# 基于路网的移动对象索引机制研究

郭景峰　王建朝　董宏宇　闫立华
（燕山大学信息科学与工程学院　秦皇岛 066004）

**摘　要**　本文基于FNR-Tree的思想提出了一种新的索引算法FNR$^+$-Tree，该算法可以实现基于轨迹的查询，而这正是FNR-Tree索引结构所欠缺的，接着给出了FNR$^+$-Tree的数据结构和插入算法，查询算法，最后给出了两种索引结构的试验对比结果。

**关键词**　路网，FNR$^+$-Tree，轨迹，索引

## Indexing Scheme of Moving Objects on Fixed Networks

GUO Jing-Feng　WANG Jian-Chao　DONG Hong-Yu　YAN Li-Hua
(College of Information Science and Engineering, Yanshan University, Qinhuangdao 066004)

**Abstract**　Based on the FNR-tree method, a new method, namely the FNR$^+$-tree method, is proposed in this paper. In this literature, the focus is the data structure, insertion algorithm and trajectory-based query algorithm of the FNR$^+$-tree. The result of the experiment is given and it shows that the FNR$^+$-tree method is efficient.

**Keywords**　Network, FNR$^+$-Tree, Trajectory, Index

## 1　序言

移动对象数据库是数据库领域近年来发展起来的一个分支，已有很多成果，其中有关移动对象索引的算法也有很多，如TPR-Tree，TB-Tree，3D R-Tree，STR-Tree，SETI等都是对移动对象的轨迹建立索引，也就是对移动对象过去的位置信息建立索引，其中TB-Tree提出了轨迹保存的概念[1]，TPR-Tree，PPR-Tree是对移动对象当前时刻位置信息建立索引，并预期将来时刻的位置。但是这些索引算法是在移动对象的运动没有限制的情况下进行的，都不能用来对基于路网的移动对象建立索引。而实际应用中，很多情况下移动对象的运动有时是受到一定条件限制的，比如火车是在铁路上运行的，汽车是在公路网上行驶的等等。目前基于路网移动对象索引结构的研究还处于初级阶段，成果比较少，有关的工作是Elias曾提到的FNR-Tree[2]索引结构。

FNR-Tree由一个2D R-Tree和1D R-Tree森林组成，2D R-Tree用来对路网建立索引，1D R-Tree对移动对象的时间信息建立索引，这种结构对于包含有空间操作的时空范围查询具有很好的性能，但由于它的各个1D R-Tree之间是孤立的，对于时间片查询和基于轨迹的查询，必须遍历1D R-Tree森林中的所有叶子节点，代价是及其昂贵的，效率也是很低的。

基于FNR-Tree思想，本文提出了一种新的基于路网的索引结构FNR$^+$-Tree，它可以很容易实现基于轨迹的查询操作。

## 2　路网的表示模型

现实世界中的路网结构是复杂的，如图1中a所示，为了便于计算机的处理，在本文中把复杂的路网结构用二维空间来表示，这样表示的路网由一系列线段组成[3]，如图1中b所示。


图1　(a)现实世界中的路网　(b)二维空间表示的路网结构

## 3　FNR$^+$-Tree数据结构

FNR$^+$-Tree由一个2D R-Tree、1D R-Tree的森林和一个哈希表＋单链表结构构成，并且每棵1D R-Tree都和2D R-Tree叶子节点中包含的一个数据项相对应，每个移动对象都对应一个单链表。在整个FNR$^+$-Tree结构的生命周期内，只要路网没有变化，那么2D R-Tree就是一棵静态树。

FNR$^+$-Tree用一个2D R-Tree对组成路网的线段建立索引。2D R树每个叶子节点包含若干个数据项，每个数据项有一个与之对应的1D R-Tree，并且每个数据项都指向路网上的一个唯一的线段。1D R-Tree用来对移动对象在一个路段(1 D R-Tree所对应的2 D R-Tree的数据项包含的线段)上运行的时间间隔建立索引。移动对象经过路网上每个结点时记录它当前的位置信息。在两个结点之间，假设移动对象的运动是匀速的，这样就可以计算某个时刻的移动对象的位置了[2]。因此，路网的建模是相当重要的。

2D R-Tree中，非叶子节点的数据项形式为(ptr，MBR)，ptr是指向孩子节点的指针，MBR是包含子节点中所有线段的最小外接矩形。每个叶子节点包含有若干个数据项，数据项形式为(ID，Route id，MBR，Orientation，ptr)，ID是一个指针，它是线段的实际存储地址。Route id是路网上的路径的

**郭景峰**　硕士生导师，教授；**王建朝**　硕士研究生，主要研究方向：移动对象数据库；**董宏宇**　硕士研究生，主要研究方向：移动对象数据库；**闫立华**　硕士研究生，主要研究方向：移动对象数据库。

编号，线段只是路径的一部分，一个路径被分成多个线段，MBR 是线段的最小的外接矩形，Orientation 是一个标志位，用来表示线段在 MBR 中的实际位置，它的取值是 0 或者 1（图 2）。ptr 指向与它对应的 1D R-Tree 的根节点[2]。

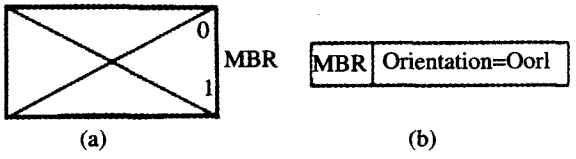


图 2 (a)线段在 MBR 中位置 (b)相应的 2D R-Tree 节点表示

1D R-Tree 中，非叶子节点数据项结构为（ptr，$T_{entrance}$，$T_{exi}$），ptr 为指向子节点的指针，$T_{entrance}$ 是其子节点中的最小时刻，$T_{exit}$ 是其子节点中的最大时刻。叶子节点的数据项结构为（Moving Object ID，$T_{entrance}$，$T_{exit}$，Direction，ptr），Moving Object ID 是移动对象编号，$T_{entrance}$ 为此移动对象进入线段的时刻，$T_{exit}$ 为此移动对象离开线段的时刻，ptr 指向此 1D R-tree 对应的数据项（2D R-Tree 中）所指向的线段，Direction 表示移动对象的运动方向，即移动对象进入线段的方向，它的取值为 0 和 1。如图 3 所示，当移动对象从线段的左边端点进入时，Direction 的值为 0，反之为 1。若线段为垂直方向，那么当移动对象从线段的上面端点进入时 Direction 取值为 0，反之为 1。

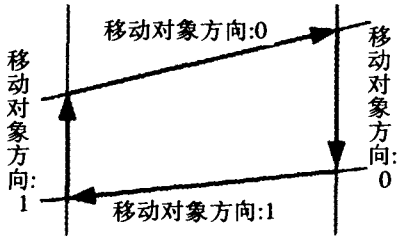


图 3 路网上方向的表示

哈希表＋单向链表结构是一个复合存储结构，主要用来实现基于轨迹的查询操作。哈希表用来存储移动对象的信息，表中的数据是每个移动对象的编号，即 Moving Obejcet ID(1D R-Tree 叶子节点数据项中的 Moving Object ID)，是唯一的。每出现一个新的移动对象，哈希表就存入这个新对象的 Moving Obeject ID，并且哈希表中每个数据项都指向一个单链表结构，如图 4 所示。该单链表按照移动对象运行轨迹（也可说是按照时间顺序）把 1D R-Tree 森林中的属于同一个移动对象的数据项连到了一起。

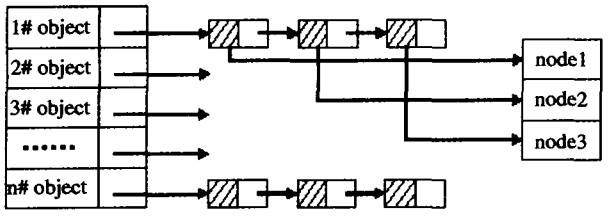


图 4 哈希表＋单链表结构

Node1，node2，node3 分别是 1＃移动对象有关的 1D R-tree 森林叶子节点中的数据项，并且它们时间上是连续的，按时间顺序依次是 node1，node2，node3。那么 1＃移动对象的哈希表＋单链表结构如图 4 所示。

## 4 操作算法

### 4.1 插入

移动对象从网路上的线段离开时执行 FNR$^+$-Tree 的插入操作。首先执行 2D R-Tree 搜索算法，在 2D R-Tree 的叶子节点中找到移动对象离开的线段，接着找到与这个线段对应的 1D R-Tree，然后执行 1D R-Tree 的插入算法，最后执行哈希表＋单链表的插入操作。下面分别介绍了三种算法操作。

(1)2D R-Tree 的搜索算法

```
Algorithm:Search(root,W)
//W 为空间查询窗口
Begin
  if(root 不是叶节点)then
      for(root 中每个数据项 entry[i])do
        if(entry[i]. rect 与 W 相交)then
          Search(entry[i]. rect,W)
        endif
      end
  else
      for (root 中每个 entry[i])do
        if(entry[i]. rect 与 W 相交)then
              output(entry[i]. rect,entry[i]. ref)
        endif
      end
  endif
End
```

(2)1D R-Tree 的插入算法

在 1D R-Trees 中，时间间隔是单调变化的，所以时间间隔是按增长的顺序插入 1D R-Tree 结构中的。每个新的数据项直接插入 1D R-Tree 最右边的叶子节点中。如果叶子节点存储空间已经满了，那么将新创建一个叶子节点，然后将新的数据项存入新的叶子节点中，新创建的叶子节点插入最右边叶子的父节点中。若最右边叶子节点的父结点的存储空间也已经满了，那么 R-Tree 将向上延伸和繁殖。在 1D R-Tree 的插入算法中，新的数据项总是插入到最右边的叶子节点中。

(3)哈希表＋单链表的插入算法

1D R-Tree 中插入新数据项后，然后要建立单链表结构。首先按照新数据项中的 Moving Object ID 在哈希表中查找，若查找结果为空，则在哈希表中存入这个新对象的编码-Moving Object ID，然后指向一个新创建的单链表，同时把 1D R-Tree 新插入的数据项的地址存入到单链表中。若查找结果不空，则通过哈希表，查到与之对应的单链表，在单链表末尾插入一项，然后将 1D R-Tree 新插入数据项的地址存储到单链表中。

### 4.2 查询

本文提出的 FNR$^+$-Tree 主要是在 FNR-Tree 基础上改进的，并且对于 2D R-Tree 和 1D R-Tree 的叶子节点的数据项进行了修改，然后加入新设计的哈希表＋单链表结构，因而 FNR$^+$-Tree 在范围查询和时间片查询方面的性能和 FNR-Tree 是相同的，但 FNR$^+$-Tree 比 FNR-Tree 优越之处就在于它具有了良好的轨迹查询性能，这是 FNR-Tree 所不具有的。

在 FNR$^+$-Tree 中，若要查询已知移动对象的轨迹，首先按照对象的编码（Moving Object ID），在哈希表中找到该移动对象的值，然后取出它所指向的单链表。由于单链表是指向 1D R-Tree 森林中所有包括该移动对象数据的叶子节点，并且是按时间顺序把所有数据项连接起来了，因此查询轨迹时就不用遍历 R-Tree 森林，而直接根据单链表中的信息，取出所有线段的值，然后连接起来就形成了该移动对象的轨迹。而在 FNR-Tree 存储结构中，若要查找一个移动对象的轨迹，必须查询所有 1D R-Tree，把包含该移动对象的所有节点查找出来，然后再根据时间的信息进行排序，代价是非常大的。

在后面我们可以用试验来证明这一点。

若查询 3D 时空窗口中移动对象的轨迹，首先需要找到包含在查询窗口中的所有移动对象(范围查询)，然后再根据哈希表＋单链表结构得到移动对象的运行轨迹。算法如下：

(1)首先要执行 2D R-Tree 的搜索算法，找到路网上包含在空间查询窗口中的线段。查找包含了这些线段的叶子节点，然后存储这些线段。

(2)在和前一步线段对应的所有 1D R-Tree 中执行搜索算法(和 2D R-Tree 搜索算法类似)。

(3)对于第二步检索到的 1D R-Tree 叶子节点的所有数据项(包含有时间区间)，结合 1D R-Tree 对应的线段(每棵 1D R-Tree 对应于 2D R-Tree 中的一个数据项，该数据项指向路网上唯一的一个线段，这个线段在第一步中已经存储了)，和 3D 时空查询窗口进行比较，把不满足条件的去掉。到这里为止，实现了范围查询。保存 1D R-Tree 中所有符合条件的数据项。

(4)在第三步查询中得到 1D R-Tree 中所有符合条件的数据项，通过这些数据项可以得到所有满足条件的移动对象编号(Moving Object ID)。根据这些编号找到与之对应的单链表，就可以得到移动对象的轨迹。

## 5 试验比较

### 5.1 数据集

实验中采用了 T Brinkhoff 的基于路网的移动对象数据生成器[4]产生的数据集和一个真实的道路网络图 Oldenbourg[5]。在以 Oldenbourg 作为路网的数据生成器中(图 5)，规定初始产生的移动对象个数为 100 个，每个时间戳新产生移动对象 5 个，最大时间戳数为 1000。那么在随后的 1000 个时刻里，每个时刻将会有 5 个新的对象产生，所有对象都将在路网上移动，并且有的对象由于运动到路网之外而消失。在整个数据集生成过程中，共产生移动对象 5100 个。

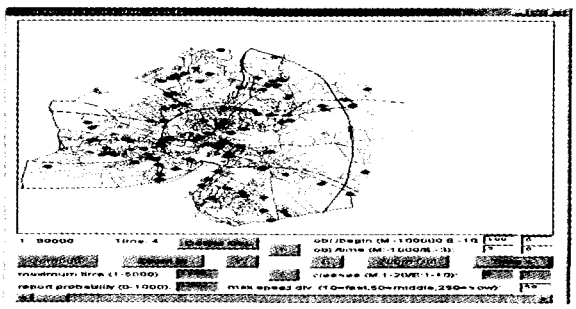

图 5 以 Oldenbourg 作为路网的移动对象数据生成器

### 5.2 试验结果

试验给出了 FNR-Tree 和 FNR$^+$-Tree 在时空范围查询和轨迹查询方面的比较结果。在时空范围查询方面，二者的性能相差无几，而对于轨迹的查询，FNR$^+$-Tree 的性能远远胜于 FNR-Tree，这主要是由于在 FNR$^+$-Tree 中采用了哈希表＋单链表结构，而这种结构将 1D R-Tree 森林中属于同一移动对象轨迹的数据项按时间顺序连接到了一起，这和 TB-Tree 中采用的轨迹保存属性(Trajectory Preservation)[3]是类似的。图 6 是时空范围查询，图 7 是对象轨迹查询。

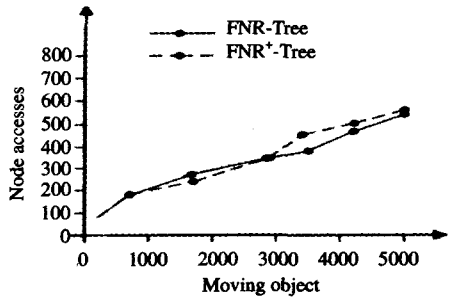


图 6 范围查询

**总结** 本文提出了一种新的基于路网的索引方法 FNR$^+$-Tree，它很好地实现了基于轨迹的查询操作。由于 FNR$^+$-Tree 是基于路网建立的一种索引方法，因此路网的建模对于索引的建立是至关重要的。本文为了便于研究，路网的建模仅仅采用了二维表示方法，而实际生活中的路网结构是相当复杂的，因此今后对于路网建模问题的研究应该是一个重点。

## 参 考 文 献


1 Pfoser D, Jensen C S, Theodoridis Y. Novel approaches to the indexing of Moving Object Trajectories. In: Proc. of the 26th Intl. Conf. on Very Large Databases, 2000. 395～406

2 Frentzos E. Indexing Objects Moving on Fixed Networks. In: Proc. of the 8th Intl. Symp on. Spatial and Temporal Databases (SSTD), 2003. 289～305

3 Speicys L, Jensen C S, Kligys A. Computational Data. Modeling for Network-Constrained Moving Objects: [DB Tech. Report]. Aalborg University, 2003

4 Brinkhoff T. Generating Network-Based Moving Objects. In: Proc. of the 12th Int'l Conf. on Scientific and Statistical Database Management (SSDBM'00), 2000. 253～255

5 http://www.fh-oow.de/institute/iapg/personen/brinkhoff/generator/


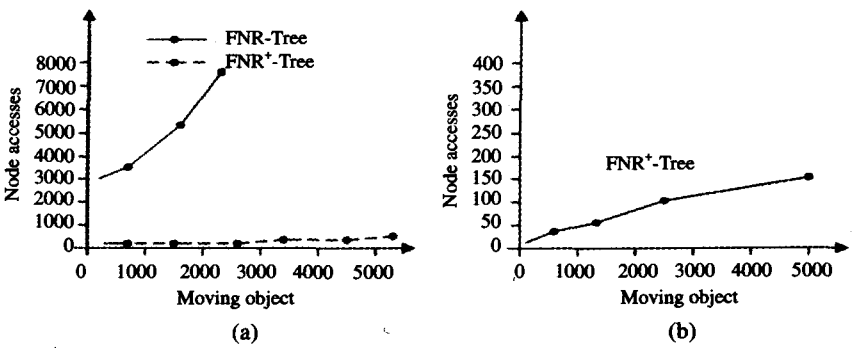


图 7 轨迹查询(a)FNR-Tree 与 FNR$^+$-Tree 比较 (b)FNR$^+$-Tree 的查询