计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2015.20131230
52(9):2002-2013,2015

支持多种查询的室内移动对象索引

贲婷婷　秦小麟　许建秋
(南京航空航天大学计算机科学与技术学院　南京　210016)
(ben_tingting@126.com)

Index of Indoor Moving Objects for Multiple Queries

Ben Tingting, Qin Xiaolin, and Xu Jianqiu
(*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing* 210016)

Abstract　Moving object index is widely used in location-based services. Since people spend large parts of their lives in indoor spaces (e.g. hospitals, shopping malls, subway systems, etc.), effective management of indoor mobile data becomes very important. Existing indoor moving object indices focus on historical data queries, and only one type of queries is supported. In this paper, we propose a novel index, called MQII (multiple queries indoor index), which supports not only history queries and present queries, but also object queries and range queries. MQII is based on graph-based model, and can index two aspects with the object list and bucket list structure, such as the object and spatial-temporal scales. In order to improve the query performance, we present a RFID (radio frequency identification) data preprocessing method to reduce the size of the input data sets for MQII. Furthermore, effective update and query algorithms are developed. Experimental results show that compared with existing indoor moving object indices, the data preprocessing can reduce the amount of data. In addition, the index we proposed not only supports history queries and present queries, but also provides efficient object location queries, trajectory queries and range queries. This method can be used in various indoor spaces such as office buildings, hospitals and hotels.

Key words　moving object index; indoor space; range query; trajectory query; indoor graph-based model

摘　要　随着室内定位技术的广泛应用,室内位置服务快速发展.移动对象索引技术作为支撑位置服务的核心技术,大多数都基于室外环境,难以直接应用于室内空间.现有的室内移动对象索引,仅关注对移动对象历史数据的查询,且支持的查询类型单一.为此,提出MQII(multiple queries indoor index)索引结构,对移动对象历史和当前位置信息进行索引,能够同时支持对象位置查询、轨迹查询以及时空范围查询.索引采用对象链表和桶链表结构,实现从对象和时空范围2个方面对移动对象数据的管理;提出针对该索引结构的有效更新、查询算法;实验结果表明,与现有室内移动对象索引相比,索引不仅能够支持历史查询和当前查询,还能够同时高效支持对象位置查询、轨迹查询和范围查询.该方法可应用于办公楼、医院等多种室内空间.

关键词　移动对象索引;室内环境;范围查询;轨迹查询;室内图模型

中图法分类号　TP311.13

收稿日期:2013-09-05;**修回日期**:2014-11-20

基金项目:国家自然科学基金项目(61373015);国家自然科学基金青年基金项目(61300052,41301407);国家教育部高等学校博士学科点专项科研基金项目(20103218110017);江苏高校优势学科建设工程项目(PAPD);中央高校基本科研业务费专项资金项目(NP2013307)

通信作者:秦小麟(qinxcs@nuaa.edu.cn)

近年来,随着移动计算、无线通信、定位技术以及物联网技术的发展与普及,移动对象的数据管理作为位置服务[1]的核心技术,得到了广泛的重视与研究.研究表明,人们一天中大多数的时间要在室内度过,他们在室内工作、休息、购物、娱乐,室内环境变得越来越复杂[2].通过定位技术获取室内对象的位置信息,挖掘移动对象行为、意图及行为方式,进而提供位置服务.有效的室内移动对象管理技术能够给人们的生活带来极大便利,应用广泛.比如:在机场,可以快速找到还没有登机的旅客,提醒旅客抓紧时间登机,降低飞机的延误率;通过查询某人某段时间内在室内的运动轨迹,寻找遗失物品的可能所在地;查询某人在办公室里一共待过的时间总和,进行考勤等.

索引是提高移动对象数据库查询处理性能的关键技术,时空索引的研究对于对象跟踪、位置服务起至关重要的作用.在过去的数十年里,国内外学者针对移动对象的管理进行了很多研究,提出了许多移动对象索引方法,但绝大多数的研究主要集中于对室外移动对象的管理,包括欧氏空间的移动对象以及基于路网约束的移动对象[3-5].由于室内环境的特殊性,室外移动对象索引技术难以直接应用到室内空间中[6-7],主要因为:

1) 模型不同.由于室内空间独特的拓扑结构,通常是由房间、走廊、楼梯、门等多种实体组成,使得传统欧氏空间模型和空间网络模型不适用于室内移动对象的表示.室外空间的网络模型可以用于抽象表达道路的拓扑结构,此类模型不能完全适用于室内空间.与自由运动对象和道路网约束的移动对象相比,由于室内空间通常表示为一组具有语义的实体,例如:房间、走廊、门等,人和物在室内空间移动时受到更多的限制,2 个室内分区之间只能通过门进行连通.因此,为了有效地管理室内移动对象,需要使用新的模型去对室内空间建模,以适应室内空间的轨迹表达.

2) 定位技术不同.室内定位技术和常使用的室外定位技术,无论是从原理还是本质上,都存在很大的不同,它通常不能连续报告室内移动对象的位置和速度.现有的 GPS(global positioning system)定位技术,可以对室外移动对象进行精确定位,获得其连续的位置信息.而在 RFID(radio frequency identification)定位系统中,每个 RFID 阅读器都具有一定的覆盖区域,阅读器每隔固定的周期才会检测出现在其覆盖区域内的 RFID 标签,室内移动对象的位置信息是离散的.当带有 RFID 标签的移动对象进入到某个 RFID 阅读器的激活范围时,可以被阅读器检测到,并通过该阅读器的部署位置和范围来估计该移动对象的位置,这种估算在很大程度上跟 RFID 阅读器的性能有关,给移动对象的位置信息带来了很大的不确定性.

由于上述问题的普遍存在,给室内移动对象索引技术的研究带来了很大的挑战,现有的室内移动对象索引技术很少,并且支持的查询类型单一.Jensen 等人[8]提出的 RTR-tree(reader-time R-tree)和 TP^{2}R-tree(time parameter point R-tree)能够高效支持范围查询,但其对象查询效率低下;2012 年,Shin 等人[9]提出的 ACII(adaptive cell-based index for indoor moving objects)索引关注解决对象位置查询、轨迹查询,而范围查询受到时间限制,性能很低.然而,在实际应用中,往往同时需要处理多种类型的查询,不仅需要能够提供时空范围查询,而且针对某个特定的对象,需要能够提供差异化、个性化的服务,时空范围查询和对象查询作为 2 类最基本的查询类型,缺一不可.

本文提出了一种新的室内移动对象索引方法,即 MQII(multiple queries indoor index)索引技术,关注移动对象的时空信息,从对象和时空范围 2 个维度出发,有效管理移动对象历史和当前位置信息的同时解决现有室内移动对象索引技术查询单一化的问题,能够同时高效支持对历史和当前数据信息的对象查询和范围查询.此外,提出针对 MQII 索引结构的有效更新查询算法;通过实验将其与现有室内移动对象索引进行对比,分析索引结构有效性和查询性能.

1 相关研究工作

1.1 室内移动对象索引技术

现存绝大多数移动对象索引技术都是针对室外欧氏或受限空间,根据处理数据来源的时间大致可以分为 3 类[10]:对历史数据的索引[11-12],如 HR-tree(historical R-tree),MV3R-tree(multi-version 3D R-tree)等;对移动对象当前位置的索引[13],如 LUR-tree(lazy update R-tree)等;对移动对象未来轨迹的预测索引[14-15],如 TPR-tree(time-parameterized R-tree),TPR*-tree(the improved time-parameterized R-tree)等.随着室内环境变得越来越复杂,人们逐渐把目光转向室内空间,室内移动对象索引日显重要.

2009 年,Jensen 等人[8]最早提出的室内移动对象索引 RTR-tree 和 TP²R-tree,采用符号化定位方式[3],通过对 R-tree 的节点组织策略进行优化,降低 MBR(minimum bounding rectangle)之间的重叠提高索引效率,其空间范围用阅读器的集合标识,要求阅读器基于空间邻近部署. 索引可以有效处理范围查询,但是却没有提供基于对象的索引,当要处理对象位置查询和轨迹查询时,在满足查询时间要求的条件下,几乎需要遍历索引中全部的节点才能获取所需的记录,效率低.

DR-tree(dual R-tree)[16]是对 RTR-tree 的改进,维护了 2 个索引树 LT-tree 和 OT-tree. 由于在 DR-tree 中,负责时空范围查询的是 LT-tree,其采用了与 RTR-tree 相同的索引结构,因此两者范围查询效率没有差别. DR-tree 在保留时空维度的基础上增加对象维,OT-tree 负责基于对象的轨迹索引,但由于 OT-tree 对轨迹的保存并不彻底,同一节点或相邻节点中不能保证存储的是同一轨迹的索引记录,查询效率一般且索引结构重复信息多.

Alamri 等人[17-19]提出的索引树结构,通过邻接表管理单元到其他单元之间的距离,首次提出用"连通"的思想表示单元间的关系. 通过一个扩展算法表示单元之间的重叠关系,基于对象位于"相似"单元中的性质创建 MBR,从而创建室内索引树. 索引结构具有良好的更新性能,能够有效处理室内空间查询,但其他基于邻接方法的查询还有待进一步研究和解决,如对象轨迹查询、时空查询等.

ACII 索引,基于象征空间模型(symbolic space model)[7],包含 2 个主要结构:MC(memory cell)和 MEMO(memory),其中 MC 是一个基于单元的可扩展结构,存放最新在单元中的移动对象信息,能够支持移动对象当前位置查询;MEMO 基于对象存储信息,依次存储移动对象不同时间段的所有信息,有效支持对象轨迹查询. 但当要处理时空范围查询时,由于 ACII 索引在存储信息的时候都是基于对象存储,没有基于单元的索引,查询某一单元存在对象时,几乎要遍历索引结构中的所有信息,才能获取满足查询条件的记录,效率极低.

1.2 移动对象查询种类

按照查询已知内容的不同,将现有室内移动查询分为 2 类,分别为对象查询(object queries)和范围查询(range queries). 如表 1 所示,对象查询包括对象位置查询和对象轨迹查询,范围查询包括时间戳范围查询和时间片范围查询,对应的查询实例如下:

1) 对象位置查询. 查询对象 o_1 在时刻 t 的位置.

2) 对象轨迹查询. 查询对象 $\{o_1,o_2,o_3\}$ 在时间区间 $[t_1,t_2]$ 内的室内运动轨迹.

3) 时间戳范围查询. 查询房间 $\{c_1,c_2\}$ 在时刻 t 的移动对象.

4) 时间片范围查询. 查询房间 c_1 在时间区间 $[t_1,t_2]$ 内的移动对象.

Table 1 Query Types for Indoor Moving Objects

表 1 室内移动对象查询类型

Classify	Query Type	Query Format
Object Queries	Position Queries	$QT([objectID_m,objectID_n],t)$
	Trajectory Queries	$QT([objectID_m,objectID_n],[t_1,t_2])$
Range Queries	Timestamp Queries	$QT([cellID_m,cellID_n],t)$
	Timeslice Queries	$QT([cellID_m,cellID_n],[t_1,t_2])$

现有的室内移动索引技术关注解决某一类型的查询,如:ACII 索引着重于解决对象轨迹查询,RTR-tree 高效支持范围查询,而 MQII 索引可以同时有效支持对象查询和范围查询.

2 模型和数据处理

2.1 室内图模型

图模型(graph-based model)[20]是对室内空间的一种常用建模方法,它为室内移动对象的管理提供基础保障. 图 1(a)是在一个简单的室内平面图上部署 RFID 阅读器得到的设备部署图.

通过室内 RFID 的部署,可以获取移动对象的离散信息,当定位设备检测到进入其覆盖范围的对象时产生数据,有效的 RFID 格式为($readerID$,$tagID$,t),表示阅读器 $readerID$ 在时刻 t 检测到带有标签 $tagID$ 的对象进入了它的激活范围. 由于 RFID 阅读器只能判断在该阅读器激活范围里对象是否存在,不能判断对象的运动方向,所以通过 RFID 阅读器成对部署判断对象在单元[20]间的进出关系,如图 1(a)中的设备 3 和 3′.

图 1(b)所示的部署图有效集成了所部署的定位设备的相关信息,图 1(b)中的顶点为被划分设备划分出来的单元,连接 2 个节点的有向边表示移动对象可以在不经过第 3 个单元的情况下,从头节点表示的单元移动到尾节点表示的单元,边上的数值代表一对划分设备的先后读数. 如从办公室 2 到走廊,对象先被设备 3 检测,再被 3′检测到,在图 1(b)中用(3,3′)进行说明.

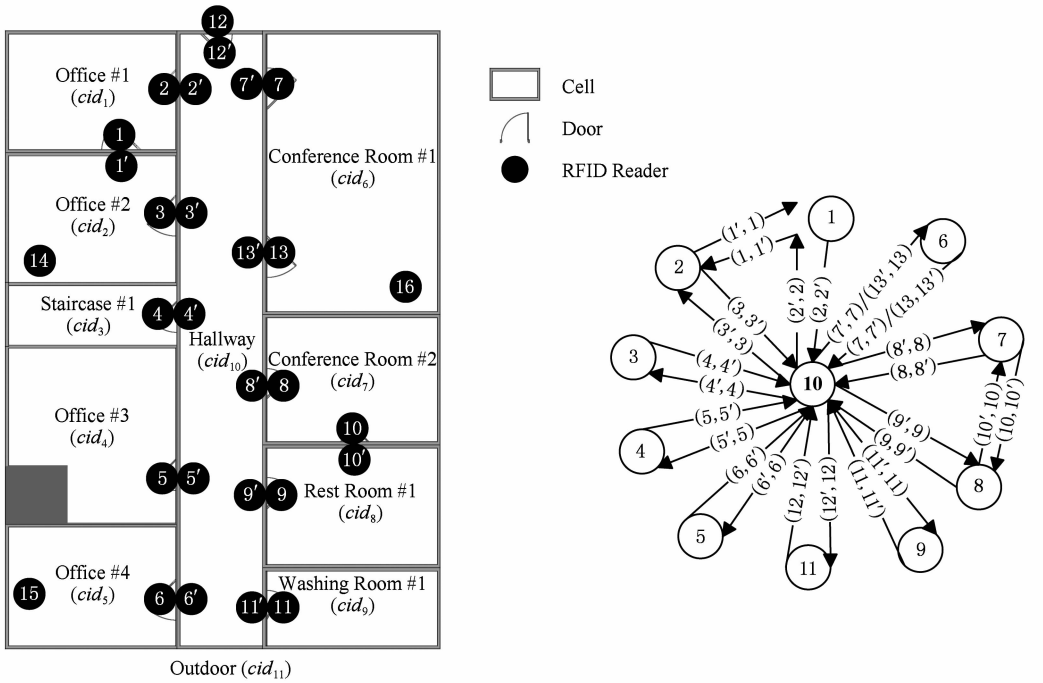


Fig. 1 RFID readers deployment graph.

图 1 RFID 设备部署图

基于部署图,通过对大量 RFID 原始数据进行分析处理,可以简化初始数据,获取室内移动对象的有效信息,数据预处理将在 2.2 节中详细说明. 基于预处理后的数据,通过有效的索引结构进行管理,可以实现对移动对象运动信息的多种查询.

2.2 RFID 数据预处理

为了提高索引的效率,在创建索引之前,首先对大量原始 RFID 数据进行预处理. 数据预处理模块基于图模型,忽略移动对象在某个具体单元内部的移动,将单元定义为最小移动单位,只有当对象移动到另一个单元的时候,才更新对象的位置信息.

对象 o 携带标签 tag_1,当 o 进入到阅读器激活范围时产生读数,使用文献[20]中的预处理模块,将 RFID 原始读数进行初步处理,得到离线轨迹如表 2 所示.

Table 2 Off-Line Trajecotry of Object o

表 2 移动对象 o 的离线轨迹

ID	$tagID$	$readerID$	t_s	t_e
1	tag_1	$reader_{12}$	t_1	t_2
2	tag_1	$reader_{12'}$	t_3	t_4
3	tag_1	$reader_{2'}$	t_7	t_8
4	tag_1	$reader_2$	t_9	t_{10}
5	tag_1	$reader_1$	t_{21}	t_{22}
6	tag_1	$reader_{1'}$	t_{23}	t_{24}
7	tag_1	$reader_{14}$	t_{30}	t_{32}
8	tag_1	$reader_3$	t_{40}	t_{41}
9	tag_1	$reader_{3'}$	t_{42}	t_{43}
10	tag_1	$reader_{4'}$	t_{46}	t_{46}
11	tag_1	$reader_{5'}$	t_{50}	t_{51}
12	tag_1	$reader_{6'}$	t_{57}	t_{58}
13	tag_1	$reader_6$	t_{59}	

第 1 步. 如果 2 个连续记录的时间属性相差少于或者等于一个采样周期,并且是由一对有向划分设备产生,根据部署图获取有向边对应的进出单元信息,从而生成的记录格式为($objectID$, $freaderID$, $sreaderID$, $OutCell$, $InCell$, t_s, t_e). 其中,$objectID$ 表示对象标识;$freaderID$, $sreaderID$ 表示一对阅读器;t_s 和 t_e 表示一对划分设备最新检测到对象的时刻和最后检测到对象的时刻. 将除此之外的数据删除,即表 2 中 ID 为 7,10,11 的记录,可以删除大量无用信息.

第 2 步. 由于只关注对象从一个单元移动到另一个单元的状态变化,根据式(1),将进出某个单元的时刻抽象为一个时间戳. 进一步简化,忽略 RFID 信息,得到最终的数据形式,如表 3 所示,此时得到的有效信息仅包含对象、单元和时间信息.

$$t = (\lfloor t_s + t_e \rfloor)/2. \quad (1)$$

Table 3 Trajectory after Step1 and Step2

表 3 2 步简化后的数据

objectID	*OutCell*	*InCell*	*t*
oid_1	cid_{11}	cid_{10}	t_2
oid_1	cid_{10}	cid_1	t_8
oid_1	cid_1	cid_2	t_{22}
oid_1	cid_2	cid_{10}	t_{41}
oid_1	cid_{10}	cid_5	t_{58}

由数据预处理,得到 MQII 索引处理的基本数据结构如下:

$$(objectID, OutCell, InCell, t),$$

其中,*objectID* 表示对象标识,*OutCell* 和 *InCell* 分别表示对象在时刻 t 离开和进入的单元标识. 例如:$(oid_1, cid_{10}, cid_1, t_8)$ 表示在时刻 t_8 对象 oid_1 从 cid_{10} 单元移动到了 cid_1 单元. 从表 3 可以清晰地知道对象 o 在这段时间内的运动轨迹,结合图 1(a),具体语义信息为对象 o 从室外进入室内后,经过走廊进入办公室 1,停留一段时间后,进入办公室 2,接着返回走廊,并由走廊进入办公室 4. 从表 2 和表 3 的对比可以看出,数据预处理能够删除很多冗余数据,降低数据量.

3 MQII 索引

3.1 MQII 索引结构

MQII 索引结构由单元 Hash 表、桶链表和对象链表 3 部分组成,结构如图 2 所示:

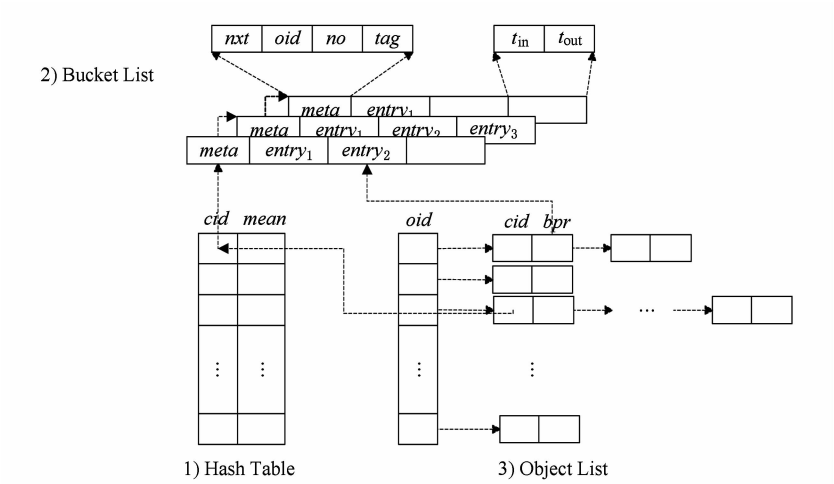


Fig. 2 MQII structure.

图 2 MQII 索引结构

1) 单元 Hash 表(Hash table, *HT*). 提供室内单元索引,将室内单元信息组织起来,包含(*cid*, *mean*)的键值对. 其中,*cid* 是单元的唯一标识;*mean* 代表单元的具体语义信息,每个单元对应一个桶链表,用来存储该单元中的移动对象信息.

2) 桶链表(bucket list, *BL*). 用来记录单元中存在过的对象的具体信息. 对象的数据信息存放于桶链表中,单元 Hash 表中仅仅存放指向桶链表第一个桶的指针. 每一个对象都有自己单独的桶,用于记录该对象在这个单元中存在的时间段,每次更新信息时,插入新项到该对象非满桶的最后面,如果桶已满则在该桶的前面新建一个桶,保存该对象的信息,这样存储使得同一个对象在某个单元中的信息连续存放,桶链表中存储的对象实际信息具备相同对象信息连续存放以及同一单元同一对象信息按照时间由近到远依次存放这 2 个性质. 时空范围查询,可以通过单元 Hash 表和桶链表快速查找,利用桶链表中相同对象信息连续有序存放的性质,配合 *tag* 标签,进行大量快速剪枝,查询效率高.

每个桶的头节点的存储结构具有如下形式:(nxt, oid, no, tag),其中,*nxt* 表示指向下一个桶的指针,*oid* 表示该桶所存对象的标识,*no* 表示当前桶中存放的有效对象信息数,*tag* 表示存储该桶后仍然存储同一对象信息的桶数.

对象数据存储为 2 元组的形式:(t_{in}, t_{out}),其中 t_{in} 和 t_{out} 分别表示进入和离开单元的时刻,每一个 (t_{in}, t_{out}) 表示对象在该单元存在过的一个时间区间.

3) 对象链表(object list, *OL*). 提供针对对象的索引,每个对象对应一个单链表,链表中依次存放对象信息存储的位置,链表的表节点为:(cid, bpr). 其中,*cid* 指向移动对象对应的单元,*bpr* 指向记录存放在桶中的具体位置. 每当更新一个对象位置时,

都需要在对象对应链表的最前端插入对应新项.对象位置查询和轨迹查询,可以通过对象链表直接定位到桶链表中存放对象具体有效信息的位置来查找内容,无需查找其余无效内容,查询效率高.

3.2 更新过程

MQII 索引接收到的更新信息经过数据预处理,数据格式为($objectID$,$OutCell$,$InCell$,t).分 3 种情况讨论:

1) 对象第 1 次进入室内空间;

2) 对象进入一个之前去过的单元;

3) 对象进入一个之前没有去过的单元.

更新过程主要包括 3 个方面:

1) 根据对象链表的第 1 项找到对象最新所在单元,更新其离开单元的时间信息.

2) 查找对象进入单元对应的桶链表中该对象的存储位置.如果在桶链表中找不到该对象信息,则新建该对象的桶,插入到桶链表的最前面;如果找到且桶未满,则直接更新信息到未满项,如果桶满则新建一个桶,插入到该桶之前.

3) 将对象新插入的单元信息和位置信息更新到对象对应链表的最前面.

算法描述如下:

算法 1. UPDATE 算法.

输入:更新信息 $new=(objectID, OutCell, InCell, t)$,用 o 表示 new 代表的对象;

输出:信息更新后的索引结构.

① If ($o\notin OL$){
② $OL\leftarrow o$;
③ $Insertnew(new, InCell.BL)$;}
④ Else{
⑤ 从 $o.OL$ 中取出 $first.bpr$;
⑥ 更新 bpr 指向的桶链表中对应项的时间属性,使得 $t.out=t$;
⑦ If (在 $InCell.BL$ 中找到对象,使其 $Bucket.oid=objectID$)
⑧ $Insert(new, BL)$;
⑨ Else
⑩ $Insertnew(new, InCell.BL)$;}
⑪ $o.OL\leftarrow new\ entry$.

其中,$Insertnew$ 函数功能为新建一个桶,存储信息并插入到 $InCell$ 单元对应的桶链表的最前面;$Insert$ 函数表示在桶中相应位置添加信息,如果原桶未满则直接将信息添加到桶后面,否则新建一个桶存储信息并插入到原桶之前.

3.3 查询处理

MQII 索引技术有效支持历史和当前数据的对象查询和范围查询.对象查询基于对象链表,通过快速定位到桶链表存储位置处理查询;范围查询利用对象在单元对应的桶链表中按时间段先后连续存储的性质实现.具体实现算法将在本节讨论.

假设移动对象集合 O,查询涉及的参数有:1)部分对象集合 $O'\subseteq O$;2)空间区域 $S'\subseteq S$;3)时间片 $T'\subseteq T$.

3.3.1 对象查询

对象查询包括对象位置查询和轨迹查询.

3.3.1.1 对象位置查询

给定对象标识集合 O',查询时刻 $t\in T$,对象位置查询返回所有对象 $o\in O'$ 在时刻 t 所处的单元信息 $cid\in S$.分 2 种情况讨论:

1) 当前时间位置查询.只需要根据查询对象对应的对象链表第 1 项的单元信息即可判断,算法时间复杂度为 $O(1)$.

2) 历史时间位置查询.需要查看对象链表项,根据 t_q 是否在(t_{in},t_{out})区间进行判断,此时时间复杂度为 $O(n)$,由于(t_{in},t_{out})按照时间由近到远有序排列,算法执行的元素比较次数取决于 t_q 位于第几个(t_{in},t_{out})里.

具体算法描述如下:

算法 2. PositionQuery 算法.

输入:待查询的对象集合 O'、查询时刻 t_q;

输出:所有对象 $o\in O'$ 在时刻 t_q 所处的单元信息 $cid\in S$,查询结果存放于 res 中.

① For each $o\in O'$ do {
② If (t_q=Now) {
③ If ($o.OL$ 中的 $firstentry.cid$ 不等于室外空间对应的 id)
④ $res=firstentry.cid$;}
⑤ Else {
⑥ While($o.OL\neq\varnothing$) {
⑦ If ($bpr.t_{in}\leqslant t_q\leqslant bpr.t_{out}$)
⑧ $res=entry.cid$ and break;
⑨ Else If ($t_q>bpr.t_{out}$)
⑩ break;
⑪ Else
⑫ 从 $o.OL$ 中取出下一个 $new\ entry$;}}}
⑬ return res.

3.3.1.2 对象轨迹查询

给定对象标识集合 O' 和时间区间 T',对象轨迹查询返回所有对象 $o\in O'$ 在时间 T' 里的运动轨迹,

即依次经过的单元集合 $S'\subseteq S$.

对象链表中的节点按照对象更新时间的先后顺序排列,找到查询的时间区间,满足时间区间的单元序列即为对象经过的单元集合. 由于每次更新都是插入新项到链表表头,所以采用栈结构存储结果集,出栈顺序即为对象轨迹.

算法 3 给出了对象轨迹查询的伪代码.

算法 3. TrajectoryQuery 算法.

输入:待查询的对象集合 O'、查询时间段$[t_{qs}, t_{qe}]$;

输出:对象在时间段内依次经过的单元序列,按对象输出.

① For each $o\in O'$ do{
② While($o.OL\neq$null){
③ If ($bpr.t_{in}\leqslant t_{qe}\leqslant bpr.t_{out}$){
④ While($o.OL\neq\varnothing$){
⑤ $push(cid,S)$;
⑥ If ($bpr.t_{in}\leqslant t_{qs}\leqslant bpr.t_{out}$)
⑦ break;
⑧ $entry=entry\rightarrow next$;}}
⑨ If ($t_{qs}>bpr.t_{out}$)
⑩ break;
⑪ $entry=entry\rightarrow next$;}
⑫ $pop(S,cid)$;}

对于每个对象,算法行③和行⑥分别从对象链表中找到查询结束和开始时刻的位置;行⑤将从找到结束查询时刻开始的单元信息压栈,直到找到查询开始时刻为止;行⑫通过出栈操作获得对象轨迹.

对于对象轨迹查询而言,MQII 索引结构通过对象链表直接定位,获取查询所需数据,查询效率很高. 算法时间复杂度为 $O(n)$,算法执行的元素比较次数取决于 tqs 位于第几个(t_{in}, t_{out})里,结果入栈的次数取决于$[t_{qs}, t_{qe}]$时间段内对象去过多少个单元,即存在的(t_{in}, t_{out})个数.

3.3.2 范围查询

范围查询包括时间戳范围查询和时间片范围查询.

3.3.2.1 时间戳范围查询

给定查询范围 S'和查询时间点 $t\in T$,时间戳范围查询返回在时刻 t 查询范围里的对象集合 O'.

时间戳查询根据查询时刻的不同,分为 2 种情况:

1) 当前时间戳范围查询. 对于查询范围内的每个单元,只需要查看单元对应的桶链表,查看桶的最后 1 项,如果 $t_{out}=\infty$,表明对象在该单元中,添加到结果集. 通过桶头节点中的 tag 属性进行剪枝,每查看过一个桶之后,跳过其之后的 tag 个桶,查看下一个对象,直至桶链表空,返回结果集.

2) 历史时间戳范围查询. 依次检索查询范围内的每个单元对应的桶链表. 顺序检索桶链表,对于每个桶,从最后 1 项向前扫描,这样可以保证检索到的同一个对象的时间信息(t_{in}, t_{out})是有序的. 当 t_q 属于(t_{in}, t_{out})时,将对象添加到结果集,跳过后面存储该对象 tag 个桶,继续检索下一个对象;由于每个对象检索到的时间(t_{in}, t_{out})是有序的,所以并不需要完全检索对象的每一项,当 $t_q>t_{in}$时,表明已超出查询范围,跳过其之后的 tag 个桶剪枝,继续检索下一个对象,直至链表为空,返回结果集.

算法的详细过程如下:

算法 4. TimestampQuery 算法.

输入:查询单元范围 S'、查询时刻 t_q;

输出:满足查询要求的对象集合 O',结果存储在 res 中.

① For each $Cell\in S'$ do {
② If (t_q=Now) {
③ While($Cell$'s $Bucket\neq$null) {
④ $object=readObj(Bucket.entries[Bucket.no-1])$;
⑤ If ($object.t_{out}=\infty$)
⑥ $res.add(object)$;
⑦ 跳过 tag 个桶;}}
⑧ Else {
⑨ While ($Cell$'s $Bucket\neq$null){
⑩ For $id=Bucket.no-1$ to 0 do{
⑪ $object=readObj(Bucket.entries[id])$;
⑫ If ($object.t_{in}\leqslant t_q\leqslant object.t_{out}$) {
⑬ $res.add(object)$;
⑭ 跳过 $tag-1$ 个桶后 break;}
⑮ Else If ($t_q>object.t_{in}$)
⑯ 跳过 $tag-1$ 个桶后 break;
⑰ $Bucket=Bucket\rightarrow next$;}}}
⑱ return res;}

对于当前时间戳范围查询算法,时间复杂度为 $O(n)$,n 取决于单元对应桶链表中存在对象的个数,算法执行的元素比较次数为查询单元中对象的个数. 对于历史时间戳范围查询算法,时间复杂度除了依赖于单元中存在过的对象个数外,还取决于查询时间戳的远近,由于桶链表中同一对象信息按照时间由近到远的顺序有序存放,所以查询时间戳越近查询效率越高.

3.3.2.2 时间片范围查询

给定查询范围 S'和时间片 T',时间片范围查询返回对象集合 $O'\subseteq O$,满足存在 $t\in T'$时 $o\in O'$落在单元 $s\in S'$里.

图 3 为某移动对象在某单元中存在过的时间片序列,时间片(t_{qs}, t_{qe})表示查询范围,对每一段时间(t_{in},t_{out}),分为 4 种情况:

1) $t_{qe}>t_{out}$ & $t_{qs}>t_{out}$,结束该对象查询;

2) $t_{ge}>t_{out}$ & $t_{qs}<t_{out}$,将对象添加到结果集中,结束该对象查询;

3) $t_{qe}<t_{out}$ & $t_{qe}>t_{in}$,将对象添加到结果集中,结束该对象查询;

4) $t_{qe}<t_{in}$,检查该对象的下一个时间片(t_{in}, t_{out}),递归继续判定.

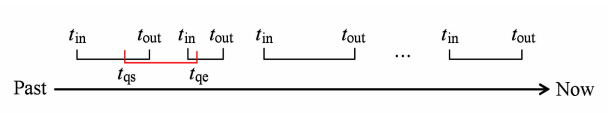


Fig. 3 Time slice sequences.

图 3 时间片序列

算法 5 给出了时间片查询的详细算法.

算法 5. TimesliceQuery 算法.

输入:待查询的单元集合 S'、时间区间[t_{qs},t_{qe}];

输出:满足查询要求的对象集合 O',查询结果存储在集合 *res* 中.

① For each $Cell\in S'$ do {
② While (*Cell*'s *Bucket*≠null) {
③ For $id=Bucket.no-1$ to 0 do {
④ $object=readObj(Bucket.entries[id])$;
⑤ If ($t_{qe}>t_{out}$ & $t_{qs}>t_{out}$)
⑥ 跳过 $tag-1$ 个桶后 break;
⑦ Else If (($t_{ge}>t_{out}$ & $t_{qs}<t_{out}$) ‖ ($t_{qe}<t_{out}$ & $t_{qe}>t_{in}$)) {
⑧ $res.add(object)$;
⑨ 跳过 $tag-1$ 个桶后 break;}
⑩ $Bucket=Bucket\rightarrow next$;}}}
⑪ return *res*.

算法 5 中行①依次查找查询范围内的每个单元;行②③从后向前检查每个桶项,直到链表为空;行⑤⑥描述的是情况 1;行⑦~⑨描述的是情况 2 和 3;行⑩描述的是情况 4;行⑪返回查询结果.

时间片查询算法时间复杂度依赖于单元中存在过的对象个数,还取决于查询时间区间范围的大小,基于桶链表中同一对象信息连续存放以及同一对象信息按照时间由近到远顺序有序存放这 2 个性质,查询效率能够通过 *tag* 属性快速排除无效信息,保障查询效率.

4 实 验

为了验证本文提出的 MQII 索引结构的有效性,采用 C++语言在 VS2010(microsoft visual studio 2010)环境下分别对 MQII, ACII[9], RTR-tree[8], DR-tree[16]索引结构进行了实现. 实验建立在同一运行环境、同一数据集下,结果均是 500 个查询的平均执行时间,查询参数基于随机选取原则. 实验环境为:2.10 GHz CPU,2.0 GB 内存,Windows 7 操作系统.

4.1 实验数据

实验中的数据集使用文献[21]中的 MWGen 数据生成器模拟生成,模拟的室内环境来源于真实场景共 8 层,包含 294 个室内单元(包括办公室、走廊、楼梯、厕所等),所有的房间和楼梯口都是通过门与走廊相通,移动对象可以在单元内部移动,也可以移动到与之相连的其他单元.

模拟 10 000~100 000 个随机移动对象的运动情况,每个移动对象有一个生命周期,在生命周期内按照随机生成的运动轨迹移动. 初始时刻每个移动对象选择一个目的地,按照一定的速度移动,从而生成运动轨迹.

将 MWGen 数据生成器生成的同一数据集,经过数据预处理,生成索引需要的数据集,数据格式为:

1) MQII 索引.(*objectID*,*OutCell*,*InCell*,*t*),表示对象 *objectID* 在时刻 *t* 从单元 *OutCell* 移动到 *InCell*.

2) ACII 索引.(*objectID*,*cellID*,*t*),表示对象 *objectID* 在时刻 *t* 进入了 *cellID* 单元.

3) RTR-tree 索引.(*objectID*,*cellID*,t_1,t_2),表示对象 *objectID* 在[t_1,t_2]时间段内在 *cellID* 单元中.

4) DR-tree 索引.(*objectID*, *cellID*, t_1, t_2),表示对象 *objectID* 在[t_1,t_2]时间段内在 *cellID* 单元中.

4.2 实验结果分析

4.2.1 数据预处理性能

MQII 索引和 ACII 索引都是基于单元实现的索引结构,图 4 是两者需要处理的数据量大小的比较. 实验结果表明,数据预处理方法能降低原始数据集大小,相比 ACII 而言,降低了约 60%. 这是由于数据预处理基于部署图,初步分析出对象的单步移动

轨迹,关注于移动对象的位置转换,将 2 条记录合为 1 条记录,降低一半的数据量,同时能够删除存在设备读取的无用信息和划分设备读取的重复信息,从而达到大大降低索引需要处理的数据量的效果.

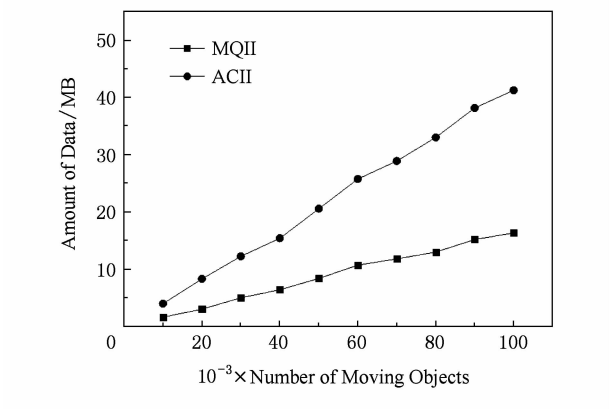


Fig. 4 Performance of data pre-processing.

图 4 数据预处理的性能

4.2.2 索引更新代价

图 5 是索引更新的开销对比图. 随着移动对象数目的增加,数据量增加,MQII,RTR-tree,DR-tree 索引的更新代价增加. MQII 索引的更新代价为毫秒级,但较其他索引略高,这是因为 ACII 索引无需查找旧记录而直接插入移动对象新的记录;RTR-tree 和 DR-tree 索引自顶向下快速找到插入位置;但 MQII 索引为了追求更好的查询性能,在向新进入的单元添加记录时,需要查找对象在该单元中原先存储信息的位置,以保持同一对象信息存储的连续性,虽然采用按标记位剪枝的方法降低更新代价,但由于数据量的增加,更新开销会相应增加.

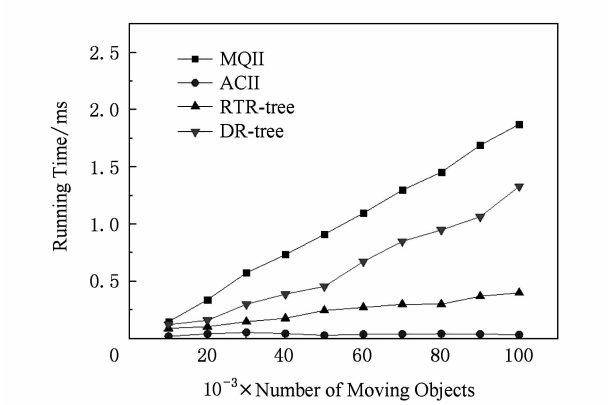


Fig. 5 Index update cost.

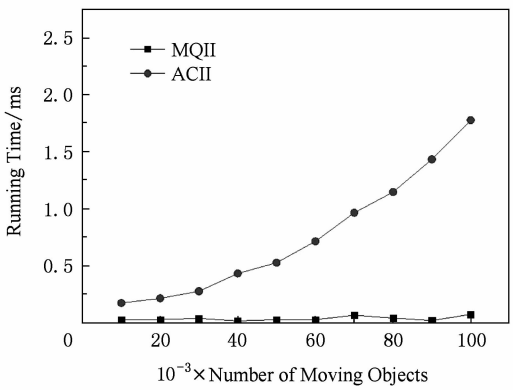
图 5 索引平均更新开销比较

4.2.3 查询性能分析

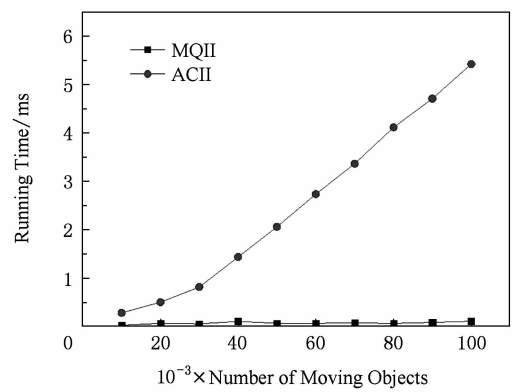
4.2.3.1 平均查询开销对比

选取 500 个自由的查询,查询时间区间为时间生命期的任意子区间,对象查询选取一个随机对象作为查询输入,范围查询选取一个随机单元作为查询输入.

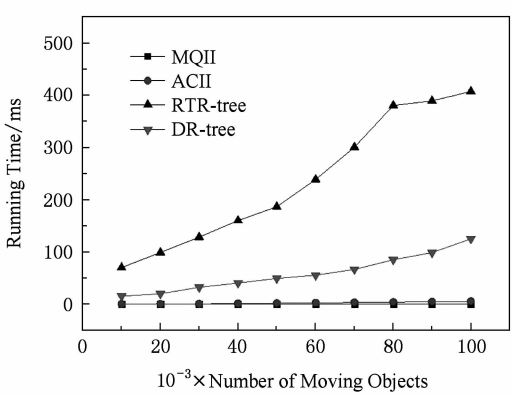
图 6 是对象查询性能对比图,图 6(a)为对象当前位置查询,图 6(b)(c)是对象轨迹查询,可知 MQII 索引查询性能最优,较 RTR-tree 和 DR-tree 索引更有明显提升. MQII 索引结构通过对象链表直接定位,获取数据;而 ACII 索引由于无法获知对象存储在 MC 结构中哪个单元,几乎需要查找所有单元的 MC 结构查找对象当前所处单元,降低了查



(a) Current position queries



(b) Trajectory queries for MQII and ACII



(c) Trajectory queries for all

Fig. 6 Query performance vs. object queries.

图 6 对象查询性能对比

询效率;RTR-tree 索引仅支持历史数据信息查询,由于其不存在对象的索引,在满足查询需求时间维的情况下,几乎要遍历所有节点才能获取需要的信息,查询效率较其他 2 个索引而言,效率极低;DR-tree 索引由于添加了基于对象的索引维度,效率较 RTR-tree 索引有所提高,但由于索引中 OT-tree 结构对轨迹的保存并不彻底,同一节点或相邻节点中不能保证存储的是同一轨迹的索引记录,且索引结构重复信息多,从而会造成对其查询效率的影响,100 000 数据集时查询时间达到 0.1 s 以上.

图 7 是范围查询的性能对比图,由于在 DR-tree 中,负责时空范围查询的是 LT-tree,其采用了与 RTR-tree 相同的索引结构,在查询效率上没有差别,为了实验结果图的清晰度,没有在有关范围查询的实验结果图中添加 DR-tree 的曲线,其曲线与 RTR-tree 重合. 从图 7 可以看出,MQII 索引和 RTR-tree 索引查询性能明显优于 ACII 索引.

在 MQII 索引中,单元对应的桶链表负责范围查询,由于桶链表中同一对象信息连续存放且每个桶的头节点中存放的 *tag* 属性,记录该桶之后仍然存放该对象信息的桶数,可以方便地用于实现大量

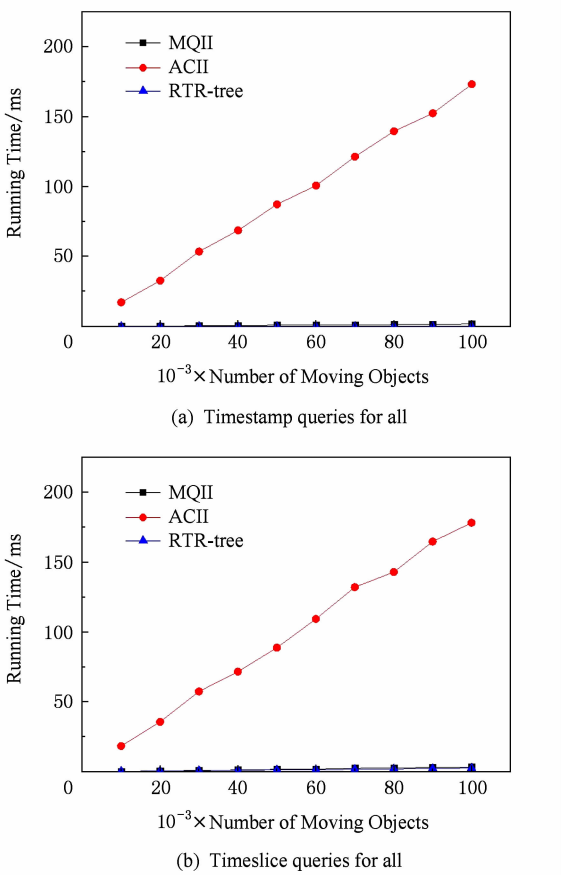


Fig. 7 Query performance vs. range queries.

图 7 范围查询性能对比

剪枝,缩小查询范围;ACII 索引由于大量信息存放于 MEMO 结构中,以对象为索引,单元对应的 MC 结构中只存储当前存在的对象信息,几乎需要遍历 MEMO 中对象的所有桶链表,效率很低;RTR-tree 索引结构能够通过 R-tree 的剪枝快速获取需要查询的对象,查询效率高.

4.2.3.2 查询参数对性能的影响

1) 时间区间大小对查询性能的影响

将单元范围固定为 1,移动对象数目选择默认大小 20 000,令时间区间选取为数据集生命周期的 10%~50%. 图 8(a)(b)表示时间区间对轨迹查询性能的影响,图 8(c)(d)表示时间区间对时间片范围查询性能的影响. 与图 7 类似,为了使图像更清晰,图 8(d)未添加 DR-tree 曲线. 从图 8 可以看出,查询时间区间越大,查询时间越长.

对于轨迹查询,时间区间的增加对于 MQII 索引和 ACII 索引而言,因为存在基于对象的索引,只需查询该对象更多的信息,影响不大且查询效率高. 而从图 8(b)可以看出,对于 RTR-tree 索引而言,由于没有基于对象的索引,需要查询满足时间范围内的所有节点,时间区间对其查询性能影响较大;对于 DR-tree 而言,虽然可以对对象进行索引,但随着查询对象数目的增加,造成查询范围增加,从而导致所需的查询时间增加.

对于时间片范围查询而言,图 8(d)所示,MQII 索引和 RTR-tree 索引明显优于 ACII 索引,ACII 索引由于没有基于单元的索引,需要检索每个 MEMO,由于时间的增加,导致检索范围增加. 从图 8(c)可以看出,RTR-tree 索引虽然查询性能比 MQII 索引高,但其开销增加的幅度却比 MQII 索引大,时间区间的增加使得 MQII 索引在单元对应的桶链表中查找每个对象的信息量增大,开销增加平缓;但 RTR-tree 索引查询时间区间的增加导致查询范围的矩形框呈线性增加,影响较大.

2) 单元范围大小对查询性能的影响

将时间区间固定为数据集生命周期大小的 10%,移动对象数目固定为 20 000,单元范围选取为数据集单元总数的 1%~5%个任意单元.

图 9(a)(b)分别表示单元范围大小对时间戳和时间片范围查询性能的影响,可以看出,查询范围对 MQII 索引和 RTR-tree 索引有一定的影响,随着查询单元范围的增加,查询性能降低. 与图 7 类似,为了使图像更清晰,图 9 未添加 DR-tree 曲线. MQII 索引范围查询基于桶链表,由于单元范围的增加,需要查询的桶链表数增加,导致查询时间增加;RTR-tree

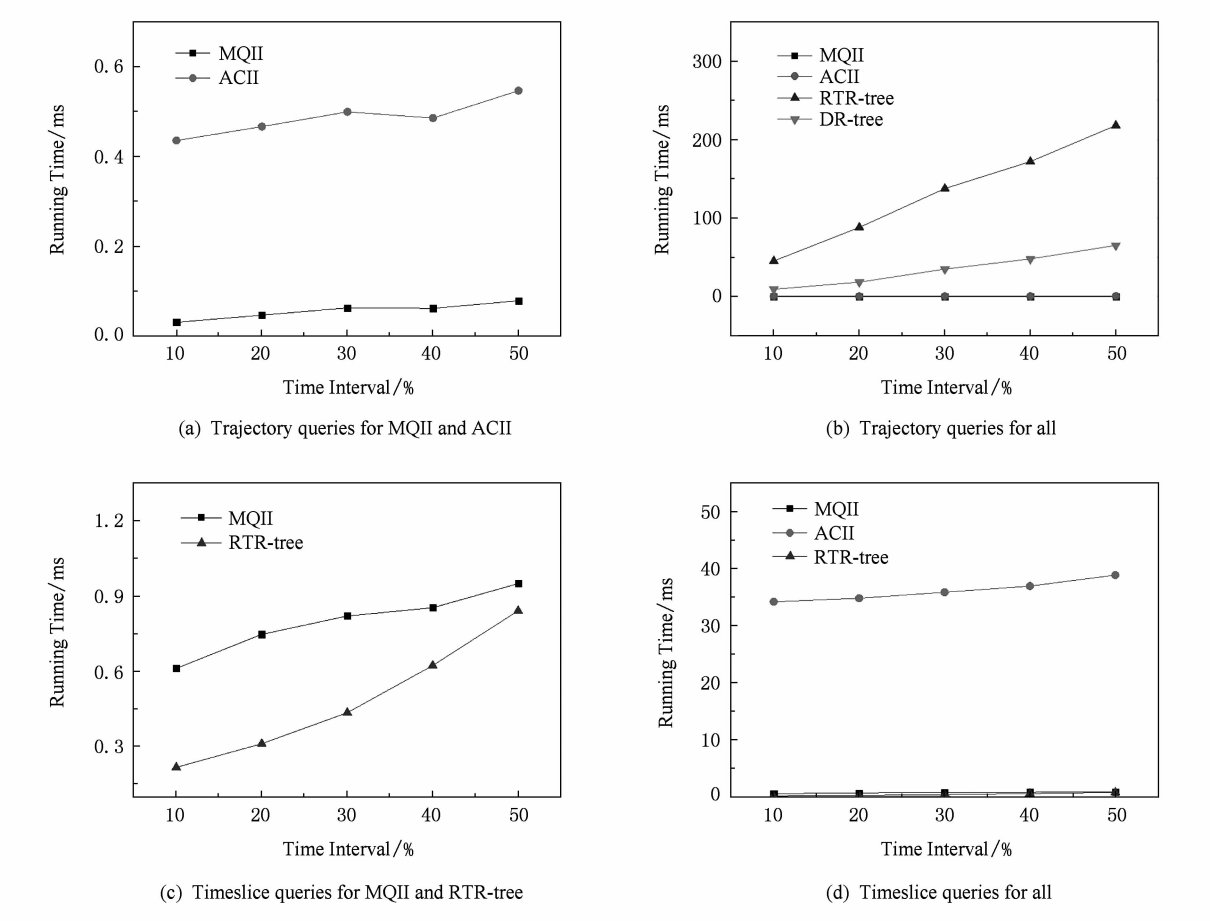


Fig. 8 Query performance vs. time interval.

图 8 时间区间对查询范围的影响

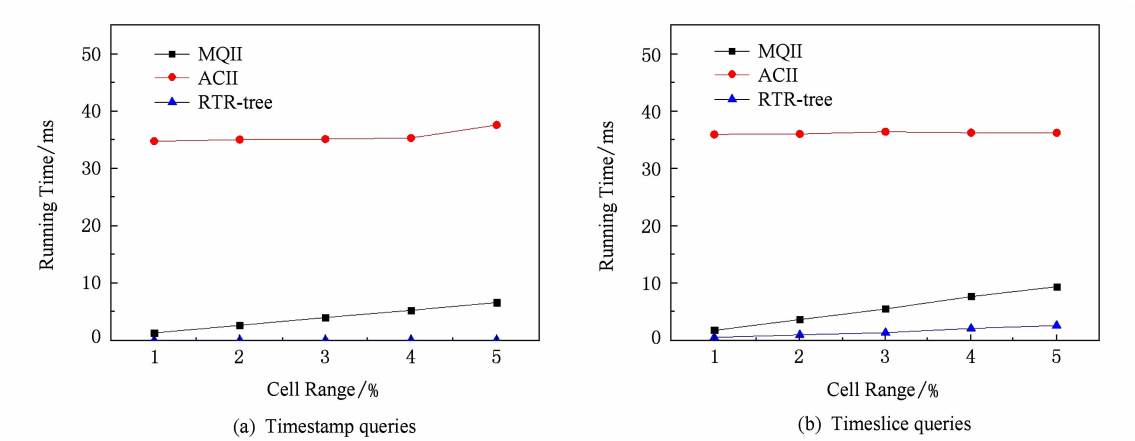


Fig. 9 Query performance vs. cell range.

图 9 单元对查询性能的影响

索引随着单元范围的增加,查询 MBR 增加,查询时间增加;ACII 索引由于不存在基于单元的检索,需要遍历所有对象的桶链表得到查询结果,虽然查询性能基本不会受单元范围的影响,但其效率最低.

综合图 6～9 可知,MQII 索引不仅能够支持历史和当前数据的查询,而且能够同时有效支持对象查询和范围查询.

5 结 论

现有室内移动对象索引方法,关注于支持对象查询或者是范围查询. 本文提出了一个新的室内移动对象索引方法即 MQII 索引,能够同时有效支持对象查询和范围查询;针对索引结构,结合剪枝技

术,提出了有效的更新和查询算法;通过实验验证了索引结构的有效性.实验结果表明,与现有室内移动对象索引技术相比,在不影响查询效率的情况下,MQII 索引不仅能够支持历史查询和当前查询,而且能够同时支持 ACII 索引的对象查询和 RTR-tree 索引的范围查询.

参 考 文 献

[1] Zhou Aoying, Yang Bin, Jin Cheqing, et al. Location based services: Architecture and progress [J]. Chinese Journal of Computers, 2011, 34(7): 1155-1171 (in Chinese)
(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] Worboys M. Modeling indoor space [C] //Proc of the 3rd ACM SIGSPATIAL Int Workshop on Indoor Spatial Awareness. New York: ACM, 2011: 1-6

[3] Güting R H, Böhlen M H, Erwig M, et al. A foundation for representing and querying moving objects [J]. ACM Trans on Database Systems, 2000, 25(1): 1-42

[4] Güting H, de Almeida T, Ding Zhiming. Modeling and querying moving objects in networks [J]. The International Journal on Very Large Data Bases, 2006, 15(2): 165-190

[5] Xu Jianqiu, Güting R H. Manage and query generic moving objects in SECONDO [J]. Proceedings of the VLDB Endowment, 2012, 5(12): 2002-2005

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

[7] Lu Hua, Yang Bin, Jensen C S. Spatio-temporal joins on symbolic indoor tracking data [C] //Proc of the 27th Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2011: 816-827

[8] Jensen C S, Lu Hua, Yang Bin. Indexing the trajectories of moving objects in symbolic indoor space [G] //Advances in Spatial and Temporal Databases. Berlin: Springer, 2009: 208-227

[9] Shin S S, Kim G, Bae H Y. Adaptive cell-based index for moving objects in indoor [J]. KSII Trans on Internet and Information Systems, 2012, 6(7): 1815-1830

[10] Fang Ying, Cao Jiaheng, Peng Yuwei, et al. Efficient Indexing of the past, present and future positions of moving objects on road network [G] //Web-Age Information Management. Berlin: Springer, 2013: 223-235

[11] Nascimento M A, Silva J R O. Towards historical R-trees [C] //Proc of the 1998 ACM Symp on Applied Computing. New York: ACM, 1998: 235-240

[12] Tao Yufei, Papadias D. The MV3R-tree: A spatio-temporal access method for timestamp and interval queries [C] //Proc of the 27th VLDB Conf. San Francisco, CA: Morgan Kuafmann, 2001: 431-440

[13] Kwon D, Lee S, Lee S. Indexing the current positions of moving objects using the lazy update R-tree [C] //Proc of the 3rd Int Conf on Mobile Data Management. Piscataway, NJ: IEEE, 2002: 113-120

[14] Šaltenis S, Jensen C S, Leutenegger S T, et al. Indexing the positions of continuously moving objects [C] //Proc of the 2000 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2000: 331-342

[15] Tao Yufei, Papadias D, Sun J. The TPR*-tree: An optimized spatio-temporal access method for predictive queries [C] //Proc of the 29th Int Conf on Very Large Data Bases. San Francisco, CA: Morgan Kuafmann, 2003: 790-801

[16] Gan Zaobin, Yuan Yongguang, Zhao Yizhu, et al. Indoor moving objects index research based on DR-tree [J]. Computer Science, 2012, 39(10): 177-181 (in Chinese)
(甘早斌, 袁永光, 赵贻竹, 等. 基于 DR-tree 的室内移动对象索引研究 [J]. 计算机科学, 2012, 39(10): 177-181)

[17] Alamri S, Taniar D, Safar M. Indexing moving objects in indoor cellular space [C] //Proc of the 15th Int Conf on Network-Based Information Systems. Piscataway, NJ: IEEE, 2012: 38-44

[18] Alamri S, Taniar D, Safar M, et al. Spatiotemporal indexing for moving objects in an indoor cellular space [J]. Neurocomputing, 2013, 122: 70-78

[19] Alamri S, Taniar D, Safar M, et al. A connectivity index for moving objects in an indoor cellular space [J]. Personal and Ubiquitous Computing, 2014, 18(2): 287-301

[20] Jensen C S, Lu Hua, Yang Bin. Graph model based indoor tracking [C] //Proc of the 10th Int Conf on Mobile Data Management. Piscataway, NJ: IEEE, 2009: 122-131

[21] Xu Jianqiu, Guting R H. MWGen: A mini world generator [C] //Proc of the 13th Int Conf on Mobile Data Management. Piscataway, NJ: IEEE, 2012: 258-267

Ben Tingting, born in 1990. Received her ME degree from Nanjing University of Aeronautics and Astronautics. PhD candidate. Her current research interests include moving object index and indoor moving objects.

Qin Xiaolin, born in 1953. Professor and PhD supervisor. Senior member of China Computer Federation. His main research interests include spatio-temporal databases, secure database, data management and disaster tolerance in distributed environment and information security.

Xu Jianqiu, born in 1982. Received his PhD degree from FernUniversitaet Hagen in Germany. Associate processor. His current research interests include spatial databases and moving objects databases (jianqiu@nuaa.edu.cn).