Research on modeling and indexing of Trajectories of moving objects in road networks

Yanling ZHENG
Depart. of Electronics and Information Technology
Jiangmen Polytechnic
Jiangmen, China
e-mail: jmptzheng@126.com

***Abstract*—Proposed a new index structure, named MG2R*, can efficiently store and retrieve the past, present and future positions of network-constrained moving objects. It is a two-tier structure. The upper is a MultiGrid-R*-Tree (MGRT for short) that is used to index the road network. The lower is a group of independent R*-Tree. Each R*-Tree is relative to a route in the road network, can index the spatiotemporal trajectory of the moving objects in the road. Moreover, moving object's query is implemented based on this index structure. It compared to other index structures for road-network-based moving objects, such as MON-Tree, the experimental results shown that the MG2R* can effectively improve the query performance of the spatio-temporal trajectory of network-constrained moving objects.**

Keywords- moving objects；database；index；trajectory

I. INTRODUCTION

For research on various aspects of moving objects, index and query technology of moving objects, is the management of moving objects, are also the key core technology of spatio-temporal database and moving object database. There are numbers of index for different situations about patio-temporal index, such as 3DR-Tree, TB-Tree, TPR-Tree, and so on. But in the vast majority of index methods assume objects to be free in space, and storing and indexing moving objects spatio-temporal information for each sampling time. However in practice, moving objects are mainly located in the road networks, which are characterized by running path is already given in advance, such as road networks of the city, cars can run on those roads identified, and cannot be changed at will. Running application based on the fixed network is the most common and value form in actual research.

In recent years, People come to realize the importance of network-constrained moving objects, and made numbers of index structure of moving objects database based on road network, such as FNR-Tree and MON-Tree, but the index structure can only deal with history, cannot support the query of moving object in the current and future position. To solve this problem, MG2R* index structure is presented, effectively support the past, present and future position of the query.

II. THE SPATIO-TEMPORAL TRAJECTORY MODEL OF NETWORK-CONSTRAINED MOVING OBJECTS

Reasonable model of road network is the key to built effective temporal and spatial index. At present, there are two main methods for modeling road network structure. One is edge-based model of road network structure; second is route-based model. From the perspective of index methods, route-based model is better than the edge-based because of the smaller expression of moving objects. Also, when the data volume is large, route-based model can reduce the amount of index data. Therefore, this paper is based on route model as an example for illustration purposes.

It is the model that this paper studied is based on the spatio-temporal trajectory model of network-constrained moving objects which the paper [1-2] suggesting. For details on the related definition of the model, refer to paper [1-2].

III. MG2R*

MG2R* using the general layered structure of the current index of moving objects based on road network, which carry out a separation processing to the static network and dynamic moving objects. On the top level, MGRT used to index road, the R* tree leaf node of the MGRT point to the bottom R* tree, which used to index object along the polyline movement. In addition, combined moving objects location updates, the MG2R* index structure realized motion vector (which contains information on current and future position of a moving object) in the index of explicit said and dynamic maintenance, thus for the moving object query provides a comprehensive range of support.

A. The upper of MG2R:MGRT*

Common grid index divides the network into M×N grid, each grid area as an index item, recording the entity identity and boundary rectangle of all fully or partially falls within the region, indexing through the entity's boundary rectangle operating space entity. As the same entity may be across multiple grids, then this entity at the crossing are to be stored in the grid, and cause data redundancy. If grid is divided too large, small geographic objects cannot be pinpointed. Conversely, the division of the grid is too small to provide higher search accuracy, but also longer time to create spatial index, the more data redundancy.

Published by Atlantis Press, Paris, France.

Based on the shortcomings of common grid index, this paper uses adaptive MultiGrid[3-4] index. First, the geographical range is divided into M × N. If the total number of spatial object in a grid exceeds the maximum tolerable default, the grid will be divided into K × L, and then recursively determine each grid of the K × L, until grid series is reached the default maximum series or the number of space object in the grid is less than the default maximum number.

R-tree is an extension of the B-tree in multidimensional space, which is a balanced tree structure. In R -tree structure, the search performance depends on two parameters: coverage and overlap [5-7]. Due to the MBR of the R-tree internal node happened overlap, the worst performance of search operation is immeasurable. To improve the search performance of R-tree requires minimal coverage and overlap, and overlaps minimized minimize even more critical than the coverage.

R*-trees [8-11] is an improved R-tree. R*-tree the most remarkable feature is that all of the MBR of internal nodes are disjoint and zero overlap is produced among the internal nodes. Although this will increase the height of the tree (although only slightly), but the search performance has been greatly improved.

Based on the above analysis, a multistage hybrid index structure is proposed based on the advantages of MultiGrid and R*-Tree, namely MultiGrid-R*-Tree, referred to as the MGRT. It is the structure that geographical space is divided into multistage grids to establish multistage index, and R*-tree index is then created for each grid which can not be divided.

As shown in Figure 1, the distribution of MBR of some routes in the multistage grid: first of all, the entire space with 2×2 division is divided into 4 first-level grids. The grid-0 does not contain MBR of any route, and grid-3 contains only one MBR of route. Therefore, grid-0 and grid-3 don't need to be performed secondary division, but it need to be established directly R*-tree index with n for the root node in grid-3. Grid-1 and grid-2 contains more routes. If directly to be established R*-tree index in the grid, R*-tree contains a greater number of node. Therefore secondary division is performed in the grid-1 and grid-2. In Figure 1, grid-1 is divided into 2×3; grid-2 is divided into 2×1. G and h completely fall into grid-15, and form R*-tree with 2 objects' MBR. B and c fall into grid-10 and grid-11 respectively, and form R*-tree with only root node respectively. Across four grids of grid-10, grid-11, grid-13 and grid-14, F is no longer stored in R tree, but stored directly in the cross-grid-list. In addition, i，j，k and l are in the same secondary grid, and form R*-tree with 4 objects' MBR. In practice, each grid may contain hundreds and even thousands of objects, so it may choose figure 1 as shown to be divided once more.

In figure 1, the coordinate of the lower-left corner of the whole region is set to be origin, and the coordinate of the upper-right corner is set to (60, 60). Therefore, the coordinate of the lower-left corner of grid-0 of the first-level is set to (0, 0), and the coordinate of the upper-right corner is set to (30, 30), which lead to the coordinate of grid-0 of the first-level to (0, 0, 30, 30). Set up the index structure According to the figure 1 as shown in figure 2, the data structure of the index that consists of several array, a single chain which contains cross-grid index information and a pointer which contains index information of the root node of the R*-tree.

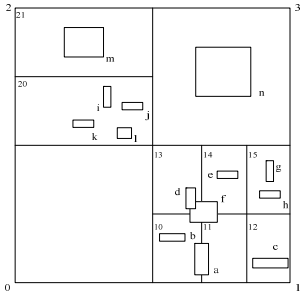

Figure 1. objects in multi-grid space

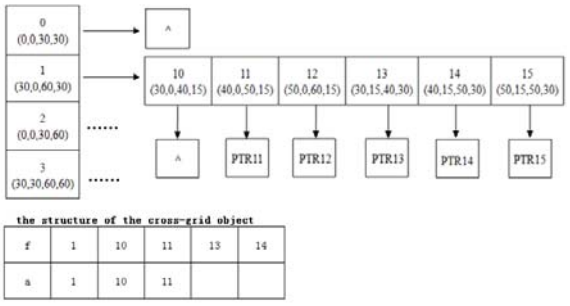

Figure 2. the index structure of MGRT

Seen from the above, it is a multistage hybrid index structure that the upper of the MG2R* structure, MGRT, is based on the MultiGrid and R*-tree and made route as the basic index unit. If the MBR of some routes completely fall into the same grid, R*-tree index is set up for the routes. Each grid stores the pointer, which points to the root node of the R*-tree. The index information of cross-grid route is stored directly in the corresponding grid. The leaf node of R*-tree contains information for (MBR, rid, PTR). MBR is the minimum boundary rectangular of rid. The rid refers to the identification of corresponding route. It is a pointer that PTR points to the bottom R*-tree of the corresponding route. Intermediate nodes is formed for (MBR, childptr), of which the MBR contains MBR of all child nodes; childptr is a pointer pointing to child node. Because of the relatively fixed topological structure of road network, the upper of the MG2R* structure, MGRT, belongs to static structure.

B. The bottom of MG2R:R*-tree*

Each R*-tree of the bottom of the MG2R* structure corresponds to a specific road named rid. The R*-tree is used to index the trajectory unit generated while the moving objects driving on the rid update the location. The intermediate nodes of R*-tree is formed for (*MBR, PTR*), which *MBR* contains *MBR* of all child nodes; *PTR* is a pointer pointing to child node. The leaf node is formed for (mid, mv_s, mv_e, MBR), which *mid* is identification for moving object; *MBR* is the *MBR* of the trajectory unit of *mid*; mv_s

and mv_e are the two continuous motion vector corresponding trajectory unit.

IV. THE ESTABLISHMENT AND THE MAINTENANCE OF INDEX

When moving object is traveling in road networks, it needs to compare continuously the current operating parameters (such as location, speed, direction, etc) with the motion vector: $mv_n = (t_n, (rid_n, pos_n), \vec{v}_n)$, which is submitted by the last location update. Once some predefined conditions of location update are met, it needs to trigger location update process, and send the latest operation parameters to the server.

The location updates in this paper use the following four strategies [2] at one time:

1) Object just enter the system, and began to run;

2) Object changes the running route;

3) The deviation of the position of object is beyond a certain threshold ε;

4) The deviation of the speed of objects is more than a certain threshold ξ.

When the object is created or the change of the speed of object is more than the threshold ξ or the deviation of the position of object is beyond the threshold ε, it will trigger update and bring a motion vector. When the object moved from a road *r1* to another road *r2*, it also need to update and bring 3 motion vectors:

$$\mathrm{mv}_i = (\mathrm{t}_i, (\mathrm{rid}_i, \ \mathrm{pos}_i), \ \vec{\mathrm{v}}_i)(\mathrm{i} = 1,2,3) . \quad (1)$$

Note that *mv1* and *mv2* respectively corresponding to the state of the moving objects in through the intersection of road *r1* and road *r2*, *mv3* corresponding to the current update state.

Insert operation included the insertion of the data record of road network into the MGRT index, as well as the insertion of trajectory unit of moving objects into R*-tree of the bottom of the MG2R* structure.

Step 1: Scan data record of the road network, and establish MGRT index. The R*-tree of the bottom of the MG2R* structure which the records of MGRT point to is empty.

Step 2: Judge the number *n* of motion vector contained in the update information, when moving object sends location update information to the server. If the number $n = 1$, turn step 3, or turn step 4.

Step 3: Determine the route which rid refers to according to the motion vector of moving objects, and get its MBR. To create or update the information of the R*-tree of the bottom of the MG2R* corresponding to the route in the suitable grid of MGRT, and update the record of the R*-tree. To insert record (*mid, mv_i, MBR*) in the R*-tree corresponding to *rid* when the mv_n of *mid* is null. Otherwise, it need to delete records (*mid, MVN, MBR*) in the R*-tree corresponding to *rid*, and then insert record *(mid, MVN, mv_i, MBR)* and *(mid, mv_i, MBR)*.

Step 4: If the update information contained 3 motion vectors, moving objects changed operation route. Suppose that the motion vector of the updated information of the moving object:

$$\mathrm{mv}_{ai} = (\mathrm{t}_{ai}, (\mathrm{rid}_{ai}, \ \mathrm{pos}_{ai}), \ \vec{\mathrm{v}}_{ai})(\mathrm{i} = 1,2,3) .$$

Determine the moving object *mid* in R*-tree corresponding to rid_{a1}, delete records *(mid, mv_n, MBR)* in R*-tree, and then insert record *(mid, mv_n, mv_{a1}, MBR)*. Determine the moving object mid in R*-tree corresponding to rid_{a2}, and then insert record (*mid, mv_{a2}, mv_{a3}, MBR*) and (*mid, mv_{a3}, MBR*).

V. QUERY

Because the index structure contains activities trajectory unit, it can support not only the moving objects in the past position query but also the current and future position query.

1) Global history query: according to the id of the moving object, Traversing the lower R*- tree you can get.

2) The time query: the leaf node of the lower R*-tree stores the trajectory unit of the moving object which contains time information. It meets the requirements when the time information intersects with query time *(t1, t2)*.

3) Spatial query: The rectangular box *(x1, x2, y1, y2)* compared with the MBR of grid index. If the rectangular box intersected with the MBR, recursive search should go on, and until all routes which met the space requirements, were obtained. Get the leaf node records for R*-tree of each route in order to collect the history trajectory of all the moving objects in the space.

4) Spatio-temporal query: Give spatio-temporal query window *Q = (x1, x2, y1, y2, t1, t2).* All moving objects, which are located in the area of space *r = (x1, x2, y1, y2)*, are found out in time interval *t = (t1, t2)*. First, use the inquired frame and some already existing grid index, get the intersection grid of the space and query box, take out the route in the grid, and remove repetitive route. Recursive search the route which met the time requirement in R* tree of each route at time *(t1, t2)*. Note that a route may intersect with the part of query space window. It needs to use the trajectory unit of objects in the intersection to obtain the accurate trajectory information.

5) The future position query: Because of topological relationship of road network (each road to maintain its adjacency road information), it can predict a location of the moving objects in the future time according to the speed and direction.

VI. INDEX PERFORMANCE ANALYSIS

Testing environment used the hardware to Pentium® 4 2.66GHz CPU and 1G memory, the operating system to Windows XP, all the procedures implemented by the JAVA language and used the development environment for JDK6.0.The experimental data is generated by the moving object based on the road network which is presented in the paper [12], and road network data come from the German road traffic network from reference [12]. Moving object data set are used for the 10000, 15000, 20000, 25000, 30000,

35000, and 40000 objects, and the life cycle of moving objects set to [0,500] seconds. For reasons of simplification, assumes that the moving object speed running. This section of the experiment mainly to index storage and create spending, window query efficiency were compared and analyzed between MG2R* and MOTN-Tree.

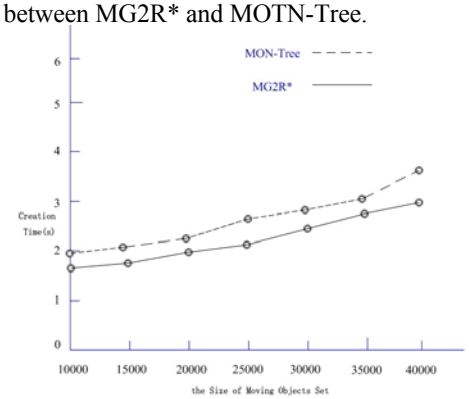

Figure 3. The influence of the creation time

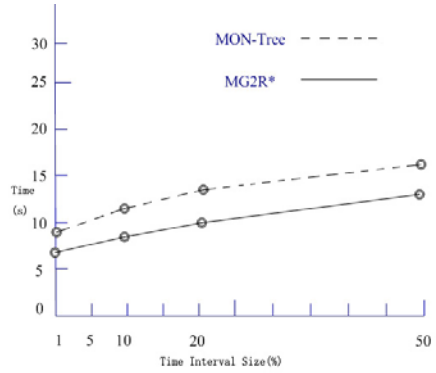

Figure 4. Time interval size on the influence of the query performance

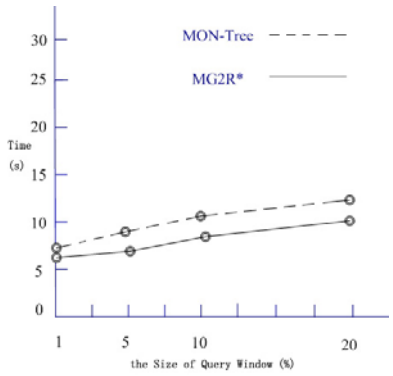

Figure 5. The size of query window on the influence of the query performance

As Figure 3 shown, the changes in the size of the moving object datasets have an influence on the creation time of the two indexes. From the Figure 3, creation time of MG2R* is a little less than the creation time of MON-Tree.

Figure 4 and Figure 5 pointed out the difference in performance on window query of the two indexes. From the figures, the efficiency in window query of MG2R* had a slightly better than MON - Tree.

VII. CONCLUSION

Moving object index based on road network is a key technology of support mass moving object management. However, the methods of current constrained network moving objects index such as FNR-Tree, MON-Tree and other major according to historical trajectory index, can't support the operation of moving objects in the current and future position query, thus greatly influenced its practicability.

To solve the above problems, this paper puts forward MG2R* index suitable for constrained network moving objects database. The experimental results show that MG2R* index provides a good retrieval and maintenance process performance.

REFERENCES

[1] Ding Zhiming,Li Xiaonan,Yu Bo. Indexing the Historical, Current, and Future Locations of Network-Constrained Moving Objects [J]. Journal of Software,200912(20):3193-3204

[2] Huang Qunshan. Research and Implementation Of Digital map editing system and its supporting technology [D].ZheJiang University.2008

[3] Ding Z, Güting RH. Managing moving objects on dynamic transportation networks. In: Proc. of the 16th Int'l Conf. on Science and Statistical Database Management (SSDBM 2004). Santorin: IEEE Computer Society, 2004. 287−296.

[4] T. Brinkhofi. A framework for generating network-based moving objects. GeoInformatica, 6(2):153 - 180, 2002.

[5] Chen J, Meng XF. Indexing future trajectories of moving objects in a constrained network. Journal of Computer Science and Technology, 2007,22(2):245−251.

[6] Guo J, Guo W, Zhou DR. Indexing approach for querying about present and future based on constrained moving objects in spatial-temporal databases. Journal of Chinese Computer Systems, 2007,28(2):128−131 (in Chinese with English abstract).

[7] A. Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. Proc. of theACM SIGMOD, Boston, MA, June 1984: 47-57.

[8] Shi Shaoyu,Tang Xinming,Wu Fan,Lei Bing,Wang Huibing. Multi-level grid spatio-temporal index [J]. science of surveying and mapping,20063(31):54-55.

[9] Hao Zhongxiao. Spatio-temporal Database Query and inference [M].Beijing:Science Press,2010:6-7.

[10] T. K. Sellis, N. Roussopoulos, C. Faloutsos. The R*-tree : A Dynamic Index for Multidimensional Objects. Proceedings of the Thirteenth International Conference on Very Large Data Bases : t987 13th VLDB, Brighton, UK, 1987. Los Altos, CA, USA, Morgan Kauthaarm,1987:507-518.

[11] D. Pfoser, Indexing the Trajectories of Moving Objects. IEEE Data Engineering Bulletin,2002,25(2):3-9.

[12] http://data.geocomm.com/catalog/c_index.html.