Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
September 2008, 15(Suppl.): 64–67
The Journal of China Universities of Posts and Telecommunications
www.buptjournal.cn/xben

Indexing structure for moving object databases based on R-tree

HE Yun-bin[1], HAO Zhong-xiao[1, 2]

1. College of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China
2. College of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China

Abstract

A general trajectory model based on moving point is introduced, which can model low dimension and high dimension moving objects. Discrete update strategies are discussed. On this basis a 2-level indexing structure based on R-tree is put forward. It indexes the object's current and past moving trajectory by R-tree and dynamic array connected with its leaf node. The method of update, insertion and deletion based on the structure has been discussed in detail. It is implied that the structure supports queries on past, now and future, and the space intersection is very small.

Keywords moving object databases, R-tree, indexing structure

1 Introduction

Moving object database is a rising branch in database research field. Moving objects' positions of different times are recorded in moving object database, thus users can effectively manage moving objects, for example, users can query about moving objects' information of a certain time in past, present and future. Moving object databases' application is very wide, including civil aviation control, traffic control, military command and position-based information service, etc. For example, a taxi corporation can query about taxis that will be at some place ten minutes later, a military command center can query about enemy helicopter's flight route during the past half hour.

There is a great deal of researches on moving object databases. Several models have been presented, including R-tree [1] and its transmutations (STR-tree, TPR-tree [2] and LUR-tree [3], etc.), quadtree [4] and its transmutations (PMR-quadtree, XBR-quadtree, etc.), grid structure [5], etc. They all have shortcomings. There exists space overlap (dead space) problem in R-tree and its transmutations. Quadtree and its transmutations have the problem of inflexibility. Grid structure can't effectively manage dynamic data. A two level index structure based on R-tree and dynamic array in presented is this paper, which can support query on past, now and future, and effectively reduce space overlap. A trajectory modeling method based on moving points is introduced in Sect. 2, which can model low dimension and high dimension moving objects. The 2-level index structure is discussed in Sect. 3. The update, insertion and deletion algorithms of the structure are detailedly discussed. Section 4 is the conclusion, it gives future research emphases.

Received date: 08-02-2008
Corresponding author: HE Yun-bin, E-mail: hybha@163.com

2 Object trajectory model based on moving points

When we manage and query about real moving objects, such as cars, airplanes and foot passengers, we can take no account of their geometry. Thus we use space points to represent various moving objects in this paper. We define moving objects' attributes at a certain time point including an exclusive identifier, position, velocity and time, etc. as follows.

Definition 1 Moving object (MO) is a quadruple, MO = (oid, $p(p_1, p_2,\ldots, p_i)$, $v(v_1, v_2,\ldots, v_i)$, t), in which oid represents exclusive identifier, i represents space dimension, p represents position, v represents velocity, t represents time.

Different moving objects have different moving space dimensions. For example, cars move on 2-dimension plane, airplanes move in 3-dimension space. At time point t, position p, we can represent the position of a car at time point t' $(t' > t)$ with velocity of v as follows.

$$p = (p_1, p_2) + (v_1, v_2)(t' - t) \tag{1}$$

The position of an airplane under the same condition can be represented as

$$p = (p_1, p_2, p_3) + (v_1, v_2, v_3)\ (t' - t) \tag{2}$$

For moving object's trajectory, we use interpolation by sampling points to describe whole moving trajectory. We can sample moving object's positions of different times, and then

apply linear interpolation between adjacent positions.

Definition 2 Moving object trajectory (MOT) consists of a sampling points sequence. MOT = (m_1, m_2,…, m_n), in which m = (oid, $p(p_1, p_2,\ldots, p_i)$, t), n is number of sampling points.

The trajectory is a polygonal line consists of segment sequences, in which sampling points represent terminal vertexes of segments. We combine time dimension with space dimension in building the model. A car's moving trajectory during a certain time quantum is shown in Fig. 1, in which bold polygonal line represents the trajectory, points corresponding to t_1, t_2, t_3 and t_4 represent sampling points, the dotted polygonal line on the plane represent actual moving route.

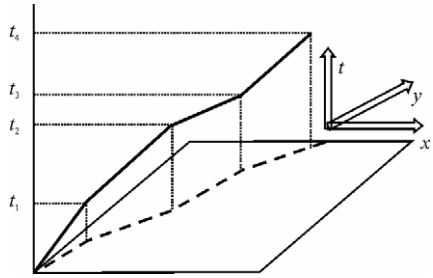

Fig. 1 Trajectory of moving object

Moving object's position varies continuously over time and need to be updated continuously [6,7]. The update strategy used in database can trace not only moving object's current position, velocity and direction, but also moving object's trajectory. It associates an appropriate uncertain value with each past trajectory. The query result involving past trajectory relates to possibility (namely the result may satisfy known query predication). Discrete update strategies are classified as fixed time interval update strategy, simple absolute estimate update strategy and adoptable absolute estimate update strategy.

With simple absolute estimate update strategy, a maximum deviation d is set, and moving object's position is sampled timely. The database won't be updated unless the deviation between actual position and the position described in database exceed d. d is also determined according to real situation.

The adoptable absolute estimate update strategy is almost same with simple absolute estimate update strategy except that moving object can alter update limit. Each moving object has time and space record and the maximum deviation d of the newest report. When moving object reports new time and space record, limit value can be altered for future update.

3 2-level index structure based on R-tree

3.1 Model of 2-level index structure

A 2-level index structure is introduced here. It uses R-tree to index current position of moving object. Each leaf node corresponds to a moving object. Only current position of the object is stored in leaf node. A dynamic array is used to store its historic points' positions. Pointers are used to connect leaf node and its corresponding dynamic array. Moving objects in Fig. 2 are represented by index structure shown in Fig. 3. In Fig. 3, rectangle represents dynamic array, connected with corresponding leaf node by pointer.

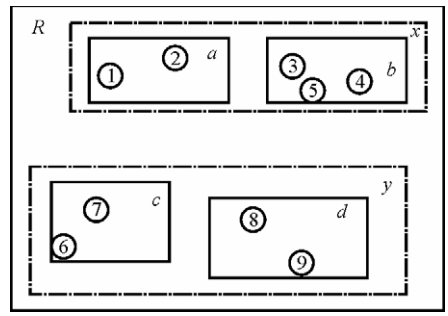

Fig. 2 A space objects' collection

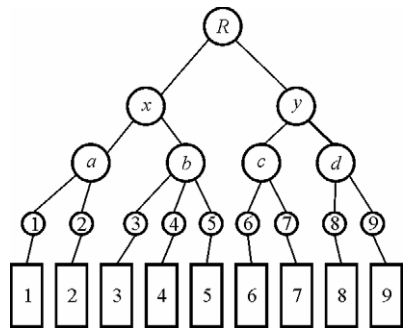

Fig. 3 2-level index structure

Each non-leaf node in R-tree is represented as

$$N = (\text{MBR, pointer [ ]}) \quad (3)$$

in which pointer[ ] is pointer pointed to sub nodes, MBR is minimum bounding rectangle of the sub nodes it contains.

Each leaf node is represented as

$$\text{Leaf} = (\text{oid}, p(p_1, p_2,\ldots, p_i), v(v_1, v_2,\ldots, v_i), t, \text{pointer}) \quad (4)$$

Oid is an exclusive identifier of object, p is latest updated object's position, v is latest updated object's velocity, t is latest update time, pointer points to the dynamic array which stores its corresponding historic position.

Each data item in the dynamic array pointed by leaf node is represented as

$$D = (p(p_1, p_2,\ldots, p_i), t) \quad (5)$$

At a time point t in the past, the object is located at p, for n historic sampling position of the object, there are nD values in the corresponding dynamic array.

The 2-level index structure can dynamically update, insert and delete nodes. When a new object node is inserted, a dynamic array is allocated. Node points to the array by pointer. When a node is deleted, the corresponding dynamic array is removed. When an object node is updated, because