

引用格式:金海,郝晓丽,牛保宁. 结合否定关键词的空间关键词查询[J]. 微电子学与计算机,2021,38(9):54-60. [JIN H, HAO X L, NIU B N. Combination of negative keywords spatial keyword query [J]. Microelectronics & Computer, 2021, 38 (9):54-60.]

结合否定关键词的空间关键词查询

金 海,郝晓丽,牛保宁

(太原理工大学 信息与计算机学院,山西 晋中 030600)

摘 要: 面向个性化约束的空间关键词查询是数据库查询领域的热点问题,其中快速性和匹配性是衡量此类查询优劣的核心问题.传统空间关键词范围查询无法匹配除地理位置和关键词信息以外的带有个性化约束条件下的查询,且大多数二维空间下的索引结构的构建更新速度和查询效率较低.针对上述问题,提出了一种带否定关键词约束(即用户不喜欢的关键词)的查询模式.采用 Geohash 字符串表示兴趣点对象,对字符串排序后构建 B+树作为二叉树的叶节点,通过二叉树过滤带否定关键词的对象,构建了基于 Geohash 的混合索引结构 BGIB-Tree.在此基础上,依靠 Geohash 编码的递归性,设计了前缀匹配搜索算法.以区域编码和对象编码前缀匹配为剪枝策略,快速找到满足空间约束的兴趣点,最后在倒排索引中双向搜索即可完成查询.通过与 IR-Tree 和 BIR-Tree 对比,在真实数据集上对 BGIB-Tree 的构建时间与相关参数对查询算法的影响做出验证,实验证明结果表明在索引构建时间上减少 30%,算法查询效率提高 29%.

关键词: 否定关键词;空间关键词范围查询;个性化约束;混合索引;Geohash;前缀匹配

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2021)09-0054-07

Combination of negative keywords spatial keyword query

JIN Hai, HAO Xiaoli, NIU Baoning

(College of Information and Computer, Taiyuan University of Technology, Jinzhong 030600, China)

Abstract: Spatial keyword query oriented to personalized constraints is a hot issue in the field of database query, in which the rapidity and matching are the core issues to evaluate the merits of such query. The traditional spatial keyword range query is unable to match the query with personalized constraints except geographical location and keyword information, and the construction and update speed and query efficiency of most index structures in two-dimensional space are low. Aiming at these problems, puts forward a constraint with negative keywords (i. e., users don't like keywords) query model, USES the Geohash string representation point of interest object, built after the string sorting B+Tree as a binary Tree leaf nodes, through the binary Tree to filter object with negative keywords, to build a hybrid index structure based on Geohash BGIB-Tree. On this basis, the prefix matching search algorithm is designed based on the recursion of Geohash encoding. The pruning strategy is based on the region encoding and the object encoding prefix matching to quickly find the points of interest that meet the spatial constraints. Finally, the query can be completed by two-way search in the inverted index. By comparing with IR-Tree and BIR-Tree, the influence of BGIB-tree's construction time and related parameters on the query algorithm was verified on the real data set. The experimen tresults proved show that the index construction time was reduced by 30% and the query efficiency of the algorithm was improved by 29%.

Key words: negative keywords; spatial keyword range query; personalization constraint; hybrid index; geohash; prefix matching

收稿日期: 2020-12-01; 修回日期: 2021-01-02

基金项目: 山西省应用基础研究项目(201901D111100);山西省重点研发计划(201903D121132)

1 引言

空间关键词查询(Spatial Keywords Query, SKQ)是空间查询与文本搜索的结合,它不仅要求快速地获取对象的空间位置信息,并尽可能保证所获取对象在文本信息上的匹配性^[1-2]。其中,匹配性是保证在空间范围上的匹配,保证对象在查询点附近,以及对象文本信息要与查询关键词完全匹配;而快速性是要求相关索引构建时间短,查询效率高,能够满足实时查询需要。文献[3]研究了基于线索的空间文本查询,通过设计线索与上下文之间相似度的度量方式,使得与线索完全匹配的对象都可被返回,然而无法匹配带有否定关键词约束的对象。快速性方面,现有方法根据索引划分策略的不同可以将其分为基于数据划分^[4]和基于空间划分^[5]两类。基于数据划分主要以 R-Tree 索引及其变体为代表。文献[6]提出了 IR-Tree,与 R-Tree 索引相比,它使得空间剪枝和文本过滤能够在查询处理中同时被执行,能够快速排除不相关或者相关性较低的对象,可是在面对大量数据搜索时它的查询效率会显著降低。文献[7]率先在查询中增加了用户排斥属性,通过构建 BIR 树来索引兴趣点数据,在每一个节点都设计两种倒排文件用来分别记录两种类型关键词,然则加大了每个节点的存储空间。而基于空间划分则根据经纬度或网格来划分空间^[8-9]。文献[9]提出了 binaryquadtree,能够保持兴趣点在空间位置上的邻近性,但是当空间文本对象分布不均匀时,随着对象的插入会使得 quadtree 深度不断加深造成树的结构严重不平衡,导致查询效率急剧下降。

基于此,本文提出了一种新的查询,即结合否定关键词约束的空间关键词查询,其中否定关键词指用户在兴趣点文本信息中排斥的关键词。该查询要求返回结果中的每一个兴趣点都完全包含查询中携带的关键词并且不包含任意一个否定关键词,属于完全匹配类型。而在快速性方面,提出一种支持否定关键词约束、空间位置以及文本约束的三维信息混合索引 BGIB-Tree(Binary Geohash Inverted file B-Tree),并提出基于此索引的前缀匹配查询算法,采用字符串前缀匹配原则作为算法剪枝策略。实验结果表明,BGIB-Tree 与前缀查询算法在构建时间和查询效率上均具有良好的

表现。

2 基于否定关键词的一维混合空间结构设计

2.1 问题及解决思路

在现有查询框架返回对象中仍包含了用户不喜欢的关键词,而当前方法无法排除这些对象,造成了冗余查询。因此,目前空间文本查询依然存在以下问题:

(1)在查询时只能查询需要的关键词信息,而无法解决带有否定关键词约束的空间关键词查询。

(2)空间关键词的索引大多建立在二维空间,以最小外接矩形或网格代表对象,同时每个节点都关联一个倒排文件。随着兴趣点的属性和数量不断增加,导致二维空间下的节点更新占用更多内存空间,查询时 I/O 开销较大。

(3)以数据划分为基础的方法通常所有的兴趣点都存储在一颗 R 树内,当节点过多时树的深度增加,导致在大数据量时查询效率不高。而以 quad 树或网格等划分空间的方法会在兴趣点密度不均时造成结构失衡。

为了解决上述问题,本文的这种新的查询方法创新点如下:

(1)针对现有框架无法处理带有否定关键词约束查询的问题,提出了一种结合否定关键词的空间文本查询框架,高效地解决了带否定关键词的查询问题。

(2)针对目前对象只能以最小外界矩形或网格来表示的问题,本文以 Geohash 编码技术为基础,利用 Geohash 编码的全球唯一性,将对象以字符串的形式表示。同时,由于 B+ 树在非叶子节点中存储的信息较少,读入内存时可以一次性读取更多的兴趣点信息,因此本文以 B+-Tree 为载体,构成 GB-Tree,可以有效地对数据结构降维。

(3)针对目前空间关键词索引在大数据量下兴趣点密度不均导致结构失衡的问题,本文以 GB-Tree 作为二叉树的叶节点,构建 BGIB-Tree 一维索引来代替二维空间下的索引,此外设计了相应算法来使查询更加高效。

图 1 展示了基于否定关键词的查询框架。由迭代更新模块、存储管理模块和查询引擎模块三个部分构成。其中迭代更新模块用来支持索引的更新及删除功能,存储管理模块包含 POI 数据库及 BGIB-

Tree,用户可以通过查询引擎模块进行带否定关键词的空间文本查询.

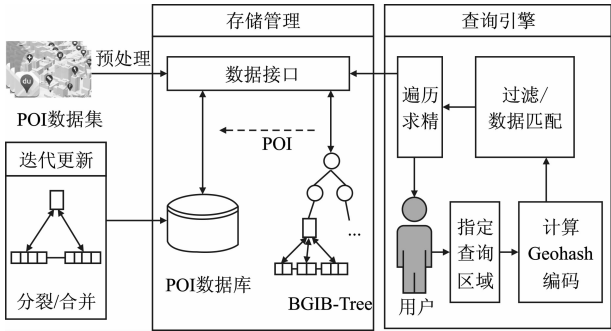


图 1 结合否定关键词的空间文本查询框架

2.2 混合索引结构设计

建立 BGIB-Tree 索引前需要对兴趣点集 S 中的数据进行预处理,为空间文本对象 o 中的每一个关键词都赋予一个权重 $w_{i,o}$,代表该关键词在一个空间文本对象 o 中的重要程度.受 Aizawa 等^[5]启发引入一种在自然语言处理领域中使用很广泛的关键词统计方法 TF-IDF 来计算关键词的权重,TF-IDF 分为基于术语频率(TF)和逆文本频率(IDF),关键词的权重即:

$$w_{i,o} = tf_{i,o} \times idf_{i,o} \tag{1}$$

$$tf_{i,o} = \frac{n_{i,o}}{\sum_k n_{k,o}} \tag{2}$$

式(2)中 $n_{i,o}$ 表示关键词 o , k_i 在对象 o 中出现的次数, $\sum_k n_{k,o}$ 表示对象 o 中关键词的总个数;

$$idf_{i,o} = \log \frac{|O|}{|\{j:k_i \in o_j\}|} \tag{3}$$

式(3)为了保证逆文本频率的普适性,使 $|O|$ 表示整个待查询数据集中空间文本对象的个数, $|\{j:k_i \in o_j\}|$ 表示在空间文本对象全体中包含关键词 k_i 的对象的个数.

对关键词的权重 $w_{i,o}$ 进行降序排序取出前 z 个权重最大的关键词对应于准备构建的前 z 层二叉树,每层二叉树代表一个关键词筛选层,在这层所有的节点中的左子树代表包含此关键词的兴趣点集,右子树反之.二叉树的叶子节点代表一个以此叶节点为根的 GIB-Tree,经过筛选后每一个兴趣点都会唯一属于一个二叉树的叶节点.将兴趣点集 S 中的对象位置信息以特定精度的 Geohash 编码表示,形成长度为 n 的字符串 LineString,以 LineString 为对象对兴趣点集 S 按照字典序排序,形成键值 key 和能独一无二表示每个兴趣点的时间戳 Timestamp,其中 Timestamp 以整形形式表示, o . K 代表关键词.由此可得一条数据的基本类型为:

$$(key,o.L_{lon},o.L_{lat},o.K,LineString,Timestamp) \tag{4}$$

具体表示形式如图 2 所示.

Key	$o.L_{lon}$	$o.L_{lat}$	$o.K$	LineString	Timestamp
0	30.235 909 1	-97.795 139 58	Indian Restaurant	9v6kn87b	1
1	30.244 859 7	-97.757 163 05	Coffee Shop	9v6kp3zm	2
2	30.254 141 5	-97.762 317 22	Residential Building(Apartment/Condo)	9v6kp7ee	3
3	30.263 774 8	-97.770 659 92	pool	9v6kpjme	4
4	30.263 4181	-97.757 596 67	Spa	9v6kpmrl	5

图 2 预处理数据的例子

每一个矩形单元视为一个容器,在这个容器中使用他们的 key 值来标记内部对象,使在空间位置上临近的对象也可以被分配在相同的存储区域当中,这样的特性将会使查询性能得到提升.按照 LineString 的顺序插入到 B+-Tree 中,并且在插入过程中最关键的是要确定节点分割的规则和中间键的选取,为了方便节点分裂将 B+-Tree 的阶数设为 5,所以在 5 阶 B+-Tree 中一个节点最少有两个 LineString,最多四个 LineString,将溢出节点从中间分隔,并且选择右边第一个键作为中间键,在 B

+-Tree 的每个叶子节点上都至少存储两个到四个 $\langle LineString, key \rangle$ 键值对,因此每个二叉树的叶节点结构如图 3 所示.

在关键词检索阶段构造位 - 关键词倒排索引结构,由于在数据中 Timestamp 的唯一性,对所有 Timestamp 建立倒排索引,并同时为这条数据中的每一个关键词 $(k_1,k_2,\cdots,k_i)$ 构建倒排列表,例如一条倒排索引的结构可以表示为:

$$Timestamp \rightarrow (k_1, key), (k_2, key), \cdots, (k_i, key) \tag{5}$$

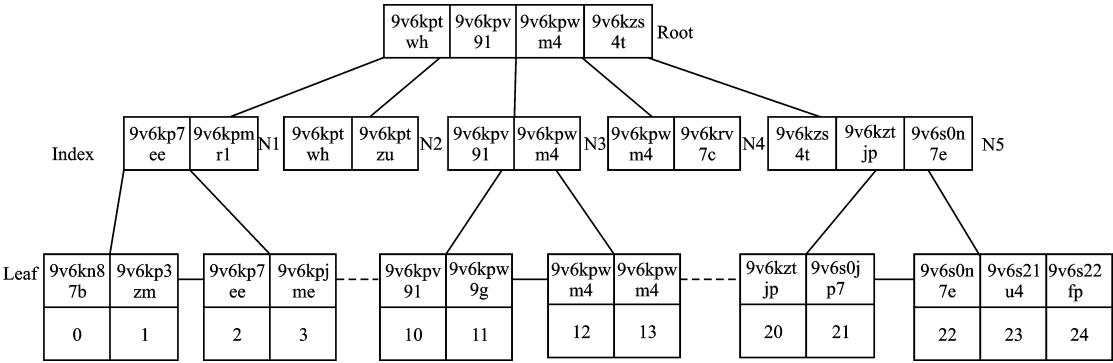


图 3 GB-Tree 索引结构

将 GB-Tree 作为二叉树叶节点并结合倒排索引就构成了 BGIB-Tree 索引结构,如图 4 所示.

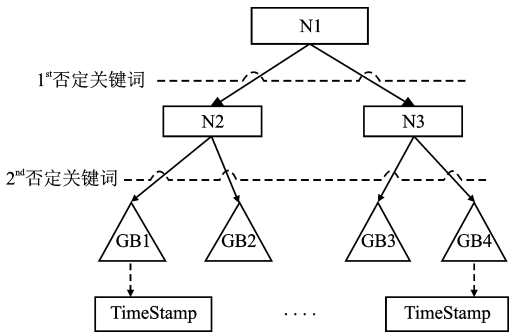


图 4 BGIB 索引结构

3 前缀匹配查询算法

3.1 算法详述

前缀匹配算法采用模糊范围精确查询的解决思路,首先确定查询的矩形区域,在区域内存在有限数量的兴趣点集,经过二叉树过滤后通过字符串匹配判断对象是否在查询范围内,然后查询倒排索引即可.

本文的查询匹配算法步骤如算法 1 所示:

算法 1 前缀匹配搜索算法

BGIB-Tree, 查询范围 R , 包括查询的位置点 Ql 和查询距离约束 Qd ; 需要的关键词 $Q.K^n$; 不需要的关键词 $Q.K^h$.

Output: 一组满足查询条件的对象集合 C .

Begin
1 root = BGIB-Tree;
2 cur = root;
3 While cur≠leaf do;
4 If $Q.K^h \supseteq \text{cur}$, 否定关键词 then;
5 cur = cur.rightChild;
6 Else cur = cur.leftChild;

7 计算 Ql 及周围八区域编码为 p 位精度 Geo-hash 字符串;
8 For 遍历 cur 子树;
9 找到最左边节点;
10 While cur.next != null do;
11 If 当前区域编码 = cur 前 p 位编码 then;
12 获取节点 Timestamp;
13 查询倒排索引交集获得 key 值;
14 If key 兴趣点满足距离约束;
15 加入集合 C ;
16 Return C ;
End

第 1 ~ 6 行在二叉树上进行查找,经过二叉树筛选会得到一组叶子节点,每一个二叉树叶子节点都包含着若干满足否定关键词条件的兴趣点,也就是 GIB-tree 的根节点.第 7 行根据查询距离约束 d 与 geohash 精度表对比选定与 d 相对应的编码长度 p ,并获取 Ql 代表区域的周围八个区域的 geohash 编码,共九个区域共同作为查询的范围约束,以保证查询的精确性.第 8 ~ 13 行通过遍历 GIB-Tree 与倒排索引获得在矩形区域内的对象.第 14 ~ 16 行筛选满足距离约束的对象加入集合 C 并返回.

其中否定关键词快速剪枝如算法 2 所示:

算法 2 否定关键词快速剪枝

Input: BGIB-Tree, 否定关键词 $Q.K^h$.

Output: GIB-Tree 根节点

Begin
1 root = BGIB-Tree;
2 If root == null ;Return null;
3 Else if root != null & $Q.K^h \supseteq \text{root}$, 否定关键词;
4 Return root;
5 Else Node leftnode = this.find($Q.K^h$, leftChild);

```
6      Node rightnode = this.find(Q.Kh, right-
      Child);
7      递归查询左右子树;
End
```

实际生活中,有 20%的关键词基本可以包含 80%兴趣点,通过否定关键词的快速剪枝可以精简大部分无关数据,在二叉树中查找这些否定关键词,包含否定关键词的节点将被剪枝,不会继续向下遍历查询。

3.2 算法分析

以最差时间复杂度为标准,最先采用先序遍历二叉树的最差时间复杂度为 $O(n)$,令 m 等于叶节点中的对象个数, n 等于在 $B+$ 树中的对象总数,在其中查询字符串的复杂度与 $B+$ 树相同为 $O(\log_m^n)$,并且设在倒排文件中满足查询要求的时间戳个数为 a ,总的时间戳个数为 b ,所以在倒排索引中查询复杂度为 a 。综上所述,本文所提出查询算法的时间复杂度为:

$$O\left(n + \log_m^n + \frac{a}{b}\right) \tag{6}$$

4 实验

4.1 实验环境设置

实验在 Lenovo-PC y50 上进行,配置如下: CPU 为 Intel(R) Core (TM) i5-4210H CPU @ 2.90 GHz、内存为 8GB,WINDOWS 10 的操作系统,IntelliJ IDEA 2019.1×64 作为集成开发环境.实验数据采用 Foursquare 网站上真实开源用户签到数据集.数据集统计信息如表 1 所示,实验采用经纬度作为位置点信息,将记录的标签作为关键词信息来测试我们提出的存储模型和算法。

表 1 数据集基本统计信息

城市	兴趣点的数量	关键词的数量	平均兴趣点携带关键词数量
Austin	183 795	371 109	2.019 15
San Francisco	161 996	305 169	1.883 81

4.2 实验结果与分析

本文从匹配性和快速性两个方面衡量索引和查询算法效率。

(1)在匹配性实验中,通过设定否定关键词个数,在不对返回结果数做约束的情况下,默认需要的关键词数为 3,查询范围设定为 $0.05^\circ \times 0.05^\circ$,从图 5 中可以看出随着否定关键词数量的增加,查询结果逐渐减少却更加精确,平均在查询中每增加一个

否定关键词,平均匹配率提高 39%。但可以预见的是如果查询加入的否定约束过多,有可能会出现没有返回结果的情况。

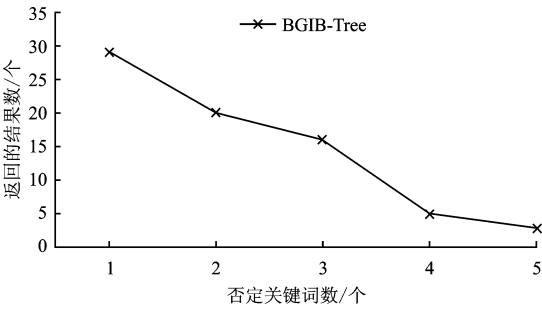


图 5 否定关键词约束对匹配性的影响

(2)在验证快速性的实验中,以 IR-Tree 和 BIR-Tree 为基线算法与本文提出的 BGIB-Tree 算法作对比,所有试验均采用两个数据集的平均值来表示.实验中返回的结果数默认为 3,需要的和否定关键词数量为 3,查询范围为 $0.1^\circ \times 0.1^\circ$ 的经纬度查询矩形。

图 6 显示了在兴趣点数量不同的情况下,各索引构建时间的大小,从图中可以看出本文索引构建时间要小于另外两种索引,平均构建时间要快 30%,这是因为在同等数据量下基于 Geohash 字符串的对比速度要比计算 R 树中最小外界矩形更快。

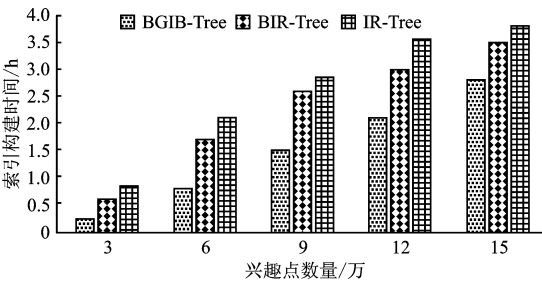


图 6 索引构建性能对比

图 7 显示了结合否定关键词的查询框架的可扩展性.可以看出,随着兴趣点数量增加 BGIB 树总的查询效率明显高于其他两种算法并且显示出较为稳定的趋势,在数据量最大的时候查询效率提高 35%。这是由于随着数据集的增大,BGIB-Tree 从叶节点链表即可直接查询,比重复遍历索引要快得多.说明 BGIB 树能更好的适应较大的数据集。

图 8 展示了在不同否定关键词数量下查询响应时间的变化情况.可以看到 BIR-Tree 和 BGIB-Tree 算法随着否定关键词数量的增加使查询效率显著提高,否定关键词数量为 3 时查询效率提高 29%,这是由于否定关键词是在二叉树中存储的,所以随着

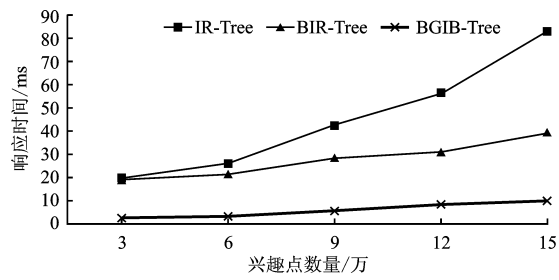


图 7 查询框架的可扩展性

二叉树层数增多,数据集被进一步划分,可以极大程度上减少搜索空间;但是后续查询效率逐渐降低,这是因为二叉树随层数增多,复杂度及存储空间呈指数级上升,导致整体索引结构更加复杂从而影响查询效率。

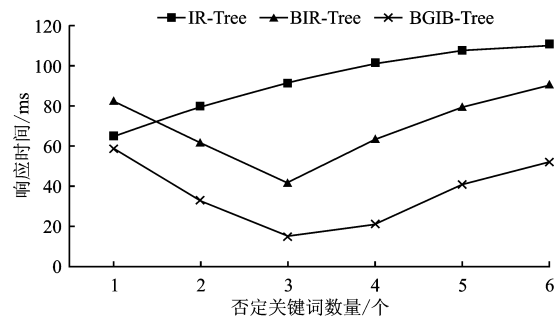


图 8 否定关键词数量对查询时间的影响

由图 9 可以看出无论是大范围查询还是小范围查询,BGIB-Tree 的查询效率均高于另外两种算法,这是由于在范围扩大时,本文通过预处理数据即可直接通过字符串比较即可快速建立索引,从而提高查询效率,在较大范围中查询时间提高 48%,验证了本文算法的优越性。

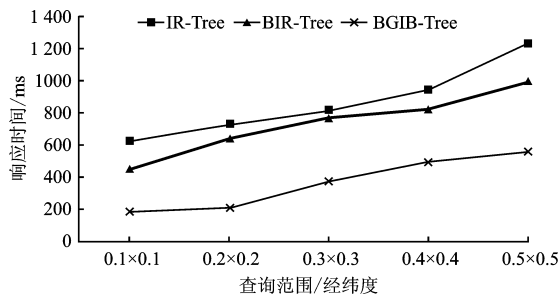


图 9 查询范围对查询时间的影响

5 结束语

本文针对传统空间关键词查询中无法有效解决带否定关键词约束查询的问题,提出了一种带否定关键词的查询模式.该方法通过 Geohash 算法对兴趣点降维处理后插入 B+ 树中,以其作为二叉树叶

节点构建 BGIB 混合索引结构,将二维空间计算转移到一维空间中,大大加快了构建速度且很好地存储和组织了世界范围内的兴趣点数据.基于 Geo-hash 编码递归特性,提出了前缀匹配算法,将经纬度计算改为字符串比较,加快了空间约束的匹配速度,降低了 I/O 损耗,能够支持空间领域中布尔范围查询及基于否定关键词查询.通过构架二叉树、GB-Tree 和倒排索引的三层过滤结构,有效地减少了搜索空间,排除了不需要的结果,提高了空间关键词查询的效率和精确度.实验证明,与 BIR-Tree 索引和传统 IR-Tree 相比,该方法拥有更快的索引构建速度和更高的检索效率.在如今海量数据爆发的情况下,更快的查询速度与更高的查询精确度无疑能为用户带来更好的体验,同时 BGIB 混合索引结构在一维空间中占用更少的内存,从而一次性可以读入更多节点,前缀匹配算法节省了 CPU 在运算过程中的算力,为空间数据库查询领域提供了更优的解决方案.在后续工作中将继续研究相关模糊语义的空间关键词查询,使算法更具普适性。

参考文献:

[1] XU JJ, CHEN J, YIN L H. Multi-objective spatial keyword query with semantics: a distance-owner based approach[J]. Distributed and Parallel Databases, 2020, 38 (3): 625-647. DOI: 10.1007/s10619-020-07283-1.

[2] ZHU L, SONG J Y, YU W R, et al. Reverse spatial visual top-k query[J]. IEEE Access, 2020, 8(8): 21770-21787. DOI: 10.1109/ACCESS.2020.2968982.

[3] LIU J L, DENG K, SUN H L, et al. Clue-based spatio-textual query[J]. the VLDB Endowment, 2017, 10(5): 529-540. DOI: 10.14778/3055540.3055546.

[4] MENG X F, LI P, ZHANG X Y. A personalized and approximated spatial keyword query approach [J]. IEEE Access, 2020(8): 44889-44902. DOI: 10.1109/ACCESS.2020.2977996.

[5] AIZAWA K, MOTOMURA K, KIMURA S, et al. Constant time neighbor finding in quadrees: An experimental result[C]//2018 3rd International Symposium on Communications, Control and Signal Processing. Saint Julian's, Malta:IEEE, 2008.

[6] LI Z S, LEE K C K, ZHENG B H, et al. IR-Tree: an efficient index for geographic document search[J]. IEEE Transactions on Knowledge and Data Engineering, 2011, 23 (4): 585-599. DOI: 10.1109/TKDE.2010.149.

[7] 屠川川. 带排斥关键字的空间关键字查询[J]. 微型电脑应用, 2015, 31(4): 19-22. DOI: 10.3969/j.issn.1007-757X.2015.04.006.

TU C C. Exclusive spatial keyword query[J]. Micro-computer Applications, 2015, 31(4): 19-22. DOI: 10.3969/j.issn.1007-757X.2015.04.006.

[8] YANG M, MA K, YU X H. An efficient index structure for distributed k-nearest neighbours query processing[J]. Soft Computing, 2020, 24(8): 5539-5550. DOI: 10.1007/s00500-018-3548-4.

[9] PARK K. A hierarchical binary quadtree index for spatial queries[J]. Wireless Networks, 2019, 25(4):

1913-1929. DOI: 10.1007/s11276-018-1661-z.

作者简介:

金海男, (1996-), 硕士研究生. 研究方向为空间数据查询.

郝晓丽(通讯作者)女, (1973-), 博士, 副教授. 研究方向为视频图像处理、情感计算、数据挖掘等.

E-mail: 951001341@qq.com.

牛保宁男, (1964-), 博士, 教授. 研究方向为大数据管理与分析、数据库系统性能管理、区块链数据管理.