

# 时空数据库索引方法研究

祝蜀平,赵瑾瑾

(上海师范大学 数理信息学院,上海 200234)

**摘要:**时空数据库作为数据库研究领域中的一个重要分支,经过近十年的发展,在时空数据模型、时空查询优化与索引和时空本体论等方面取得了许多成果。现实世界中的许多实体都具有空间特性和时态特性,需要数据库管理系统提供有效的时空数据管理能力,如地籍管理系统中的地块、交通管理系统中的车辆等。时空数据库用于管理形状和位置随时间变化的对象。为了快速访问其庞大的数据量,必须建立有效的时空索引以提高各类时空查询的效率。提出了一种新的时空索引方法(SEST索引),它综合了快照和事件这两种时空信息建模方法。不仅能够处理时间片查询和时间段查询,而且能够进行事件查询。SEST索引使用R-tree结构来存储快照,用一种日志数据结构来存储发生在两次相邻快照之间的事件。通过实验对比SEST索引和HR-tree,结果表明:当变化频率在1%到13%之间时,SEST索引比HR-tree需要的存储空间少;当变化频率在1%到7%之间时,在时间段查询方面,SEST索引比HR-tree要好。因为SEST索引是一种面向事件的结构,所以事件查询时效率很高。

**关键词:**时空索引;R-tree;时态事件

**中图分类号:**TP311

**文献标识码:**A

**文章编号:**1673-629X(2008)07-0056-04

## Research of Spatio-Temporal Access Method

ZHU Shu-ping, ZHAO Jin-jin

(Mathematics and Sciences College, Shanghai Normal University, Shanghai 200234, China)

**Abstract:** Spatiotemporal database is an important branch in the research field of database. After about ten year's development, researchers have made significant progress in areas of spatiotemporal ontology, spatiotemporal model, spatiotemporal query optimization and index. Many entities in real world have both spatial and temporal characteristics, such as parcels in land management systems, vehicles in traffic management systems and so on. As more complicated applications concerning these entities appear, it is necessary for modern database management systems to be capable of providing efficient means to manipulate spatiotemporal data. Spatiotemporal database is used to manage objects that change their locations and shapes as time passes. It is essential to build effective spatiotemporal indices in order to improve query performance because of the huge mount of spatiotemporal data in the applications. Describes a new spatiotemporal access method (SEST-Index) that combines two approaches for modeling spatiotemporal information: snapshots and events. This method makes it possible to not only process time slice and interval queries, but also queries about events. The SEST-Index implementation uses an R-tree structure for storing snapshots and a log data structure for storing events that occur between consecutive snapshots. Experimental results that compare SEST-Index and HR-tree show that, for a change frequency between 1% and 13%, SEST-Index requires less storage space than HR-tree, and for a change frequency between 1% and 7%, SEST-Index outperforms HR-tree for interval queries. In addition, as SEST-Index is an event-oriented structure, event queries are efficiently answered.

**Key words:** spatiotemporal access methods; R-trees; temporal events

## 0 引言

现实世界中任何事物都有时间和空间两个固有的属性,因此,一个事物可以由其在任一时刻的位置和范围所确定<sup>[1]</sup>。这些事物就组成了在一些计算机应用领域需要被管理的各种时空数据。比如交通监控的车

辆、森林火灾的火灾区域和动物研究中的候鸟迁移等,都需要使用时空数据库来管理。

时空数据应用根据它们所管理的数据类型可以分为以下三种<sup>[2]</sup>:

(1)处理连续变化的应用,例如在高速公路上行驶的汽车的移动。

(2)有一些空间中存在的事物可以通过改变自身的几何形状来改变自身的位置,如一个城市随时间发展它的管理区域发生变化。在包含此类事物的应用

收稿日期:2007-10-03

作者简介:祝蜀平(1978-),女,河南三门峡人,硕士研究生,研究方向为网络与多媒体;导师:胡金初,教授,研究方向为网络与多媒体。

中,事物的几何形状的改变是以一种离散的方式发生的。

(3)还有一种应用包括了以上两种行为。这种应用可能出现在环境领域,即要对事物的连续移动进行建模,又要对它们的随时间而变化的几何形状进行描述。

文中提出的方法适合于以上提到的第二类应用,即支持事物在位置和形状上的离散变化。本方法基于这样一种思想:在事物发生一系列变化后产生快照,并且用一种叫做日志的数据结构来保存引起这些变化的事件。因此,它采用了两种表示方法:时态快照和事物

上发生的事件,这两种表示法在文献[3]里都有描述。快照的数量需要权衡存储空间和查询响应时间而定,因为大量的快照可以减少查询响应时间,却要求更多的存储空间。反过来,少量的快照会降低存储空间要求,却增加查询响应时间。

目前为止,综合考虑时间和空间因素的索引算法仅有四种,即 MR-trees<sup>[4]</sup>, RT-trees<sup>[5]</sup>, 3D R-trees<sup>[6]</sup>和 HR-trees<sup>[7]</sup>。RT-tree 时态信息被保存在 R-tree 的节点里,也就是对通常的 R-tree 的信息内容的一个扩展。由于检索主要还是根据空间信息进行,所以时态信息处于次要角色。这种方式下,含时态信息的查询效率不可能高。3D R-tree 认为时间是二维空间之外的另一维,用这种方法,对于一个在时间段  $(t_i, t_j)$  时位于  $(x_i, y_i)$  处,而后在时间段  $(t_j, t_k)$  时位于  $(x_j, y_j)$  处的对象,可以用三维空间的两个线段表示:  $[(x_i, y_i, t_i), (x_i, y_i, t_j)]$  和  $[(x_j, y_j, t_j), (x_j, y_j, t_k)]$ , 这种结构可以用 3D R-tree 来索引。这种方法的缺点之一是不适合时间片查询,优点是能够有效利用存储空间和有效处理时间段查询。HR-tree 和 MR-tree 都基于重叠的概念。基本思想就是,给定两棵树,最近的一棵树始终对应前一棵树的一次演化。HR-tree 的主要优势在于处理时间片查询时效率很高。最主要的缺点是需要额外的空间来存储它的结构。和以前的研究不同,目标是定义一种新的数据索引方法来有效地进行时间片、时间段和基于事件的查询。

## 1 文中提出的方法:SEST 索引

此方法的思想是维护数据库在某些时刻的快照,并用日志来存储发生在两个连续的快照之间发生的事

件。当一个对象的空间属性在某个特定的时刻发生改变时,就会产生一个事件。日志按时间顺序存储,可以重构两个连续快照之间的数据库的任何状态(见图1)。笔者认为,对于每个快照,数据库中存活的对象的空间组件都存储在 R-tree 数据结构中。就像前面所提到的变化在日志中体现。例如图1中,快照  $t_0$  中对象的状态存储在  $R_0$  中,在  $(t_0, t_i)$  时间段内使对象发生改变的事件存储在日志  $L_0$  中。因此,如果要恢复满足  $t_0 < t < t_i$  的时刻  $t$  的数据库状态,就从  $t_0$  时刻的 R-tree 开始,用日志  $L_0$  中的信息来修改对象的属性(即位置)。

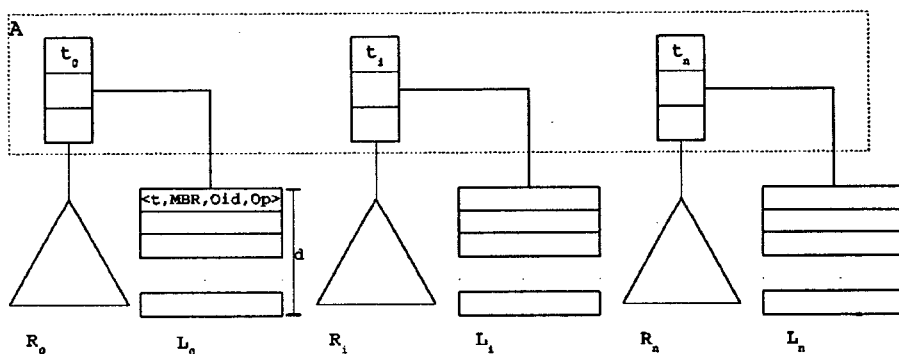


图1 SEST 索引整体纵览图

### 1.1 结构描述

R-树的结构同文献[8]中提到的一样,日志的数据结构可以用块的链表来实现。块的结构为  $\langle t, \text{Geometry}, \text{Oid}, \text{Op} \rangle$ ,  $t$  是变化(如插入一个新对象)所发生的时间, Oid 是对象标识符, Geometry 是对象的空间组件的值,存储的空间对象的类型不同这个值也不同,对象如果是二维空间的一个点,那么 Geometry 是这个点的坐标  $(x, y)$ , 对象如果是多边形,那么 Geometry 是定义多边形或它的近似值的点的集合,比如多边形的 MBR(最小包围矩形)。Op 表示操作的类型(也就是事件的类型或者说变化的类型)。文中只考虑两类操作: move\_in(对象移到一个新的位置)和 move\_out(对象离开它当前的位置),因此一个对象的创建可以用 move\_in 来表示,对象的删除可以用 move\_out 来表示,对象改变位置或形状可以用先 move\_out 再 move\_in 来表示。日志里的条目按照属性  $t$  的顺序排列。整个索引的数据结构称为 A, 是快照  $A_i$  的序列,  $A_i.t$  是第  $i$  个快照对应的时刻,  $A_i.R$  是第  $i$  个快照的 R-tree,  $A_i.L$  是  $A_i.t$  时刻与  $A_{i+1}.t$  时刻之间发生的事件日志。用参数  $d$  来定义两个连续的快照之间的最大距离(可以用磁盘块或发生的变化数量来衡量)。

### 1.2 操作

#### 1.2.1 时间片查询

时间片查询的第一步就是根据查询条件中指定的

时刻  $t$  找到相应的快照,也就是区间  $[0, t]$  之间最靠近时刻  $t$  的一个已存储的快照。用这个快照,根据文献 [8] 中定义的方法对相应的 R-tree 进行空间检索,从而得到一个对象的集合,然后根据相应的日志中的条目对这些对象进行更新(见图 2)。

```

1: TimeSliceQuery(Rectangle q, Time t)
2: {R will be a set that stores the objects of the answer}
3: Find the last entry i in A such that  $A_i.t \leq t$ 
4:  $R = \text{SearchRtree}(A_i.R, q)$  {All the objects in  $A_i.R$  that
intersect q at instant  $A_i.t$  are retrieved}
5: for each entry  $e \in A_i.L$  so that  $e.t \leq t$  do
6:   if e.Geometry Intersect(q) then
7:     if e.Op = Move in then
8:        $R = R \cup \{e.Oid\}$ 
9:     else
10:       $R = R - \{e.Oid\}$ 
11:   end if
12: end for
13: end for
14: return R

```

图 2 时间片查询算法

### 1.2.2 时间段查询

与时间片查询类似,找到离查询条件中的时刻  $t_1$  最近的快照,从那一点开始恢复从 R-tree 中查询到的对象,对所有时间小于等于  $t_2$  的日志中的条目进行处理(见图 3)。

```

1: IntervalQuery(Rectangle q, Time t1, Time t2)
2:  $G = \emptyset$  {G is a set that stores objects of the answer}
3: Search last entry i in A such that  $A_i.t \leq t_1$ 
4:  $R = \text{SearchRtree}(A_i.R, q)$  {All objects of  $A_i.R$  that inter-
sect q in the instant  $A_i.t$  are retrieved}
5:  $L = A_i.L$ 
6:  $k = i$ 
7: Update R with the changes in the log L which are found in
the interval  $[t, t_1]$  (just like a TimeSliceQuery)
8:  $G = R$ 
9: if all the entries in log L were processed then
10:  $k = k + 1$ 
11:  $L = A_k.L$ 
12: end if
13: Let  $t_s$  be the next instant after  $t_1$  stored in Log L
14: while  $t_s \leq t_2 \wedge$  entries remain to be processed in L do
15: Update R with the changes in Log L occurred in  $t_s$ 
16:  $G = G \cup R$ 
17: if all the entries in log L were processed then
18:  $k = k + 1$ 
19:  $L = A_k.L$ 
20: end if
21: Let  $t_s$  be the next instant stored in Log L

```

22: end while

23: return G

图 3 时间段查询算法

### 1.2.3 事件查询

用 SEST 索引可以进行事件查询。例如,给定某时刻  $t$  和一个区域  $q$ ,查询在  $t$  时刻多少对象进入  $q$  区域,多少对象离开  $q$  区域。这类查询的一个简单的实现是使用一个数组  $B$  来定位包含  $t$  时刻发生的事件的第一个日志块 Bid,数组  $B$  中的每一项占用空间很少,因此在一个磁盘块中可以存放很多项。1024 字节大小的磁盘块大约可以存放 128 项。值得注意的是这种实现方法不需要访问 R-tree。图 4 描述了这种算法。

```

1: EventQuery(Rectangle q, Time t, Array B)
2: Search the entry i in array B such that  $B_i.t = t$ 
3:  $L = B_i.Bid$ 
4: terminated = false
5:  $oi = 0$  {quantity of objects that entered q at time t}
6:  $os = 0$  {quantity of objects that left q at time t}
7: while not terminated do
8:   for each entry  $e \in L$  so that  $e.t \leq t$  do
9:     if  $e.t = t \wedge e.$ Geometry Intersect(q) then
10:      if e.Op = Move in then
11:         $oi = oi + 1$ 
12:      else
13:         $os = os + 1$ 
14:      end if
15:    end if
16:  end for
17: if  $e.t > t$  then
18:   terminated = true
19: else
20:    $L = \text{next log block}$ 
21: end if
22: end while
23: return (oi, os)

```

图 4 事件查询算法

### 1.2.4 更新结构

本算法的目的是在某时刻对象发生变化后更新索引。假设有一个在某时刻发生的所有变化的列表,和在这一时刻之前数据库中“存活”的所有对象的集合。本算法就是更新这些对象在新的时刻的状态,将它们插入数据库。然后验证参数  $d$  的阈值条件,也就是,由于这些变化的发生是否日志的大小已经达到了最大值的限制。如果达到最大值,就会创建一个新的快照,也就是创建一个新的 R-tree 和一个 A 的数据项。如果还没有达到最大值,这些变化只存储在日志里。图 5 描述了这种算法。

1: InsertChanges(Time  $t$ , Changes  $c$ , SnapShot  $F$ ) |  $t$  is the instant of time in which the changes happen,  $c$  is a list with the changes occurred at  $t$  and  $F$  corresponds to the "live" elements in the instant immediately preceding  $t$  |

2: Let  $i$  be the last entry of  $A$ .

3: Update  $F$  with the changes of  $c$

4: Insert the elements of  $c$  at the end of  $\log A_i \cdot L$

5: if the total changes in  $c$  plus the stored changes in the  $\log A_i \cdot L$  is greater than  $d$  then

6: {Create a new snapshot in SEST-Index}  $A_{i+1} \cdot t = t$

$A_{i+1} \cdot R = R$ -tree with the live objects in  $F$

$A_{i+1} \cdot L = \emptyset$

7: end if

图5 更新结构算法

## 2 结束语

提出了一种新的时空索引方法(SEST索引),它整合了快照和事件两种存储方式,试图在查询响应时间和索引空间占用率两方面采用较好的折衷。

虽然SEST索引在变化频率比较低时性能好,但是它有一个主要的缺点就是随着变化频率的增长它的规模也快速增长。在进一步的研究中,计划实现SEST索引的两个变体以克服这个缺点。

### 参考文献:

[1] Manolopoulos Y, Theodoridis Y, Tsotras J V. Advanced

Database Indexing[M]. Dordrecht, the Netherlands: Kluwer Academic Publishers, 1999.

[2] Pfoser D, Tryfona N. Requirements, definitions, and notations for spatiotemporal application environments[C] // In GIS '98: Proceedings of the 6th ACM International Symposium on Advances in Geographic Information Systems (1998). New York: ACM Press, 1998: 124-130.

[3] Worboys M. Event-oriented approaches to geographic phenomena[J]. International Journal of Geographical Information Science, 2005, 19(1): 1-28.

[4] Theodoridis Y, Sellis T, Papadopoulos A N, et al. Specifications for Efficient Indexing in Spatiotemporal Database[C] // In Proceedings of SSDBM'98. Capri, Italy: [s. n.], 1998: 123-132.

[5] Xu X, Han J, Lu W. RT-tree: An Improved R-tree Index Structure for Spatiotemporal Database[C] // Proceedings of the 4th International Symposium on Spatial Data Handling (SDH'90). Zurich, Switzerland: [s. n.], 1990.

[6] Theodoridis Y, Vazirgiannis M, Sellis T. Spatial Temporal Indexing for Large Multimedia Applications[C] // Multimedia Computing and Systems (ICMCS'96). Hiroshima, Japan: [s. n.], 1996.

[7] Nascimento M, Silva J. Towards Historical R-trees[M]. SAC: ACM, 1998.

[8] Guttman A. R-trees: A dynamic index structure for spatial searching[C] // In ACM SIGMOD Conference on Management of Data (1984). New York: ACM Press, 1984: 47-57.

(上接第55页)



(a) 指纹原图



(b) 文献[3]方法滤波结果



(c) Gabor 算法滤波结果



(d) 文中方法滤波结果

图6 指纹增强结果对比

### 参考文献:

[1] Kovacs-Vajna Z M, Rovatti R, Frazzoni M. Fingerprint ridge distance computation methodologies[J]. Pattern Recognition, 2000, 33(4): 69-80.

[2] Chikkerur S, Govindaraju V, Cartwright A N. Fingerprint Image Enhancement Using SIFT Analysis[C] // Proceedings of 3rd International Conference on Advances in Pattern Recognition and Image Analysis, Lecture Notes in Computer Science. [s. l.]: [s. n.], 2005: 20-29.

[3] Hong L, Jain A K, Pankanti S. Fingerprint enhancement[C] // In Proc. 1st IEEE WACV. Sarasota, FL: [s. n.], 1996: 202-207.

[4] Sherlock B G, Monro D M, Millard K. Fingerprint Enhancement by Directional Fourier Filtering[J]. IEEE Proceedings - Vision, Image and Signal Processing, 1994, 141(2): 87-94.

[5] Hong L, Wang Y F, Jain A K. Fingerprint Image Enhancement Algorithm and Performance Evaluation [J]. IEEE Trans. Pattern Anal. Machine Intell., 1998, 20(8): 777-789.