Advances in Intelligent Systems Research (AISR), volume 151
2018 International Conference on Computer Modeling, Simulation and Algorithm (CMSA 2018)

A Moving Objects Index Method Integrating GeoHash and Quadtree

Ya Ban[1], Rui Wang[1], Hongjing Liu[1], Jing Yuan[1], Hao Luo[1], Fuli Yang[2], Ling Yu[1] and Xinping Xu[1]
[1]Chongqing Academy of Metrology and Quality Inspection, Chongqing 401120, China
[2]State Grid of Chongqing Electric Power Co. Electric Power Research Institute, Chongqing 401123, China

***Abstract*—To take into account of frequent update, efficiency and query capability, a new full-time index structure was proposed with help of one-to-one match between Quad-tree nodes and GeoHash codes, named GQ-tree, which supported moving objects updating frequently. An efficient index update algorithm based on GeoHash encoding was proposed to improved efficiency. Algorithms of efficient nearest neighbor query, spatiotemporal query, and continuous query were proposed with the help of Hash table, linked list and GeoHash of moving object position. Experimental results prove GQ-tree is better than existing methods in terms of index update, spatiotemporal query and so on.**

Keywords—GeoHash; bottom-up update; Quadtree; nearest neighbor query; moving object index

I. INTRODUCTION

Research of moving object database was mainly focused on: representation and modeling of moving object, moving object index, query technologies and privacy protection. As one of the key technologies of moving object, index played an important role in spatiotemporal query[1].The research of moving object spatio-temporal index can be divided into two kinds: One is the index of the historical trajectory of the moving object[2],such as RT-Tree[3],STR-tree[4],HR-tree[5]; the other is the index of the current and future position of the moving object, such as PMR-quadtree[6],TPR-tree[7],TPR*-tree[8],VCI R-tree[9], ATPR-tree[10],STRIPES-tree[11], improved Quadtree [12], associated tree[13].In particular, the study on the full-temporal indexing of moving objects is less successful in academia [14,15,16].

Location of moving objects location update frequently, which results in dramatic changes in index structure [17]. Due to inherent top-down update model, methods above had a large I/O cost, and did not support nearest neighbors query. GeoHash can transform two-dimensional coordinates of latitude and longitude into a code, as the world's unique identification to express points. We can reuse Hash table, B-tree to query data at one-dimensional space by GeoHash, and it is more simple and efficiency than two-dimensional index. Beside, GeoHash can effectively prevent geographic information leakage.

To solve the problems, we focus on reducing the cost of various queries and frequent updates. This paper presents an efficient index for moving objects, which integrates Quad-tree, GeoHash and linked list, named GQ-tree. Efficient index update algorithm based on GeoHash (GEIU) is proposed.

II. GQ-TREE AND ALGORITHM

A. Principle

GQ-tree divides space based on the similar zoning rules between GeoHash and Quad-tree, and there is a consistent one-to-one match between Quad-tree nodes and GeoHash codes, the principle as shown in Figure Ⅰ. Quadtree leaf nodes, brothers and parent node can be quickly search by GeoHash, and no need frequent access to the Quadtree. Moving object locations are stored in forms of GeoHash encode, not only for index update, but also indicating location ,and helping to protect privacy. GeoHash encoding length controls region area, the shorter the encoding, the larger the area. Adjacent locations have the same prefix, which has obvious advantages in solving problem of nearest neighbors query.

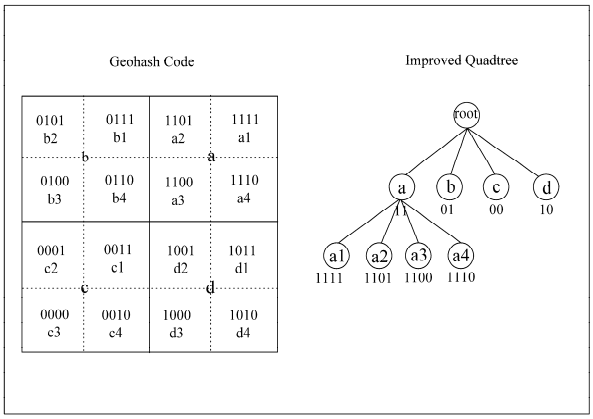

FIGURE I. PRINCIPLE

B. GQ-Tree Index Structure

GQ-tree index structure as shown in Figure Ⅱ. In order to store full-time trajectory data of moving objects, GQ-tree consists of two parts: one is an improved Quadtree for storing present and future trajectories of moving objects; the two is hash table and linked list to store historical trajectory data.

According to BUU [18], GeoHash efficient index update (GEIU) algorithm is proposed in this paper. The key issues to performance of GEIU are brother's leaf nodes search algorithm and reverse backtracking search algorithm. We introduce a direct access table used to point to the Quadtree nodes, so we can access Quadtree nodes directly without extra disk I/O. At the upper-left corner is the direct access table. GeoHash in the table is leaf node code, through GeoHash code, leaf node contained moving object can be found, at the same time, through the

GeoHash encoding can brothers and parent node can be found; *MBR* as a node of minimum bounding rectangle; *Ptr* as a node pointer, which points to the disk address of the node; hash table at left-lower corner is used to store list of moving objects history trajectories. The hash table node structure is <*OID, LeafNodeLink, LinkedList*>.*OID* is moving object identification; LeafNodeLink is a pointer pointing to address of moving objects in Quadtree;*LinkedList* points to the head node of the one-way linked list.The one-way link node structure is <*OID,X,Y,Geohash, StartTime, EndTime, V_x, V_y, NextLink*>. *OID* is moving object identification; GeoHash is the start point of trajectory. *X* and *Y* are longitude and latitude at *StartTime* respectively; *StartTime* and *EndTime* are start time and end time of trajectory respectively; V_x, and V_y are velocity vector at *X* and *Y* direction; *NextLink* points to next node.

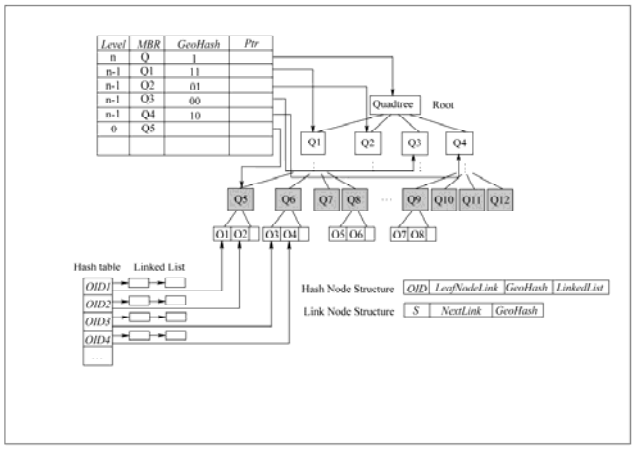

FIGURE II. INDEX STRUCTURE

C. Update Algorithm

When the speed and direction of change exceed the threshold, whether moving object is within MBR of leaf node, and data of leaf node is updated directly; Otherwise, algorithm deletes old records from the leaf nodes and selects a suitable leaf node from its sibling node to insert a new record; if a suitable sibling leaf node cannot be found, a subtree containing MBR of moving object can be found backtracking on tree from leaf nodes, then standard insertion algorithm is executed to the subtree. The *GetParentNode* algorithm is introduced.

Given Quadtree as Q, Hash auxiliary table as *H*, direct access table as *DAT*, moving object identification as *oid*, moving object GeoHash as *newGeohash*, intermediate node variable as *entry*, direct access table record variables as *node*, Hash auxiliary index records as *h*, Quadtree record as *Q*, pseudo code is as follows:

```
entry=DAT[0];
if (entry.Geohash⊄ newGeoHash.substring(0,
newGeoHash.length-2)) {
 Update ( T , oid , newGeohash)
 return ;
}
foreach(h in H)
if (h. oid ==oid){
    node = h. LeafNodeLink;
}
if (node.Geohash ⊂ newGeoHash.substring(0,
newGeoHash.length-2))
   write out node ;
   else
{
 node =node. parent ;
 foreach(q in node) {
   if (q.Geohash⊂ newGeoHash.substring(0,
newGeoHash.length-2)){
   Insert an index entry with newGeohasha into node;
          return ;
       }
     }
  }
 node =GetParentNode (node, newGeohash);
 Insert(node, oid , newGeohash);
```

According to the input intermediate node of Quadtree, *GetParentNode* algorithm can find a subtree containing updated moving object and the node from the direct access table. Given the direct access table as *DAT*, tree height as *H*, the intermediate variable node level as *l*, the direct access table record as *entry*, *GetParentNode* algorithm is described as follows:

```
foreach(entry in DAT ) {
    if (entry.Geohash == newGeoHash.substring(0,
newGeoHash.length-2)){
        node =entry. Ptr
       foreach(entry in DAT )
       {
       if(node.Geohash == newGeoHash.substring(0,
 newGeoHash.length-2))
        node =entry. Ptr
       return node;
   }
}
```

III. QUERY FOR MOVING OBNECTS IN ADVANCED

A. K-Nearest Neighbors

K-Nearest Neighbors (KNN) was to query k nearest moving objects at a given time. KNN query needed to process a large amount of data, so a lot of spatiotemporal overload was generated based on the existing indexes. Based on the characteristic of the same GeoHash prefix in the same region, GeoHash encoding could be applied to the nearest neighbor query. For example, in order to query nearest neighbors of the Point (116.389550, 39.928167), the following query SQL sentence can be executed: SELECT * FROM place WHERE GeoHash LIKE 'wx4%'.

Since GeoHash encoding is based on the space filling curve, this space filling curve has mutations. Therefore, the case GeoHash code adjacent, but the distance far away may appear. AS shown in Figure Ⅲ, the point A and C of encoding is more similar to A and B, but the distance between A and C is far more than B and A. In order to solve the problems above, eight regions around the area can be applied to query.

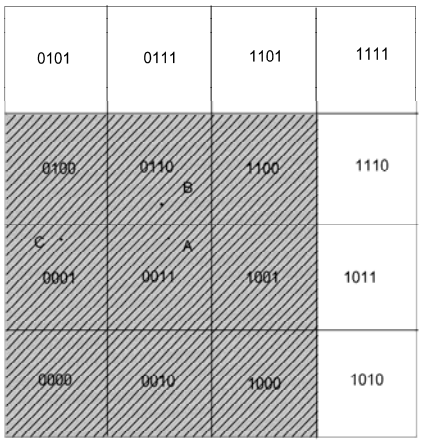

FIGURE III. GEOHASH ENCODING

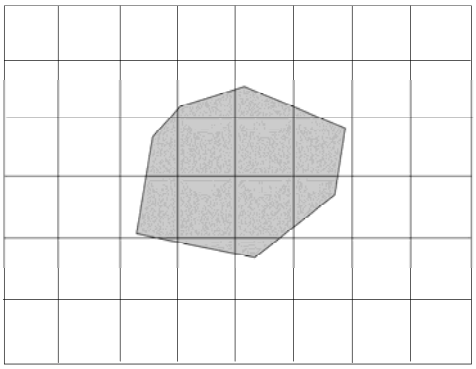

FIGURE IV. QUERY REGIONAL DIVISION METHOD

B. *Time Range and Region Query*

Time range and region query is to query moving objects in the query window within a given time. In the paper, we propose to implement the regional query by query area of GeoHash. Firstly, the query region is divided, secondly the covered grids are encoded (Figure Ⅳ); finally, fuzzy query is executed to each GeoHash codes.

C. *Continuous Query*

Continuous query is to query the moving object trajectory over a period of time. As shown in Figure Ⅱ, the GQ-tree had built a one-way linked list for all moving objects, and each object trajectory at any time can be query by OID.

IV. EXPERIMENTAL EVALUATION

In order to examine the performance of our proposed index structure, the GQ-tree, we did an experimental evaluation. We also implemented the TB*-tree and compared to our results.

A. *Experiment*

For our experiments we used a personal computer with 64 Window 7 system, Intel Core I7-3370M, 3.30GHz CPU, 4G main memory. The index structure were implemented in C#, and stored in MongoDB database. In our experiments, we used moving objects generator proposed in [19].

B. *Exoerimental Contents and Results Analysis*

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

1). *Updating Performance*

100000 moving objects were generated and the nodes access were evaluated by the updates of 10k, 30k, 50k, 70k, and 90k.Results shown in Figure Ⅴ. GQ has better dynamic update performance, and the number of update nodes access maintained at 15. Most of the GQ index updates happened in local, and only a few searched from top to bottom.

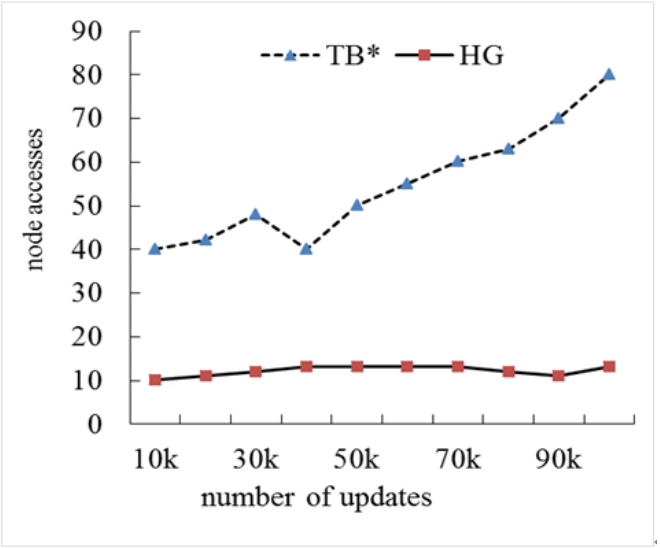

FIGURE V. UPDATING PERFORMANCE

2). *Query Performance*

The number of moving objects was 100000.Spatial areas queries were 1%,3%,5%,7% and 9% of the whole space. The simulation evaluated the cost of queries, query results shown in Figure VI. The query performance of GQ-tree had better efficiency. Through GeoHash encoding to reduce the dimension of two-dimensional coordinates, query efficiency greatly improved.

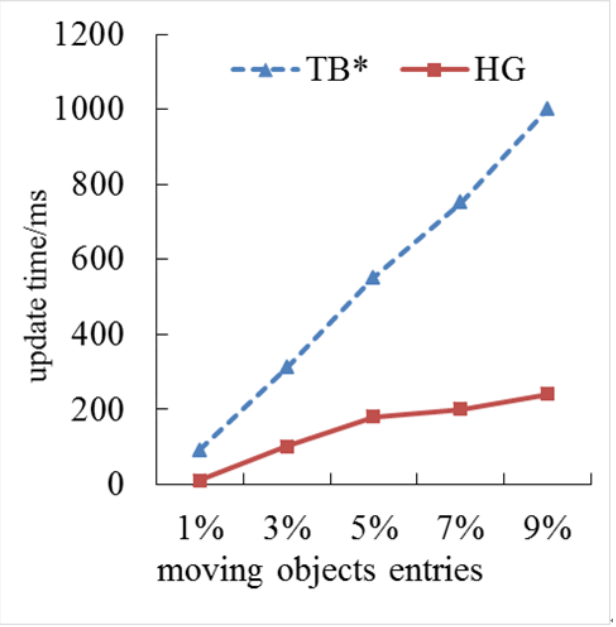

FIGURE VI. QUERY PERFORMANCE

V. SUMMARY

In the paper, a new method of moving object trajectory is proposed, which makes full use of the advantages of four fork tree, GeoHash encoding and hash table, etc. The update efficiency is improved by the strategy of GEIU. Privacy protection is a problem considered and researched to database system. However, current privacy protection technology of moving objects database is not mature. In virtue of GeoHash encoding, it both can improve search efficiency and effectively prevent location privacy leakage.

Many temporal and spatial applications (such as fire simulation, etc.) need to efficiently query the changing range of moving objects, but, the index proposed in the paper is designed for moving point objects. The moving objects trajectory for polyline and polygon were not mentioned, and this will be the future research area.

ACKNOWLEDGMENT

This work was financially supported by the National Power Grid Corp headquarter science and technology project(No. 2017 Yu electricity science and technology management 7#).

REFERENCES

[1] L. Zhang , D. Q. Tang, C. Zhang, et al. “Evolvement and Progress of Spatio-temporal Index”. Computer Science, no.4,vol.37,pp.15-20,2010.

[2] X. L. Yu, Y. Chen, X. C. Ding, et al. “A Moving Object Database Model Based on Road Network”. Journal of Software, no.14,vol.14,pp.1600-1607,2003.

[3] W. Liao,W. Xiong, N. Jing,et al. “Research on indexing moving objects methods”. Computer Science, no.8,vol.33,pp.166-169,2006.

[4] D. PFOSER, C. S. JENSEN, Y. THEODORIDIS. “ Novel Approaches in Query Processing for Moving Object Trajectories”.// VLDB 2000, Proceedings of, International Conference on Very Large Data Bases, September pp.10-14, 2000, Cairo, Egypt. DBLP.

[5] Z. C. Yin, L. Li,Z. Wang. “Spatio-temporal index based on extended HR-tree”. Geomatics and Information Science of Wuhan University, no.12,vol.32,pp.1131-1134,2007.

[6] H. Xiao, Q. Q. Li. “Access methods in moving objects databases”. Journal of computer applications, no.4,vol.30,pp.1064-1067,2010.

[7] J. PATEL, Y. CHEN, V. CHAKKA, et al. “An Efficient Index for Predicted Trajectories”.//Proc. of ACM SIGMOD International Conference on Management of Data. Paris, France: [s. n.], pp.637-646, 2004.

[8] Y. Tao, D. PAPADIAS, J. Sun. “The TPR*-Tree :An Optimized Spatio-Temporal Access Method for Predictive Queries”.// International Conference on Very Large Data Bases. VLDB Endowment, pp.790-801, 2003.

[9] S. PRABHAKAR, Y. Xia, D. V. KALASHNIKOV, et al. “Query Indexing and Velocity Constrained Indexing: Scalable Techniques for Continuous Queries on Moving Objects ”. Computers IEEE Transactions on, no.10,vol.51,pp.1124-1140,2002.

[10] Y. H. Wang, E. R. Zhang. “ATPR-Tree: spatio-temporal index with attribute dimension”.Computer Engineering and Application, no.7,vol.53,pp. 79-87,2017.

[11] J. M. PATEL, Y. CHEN, V. P. CHAKKA. “STRIPES: an efficient index for predicted trajectories”.// ACM SIGMOD International Conference on Management of Data, Paris, France, June. DBLP, pp. 635-646,2004.

[12] J. Zeng, X. L. Ke. “Improved Quadtree based indexing method for moving objects”. Geomatic Science and Engineering, vol.3,pp.40-43,2012.

[13] J. Gong, S. N. Ke, Q. Zhu, et al. “An Efficient Trajectory Data Index Integrating R-tree, Hash and B*-tree”. Acta Geodaeticaet Cartographica Sinica, no.5,vol,44,pp. 570-577,2015.

[14] J. Guo, G. J. Liu, L. Guo, et al. “A Whole-time Index Design Based on 3D+ -TPR-tree for Moving Point Targets”. Acta Geodaeticaet Cartographica Sinica ,No.3,vol.35,pp. 267- 272, 2006.

[15] X. C. Meng, S. Z. Ye. “New spatio-temporal index method based on real-time data and query log distribution”.Journal of Computer Applications, no.3,vol.37,pp.860-865,2017.

[16] L. B. Ma, X. C. Zhang. “Research on full-period query oriented moving objects spatio-temporal data model”. Acta Geodaeticaet Cartographica Sinica ,no.2,vol.37,pp.207-211,2008.

[17] X. T. Yi, Z. Xu,K. Zhang. “Index structure for moving objects based on restricted network”.Computer Science,no.5,vol.42,pp.211-214,2015.

[18] L. L. Mong,H. Wynne,S. J. Christian , et al. 2003. “Supporting frequent updates in R-trees:A bottom-up approach,” VLDB2003,Berlin.

[19] T. Brinkhodd. “A Framework for Generating Network-base Moving Objects,” Geoinformatica,no.2,vol.6,pp.153-180 , 2002.