基于DR-tree的室内移动对象索引研究

甘早斌 袁永光 赵贻竹 鲁宏伟
(华中科技大学计算机科学与技术学院 武汉 430074)

摘 要 对于移动对象历史轨迹索引,现有的方案绝大多数都基于室外空间,难以直接应用于室内空间中;同时,未将对象本身作为一个独立的维度加以索引,无法提供高效的对象轨迹查询方式。对此,提出了一个室内环境下的移动对象索引结构DR-tree来对移动数据的位置、时间、对象三个维度进行索引,并将位置维与对象维解耦,将三维索引转换为两个二维索引,同时给出查询优化方案。实验结果表明,与现有的室内环境下的索引方案RTR-tree相比,该结构不仅能够提供高效的时空查询,而且还能提供高效的对象轨迹查询。

关键词 移动对象索引,室内空间,DR-tree,对象轨迹查询
中图法分类号 TP311 **文献标识码** A

Indoor Moving Objects Index Research Based on DR-tree

GAN Zao-bin YUAN Yong-guang ZHAO Yi-zhu LU Hong-wei
(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract For the index of historical trajectories of moving objects, most of the schemes are based on outdoor space, which are hard to be directly applied to indoor space. Moreover, the object itself is not indexed as an independent dimension and the efficiency of the queries based on objects is quite low. Thus, this paper proposed an index structure DR-tree (Dual R-tree) which can index three dimensions, such as the localization, the object and the time. This scheme can convert the three-dimension index into two two-dimension index by decoupling the location and object dimension, and provide query optimization method. The experimental results show that compared with RTR-tree, DR-tree, the scheme can not only support the efficient spatiotemporal query, but also provide the trajectory query based on objects.

Keywords Moving objects index, Indoor space, DR-tree, Object trajectory query

1 引言

伴随着移动计算、位置服务的兴起,位置信息的需求越来越多。GPS及各种无线通讯技术的发展,使得获取位置信息成为可能[1]。随着人们的室内活动越来越广泛和深入,在室内环境下,通过定位技术获取对象的移动位置信息,挖掘出移动对象行为、意图及行为方式[2,3],进而提供各种服务。移动对象的位置随时间不断变化,其移动数据通常具备时间和空间多维性,传统关系型数据库难以对其进行有效的组织。位置服务具有极大的经济潜力和应用价值,近些年来,国内外学者针对移动对象的管理进行了大量研究,提出了一系列针对移动对象的索引方案[4,12]。移动对象索引通常建立在空间索引的基础上,并增加了时间维。总的来说,这些方案绝大多数是基于室外自由空间和网络空间下的索引,通过对移动数据时空维度进行索引,提供各种基于时空的查询处理。

室外空间是无限的,而室内空间是有限的,通常由房间、走廊、楼梯、门等实体组成。在有限的室内空间中,实体间存在各种障碍和可达性限制,如房间之间通过墙阻隔,通过门相连通,其距离计算与室外空间下的欧氏距离有所不同。此外,由于难以建立统一的坐标模型,室内空间下通常采用符号定位[13]的方式获取移动对象的逻辑位置,而不是具体位置坐标信息,这使得室外空间下普遍采用移动对象位置坐标建立索引的方案无法适用。

另一方面,现有的索引结构考虑到移动数据的时间和空间属性,未将对象作为一个独立的维度加以索引,难以提供基于对象的高效查询方式。在实际应用中,需要针对某些具体对象的信息提供差异化、个性化的服务。如文献[12]中提出的RTR-tree,由于缺少对象本身的索引,在查询特定对象的移动轨迹时,效率很低,几乎需要遍历整个索引树。如果增加对象本身的索引,可以显著减少结点的访问数目,提高查询效率。

针对以上问题,本文提出了基于R-tree[14]的索引结构

到稿日期:2011-12-12 返修日期:2012-04-04 本文受国家自然科学基金(61173045),湖北省自然科学基金(2007ABA307),中央高校基本科研业务费(2010MS112)资助。

甘早斌(1968-),博士,副教授,CCF高级会员,主要研究方向为电子商务、信任计算、物联网,E-mail:zgan@mail.hust.edu.cn;袁永光(1984-),硕士,主要研究方向为RFID、移动对象索引;赵贻竹(1976-),女,博士,讲师,主要研究方向为物联网、社会网络等;鲁宏伟(1964-),男,教授,博士生导师,主要研究方向为物联网、社会网络等。

DR-tree(Dual R-tree)，其由两个二维的 R-tree 组成，用于索引移动对象历史轨迹。室内空间中采用符号化定位技术，将定位设备的部署位置作为移动对象的位置。DR-tree 在保留时空维度的基础上，增加对象维，同时将位置维与时间维解耦，将位置、时间、对象三维索引转换为位置、时间与对象、时间两个二维的索引。通过数据分析表明，该结构不仅可以支持高效的时空查询，还支持基于对象的高效查询。

2 相关研究工作

针对室外空间的移动对象索引，文献[4]提出了基于轨迹查询的 TB-tree。该算法对 R-tree 的结点插入算法进行了改进，在插入新的索引项时，插入位置的选择不是以索引项 MBR 面积增加最小为标准，而是以是否属于同一轨迹为标准，使得叶结点上只保存属于同一轨迹的数据，对轨迹的查询效率较高。但这也造成分属不同轨迹的空间上近邻的记录被分配在不同叶结点上，牺牲了 R-tree 的组织特性。

Chakka 等人提出了可扩展的索引方案 SETI[5]，指出空间维度的变化是有限的，而时间维度是不断扩展的，因此将空间维度分区，每个分区中采用 R-tree 对轨迹进行索引，从而对时空进行解耦，良好的分区方案是 SETI 的关键。Chakka 等人采用的方案是将属于同一个轨迹的数据尽量存储在同一个分区中，同时将跨越不同分区的轨迹片段进行分割，并分别存储，这可能导致查询时出现重复数据，因此查询时可通过精化步骤对重复的轨迹片段进行合并。

网络空间中的索引方案结合了网络的结构特性，从而提高了查询效率。文献[6]提出了在道路网络环境下的索引结构 FNR-tree，它是一个 2+1D 的混合索引结构，将 3D 索引转化为 2D 索引和一系列 1D 索引，2D 索引用于索引道路网络，每一个叶节点上有一个 1D 索引，1D 索引则按照时间维度进行索引。这种方法的优点在于有效利用了路网的结构，从而可以有效支持空间查询，但在轨迹查询及较大的时间窗口查询方面，其表现不佳，可能需要遍历多个 1D 索引。文献[7]采用了一种带有环形交叉口的元胞自动机模型模拟移动对象的将来轨迹，并用线性回归和圆弧曲线拟合分别得到对象在规则路段和交叉口的轨迹预测方程；根据移动对象的运动特性，采用了一种新的自适应单元(AU)作为索引结构的基本单位。

为了高效处理基于范围和对象轨迹的查询，Yang 与 Ma 等人提出了基于 MapReduce[8] 的分布式处理方案[9,10]，即结合现有的时空分区思想，采用空间分区和对象分区的方法，将数据存储在集群的不同结点上，从而提供高并发的查询处理。

针对基于室内空间的移动对象，Jensen 等人提出了室内空间移动对象数据管理。文献[11]将室内空间抽象为一个图模型，门、走廊、房间等实体被定义为图中的结点，实体间的联通关系定义为图中的边。该模型考虑了室内空间下的各种实体，对实体的划分比较详细。在文献[12]中，结合了室内 RFID 定位技术，采用了符号化定位方式，这与室外空间下采用二维坐标表示对象移动位置有所不同。在此基础上，Jensen等人提出了针对室内移动对象历史轨迹的索引结构 RTR-tree 与 TP2R-tree，对 R-tree 的结点组织策略进行了优化，以减少面积为 0 的 MBR；为了减少 MBR 的面积，提高查询效率，TP2R-tree 将代表一条轨迹记录的水平线段转化为一个点和一个时间参数，从而将线段的索引转化为对点的索引。

3 DR-tree 索引

3.1 室内定位与数据类型

在室内空间中，采用 RFID 定位技术获取对象移动信息。RFID 读写器部署在室内各个区域，移动对象通过佩戴有标识对象相关信息的标签，在移动过程中进入读写器的工作范围，即可被识别和定位。对象的位置可由室内逻辑区域表示，若对象处于房间 A 中，那么房间 A 就代表对象的移动位置；也可以采用读写器的部署位置表示对象的位置，在读写器识别范围较小或者一个室内区域中部署了多个读写器时，采用这种方式表达位置更为精确。本文采用读写器的位置作为对象的移动位置。

由于 RFID 读写器会频繁发送对象的位置信息，当对象长时间停留在某一位置时，将会产生大量的重复数据。为了压缩数据量，将对象处于某一位置的时间作为一条记录存储。每条轨迹记录的格式如下：

(id, L, O, ts, te)

式中，id 表示该条记录的编号，L 表示对象移动的位置，代表室内某一区域的标识，O 代表一个对象，ts，te 表示对象 O 进入及离开 L 的时间。

3.2 DR-tree 索引思想

DR-tree 通过对移动对象的位置、时间和对象本身 3 个维度进行索引，提供了基于时空和对象的高效查询。

DR-tree 是基于 R-tree 的索引结构。在 R-tree 中，随着维度的增加，索引的复杂性也随之上升，为了包围一条记录，需要占用的空间也会增加。如果直接对移动轨迹的位置、时间、对象进行三维索引，由于每一条索引记录代表了三维空间的一条线段，这将导致索引记录占据的空间过大，从而形成大量的“死空间”，增加了结点间重叠的几率，使得查询路径增多，查询性能下降。

另外，R-tree 以空间邻近性组织结点，空间上邻近的对象会被存放在相邻或同一结点上。位置维的空间邻近性可通过 RFID 读写器的顺序编号来表示，而对象维与位置维不同，其邻近性表示需根据具体情况而定。如果直接将对象维作为一个空间维度与位置维并列进行索引，会导致移动记录在索引树中的分布比较分散，不利于对象轨迹的保持，同时也降低了索引效率。因此，需要将空间维与位置维分开索引，将三维空间(L, O, t)下的移动记录映射到两个二维(L, t)、(O, t)的空间中，并分别索引。

所以，DR-tree 是一个双索引树结构，维护了两个索引树 LT-tree 和 OT-tree。LT-tree 用于索引移动轨迹的位置和时间信息，支持面向时空的查询方式；OT-tree 则对对象和时间进行索引，支持面向对象的查询。

3.3 DR-tree 的结构

DR-tree 由 LT-tree 和 OT-tree 组成。在 LT-tree 结构中，中间结点的存储结构具有如下形式：

$(ID, index_1, index_2, index_3, \cdots, index_n)$

式中，ID 为结点的编号，$index_i$ 为该结点中的第 i 个索引项，每个索引项的表示形式为：

$(MBR, child\text{-}pointer)$

式中，$child\text{-}pointer$ 为指向该索引项所包含的子结点的指针。MBR 为该索引项所包含的所有子结点的最小外接矩形，其存储形式为 $(L_{min}, L_{max}, t_{min}, t_{max})$，$L_{min}$、$L_{max}$ 代表该索引项所包含的子结点的移动位置范围，t_{min}、t_{max} 则表示该索引项包含的子结点处于位置区域 $[L_{min}, L_{max}]$ 的时间范围。MBR 等价于由位置和时间组成的二维坐标轴上的矩形。叶结点具有如下形式：

$(ID, leaf\text{-}entry_1, leaf\text{-}entry_2, \cdots, leaf\text{-}entry_n)$

式中，ID 为该结点的编号；$leaf\text{-}entry_i$ 为该结点第 i 个索引项，其存储形式为：

(ID, MBR)

式中，MBR 存储形式为 $(L_{min}, L_{min}, t_s, t_e)$，空间上矩形退化为一条线段；$ID$ 则是指向存储在数据库中的对象实际移动记录的 ID。所有移动记录的索引项均存放在叶结点中。

OT-tree 的索引结构与 LT-tree 类似，不同之处在于 MBR 的组成。中间结点中索引项的 MBR 为 $(O_{min}, O_{max}, t_{min}, t_{max})$，等价于对象 O 与时间 t 组成的二维空间上的一个矩形；叶结点中索引项的 MBR 为 $(O_{min}, O_{min}, t_{min}, t_{max})$。

一个简单的 LT-tree 结构如图 1、图 2 所示。

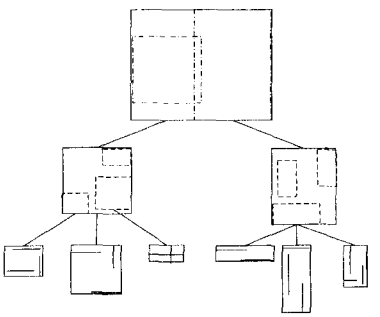


图 1　LT-tree 逻辑图

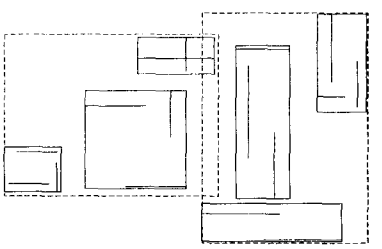


图 2　LT-tree 的 MBR 分布图

4　室内环境下 DR-tree 的查询处理

4.1　室内查询类型

移动对象历史轨迹数据的查询可分为两个基本的类型。

(1) 时空查询方式，如点查询、范围查询等。点查询可分为空间点查询和时间点查询。对于一个典型的空间点查询方式，如“查询房间 A 内的移动对象记录”，给出一个空间点 E_s，$Q(E_s)$ 返回位于该查询点的所有移动记录。

$Q(E_s) \rightarrow \{Records\}$

范围查询方式包括空间范围查询、时间范围查询及时空范围查询。空间范围查询给出一个空间范围，查询该范围内的对象移动记录，如“查询 A、B、C 区的对象移动记录”；点查询是范围查询的一个特例，其表达方式与范围查询是一致的。时空范围查询是空间维与时间维联合进行的查询，给出一个空间范围 E_s 和一个时间范围 E_t，$Q(E_s, E_t)$ 返回在 E_t 范围内位于 Es 的移动对象记录。

$Q(E_s, E_t) \rightarrow \{Records\}$

(2) 对象轨迹查询方式，如“查询移动对象 a、b、c 等在时间 t 内的移动轨迹”。给出一个对象集 E_o 和一个时间范围 E_t，$Q(E_o, E_t)$ 返回该对象集的移动轨迹记录。

$Q(E_o, E_t) \rightarrow \{Trajectory\ Records\}$

DR-tree 对移动记录的位置、时间、对象进行了索引，因此，DR-tree 除支持典型的查询方式，如空间查询、时间查询、时空范围查询外，还支持基于对象的轨迹查询。针对不同的查询方式，DR-tree 在 LT-tree 与 OT-tree 中进行选择，以达到更高的查询效率。

4.2　查询优化

与室外空间下的位置信息采用二维坐标表示不同，室内移动记录的位置与对象采用一维的数值表示。由于对象之间不具备直接邻近性，并且对象的编号带有一定的随意性，在对象轨迹查询中，给出的查询窗口中包含的对象集通常是一系列不连续的数值。例如，对于一个查询窗口 $S=\{O, T\}$，$T=\{t_1, t_2\}$，$O=\{1,2,3,4,8,9,10,16,17\}$，即查询对象集 O 所含的对象在时间区间 $[t_1, t_2]$ 内的移动记录，显然，对象集是不连续的，根据 R-tree 的查询策略，直接组合对象集和时间集形成的查询窗口如图 3 所示。由于对象 5、6、7、11、12、13、14、15 并不在查询需求中，因此按照图 3 产生的窗口 S 进行查询，将产生这些对象的查询路径，从而增加了结点的访问数目，降低了查询效率。

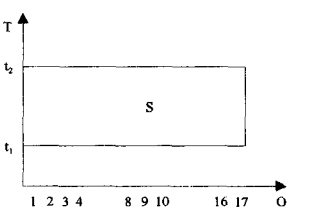


图 3　直接组合产生的查询窗口 S

因此，为了使查询更加精化，需对初始的查询窗口进行分割处理，把不连续的对象集分割为若干包含连续对象的集合，分割后将形成的对象集与时间集进行组合，组成一个包含多个查询窗口的窗口集。如将对象集 O 分割为 $O'=\{O_1, O_2, O_3\}$，其中 $O_1=\{1,2,3,4\}$、$O_2=\{8,9,10\}$、$O_3=\{16,17\}$。分割后将对象集和时间集进行组合，产生查询窗口集 $\{S_1, S_2, S_3\}$，如图 4 所示。显然，分割后的查询窗口更小。

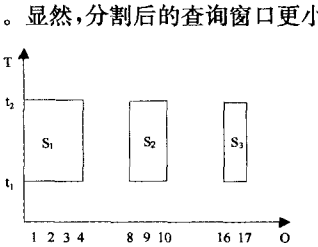


图 4　对象集合划分产生的查询窗口

在实际的查询中，空间查询、时空范围查询同样存在初始查询窗口中位置集不连续的问题，为了统一这几种查询的匹配策略，查询前统一将初始查询窗口进行分割处理。算法 1 给出了对象集的窗口分割算法伪代码描述。

算法 1 窗口分割算法 WindowSplit(S)

```
1. ObjectSet←S. os
2. if ObjectSet≠∅ then
3.     create a location set Os and add the first element of ObjectSet
         into Os, then delete this element
4.     while ObjectSet ≠∅ do
5.       choose an element E in ObjectSet that has proximity relation
           with elements of Os
6.       if E exists then
7.           add E into Os and delete E in ObjectSet
8.       else
9.           add Os into Oset then empty the Os
10.          choose an element E in ObjectSet and add E into Os
               and delete E in ObjectSet
11.      end if
12.    end while
13.    add Os into Oset
14. end if
15. rebuild search window S with Oset and time set
16. return S
```

算法 1 第 1 行获取原始查询窗口中存在的对象集，第 2—15 行描述对象集的分割过程。对象之间不存在显式的距离上的邻近关系，这里，根据对象的编号定义对象间的邻近性，两个对象编号相邻，就表示具备邻近关系。最后，算法在第 16 行对分割后的集合进行重组，形成多个查询窗口。

分割后的查询窗口变成了窗口集，对原始的查询匹配策略进行了修改。在遍历任一结点时，对于该结点中的每一条索引项，比较其 *MBR* 与窗口集中任一窗口，如果存在重叠，即认为匹配成功。算法 2 给出了对象轨迹查询的具体流程。

算法 2 ObjectTrajectoryQuery(Root,S)

```
1. N←Root
2. if N is not leaf node in OT-tree then
3.     for each entry E in N
4.       for each search window S′ in S
5.         if E.MBR overlaps S′ then
6.           ObjectTrajectoryQuery (E.p,S )
7.         end if
8.       end for
9.     end for
10. else
11.   for each entry M in N
12.     for each search window S′ in S
13.       if M overlaps S′ then
14.         add data that pointered by M to results
15.       end if
16.     end for
17.   end for
18. end for
```

算法 2 第 2 行从 OT-tree 的根结点开始遍历，检查遍历的结点是否为叶结点。第 3—9 行描述中间结点的遍历过程。对于一个中间结点，检查其存储的每一个索引项的 *MBR* 是否与查询窗口集 *S* 中的任一窗口相匹配，匹配的策略是检查索引项的 *MBR* 表示的窗口与查询窗口之间是否存在重叠。如果重叠，则表示匹配成功，然后递归地遍历索引项指向的子结点，直至遍历到叶结点。第 11—17 行是对遍历到的叶结点的处理过程。检查叶结点中的每一个索引项，如果该索引项指向的实际数据是符合查询要求的数据，则将该数据加入到结果集中。

5 仿真实验及结果分析

为了验证本文提出的双索引结构的有效性，采用 Java 语言对 DR-tree 和 RTR-tree 进行了实现，结合一定的数据集，对 DR-tree 与已有的室内环境下的索引方案 RTR-tree 的查询效率进行了比较。索引结构的实现基于 Spatial Index Library for JAVA[15]，实验环境为：Windows 操作系统，Pentium 处理器 3.00GHz，2GB 内存。

5.1 实验数据

实验数据来源于澳门某办公楼的人员移动数据。该办公楼在各个楼层、办公室、会议室、相距很近的不同门店中安装了 200 多个 RFID 读写器，进出办公楼以及不同门店的人员佩戴有相关标签，在移动过程中其位置信息被收集起来。本实验收集了 1000 名人员共计 200k 条移动记录。在移动数据集生成的 DR-tree 和 RTR-tree 中，统一设置每个结点最多存放 200 个索引项，最少存放 100 个索引项。表 1 为数据集。

表 1 数据集

数据类型	数据
数据记录数	200k 条移动记录
移动对象编号	1001,1002,1003,…,2100
位置编号	1,2,3,4,5,…,200

5.2 查询性能分析

5.2.1 性能评估方式

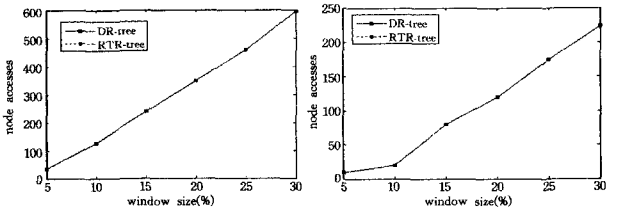
由于索引树通常存储在外存磁盘中，结点的遍历需要访问磁盘中的磁盘页，而 CPU 对外存的访问时间远大于对内存的访问。因此，查询时结点访问越少，查询效率就越高。另外，DR-tree 在查询时虽然增加了窗口分割的操作，但窗口分割算法耗费的时间远小于查询算法所耗费的时间，即使在给出的查询集合连续的情况下，执行窗口分割操作对整个查询时间而言，其影响也可以忽略不计。所以，这里采用结点访问数目作为查询性能评估方式，而不考虑窗口分割算法对查询性能的影响。

5.2.2 时空查询性能比较

查询窗口采用百分比的形式表示，如 10%表示给出的查询窗口在位置维和时间维上分别占索引树相应维度的百分比。

图 5 给出了查询窗口中相关维度连续的情况下，DR-tree 与 RTR-tree 的查询效率比较。从图中可以看出，两者在空间

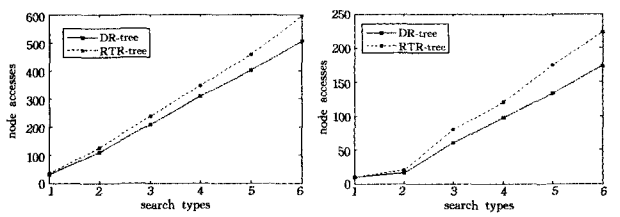
查询、时空范围查询效率方面几乎没有差别。这是由于在DR-tree中，负责空间查询、时空范围查询的是LT-tree，且LT-tree采用了与RTR-tree相同的索引结构，因此在插入相同移动记录的情况下，索引树的结构是一致的。由于给出的查询窗口相关维度是连续的，因此查询效率上也就没有差别。



(a) 空间查询　　(b) 时空范围查询

图5　相关维度连续的情况下两者在范围查询方面的比较

图6给出了查询窗口不连续的情况下两者的查询效率比较。由于DR-tree执行了窗口分割技术，对原始查询窗口进行了分割处理，分割后的查询窗口与RTR-tree采用的原始查询窗口相比，其面积更小，在给出的6种相同查询条件下，DR-tree的查询效率较RTR-tree有了一定的提高。



(a) 空间查询　　(b) 时空范围查询

图6　窗口分裂后两者在范围查询方面的比较

5.2.3　对象轨迹查询性能比较

对象轨迹查询中，两者在查询效率上的差距比较明显(见图7)。在DR-tree中，OT-tree负责基于对象的轨迹查询。由于OT-tree对移动对象进行了索引，查询过程中可直接对对象进行匹配，因此大大缩小了查询范围；而在RTR-tree中，并不存在对象的索引，在满足时间维的情况下，几乎需要遍历全部结点才能保证获取所需的记录。

图8给出了使用原始窗口查询和使用分裂后的窗口集对对象轨迹查询的影响。初始查询中的对象集通常是不连续的，执行窗口分割后，查询更精化，效率更高。

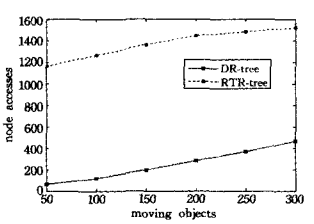


图7　两者在对象轨迹查询方面的比较

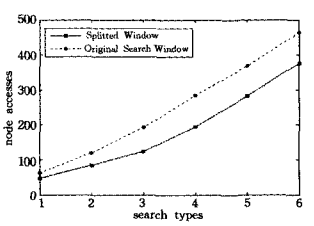


图8　窗口分割对对象轨迹查询的影响

结束语　本文提出了基于对象的轨迹查询方式，指出现有的索引结构缺乏对对象本身的索引，使得索引结构无法支持特定对象的高效查询。为此，提出了基于R-tree的双索引结构DR-tree，并对查询算法进行了优化。实验结果表明，与RTR-tree相比，在不影响时空查询效率的情况下，DR-tree提供了高效的对象轨迹查询方式。

由于OT-tree对轨迹的保存并不彻底，同一结点或相邻结点中不能保证存储的一定是同一轨迹的索引记录。下一步将研究如何把属于同一轨迹的记录链接起来，从而进一步提高对象轨迹查询的效率。

参 考 文 献

[1] 周傲英，杨彬，金澈清，等. 基于位置的服务：架构与进展[J]. 计算机学报，2011，34(7)：1155-1171

[2] 郑宇，谢幸. 基于用户轨迹挖掘的智能位置服务[J]. 中国计算机学会通讯，2011，6(6)：8-9

[3] 赵亮，景宁. 移动对象数据库关键技术[J]. 中国计算机学会通讯，2011，7(5)：52-60

[4] Pfoser D, Jensen C S, Theodoridis Y. Novel Approaches in Query Processing for Moving Object Trajectories[C]// Proceedings of the Intl. Conf. on Very Large Data Bases(VLDB). 2000：395-406

[5] Chakka V P, Everspaugh A, Patel J M. Indexing Large Trajectory Data Sets with SETI[C]// First Biennial Conf. on Innovative Data Systems Research (CIDR). 2003：164-175

[6] Frentzos E. Indexing objects moving on fixed networks[C] // Proceedings of the 8th Intl. Symp. On Spatial and Temporal Databases (SSTD). 2003：289-305

[7] 宋广军，郝忠孝，王丽杰. 一种基于受限网络的移动对象索引[J]. 计算机科学，2009，36(12)：138-141

[8] Dean J, Ghemawat S. Mapreduce: Simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1)：107-113

[9] Yang B, Ma Q, Qian W, et al. TRUSTER: Trajectory data processing on clusters[C] // Proceedings of the 14th International Conference on Database Systems for Advanced Applications. 2009：768-771

[10] Ma Q, Yang B, Qian W, et al. Query Processing of Massive Trajectory Data based on MapReduce[C]// Proceedings of the first international workshop on Cloud data management. 2009：9-16

[11] Christian S J, Lu H, Yang B. Indoor-A New Data Management Frontier[J]. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2010, 33(2)：12-17

[12] Jensen C, Lu H, Yang B. Indexing the trajectories of moving objects in symbolic indoor space[C]// International Symposium on Spatial and Temporal Databases(SSTD). 2009：208-227

[13] Hightower J, Borriello G. Location systems for ubiquitous computing [J]. IEEE Computer, 2001, 34(8)：57-66

[14] Guttman A. R-Trees: A Dynamic Index Structure for Spatial Searching[C]// Proceedings of the ACM International Conference on Management of Data (SIGMOD). 1984：47-57

[15] Spatial Index Library [EB/OL]. http://sourceforge.net/projects/jsi