR: MBR of aCN-RB-tree segment;

```
Begin
01: Search (MBR,R-tree); // find corresponding MBR in R-tree
02: do{
03:    get(aCN_RB_Tree.Segment);}
04: while(New_Segment.MBR>=Segment.MBR)
05:    Update (New_Segment_ID, New_Segment.MBR)with get.Segment;
       build New_segment_extended_B-tree;
06: end while
07: return aCN-RB-tree;
End
```

The temporal-related update is an algorithm to update the index using the time stamp as the key value. It processes using a fixed time point on each road segment.

The temporal-related updating means changing every road segment aggregation. It mainly relates to the timestamp and aggregation value. At first, we search the aCNR-tree for the segment, and then we find the timestamp and insert some new values. The algorithm follows the extended B-tree [Algorithm 6].

[Algorithm 6] Temporal update algorithm of aCN-RB-tree

```
Input
Operation: operation type of updating;
aCN-RB-tree: index tree from network;

Variable
Timestamp: last refresh time;
L.aggregation: aggregation value with L direction;
R.aggregaion: aggregation value with R direction;
Network_segment_pionter: pointer connecting network segment;

Begin
01: Search aCN-RB-tree(network,segment_pointer);
02: if  Search aCN-RB-S-tree(network_segment_pointer) == true
03:    Update B.timestamp;
04: end if
05: return aCN-RB-tree;
End
```

Another problem which we have to consider is how to get the aggregated directions if the query window includes different directions. For example, the general collision exists when the road direction is either horizontal or vertical. We can't define the direction and we make some average aggregations to meet this situation.

4. Performance Evaluation

The experiment involved evaluating and determining the conditions of maximal efficiency for the aCN-RB-tree. For a direct comparison with another spatiotemporal access method we chose the 3DR-tree. And the aCN-RB-tree was designed for queries such as "find the number of cars within a given area during a given time interval". The performance of various methods

was measured by the number of nodes accessed during processing of the workload; each consisted of 500 queries. Every query included two parameters that affect performance: the spatial query window(qs) denotes the percentage of its length over the whole network and the interval length (qt).

For a fair comparison, we used an ASCII file of the road network in the city of Oldenbourg and a file with moving objects. We opened the network file first, divided it into segments and constructed the spatial tree. Then we opened the moving object file, added the information to the spatial tree and constructed the extended B-tree. The page size of the leaf and non-leaf nodes is 4Kb, which could save 200 leaf and non-leaf nodes. 100 roads were included in every dataset and were divided into 420 segments by the point of intersection using 1000 timestamps.

234Kb of spatial data was used in the leaf node, which contained 7035 segments. Each segment had a sign and a MBB value. In the non-leaf node, we had 558Kb of non-spatial data including the two-direction aggregation value and the volume of timestamps. There were 20,275 tuples in total and the timestamps ranged from 0 to 50.

In comparison with the a3DR-tree, the timestamp increases in proportion to the primary parameters of memory capacity and frequency of node access.

For example, the aggregate data of 1000 uniformly distributed regions collected over 100 timestamps. At each timestamp, the aggregate data of the regions' aggregate agility was modified. This is a dataset parameter. The aggregate agility is denoted as AA in the following session. AA=5% means 50 regions update their aggregate data per timestamp.

For the first test, we increase the timestamp by percentages of 15%, 30%, 45%, 60% and 75%, respectively. Variations in the generation of tree size and access times can be monitored. **Fig. 15** shows the relationship between the storage and AA.

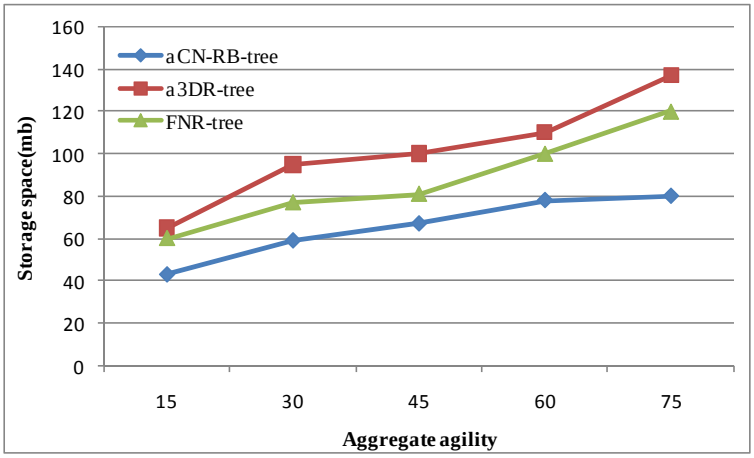


Fig. 15. Results of storage space and aggregate agility

Fig. 15 compares variations in three kinds of trees, FNR-tree, a3DR-tree and aCN-RB-tree. When the AA is relatively low, the storage space overhead of the three trees is quite similar. As the AA increases, the a3DR-tree uses the largest size followed by the FNR-tree. And the aCN-RB-tree uses the smallest space, because for the timestamp, the aCN-RB-tree used an extended B-tree, thereby sharply minimizing the number of temporal nodes.

Fig. 16 presents the average number of node accesses of query processing with different AA values. The condition is qs=5%, qt=50. The number of node accesses of the aCN-RB-tree is lower than that of the a3DR-tree because the FNR-tree (motivation of aCN-RB-tree) is

specified for routing. Also, the accuracy of aCN-RB-tree is higher than that of the a3DR-tree based on region. The adoption of the extended B-tree also minimized visiting times. When the AA is low, the aCN-RB-tree's spatial part is similar to the FNR-tree, which are both based on the road segment. And because the amount of temporal node updating is low, they have the same number of node accesses at the beginning.

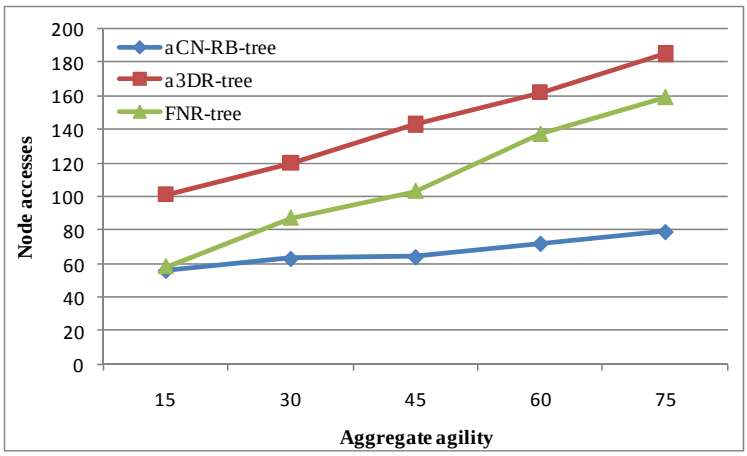


Fig. 16. Results of node accesses and aggregate agility

The aRB-tree was omitted because the aRB-tree corresponds to the region. The aRB-tree and aCN-RB-tree are basically the same in the aspects of the organization and performance except that the aCN-RB-tree corresponds to the network. It clarifies a direction in each segment which remains within non-leaf node, as a result of which it cannot be taken as a quantized comparison.

The next experiment involved evaluating the update processing performance as the number of MBRs increased. **Fig. 17** shows the results of this experiment.

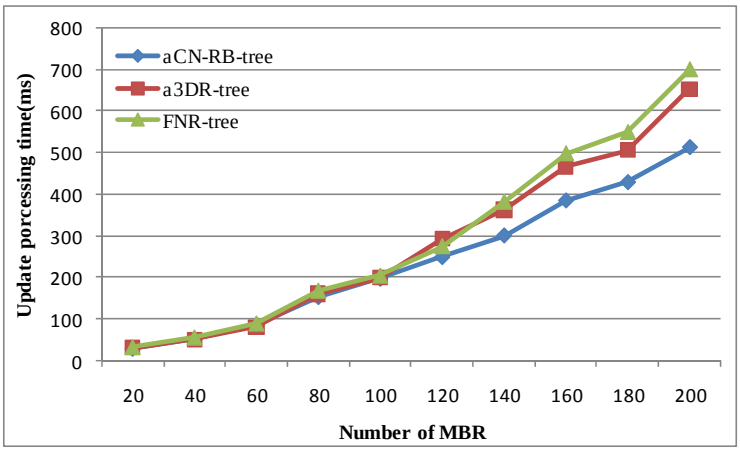


Fig. 17. Results of update processing experiment by number of MBRs

In this experiment, when the number of MBRs to be updated was increased, the proposed index showed better performance than the related indexes. In case where 140 MBRs were updated, it was 28% and 20% faster than the FNR_tree and a3DR-tree, respectively. In the

experiment to update 200 MBRs, it showed a performance gain of 36% and 29% over the FNR-tree and a3DR-tree respectively. The a3DR-tree was proposed to support index range updating, which minimizes the overlap by managing units using road segments within leaf nodes.

The last experiment was evaluated by increasing the number of road segments by 10 moving objects per segment and the results of the experiment are shown in **Fig. 18**.

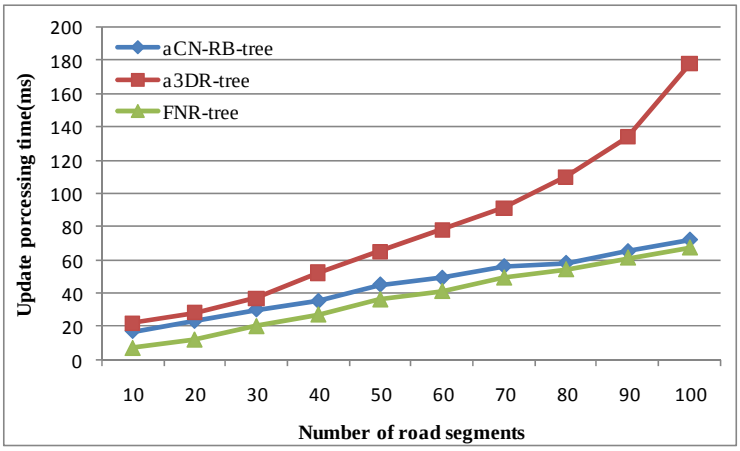


Fig. 18. Results of update processing experiment by time stamp

The a3DR-tree has the poorest performance, because it finds all road segments by MBR. The road segment information in the proposed method only consists of leaf nodes. So, search processing in the non-leaf nodes of the aRB-tree is needed. The proposed index showed almost the same performance as the FNR-tree and also supported the aggregation value with direction.

In summary, the aCN-RB-tree shows good performance in range query (both temporal and spatial). Because of the data structure, it is unnecessary to access many nodes to find an exact value. Given a time searching period, it will reach a veracious value when the data granularity becomes rough, which is very important for data warehousing. In other words, unlike the a3RDtree, the aCN-RB-tree is an on-line structure.

5. Conclusions

It is critical for applications to access spatio-temporal data. Data warehousing can solve the problems of relational and non-spatial databases. However, it is a challenge to store and obtain an exact result with spatial relationships efficiently for spatio-temporal aggregations.

In this paper, we presented the aCN-RB tree index. The tree is used in querying the general instance of a trajectory network in order to provide support for better decisions. This kind of tree uses a fix constrained network index to describe trajectory information. The extended B-tree is used to express the timestamp index in every node and also includes aggregation with two directions. This kind of tree using a leaf node not only connects spatial segments but also spatial and temporal data. Many operations are possible, such as search, insert, delete and update.

To conclude, aCN-RB-tree can support more efficient query answering requirements and can control granularity changes in the data warehouse during a given time interval, which

decreases the query time and uses little storage space. Because it is an on-line structure, it can support PDA and mobile industry applications.

This kind of indexing is tested in a simulation system. Experimental results from the simulation system show that the performance of the proposed technique is better than that of the a3DR-tree and FNR-tree, which is used by general systems.

There are two main issues that are interesting topics for future study. One is how to take into account real-time processing of aCN-RB-tree operations and the other is how to find the exact direction of the trajectory without an application program.

References

[1] J.J. Li, D.W. Lee, B.S. You, Y.H. Oh, H.Y. Bae, "Constraint Network Based Index for Spatio-Temporal Aggregation of Trajectory in Spatial Data Warehouse," *Journal of Korea Multimedia Society*, Vol. 9, No. 12, Dec. 2006.

[2] B. M. I. Lopez, R. Snodgrass, "Spatiotemporal aggregate computation: A survey," *IEEE TKDE*, 2005.

[3] D.W. Lee, S.H. Baek, H,Y, Bae, "aCN-RB-tree: Update Method for Spatio-Temporal Aggregation of Moving Object Trajectory in Ubiquitous Environment," *Proc. of the 7th International Conference on Computational Science and Applications* (*ICCSA*), Yongin, Korea, 2009.

[4] Y. Nakamura, H. Dekihara, "An Efficient Management Method of Moving Spatial Objects", *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing 1999*, Victoria, BC, Canada, 1999.

[5] D. Pfoser, "Indexing the Trajectories of Moving Objects," *IEEE Data Engineering Bulletin*, Vol. 25, No. 2, pp. 2-9, 2002.

[6] D. Pfoser, "Novel approaches to the indexing of moving object trajectories," *Proc. of the 26th Int'l Conf.Very Large Databases*, San Francisco, 2000.

[7] N. Meratnia, N. "Aggregation and Comparison of Trajectories," *Proc. of the 10th ACM International Symposium on Advances in Geographic Information System*, McLean, pp. 6, Nov. 2002.

[8] K.S. Kim, S. Kim, T. Kim, and K. Li, "Fast indexing and updating method for moving objects on road networks," *Proc. of the Fourth International Conference on Web Information Systems Engineering Workshops* (*WISEW'03*), 2003.

[9] V. Teixeira de Almeida, Ralf Hartmut Guting, "Indexing the Trajectories of Moving Objects in Networks (Extended Abstract)," *Proc. of the 16th International Conference on Scientific and Statistical Database Management* (*SSDBM'04*), pp. 219, 2004.

[10] Y. Xia, S. Prabhakar, "Q+Rtree: Efficient Indexing for Moving Object Databases," *Proc. of the 8th International Conference on Database Systems for Advanced Applications* (*DASFAA*) Kyoto, Japan, 2003.

[11] D. Papadias, Y. Tao, P. Kalnis and J. Zhang, "Indexing Spatio-Temporal Data Warehouses," *Proc. of International Conference on Data Engineering* (*ICDE*), 2002.

[12] Y. Theodoridis, M. Vazirgiannis, and T. K. Sellis, "Spatio-Temporal Indexing for Large Multimedia Applications," *Proc. of IEEE International Conference on Multi-media Computing and Systems*, 1996.

[13] E. Frentzos. "Indexing objects moving on fixed networks," *Proc. of the 8th Int'l Symposium on Spatial and Temporal Databases*, Berlin, pp. 289, 2003.

[14] D. Pfoser, "Indexing of network constrained moving objects," *Technical Report*, Data and Knowledge Engineering Group, Computer Technology Institute, Greece, 2003.

[15] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," *Special Interest Group On Management Of Data* (*SIGMOD*), pp. 322-331, 1990.

[16] R. Bayer, "Binary B-Trees for Virtual Memory," *ACM-SIGFIDET Workshop*, 1971.

[17] P. Revesz, Y. Chen, "Efficient Aggregation over Moving Objects," *Proc. of the 10th Int'l*

Symposium on Temporal Representation and Reasoning, 2003.

Dong Wook Lee received B.S. degree from Sangji University, South Korea, in 2003, and M.S. degree from Inha University, South Korea, in 2005. He is currently a Ph.D. Candidate with the Department of Computer Science & Information Engineering at Inha University, S. Korea. His research interests include spatial DBMS, spatial DSMS and spatial data warehousing for ubiquitous environments.

Sung Ha Baek received B.S. degree from Inha University, S. Korea, in 2005, and M.S. degree from Inha University, S. Korea, in 2007. He is currently a Ph.D. Candidate with the Department of Computer Science & Information Engineering at Inha University, S. Korea. His research interests include spatial DBMS, ubiquitous GIS and spatial data stream management systems for ubiquitous environments.

Hae Young Bae is a Professor at Inha University, South Korea. He received B.S. degree from Inha University, South Korea, in 1974, M.S. degree from Yonsei University, Korea Republic in 1978, and Ph. D in computer engineering from Soongsil University, S. Korea, in 1989. He has worked as the Dean of Graduate School of Information Technology and Telecommunication at Inha University from 2004 to 2006 and as the Dean of Graduate School at Inha University from 2006 to 2009. Prof. Bae's areas of interest include spatial and multimedia databases and related areas.