DOI:10.16356/j.1005-2615.2018.05.005

面向多属性条件的空间对象检索算法

韩文军[1] 吉根林[2] 朱承治[3] 徐 成[2] 赵 斌[2]

(1.国网经济技术研究院有限公司,北京,102209; 2.南京师范大学计算机科学与技术学院,南京,210023;
3.国网浙江省电力有限公司,杭州,310007)

摘要:针对现有空间索引不能满足多样化的检索需求,提出两种新型空间索引,能够同时面向空间属性、文本属性与划分属性进行空间对象检索。将分类技术应用于空间对象检索中,提出了基于划分索引与 IR-Tree 的混合索引以及先划分索引再 IR-Tree 的索引方法,不仅满足了多样化的空间检索需求,而且有效地解决了传统空间索引更新维护代价大的问题。基于真实的北京市 POI 数据集进行实验,结果表明本文提出两种索引是有效的且检索效率高。与传统空间索引相比,提出的空间索引很好地解决了具有划分属性的空间对象检索问题,并且具有较高的检索效率。

关键词:空间索引;空间对象检索;划分属性;混合索引

中图分类号:TP181 **文献标志码**:A **文章编号**:1005-2615(2018)05-0611-08

Spatial Object Retrieval Algorithms for Multi-attribute Conditions

HAN Wenjun[1], *JI Genlin*[2], *ZHU Chengzhi*[3], *XU Cheng*[2], *ZHAO Bin*[2]

(1. State Grid Economic and Technological Research Institute Co, LTD, Beijing, 102209, China;
2. School of Computer Science and Technology, Nanjing Normal University, Nanjing, 210023, China;
3. State Grid Zhejiang Electric Power Co, LTD, Hangzhou, 310007, China)

Abstract: In view of the fact that the existing spatial indexes cannot meet diversified retrieval requirements, this paper proposes two new types of spatial indexes, which can search the spatial object with spatial attributes, text attributes and dividing attributes simultaneously. We apply classification techniques to the existing spatial object retrieval, and propose two spatial indexes including a hybrid index based on both dividing index and IR-Tree and first dividing index then IR-Tree index. Both spatial indexes not only meet the requirement of diversified spatial retrieval, but also solve the problem of the traditional spatial index's high maintenance cost effectively. The experiments based on the real data set show that the two indexes are effective and efficient. Compared with the traditional spatial indexes, the spatial indexes we proposed solve the problem of spatial object retrieval with dividing attribute, and achieve higher retrieval efficiency.

Key words: spatial index; spatial object retrieval; dividing attributes; hybrid index

随着全球定位技术的发展与手机应用的普及,基于位置的服务得到了广泛的应用。用户使用这些应用程序,通过当前的位置信息和输入的一系列关键词来寻找周围的兴趣点,这种服务被称为空

基金项目:国家电网有限公司科技项目(SGZJ0000KXJS1700477)资助项目;国家自然科学基金(41471371,41301142)资助项目。

收稿日期:2018-01-23;**修订日期**:2018-06-21

通信作者:吉根林,男,博士,教授,E-mail:glji@njnu.edu.cn。

引用格式:韩文军,吉根林,朱承治,等.面向多属性条件的空间对象检索算法[J].南京航空航天大学学报,2018,50(5):611-618. HAN Wenjun, JI Genlin, ZHU Chengzhi, et al. Spatial object retrieval algorithms for multi-attribute conditions[J]. Journal of Nanjing University of Aeronautics & Astronautics,2018,50(5):611-618.

间-文本检索。例如人们使用手机 App 获取当前位置,再输入关键字“酒店”来寻找附近的酒店。

针对空间-文本检索问题的研究早已展开,其中大部分工作研究检索的效率问题。2005 年,Zhou 等[1]提出一种基于位置服务的网页搜索框架,比较了 3 种不同的索引结构:基于倒排文件和 R 树[2]的双重索引、先倒排文件再 R 树索引以及先 R 树再倒排文件索引。实验表明双重索引最慢,因为双重索引是分别对空间检索和文本检索建立索引,然后将两个检索结果集融合,该过程花费大量时间。文献[3]在原有的索引基础上提出了一种新的索引结构 IR-Tree,该索引在 R-Tree 的结构基础上叠加文本信息,也就是在 R-Tree 的每个节点同时保存子节点的文本信息,解决了同时索引空间信息与文本信息的问题,具有较高地效率。Chen[4]使用分布式技术来加速空间-文本检索的效率。还有一部分工作[5-9]从另一角度出发,使用反向 k 近邻技术来解决 top-k 空间空间-文本检索问题。

此外,针对空间属性与文本属性融合排序的相关方法主要分为两类:线性融合方法和非线性融合方法。在线性融合方面,Cai[10]提出了融合地理信息检索和文本信息检索的新方法。它在地理坐标空间和词项空间对语料库中的每一篇文档进行索引,然后在这两个空间分别进行检索,将返回的结果进行线性融合。在非线性融合方面,Yu 等[11]利用查询操作的地理特殊性决定相似性的权重,以便融合检索的排序结果;Hu 等[12]通过基因规则学习排序函数实现空间信息检索的排序;Martins[13]使用 SVM 分类模型融合不同的文本相关性和地理相关性来实现检索排序。

虽然上述方法能够较好地解决已有的空间-文本检索问题,但随着技术的发展与数据的积累,空间-文本检索问题的检索方式也呈现出多样化与精细化。例如人们在搜索附近的酒店时,不仅会选择自己当前位置并且输入检索关键字,还会选择一系列附加的筛选条件,例如“星级:四星级酒店”“人均:200-300”“评分:8.5 以上”等。本文将这些条件称为除了空间属性与文本属性之外的第 3 种属性,划分属性。现有空间-文本检索方案并不能很好地解决具有多划分属性的空间-文本检索问题。空间属性、文本属性和划分属性是 3 种完全不同的属性,也都具有各自的索引结构。由 Zhou 等的实验可知双重索引效率反而较低,因为涉及到两个结果集的 join 操作从而费时。由此可以推断如果使用三重索引,再将结果集融合,效率肯定是更低的,所以问题的解决方案要从混合索引的角度出发。典型的空间文本检索混合索引 IR-Tree 具有较差的扩展性,若使用 IR-Tree 来进行 3 种属性的综合检索,只能针对其中的空间属性和文本属性使用 IR-Tree 进行检索,再用结果集匹配划分属性进行结果集过滤。但是每次不同的空间条件和文本条件检索出来的结果集都是不一样的,所以不可能对结果集提前建立划分属性的索引,这将导致检索效率的降低。由于现有方案的不足,本文解决具有空间-文本-划分属性的空间对象综合检索问题,具有以下两个挑战:(1)混合索引具有复杂性。空间索引、文本索引和划分索引是 3 种结构完全不同的索引,如何设计一种新的综合索引结构来保存所有索引信息并提供索引功能,这是需要解决的问题。(2)混合索引的效率。对数据进行分类再查找可以迅速降低数据量,提高效率。如何充分利用划分属性的特性来提高索引的效率,是需要解决的另一个问题。

为了应对以上挑战,本文提出了两种全新的索引,基于划分索引与 IR-Tree 的混合索引(Hybrid category-IRTree index,HCIR-Tree)以及先划分索引再 IR-Tree 索引(First category then iRTree Index,FCIR-Tree)来解决具有空间属性、文本属性与划分属性的空间对象综合检索问题。HCIR-Tree 借鉴了 IR-Tree 的思想,IR-Tree 的主体结构是以 R 树为主,将对象的文本属性同时存储在 R 树的节点上,以此类推,可以同时将划分属性也存储在 R 树的节点中,3 种属性同时索引。而 FCIR-Tree 是先用分类的技术对源数据进行分类,然后对分类得到的每个簇建立 IR-Tree。分类可以完美地解决划分属性匹配问题,而 IR-Tree 对剩下的空间属性与文本属性检索效率也有保证。

1 问题描述

本文将 1 个空间对象 T 定义为 $T=(T.\mathrm{loc}, T.\mathrm{text}, T.\mathrm{category})$,其中 $T.\mathrm{loc}$ 为空间对象的位置信息,$T.\mathrm{text}$ 为空间对象的文本描述信息,而 $T.\mathrm{category}$ 是 1 个形如[key1-value1,key2-value2…]的多个键值对,每一个 key-value 对中,key 代表空间对象划分属性的一种,而 value 代表空间对象此种类划分属性的取值。$D=\{T_1, T_2, T_3, \cdots\}$ 为空间数据集合。为了更好地说明空间-文本-划分属性综合检索问题,定义以下评分函数。

定义(评分函数) 1 个典型的空间检索为给

定 1 个检索区域 queryArea，返回一系列评分由高到低的空间对象。这些空间对象按空间位置与检索区域的相对位置关系进行评分[14]，表示为 spatialScore(T) = Distance(T. loc, queryArea)。同理，1 个文本检索的检索条件是一系列关键字 w_1，w_2，…返回结果是一系列评分由高到低的文本对象，评分函数为 IRScore=Sim($\boldsymbol{D},\boldsymbol{Q}$) = $\sum_{ti\in Q,D} w_{ti_Q} \cdot w_{ti_D}$。$\boldsymbol{D}$ 和 $\boldsymbol{Q}$ 向量分别为检索关键字和待匹配关键字构成的词组向量。

空间-文本-划分属性综合检索需要同时考虑空间属性的匹配度和文本属性的匹配度，而划分属性必须满足。因此本文定义综合评分函数 HybridScore 来评估空间-文本-划分属性检索结果对象的评分，定义如下：HybridScore = α_1 * spatialScore + α_2 * IRScore，且 $\alpha_1+\alpha_2=1$，α_1 和 α_2 分别为空间检索得分占比和文本检索得分占比控制因子。

给定检索条件 Q=(queryArea, queryWords, category)，空间-文本-划分属性综合检索返回完全满足划分属性并且综合评分由高到低的一系列空间对象。其中 queryArea 为检索区域，queryWords 为一系列检索关键字，category 为一系列划分属性和其对应取值。

2　算法设计

在现有的 web 引擎中，针对多种划分属性的检索是一种最常见，也是最基本的需求。例如网上购物会指定一系列筛选条件“用途：家具，品牌：苏泊尔…”，甚至随着大数据时代的来临，已经提出了分布式的多划分属性查询研究[15]。因此，本文尝试用多划分属性查询研究中最基本的方法，属性分类，来解决本文中的难题。将分类方法与已有的空间索引 IR-Tree 进行融合，提出了 HCIR-Tree 和 FCIR-Tree。

2.1　IR-Tree 空间索引

IR-Tree 是解决空间-文本属性检索问题的 1 种索引结构。IR-Tree 是 R 树索引加倒排索引的混合索引，以空间索引 R 树为主体，将其改进，同时在每个节点保存子节点的文本属性信息。

混合索引 IR-Tree 的结构如图 1 所示。O_1～O_8 是 8 个空间对象。与 R 树类似，空间中相近的对象(O_1O_2，$O_3O_4O_8$，O_5，O_6O_7)会被合并到 1 个更大的空间矩形对象中(R_1，R_2，R_3，R_4)。然后依次向上合并，最后得到的根节点所具有的空间矩形能够覆盖所有空间对象。但与 R 树不同的是，IR-Tree 的每个结点不仅保存子节点的空间信息，也保存子节点的文本信息。每次检索时由上至下，在每个节点不仅要判断当前结点的空间覆盖区域与检索区域的关系，也要判断当前结点的文本信息和检索关键字是否有交集。IR-Tree 能够解决空间-文本检索问题，但是并不能对划分属性的检索提供支撑。

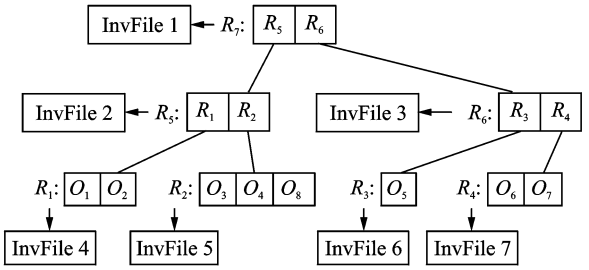

图 1　IR-Tree 索引结构

Fig. 1　Index structure of IR-Tree

2.2　基于划分索引与 IR-Tree 的混合索引

基于划分索引与 IR-Tree 的混合索引(HCIR-Tree)是以 R 树为主体结构，附加倒排索引与划分索引信息的一种混合索引。索引结构如图 2 所示。传统的 R 树在叶子节点保存空间对象信息，在非叶子节点保存子节点的最小外接矩形。HCIR-Tree 存储的每个空间对象不仅有空间属性信息，还同时具有文本属性信息及划分属性信息。在非叶子节点上，不仅要存储其子节点的空间信息，也要同时存储划分属性信息与文本属性信息。这样存储的好处是每个节点都精确保存了子节点的信息，在检索的时候从根节点开始，每个节点都同时判断 3 种属性是否同时满足，只要有任意一个条件不满足，则直接结束当前节点分支的检索。

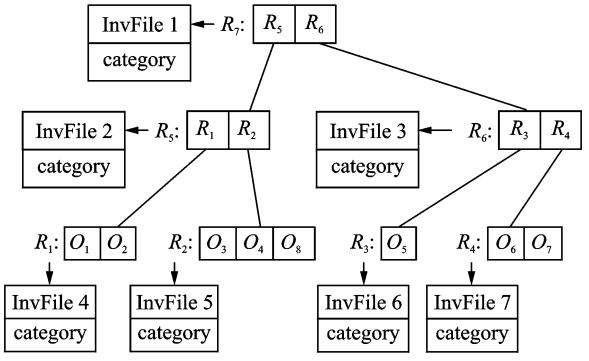

图 2　HCIR-Tree 索引结构

Fig. 2　Index structure of HCIR-Tree

HCIR-Tree 主要涉及的操作有两个：HCIR-Tree 的建立与检索。

2.2.1 HCIR-Tree的建立

HCIR-Tree的插入算法如算法1所示。建立HCIR-Tree的过程就是空间对象的插入操作。对于新来的每个空间对象,首先判断其属性,为其寻找合适的插入点。将其插入后,更新父节点向上的所有非叶子节点信息,接着调整整个HCIR-Tree的高度。

对于每一个新要插入的空间对象T,先为其选择合适的插入点(5行),选择函数与R树思路相同,只考虑空间属性的匹配。若插入点已经满了,则分列该插入点点得到新的叶子节点,否则直接将空间对象T插入该插入点(6~10行)。然后将整棵HCIR-Tree进行再平衡,调整高度(11~16行)。

算法1 HCIR-Tree插入算法

输入:空间索引HCIR-Tree的根节点root,空间对象T

输出:空间索引HCIR-Tree

(1) insertLocation←∅;//待插入的节点

(2) newLeaf←∅;//新的叶子结点

(3) newRoot←∅;新的根

(4) tmpNode←∅;临时结点

(5) insertLocation = chooseLeaf (root, T.MBR);//寻找插入点

(6) if insertLocation is Full then//判断插入点是否已经满了

(7) newLeaf ← splitNode (insertLocation, T);

(8) else if insertLocation is Insertable then//判断是否可插入

(9) insertLocation.addData(T)

(10) end if

(11) newRoot←adjustTree (insertLocation, newLeaf);//平衡整个tree

(12) if newRoot! =null then//新的根节点调整高度产生

(13) tmpNode←root;

(14) root←∅;

(15) root.add (tmpNode);

(16) root.add (root);

(17) end if

2.2.2 基于HCIR-Tree的检索

HCIR-Tree的每次检索时由根节点开始,在每个节点上同时判断空间属性、文本属性与划分属性是否满足。具体过程如算法2所示。

算法2 基于HCIR-Tree的检索算法

输入:当前结点currentNode,查询空间矩形区域rectangle,筛选属性category,查询关键字words

输出:检索结果集合resultSet

(1) if currentNode MBR is CrossingWith (rectangle) then

(2) if currentNode is Leaf ( ) then

(3) //判断每个对象是否满足检索条件

(4) for each entry∈currentNode.Data do

(5) if Contains (rectangle,entry.MBR,

(6) category, entry. category, words,

(7) entry. words) then

(8) resultSet. add (entry);

(9) end if

(10) end for

(11) else

(12) //判断当前子节点是否满足条件,递归查询

(13) for each node∈currentNode. Child do

(14) if Contains(rectangle,node.MBR,

(15) category, node. category, words,

(16) node. words) then

(17) search (node, rectangle, resultSet,

(18) category, words);

(19) end if

(20) end for

(21) end if

(22) end if

当输入检索条件进行检索,首先判断根节点的空间属性与空间检索条件区域是否有交集,如果没有则直接结束运算(2行)。若有交点,再判断该节点是叶子节点还是非叶子节点(3行)。对叶子节点,取出节点所有空间对象数据,一一判断3种属性是否都满足检索条件,满足的对象加入结果集(5~10行)。对非叶子节点,首先判断其子节点是否满足3种检索条件,若满足则进行递归查询,若不满足则放弃(13~18行)。

HCIR-Tree索引是对IR-Tree索引的一种扩展,在IR-Tree原有的数据结构上继续叠加划分属性信息。HCIR-Tree索引在每个节点精确的判断空间、文本和划分条件是否同时满足,以此来解决3种条件综合检索的问题,但是该索引也存在一些缺陷。由于每个节点都保存了所有子节点的信息,所以会造成信息存储的冗余,并且因为节点数据结

构较为复杂，也大大影响了整个索引检索更新的效率。相比之下，FCIR-Tree 具有更大的优势。

2.3　先划分索引再 IR-tree 索引

先划分索引再 IR-tree 索引(FCIR-Tree)是先使用分类的技术，对数据的划分属性进行分类，得到 N 个划分属性互不相同的簇，然后针对每 1 个簇单独建立 IR-Tree。FCIR-Tree 的索引结构如图 3 所示。最上面是 1 棵分类树，分类树的每 1 层代表 1 种划分属性的划分，第 1 层划分属性 1 有 3 种取值，则数据集被分为 3 个簇 C_1，C_2 和 C_3。第 2 层将每个簇再次根据划分属性 2 进行划分，得到更多细分的簇。依次类推，最后形成 N 个划分属性互不相同的簇集合，然后为每个簇内的空间对象建立对应的 IR-Tree。

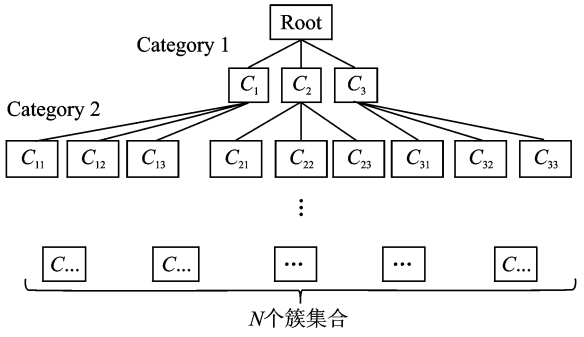

图 3　FCIR-Tree 索引结构

Fig. 3　Index structure of FCIR-Tree

FCIR-Tree 要涉及的操作有 2 个：FCIR-Tree 的建立与检索。

2.3.1　FCIR-Tree 的建立

对于每 1 个空间对象，先判断其属于哪一个簇，然后将数据插入到该簇的 IR_Tree 中。

FCIR-Tree 的建立过程就是将空间对象依次插入。该算法的主要思想是先对数据进行分类得到簇集合，再对每个簇建立 IR-Tree。对于每一个新的空间对象 T，首先计算获取其划分属性标识。若 FCIR-Tree 中已经包含具有此标识的簇，则直接将空间对象加入相关的簇(4～5 行)。若不存在此标识，则为此标识建立新的 IR-Tree 并将该空间对象插入此簇(7～8 行)。

算法 3　FCIR-Tree 插入算法

输入：空间索引 categorytree，空间对象 T

输出：FCIR-Tree

(1) category←∅;//分类标识

(2) IRTree←∅;

(3) category←T. category;

(4) if categorytree. keySet. Contains (category) then

(5) categorytree. get (category). insert(T)

(6) else

(7) IRTree. insert (T);

(8) categorytree. put(category,IRTree);

(9) end if

2.3.2　基于 FCIR-Tree 的检索

首先根据检索条件的划分属性，直接定位到所有相关的簇，然后使用 IR-Tree 进行空间属性与文本属性的综合检索。

算法 4　基于 FCIR-Tree 的检索算法

输入：空间索引 categorytree，查询空间矩形区域 rectangle，筛选属性 category，查询关键字 words

输出：查询结果集和 resultSet

(1) tree Collections←∅;//分类条件查询结果集合

(2) resultSet←∅;//查询返回结果

(3) tmpList←∅;

(4) tree Collections ← categorytree. search (category);//查询簇集合

(5) if tree Collections. size=0 then

(6) return resultSet;

(7) else

(8) for each IR Tree∈tree Collections do

(9) tmpList ← IRTree. search (rectangle, words);

(10) result. add (tmpList);

(11) end for

(12) return resultSet

(13) end if

FCIR-Tree 查找相对简单。详细算法如算法 4 所示。首先根据检索条件的划分属性计算相应的分类标识，并根据该标识去 FCIR-Tree 中搜索符合表示的簇，得到簇集合(4 行)。若簇集合为空，则直接结束算法。若簇集合不为空，则遍历簇集合中的所有 IR-Tree，进行空间－文本检索，并将符合条件的空间对象加入结果集(8～12 行)。

FCIR-Tree 索引与 HCIR-Tree 索引相比，主要有 2 个优点：(1)能够快速定位到相关的簇，缩小检索范围；(2)能够减少空间对象划分属性信息的存储占用空间，同一类空间对象只需要在簇集合的层面上存储一次划分属性信息就行了。不仅加快了索引检索速度，也同时减小了内存占用。

3 实验与分析

3.1 实验环境

本文采用通过百度地图 APi 获取的北京市 POI 信息作为实验数据，数据量大小为 274 MB，共 570 331 条记录，每条记录记录了一个 POI 的详细字段信息。实验环境为 WIN10 操作系统，处理器为 Intel（R） Core（TM） i5-6500，主频 3.20 GHz，内存大小为 4 GB。所有实验程序采用 JAVA 语言实现。

表 1 描述了实验数据 POI 的详细字段信息。一条真实记录如下："e3e2cd9d122349fac7f50dfe，北京种善堂大药店，医疗；药店；hospital，39.707158，116.384456"。"Attribute_1"是对当前 POI 的 1 级分类描述，根据 POI 所属行业划分。"Attribute_2"从属于"Attribute_1"，是对"Attribute_1"的 2 次细划分。"Attribute_3"相当于对当前 POI 的 1 个标签描述。实验中所选取的划分属性均为 POI 数据固有属性，根据同一属性具有的不同取值来进行划分。

表 1 POI 数据字段信息

Tab. 1 Detail information of POI data

ID	Name	Attribute_1	Attribute_2	Attribute_3	Location

3.2 算法性能测试与评价

将 FCIR-Tree 索引以及 HCIR-Tree 索引与 IR-Tree 索引进行实验比较。由于使用相同的检索条件和评分函数，所以通过 3 种索引进行的检索都具有相同的检索结果。实验主要对比评价 3 种索引的性能。主要从以下 3 种指标来衡量 3 种索引的性能：索引建立时间、索引检索耗时和划分属性个数对索引检索时间的影响。

理论上，假设空间对象数据量为 N，划分属性个数为 C，每一个划分属性 C_n 具有 W 个不同属性值。IR-Tree 本质是 1 颗 R 树，时间复杂度与 R 树类似，并且还要在 R 树检索的基础上对候选结果集进行 2 次遍历，匹配筛选属性。HCIR-Tree 也是在 R 树上进行的改造，但是较 IR-Tree 的检索方案，HCIR-Tree 在检索过程中就已经利用筛选属性加快了检索的效率，并且也不需要对结果集进行再次筛选。FCIR-Tree 先将数据集进行分类，检索时只需先进行筛选属性的匹配，定位到对应数据簇，立刻将数据量缩小为 $\frac{N}{\prod_{1}^{n} C_n}$，理论上检索速度明显优于 IR-Tree 与 HCIR-Tree，而 IR-Tree 与 HCIR-Tree 的性能有待实验验证。

在索引结构方面 IR-Tree 是最简单的。FCIR-Tree 也是在 IR-Tree 的基础上建立了分类树，保存了多棵 IR-Tree。HCIR-Tree 是 3 种之中最复杂的索引结构，每个节点都要同时保存 3 种属性信息。图 4 显示了随着数据量增大，不同索引建立耗时情况。

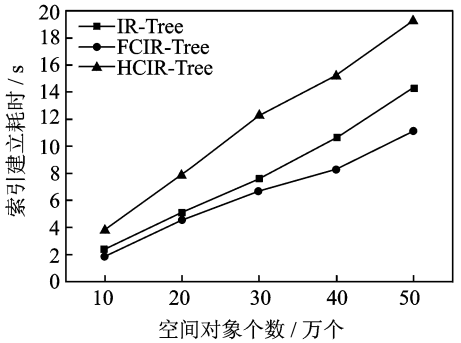

图 4 3 种索引建立时间比较

Fig. 4 Comparison for creation time of three indexes

由图 4 可以发现，3 种索引随着数据量的增加，建立耗时都有相对的增加且差距不是太大。整体上 FCIR-Tree 的建立耗时优于 IR-Tree 的建立耗时优于 HCIR-Tree 的耗时。由上面分析可知，HCIR-Tree 在每个节点都保存了子节点所有信息的汇总，所以每当插入 1 个新的空间对象之后，从此节点向上的父节点全部都需要更新，一直到 HCIR-Tree 的根节点。也就是说每次插入 1 个空间对象，就有 1 条从叶子节点到根节点的路径需要由下到上更新，将耗费大量的时间。而 FCIR-Tree 解决了要频繁更新的问题。对于每个空间对象，首先寻找其对应的簇，然后只要更新该簇内的 IR-Tree 就行，其他的簇不需要采取任何操作。

图 5 给出了不同数据量下 3 种检索用时情况。可以看出传统的 IR-Tree 索引在数据量增大之后检索时间几乎是线性增加的。而 FCIR-Tree 与 HCIR-Tree 虽然也是线性增加，但是幅度明显小于 IR-Tree 索引，检索耗时几乎没有太大的变化。IR-Tree 的主体结构是 1 棵 R 树，在数据量增大之后检索效率将会明显降低，而且由于不能解决划分属性的检索问题，只能将结果集 2 次遍历得到最终结果，也是非常耗时的。相比之下，HCIR-Tree 虽然因为在每个节点都同时保存了 3 种属性而提高的复杂性，但是在检索的时候，因为每个节点必须同时满足 3 种条件才继续往下

搜索，剪枝的效率也大大提高，提前过滤掉不满足条件的结果对象。FCIR-Tree 是 3 种索引中效果最好的，因为该索引只需要通过 1 次划分属性条件的判断，便能直接确立对应的结果簇集合，将检索数据量大大缩小接着通过二级索引 IR-Tree 快速检索空间属性和本文属性。

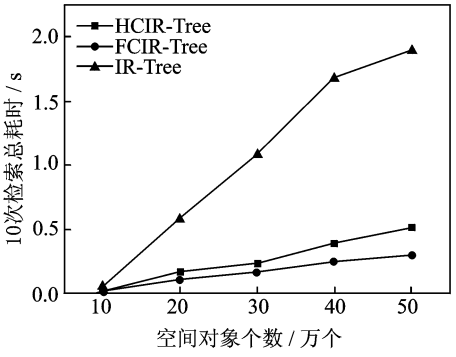

图 5　3 种索引检索时间比较

Fig. 5　Comparison for retrieval time of three indexes

图 6 分析了划分属性个数对检索时间的影响。分类数是指使用了多少个划分属性条件来建立索引。本文通过建立具有不同划分属性条件个数的索引来比较划分属性对检索效率的影响。

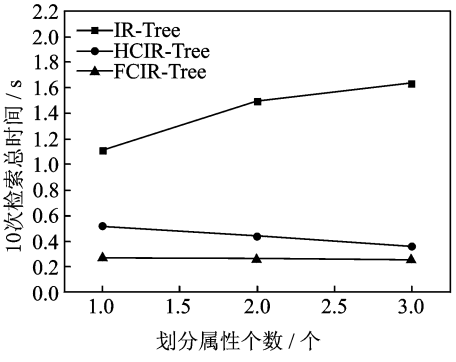

图 6　划分属性个数对 3 种检索效率的影响

Fig. 6　Effect of the number of dividing attributes on three indexes

由图 6 可以发现，随着划分属性条件个数的增加，传统的 IR-Tree 检索时间有所增加，而本文提出的 FCIR-Tree 和 HCIR-Tree 随着划分属性条件数的增加检索时间相对降低。传统的 IR-Tree 检索空间属性和文本属性，然后对结果集进行遍历匹配划分属性，随着划分属性条件数的增加，匹配时间反而略微上升。这是因为划分属性条件个数越多，对每个对象的判断次数也越多，需要同时判断所有划分属性是否都满足检索条件。HCIR-Tree 随着划分属性个数的增加，检索时间反而减少，这是因为该索引能够较好地利用的划分属性进行搜索剪枝，尽早地略去不符合划分属性的节点，减小搜索范围，快速定位候选结果集。FCIR-Tree 也同样具有划分属性条件数越多，检索效率越快的特性。因为划分属性越多，分类后得到的簇就越精细，规模越小，检索速度越快。

通过对比实验分析，发现针对具有划分属性和文本属性的空间数据，FCIR-Tree 在检索效率和索引建立时间方面都明显优于 HCIR-Tree 和 IR-Tree。FCIR-Tree 利用划分属性，能够快速低定位结果候选集，从而大大缩小检索时间。HCIR-Tree 虽然也利用了划分属性来缩小候选集，检索速度也很快，但是由于数据结构较为复杂，建立过程将会耗去大量的时间，同时也存在数据的冗余存储。而 IR-Tree 由于不能够处理划分属性，速度方面明显劣于本文提出的两种属性。所以实验证明 FCIR-Tree 是面向空间属性，文本属性和划分属性的空间对象检索的较优索引结构。

4　结束语

本文提出了两种空间索引 HCIR-Tree 和 FCIR-Tree。HCIR-Tree 将原有 IR-Tree 进行扩展，改进节点结构使之可以同时检索多重属性。FCIR-Tree 首先利用空间对象的划分属性对其分类，然后对每 1 个簇建立 IR-Tree。实验表明，与传统 IR-Tree 相比，HCIR-Tree 和 FCIR-Tree 在检索效率方面是有优势，并且随着划分属性个数的增加，效率反而越高。

参考文献：

[1] ZHOU Y, XIE X, WANG C, et al. Hybrid index structures for location-based web search[C]//International Conference on Information and Knowledge Management. [S. l.]: ACM, 2005:155-162.

[2] GUTTMAN A. R-trees: A dynamic index structure for spatial searching[C]//ACM SIGMOD International Conference on Management of Data. [S. l.]: ACM, 1984:47-57.

[3] FELIPE I D, HRISTIDIS V, RISHE N. Keyword search on spatial databases[C]//International Conference on Data Engineering. [S. l.]: IEEE, 2008: 656-665.

[4] CHEN Z, CONG G, ZHANG Z, et al. Distributed publish/subscribe query processing on the spatio-textual data stream[C]//IEEE, International Conference on Data Engineering. [S. l.]: IEEE, 2017: 1095-1106.

[5] XIE X, LIN X, XU J, et al. Reverse keyword-based location search[C]//International Conference on Data

Engineering. [S. l.]: IEEE, 2017:375-386.

[6] LIN Xin, XU Jianliang, HU Haibo. Reverse keyword search for spatio-textual top-k, queries in location-based services [J]. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(11): 3056-3069.

[7] ZHAO J, GAO Y, CHEN G, et al. Reverse top-k geo-social keyword queries in road networks[C]//International Conference on Data Engineering. [S. l.]: IEEE, 2017:387-398.

[8] BORUTTA F, NASCIMENTO M A, NASCIMENTO J N. Reverse k-nearestneighbor schedules in time-dependent road networks[J]. International Conference on Advances in Geographic Information Systems, 2015,27(11):1-10.

[9] GAO Y, QIN X, ZHENG B, et al. Efficient reverse top-k Boolean spatial keyword queries on road networks[J]. IEEE Transactions on Knowledge & Data Engineering, 2015,27(5):1205-1218.

[10] CAI G. GeoVSM: An integrated retrieval model for geographic information[C]//International Conference on Geographic Information Science. [S. l.]: Springer-Verlag, 2002:65-79.

[11] YU B, CAI G. A query-aware document ranking method for geographic information retrieval[C]//4th ACM Workshop on GeographicalInformation Retrieval. Lisbon, Portugal:[s. n.], 2007:49-54.

[12] HU Y H, GE L. Learning ranking functions for geographic information retrieval using genetic programming[J]. Journal of Research & Practice in Information Technology, 2009, 41(1):39-52.

[13] MARTINS B. Learning to rank for geographic information retrieval[C]//The Workshop on Geographic Information Retrieval. Zurich, Switzerland:[s. n.], 2010:21.

[14] SINGHAL A. Modern information retrieval: A brief overview[J]. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2001, 24(24):35-43.

[15] 周傲英, 周敏奇, 钱卫宁, 等. 大规模分布式系统中的多属性查询处理[J]. 计算机学报, 2008,31(9): 1563-1572.
ZHOU Aoying, ZHOU Minqi, QIAN Weining, et al. Complex query processing in large-scale distributed system[J]. Chinese Journal of Computers, 2008, 31(9):1563-1572.

(编辑:刘彦东)