软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(9):2278−2289 [doi: 10.13328/j.cnki.jos.005046]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一种对时空信息的 *k*NN 查询处理方法*

李 晨, 申德荣, 朱命冬, 寇 月, 聂铁铮, 于 戈

(东北大学 计算机科学与工程学院,辽宁 沈阳 110819)

通讯作者: 李晨, E-mail: neu_LiChen@163.com

摘 要: 互联网上每天都会产生大量的带地理位置标签和时间标签的信息,比如微博、新闻、团购等等,如何在众多的信息中找到在时间和空间地理位置上都满足用户查询需求的信息十分重要.针对这一需求,提出了一种对地理位置和时间信息的 *k* 近邻查询(ST-*k*NN 查询)处理方法.首先,利用时空相似度对数据对象的地理位置变量和时间变量进行映射变换,将数据对象映射到新的三维空间中,用三维空间中两点之间的距离相似度来近似代替两个对象之间实际的时空相似度;然后,针对这个三维空间设计了一种 ST-Rtree(spatial temporal rtree)索引,该索引综合了空间因素和时间因素,保证在查询时每个对象至多遍历 1 次;最后,在该索引的基础上提出了一种精确的 *k* 近邻查询算法,并通过一次计算确定查询结果范围,从而找到前 *k* 个结果,保证了查询的高效性.基于大量数据集的实验,证明了该查询处理方法的高效性.

关键词: 地理位置;时间;时空相似度;索引;*k* 最近邻查询

中图法分类号: TP311

中文引用格式: 李晨,申德荣,朱命冬,寇月,聂铁铮,于戈.一种对时空信息的 *k*NN 查询处理方法.软件学报,2016,27(9): 2278−2289. http://www.jos.org.cn/1000-9825/5046.htm

英文引用格式: Li C, Shen DR, Zhu MD, Kou Y, Nie TZ, Yu G. *k*NN query processing approach for content with location and time tags. Ruan Jian Xue Bao/Journal of Software, 2016,27(9):2278−2289 (in Chinese). http://www.jos.org.cn/1000-9825/5046.htm

*k*NN Query Processing Approach for Content with Location and Time Tags

LI Chen, SHEN De-Rong, ZHU Ming-Dong, KOU Yue, NIE Tie-Zheng, YU Ge

(School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China)

Abstract: Large amounts of content with location and time tags are generated every day on webs such as microblog, news, and group-buying. Thus, it is important to find top-*k* results that satisfy users' temporal and spatial requirements from the contents. In this paper, a novel *k*NN query (called ST-*k*NN query) processing approach is proposed for content with location and time tags. First, location variables and time variables of data objects are transformed via temporal & spatial similarity in order to map data objects to a new three-dimensional space. Next, the spatial similarity between two objects in the three-dimensional space is used to approximate the actual temporal & spatial similarity. Then, a new index called ST-Rtree is designed in this three-dimensional space. The index combines location variables & time variables, and ensures every object is traversed no more than once. At last, an exact *k*NN query algorithm is proposed. The region is determined by computing only once to find top-*k* results, which guarantees high-efficiency in the query processing. Experiments on large datasets demonstrate that the presented query processing approach is very efficient.

Key words: location; time; temporal & spatial similarity; index; *k* nearest neighbor query

随着互联网上信息数量的不断增长,人们呈现出了对带有地理位置标签的信息产生的查询需求,例如查找

* 基金项目: 国家自然科学基金(61472070); 国家重点基础研究发展计划(973)(2012CB316201)

Foundation item: National Natural Science Foundation of China (61472070); National Basic Research Program of China (973) (2012CB316201)

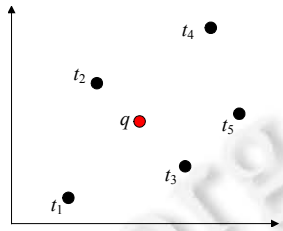
收稿时间: 2015-10-09; 修改时间: 2016-01-12, 2016-02-22; 采用时间: 2016-04-25

位置距离自己比较近的微博、看看身边发生的一些趣事、查找距离某个地点比较近的餐厅的团购信息等,因为考虑这些信息的空间位置的服务更符合人们的现实需求.近几年来,关于空间文本信息查询方面的研究已经成为数据库方向的研究热点[1−3],对空间文本信息的查询是在对文本信息查询的基础上考虑地理位置因素.但是在很多情况下,用户往往需要查询满足一定时间约束的信息,比如用户可能需要查找某个特定时间段内的某些商家的团购信息,或浏览具有一定时效性的关于某地区的新闻报道等,此时,所需的查询信息不仅需要满足一定的地理位置约束,同时还要满足一定的时间约束.通过上面的描述我们可以看出:根据这些信息包含的地理位置属性和时间属性来查询过滤大量的团购、新闻等信息,更符合用户的查询需求.如何从不断累积、持续增长的时空信息数据中快速返回用户所需的信息,成为迫切需要解决的问题.本文针对满足特定的时空约束条件的信息这一需求,提出了一种基于时空信息的 k 近邻查询处理方法.并通过利用时空相似度对数据对象的时空信息进行映射变换,设计了一种新型的三维空间的 ST-Rtree 索引,并设计了相应的查找算法,实现了对信息的时空标签进行过滤和查找,从而找到用户需要的团购、新闻等带有时空属性的信息.本文提出的索引和查找方法极大地提高了对时空信息的查询效率,能够满足用户高效性和精确性的查询需求.

图 1(a)中列出了部分美食团购信息的地理位置和时间,其中,t_i 表示发布团购信息的商家,location 一栏表示的是商家的地理位置坐标,time 一栏表示的是该商家的团购开始的时间.图 1(b)展示的是这些商家的地理位置分布.假设查询信息 q 的查询时间是 10:00,查询位置如图所示,查询内容是找到离查询位置和查询时间都比较近的一些团购信息.当 k=2 时,如果仅考虑地理位置这一因素,那么距离用户比较近的商家是 t_2 和 t_3;但如果同时考虑时间因素,商家 t_2 由于其团购开始时间是 16:30,离用户所要求的时间 10:00 相差太远,使得其很难满足用户要求.所以,按照一定的权重分配时间因素和空间因素的重要性,从而综合考虑时间因素和空间因素,最后得到的结果是商家 t_3 和 t_5.

Location		Time
t_1	(2,1)	12:00
t_2	(3,5)	16:30
t_3	(6,2)	11:30
t_4	(7,7)	14:30
t_5	(8,4)	11:00
q	(4.5,3.5)	10:30

(a) 地理位置和时间



(b) 位置分布

Fig.1 An example of ST*k*NN query

图 1 一个 ST*k*NN 查询的例子

本文首先利用两个数据对象之间的相似度对数据对象的空间地理位置变量和时间变量做映射变换,然后给映射变换后的数据对象建立 ST-Rtree 索引[4],并提出一种在 ST-Rtree 索引上的精确查找算法,找到 top-k 个离查询点最近的对象.本文的主要贡献点有:

(1) 给出两个对象之间的时空相似度的计算方法;利用对象之间的时空相似度对对象的地理位置变量和时间变量进行映射变换,从而将它们映射到三维空间中.这样可以用三维空间中两点之间的距离相似度来近似代替两个对象之间实际的时空相似度;

(2) 提出了混合索引 ST-Rtree 索引[3],ST-Rtree 索引是给映射变换后的三维空间中的数据对象建立一个空间和时间混合的三维 Rtree 索引,该索引有效地结合了数据对象的空间变量和时间变量,能够保证在查询处理时,每个数据对象至多遍历一次;

(3) 设计一种在 ST-Rtree 索引上的精确查找算法,即,根据相似度查找到 top-k 个在空间和时间上距离查询信息都很近的时空对象.该精确查找算法能够通过一次计算找出包含查询结果的区域范围,从而找到 top-k 个查询结果,极大地提高了查询效率;

(4) 设计了对比实验,通过测试大量数据集,验证了本文的查找方法的高效性.

本文第 1 节介绍相关工作.第 2 节介绍一些问题的定义,比如时间和空间的混合相似度的计算、ST-*k*NN 查询的定义等.第 3 节介绍对数据对象的归一化处理和变换以及混合索引的构建.第 4 节介绍 ST-*k*NN 查询算法的设计和实现.第 5 节介绍对比实验,并且通过实验验证本文的查询方法的高效性.

1 相关工作

随着移动通信网络和计算机网络的有效结合,为用户提供了很多便捷的基于位置的服务[21].同时,随着各种社交网站的发展,互联网上每天都会产生大量的文本信息,比如微博,各种关于文本信息中的关键字查询已经被广泛研究[5].于是,将空间地理位置和关键字结合的查询服务被越来越多的用户所接受,因为既考虑空间位置信息又考虑关键字信息的服务更符合人们的现实需求.2010 年,Cao 等人[6]提出了 L*k*PT(location-aware top-*k* prestige-based text retrieval)查询,该查询既考虑文本信息的相关性,又考虑了对象的空间信息.近些年来,关于空间文本的查询已经成为数据库领域新的研究热点[7].然而,有时仅仅包含空间和文本的查询是不能够满足用户需求的.比如,新闻要求有时效性,并且可能每天都会产生大量的信息,所以一般用户都会查询最新的或者离他指定的时间最近的新闻.于是,在空间文本的基础上加入时间因素,可能更符合用户的查询需求.同时,网上的一些对象如微博和新闻信息都是带有时间标签,这使得对时空文本的查询变得可能.

2013 年,Magdy 等人[8]提出的 Mercury 系统就是在空间关键字查询的基础上加入时间因素,它是关于微博基于内存限制的实时的关于时间和空间的查询系统.2014 年,Skovsgaard 等人[9]提出了可扩展的时空关键字的 top-*k* 查询,该查询返回给用户在用户指定的时间和空间内产生的最频繁出现的 *k* 个关键字.2015 年, Chen 等人[10]提出了考虑时间因素的空间关键字的 top-*k* 的发布和订阅.文章中提出了一种新的查询类型,即 top-*k* 的订阅查询,该查询在考虑空间本文相似度的基础上还考虑了时间因素.

本文针对信息的时间和空间信息进行过滤和查找,从而找到距离用户给出的时间和空间位置都比较近的一些信息对象.为方便查找,需要给数据对象的空间和时间建立索引.对于时空数据索引的研究有很多[11],常见的有 RT-tree[11],3DR-tree[12],MR-tree[11],HR-tree[13]等.RT-tree 索引将数据对象的时间参数和空间参数分开存储,多面向移动的数据对象;3DR-tree 索引对范围查询的效率较高;MR-tree 索引和 HR-tree 索引是给时空数据建立两级索引,先过滤时间,再查询,它们对带有时空标签信息的 *k*NN 查询效率都不是很高.在移动数据对象查询中,由于移动对象的空间位置是随着时间变化而变化的,因此也涉及到对时间和空间信息的查询.2001 年,Song 等人[14]提出了利用已有的数据信息来解决对移动对象 *k* 近邻查询的一些方法,代替给数据对象的时间因素和空间因素建立有效的混合索引,仅利用 R-tree 及其变种给数据对象建立空间索引.2002 年,Lazaridis 等人[15]将时间因素单独作为一个维度和空间因素共同建立多维 R-tree 来给移动对象建立索引,但是由于时间和空间的度量单位不同,所以在查询的时候还是要分开考虑两个因素,分别过滤,效率不是很高.Benetis 等人[16]提出了 TPR-tree 索引,在空间 R-tree 索引的基础上加入速度向量,根据位置函数和时间来确定移动对象的位置.但 TPR-tree 索引本质上只是对空间位置的索引,查询时也只是根据移动对象的空间位置查询,时间不作为查询过滤条件.本文利用相似度计算公式对时间因素和空间因素做映射变换,规定统一的衡量相似度的标准,从而建立混合的三维 ST-Rtree 索引,提高查询效率.

2 问题定义

假设 *D* 是对象数据集,对于任意属于集合 *D* 的对象 *o*,*o* 可以表示成⟨*o.t*,*o.l*⟩,其中,*o.t* 表示时间标签, *o.l*(*x*,*y*)表示对象 *o* 的地理坐标.下面首先给出对带有时空信息的 *k* 最近邻查询(ST-*k*NN 查询)的概念.

对于空间相似度 $SimSpatial(o_1,o_2)$的度量,本文采用欧氏距离来衡量两个对象之间的相似度,计算公式见公式(1).

$$SimSpatial(o_1,o_2)=\frac{\sqrt{(o_1.x-o_2.x)^2+(o_1.y-o_2.y)^2}-d_{\min}}{d_{\max}} \tag{1}$$

其中,$\sqrt{(o_1.x-o_2.x)^2+(o_1.y-o_2.y)^2}$ 表示两个对象之间的欧式距离,$d_{\min}$ 表示集合 D 中任意两个对象之间的最小距离,$d_{\max}$ 表示集合 D 中任意两个对象的最大距离.应用公式(1)可以在已知任意两个对象之间的距离时,计算出它们的空间相似度.

对于时间相似性 $SimSpatial(o_1,o_2)$的度量,计算公式见公式(2).

$$SimTemporal(o_1,o_2)=\frac{|o_1.t-o_2.t|}{T} \tag{2}$$

其中,T 表示一个合理的时间段.

给出查询 $q(q.t,q.l)$,如果查询对象 o 是带有特别时间约束的像团购这类的信息,那么要考虑团购的开始时间和用户的查询时间的前后顺序.在公式(2)的基础上改进,得到公式(3).

$$SimTemporal(o,q)=\begin{cases}\dfrac{o.t-q.t}{T}, & o.t\geqslant q.t\\ \infty, & o.t<q.t\end{cases} \tag{3}$$

即:商家发布的团购开始时间在用户的查询时间之前,该团购信息可以满足用户需求;否则,该信息是无用的.

定义 1(ST-*k*NN 查询). 给出查询点 q,ST-*k*NN 查询是在数据集 D 中找到离查询点 q 在时间和空间上都最近的 k 个对象.对于“最近”,我们根据文献[17−19]给出公式(4)的综合时间空间的相似度量标准.

$$Sim=\alpha SimSpatial(o,q)+(1-\alpha)SimTemporal(o,q) \tag{4}$$

其中,α是一个调节权重的参数.

为了解决 ST-*k*NN 查询,本文提出一种混合索引 ST-Rtree,即:给出数据集 D,通过对数据集中数据对象的地理位置坐标和时间标签进行映射变换,将数据对象映射到三维空间中,从而建立一个三维的混合 Rtree 索引,即为 ST-Rtree 索引.

通过映射变换,可以用三维空间中任意两个对象之间的空间相似度来代替实际的时空相似度,这样就可以将空间相似度看成是空间距离,从而根据对象之间的空间相似度来建立索引,保证离得近的两个对象划分到 ST-Rtree 的同一个节点中.建立混合索引 ST-Rtree 后,设计合理的 ST-Rtree 的查询算法来实现 ST-*k*NN 查询.下面将详细介绍 ST-Rtree 索引的构建过程.

3 构建索引

查找时,提前给数据建立索引会提高查询效率.但是如果给每一个因素都单独建立索引,那么查询时要遍历所有的索引,多次遍历的时间代价会比较大.如果能综合考虑各个因素,从而建立混合索引,会很大程度上提高查询效率.本文提出一种高效的混合索引 ST-Rtree,下面将详细介绍 ST-Rtree 索引.

3.1 数据归一化处理

为了提高查询效率,我们给时间变量 t 和空间变量 x,y 建立统一的索引,这样在查询时,只需要遍历一遍数据对象.首先,将它们映射到三维空间中的三个维度上;然后,统一建立 R 树索引.由于空间和时间是两个不同的概念,有不同的度量标准,不能直接放到一个空间索引中,需要给这两种变量都做归一化处理,让它们都变成 0~1 之间的变量.对于空间变量的归一化处理,首先确定一个正方形的区域范围,使这个正方形的区域能够完全包含数据集 D 中的所有对象;然后计算这个正方形的边长,设定它的一个顶点为原点,并设定坐标轴 x 和 y;接着再计算数据集中的所有数据对象相对于原点的坐标;最后,对这个坐标进行如公式(5)的归一化处理,得到归一化之后的坐标.

$$x'=\frac{x-x_0}{\sqrt{2}d},y'=\frac{y-y_0}{\sqrt{2}d} \tag{5}$$

其中,d 表示能包含所有数据对象的正方形区域的边长,$\sqrt{2}\,d$ 表示该正方形区域的对角线长,x_0 和 y_0 分别表示选定原点的横坐标和纵坐标.对数据集中的所有数据对象做公式(5)的处理,可以保证任意一个数据对象的横坐标和纵坐标经过处理后都能变成 0 到 1 之间的数值,并且任意一个数据对象到原点的距离值都不会超过 1.对于时

间变量也做类似的处理,给出合理的时间区间 T 和初始时间 t_0,对时间变量做如公式(6)的归一化处理.

$$t' = \frac{t - t_0}{T} \tag{6}$$

3.2 数据映射变换

数据对象的时间变量和空间变量经过归一化处理后都变成了 0 到 1 之间的数值,这样就可以将这两个变量放在一起计算相似度.然后,结合公式(1)中的相似度计算公式对归一化处理后的变量做映射变换,将它们映射到三维空间中.映射变换见公式(7).

$$x'' = \alpha\sqrt{2}x', y'' = \alpha\sqrt{2}x', t'' = (1-\alpha)\sqrt{2}t' \tag{7}$$

经过映射变换后,我们得到三维空间中 3 个维度上的变量 x'',y''和 t''.通过证明,可以得到变换后的三维空间中任意两点的距离相似度是原来根据实际相似度函数得到的得分的上确界.

定理 1. 假设数据集中的任意两个对象 o_1 和 o_2,这两个对象经过映射变换后,三维空间中的距离相似度 $SimS(o_1'', o_2'')$ 和实际相似度 $Sim(o_1', o_2')$ 存在这样的关系,即: $SimS(o_1'', o_2'') \geqslant Sim(o_1', o_2')$.

证明:对于数据集中的任意两个数据对象 o_1 和 o_2,对它们做如公式(5)、公式(6)的归一化处理,得到 $o_1'(x_1', y_1', t_1')$ 和 $o_2'(x_2', y_2', t_2')$,然后再对 o_1 和 o_2 做如公式(7)的映射,得到 $o_1''(x_1'', y_1'', t_1'')$ 和 $o_1''(x_2'', y_2'', t_2'')$.我们用三维空间中归一化处理和映射变换后的两个数据对象之间的距离来表示这两个对象之间的空间相似度,则对于映射后的三维空间中 o_1 和 o_2 之间的空间相似度为 $SimS(o_1'', o_2'') = \sqrt{(x_1''-x_2'')^2 + (y_1''-y_2'')^2 + (t_1''-t_2'')^2}$.利用公式(7)将映射后的变量替换成归一化后的变量,再对空间相似度做平方处理得:

$$SimS(o_1'', o_2'')^2 = 2\alpha^2(x_1'-x_2')^2 + 2\alpha^2(y_1'-y_2')^2 + 2(1-\alpha)^2(t_1'-t_2')^2.$$

接着,通过公式(1)~公式(3)得到实际时空相似度 $Sim(o_1', o_2') = \alpha\sqrt{(x_1'-x_2')^2 + (y_1'-y_2')^2} + (1-\alpha)|t_1'-t_2'|$,平方得 $Sim(o_1', o_2')^2 = \alpha^2(x_1'-x_2')^2 + \alpha^2(y_1'-y_2')^2 + (1-\alpha)^2(t_1'-t_2')^2 + 2\alpha(1-\alpha)\sqrt{(x_1'-x_2')^2 + (y_1'-y_2')^2}\,|t_1'-t_2'|$,进一步对两个数据对象的空间相似度和实际相似度做差,得到:

$$SimS(o_1'', o_2'')^2 - Sim(o_1', o_2')^2 = (\alpha\sqrt{(x_1'-x_2')^2 + (y_1'-y_2')^2} - (1-\alpha)|t_1'-t_2'|)^2 \geqslant 0.$$

由于 $SimS(o_1'', o_2'') \geqslant 0, Sim(o_1', o_2') \geqslant 0$, 所以 $SimS(o_1'', o_2'') \geqslant Sim(o_1', o_2')$. 由此可得,经过映射后两个数据对象之间的三维空间相似度大于等于两个对象之间的实际的相似度.映射变换的目的是用任意两个对象之间的空间相似度来近似代替他们的实际的时空相似度,这样可以根据空间的相似度来建立索引,并根据空间相似度来确定一个查询范围,保证计算出的查询范围内一定是可以包含根据实际的时空相似度计算出 k 个对象.

3.3 构建ST-Rtree索引

数据集中数据对象的时间变量和空间变量经过上面介绍的归一化处理和映射变换后,已经将数据对象从二维空间全部映射到了三维空间.由于经过映射后的两个数据对象之间的三维空间距离大于等于两个对象之间的实际的相似度,所以我们依据这种关系用三维空间中任意两点之间的空间距离来近似表示这两个点代表的数据对象之间的实际的相似度,从而保证了映射变换后的数据对象是有意义的.

由于数据集中的任意一个数据对象都可以在三维空间中被唯一的三维坐标点表示,所以我们只需要在三维空间中给数据集中的数据对象建立一个三维的空间索引.对于空间索引,常见的有 R 树索引、网格索引、四叉树索引等.由于 R 树索引是 B 树索引在多维空间的扩展,它具有 B 树索引的一些优点,如动态平衡、空间利用率较高等,并且 R 树索引不需要遍历所有的数据对象,只需遍历少数的叶子节点就能找到满足条件的查询结果,它的效率很高.所以本文为经过处理后的数据对象的时间变量和空间变量建立一个三维的 ST(spatial and temporal)-Rtree 索引,如图 2 所示.

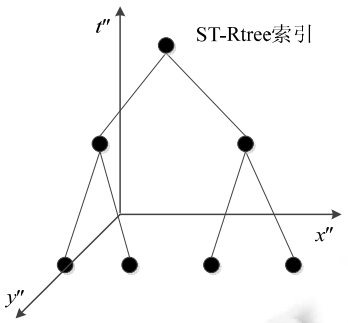


Fig.2 Index of ST-Rtree

图 2 ST-Rtree 索引

4 ST-*k*NN 查询

对于 ST-*k*NN 查询,主要思路是从根节点开始遍历 ST-Rtree,找到距离查询点最近的 *k* 个映射后的对象,然后根据第 *k* 个对象到查询点的距离确定一个范围,保证在这个范围内,一定能够包含根据实际的时空相似度计算出的最终的结果对象.虽然确定了范围,但是由于映射后的三维空间两个对象之间的空间相似度大于等于两个对象之间实际的相似度,所以最后还要分别计算在这个范围内的所有对象和查询点之间实际的相似度,从而在这个确定的范围内找到实际的距离查询对象最近的 *k* 个对象,见算法 1.

算法 1. ST-*k*NN 查询算法.

输入:查询点 *q*,ST-Rtree 索引;

输出:*k* 个数据对象.

```
1:  U←new min-Priority queue;  //用来存放 ST-Rtree 的节点
2:  V←new min-Priority queue;  //用来存放过程中三维空间中的数据对象点
3:  Result←new min-Priority queue;  //用来存放带有实际相似度的数据对象
4:  Rset←∅;  //用来存放结果
5:  r←0; R←0; U.Enqueue(ST-Rtree.root,0);
6:  e←U.Dequeue();  //最小优先权为节点到查询点的距离
7:  while (V.size<k||e.key<V(k).key)
8:      if e is a leafnode then
9:          for each point p in e do
10:             p.key←dis(p,q);  V.Enqueue(p,p.key);  //最小优先权为点到查询点的距离
11:     else
12:         for each e′ in e do
13:             e′.key←Distance(e′,q);  U.Enqueue(e′,e′.key);
14:     e←U.Dequeue();
15: end while
16: r←V(k).key;  R←√2 r;  create a ball O(q,R);  //创建一个以查询点 q 为球心、R 为半径的球
17: while (e∩O≠∅)
18:     if e is a leafnode then
19:         for each point p in e do
20:             p.key←dis(p,q);  V.Enqueue(p,p.key);
21:     else
```

```
22:             for each e′ in e do
23:                 e′.key←Distance(e′,q);   U.Enqueue(e′,e′.key);
24:         e←U.Dequeue();
25:     end while
26:     while (V.size≠0)
27:         point p←V.Dequeue();
28:         if O contains p then
29:             p.Rkey←Sim(q,p);   Result.Enqueue(p,p.Rkey);   //最小优先权为数据对象的实际相似度
30:     end while
31:     for each point in Result from 1 to k do
32:         Rset.add(Result.Dequeue());
33:     return Rset;
```

算法中,第 1 行~第 5 行是初始化阶段,其中,r 表示根据三维空间中的距离计算出的第 k 个距离查询点最近的点到查询点的距离,R 表示能包含 k 个最优对象的球的半径.第 6 行~第 15 行是遍历 ST-Rtree,找到三维空间中距离查询点最近的 k 个点.当最小优先权队列 V 中点的个数大于等于 k,并且当前 e 到查询点的距离大于队列 V 中第 k 个点到查询点的距离时,遍历终止.这样能保证 ST-Rtree 中的剩余节点中,不会再含有比当前第 k 个点到查询点的距离小的节点.其中,计算距离的函数 $Distance(e,q)$见公式(8).

$$Distance(e,q)=\begin{cases}0, & \text{if } q\in e.cuboid\\ \text{Min}Dis(e.cuboid.plane,q), & \text{otherwise}\end{cases} \tag{8}$$

MinDis($e.cuboid.plane$,q)表示的是查询点 q 到 e 节点的最小外包长方体的任意一个面的最小距离,这样保证了 MinDis($e.cuboid.plane$,q)小于等于查询点 q 到该最小外包长方体内包含的任意点之间距离.

第 16 行是在找到距离查询点最近的 k 个数据对象点后,计算第 k 个点到查询点的距离 r,从而确定一个范围,即:以 q 为球心、R 为半径的球体区域,并且通过计算证明我们得到 $R=\sqrt{2}\,r$,这样能够保证这个范围内一定包含根据实际相似度计算出的 k 个最优对象的范围.

在确定能包含 k 个最优对象的区域范围时,由于经过映射后的三维空间中的任意两点之间的空间相似度大于等于两个对象实际的相似度,所以,在三维空间得到的离查询点 q 最近的 k 个对象不一定是 q_0 的 k 最近邻对象,通过二维空间中圆的变化来说明这一问题.如图 3 所示,假设 k=3,图 3(a)展示的是根据实际的相似度得到的 k 个最优对象,图 3(b)展示的是映射后各个对象的位置变化,映射后最好的 k 个对象可能就不是原来的 k 个了,这样查询结果不准确.但是如果我们将半径扩大,确定一个新的半径 R,然后证明出扩大后的范围内一定是包含原来通过实际计算得到的最相似的 k 个对象的.通过推理证明,我们可以得到 $R=\sqrt{2}\,r$.

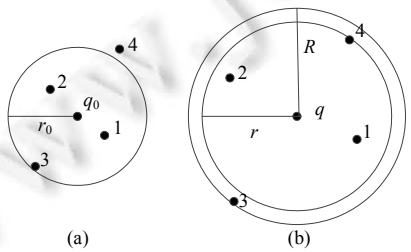


Fig.3 An example of query ranges' change

图 3 查询范围变化示例

定理 2. 查询 q 为球心,R 为半径的球内,一定能包含 k 个最优的结果对象.

证明:假设根据实际的相似度计算得到的结果集为 $O=\{o_1,o_2,\ldots,o_k\}$,由于经过映射后的三维空间中任意两

个对象之间的空间相似度大于等于实际的相似度,则$\forall o\in O, Sim(o,q)\leqslant r$.假设 $o(x,y,z)$映射到三维空间后为$o'(x',y',t')$,计算对象 o 映射后的增量如下:

$$\begin{aligned}\Delta &= SimS(o',q')-Sim(o,q)\\ &=\sqrt{(o'-x'_q)^2+(y'-y'_q)^2+(t'-t'_q)^2}-(\alpha\sqrt{(x-x_q)^2+(y-y_q)^2}+(1-\alpha)|t-t_q|)\\ &=\sqrt{2\alpha^2(x-x_q)^2+2\alpha^2(y-y_q)^2+2(1-\alpha)^2(t-t_q)^2}-(\alpha\sqrt{(x-x_q)^2+(y-y_q)^2}+(1-\alpha)|t-t_q|)\\ &\leqslant(\sqrt{2}-1)(\alpha\sqrt{(x-x_q)^2+(y-y_q)^2}+(1-\alpha)|t-t_q|)\\ &=(\sqrt{2}-1)Sim(o,q)\\ &\leqslant(\sqrt{2}-1)r.\end{aligned}$$

由此,$SimS(o',q')=Sim(o,q)+\Delta\leqslant r+(\sqrt{2}-1)r=\sqrt{2}r$,所以我们得到:当 $R=\sqrt{2}\,r$ 时,以查询点 q 为球心、R 为半径的球内,一定能包含 k 个最优的结果对象.

第 17 行~第 25 行是继续遍历 ST-Rtree,对所有和上面确定的球有交集的节点都要遍历,即,保证遍历到所有有可能包含结果的节点并计算处理.最后,第 26 行~第 33 行是处理上面所有过程中得到的有可能是结果对象的数据对象点,根据公式(3)计算这些点代表的数据对象和查询对象之间的实际相似度,然后按照实际相似度得分排列这些数据对象,从而得到最后的 k 个最优结果. □

5 实验测试

本节针对第 3 节中提出的 ST-Rtree 索引和第 4 节中的查询算法进行实验测试,并设计对比实验,证明该混合索引和查询的高效性和准确性.

5.1 代价分析

对于时空数据的索引的研究由来已久[11,20,22],并取得了很多成果,比如 RT-tree[11],3DR-tree[12],MR-tree[11],HR-tree[13]等.下面通过表 1 来对比说明一下这些索引和本文提出的 ST-Rtree 索引的优缺点.

Table 1

表 1

	索引大小	构建代价	时间变量类型	适合的查询类型
RT-tree	$O\left(\frac{N}{B}\right)$	$O\left(\frac{N}{B}\log\frac{N}{B}\right)$	时间段	对某个时间段的范围查询效率较高,kNN 查询效率低
HR-tree	$O\left(\frac{N^2}{B}\right)$	$O\left(\frac{N^2}{B}\log\frac{N}{B}\right)$	时间点	对某个时间段内的时间点查询效率较高,kNN 查询效率很低
3DR-tree	$O\left(\frac{N}{B}\right)$	$O\left(\frac{N}{B}\log\frac{N}{B}\right)$	时间段	范围查询效率很高,kNN 查询效率低
ST-Rtree	$O\left(\frac{N}{B}\right)$	$O\left(\frac{N}{B}\log\frac{N}{B}\right)$	时间点	范围查询效率很高,kNN 查询效率很高

RT-tree 是先给数据对象的空间信息建立二维的 R-tree 空间索引,然后在每个节点上汇总时间信息,时间信息和空间信息是分开存储的,它更适合移动对象的范围查询.在查询时,先过滤时间信息,再查询空间信息.对于本文提出的对带有地理位置和时间点对象的 kNN 查询,效率不高.MR-tree 和 HR-tree 都是利用重叠的思想,采用两级索引,建立多个不同时间戳的 R-tree,通过时间戳确定对应的 R-tree,然后利用空间 R-tree 索引进行查找.对于像本文这样的精确的 k 近邻查询,需要两重过滤查找,效率很低;且需要建立多个 R-tree,空间代价很大.3DR-tree 是将时间因素作为二维空间之外的第三个维度,虽然给时间单独建立一维索引,但由于时间和空间的度量单位不同,所以不能简单地像三维空间 R-tree 那样通过计算空间距离来查找对象,查询时先确定包含或者相交查询条件的时间区间,再查找在这个时间区间内的空间位置信息,分开过滤时间和空间两个因素,更适合范围查询,而且时间变量大多是一个时间区间,对带有地理位置和时间点对象的精确 kNN 查找效率很低.TPR-tree

索引[16]虽然考虑了时间因素,但主要是用来根据位置函数确定移动对象的位置,本质上还是对空间位置的索引,查询时也只是对移动对象的空间位置查询,时间因素不作为查询条件,不适用于文本的 *k*NN 查询.文献[15]中对移动对象查询时,虽然将时间因素作为一个维度和空间因素共同建立多维 R-tree 索引,但在查询时,先利用查询的时间区间过滤,然后再处理对象的位置信息,相当于两次查找,其索引的效率和 RT-tree 索引相似.但是文章中提出的查询算法主要是针对范围查询和单一维度上的近邻查询,对于综合时间因素和空间因素的 *k*NN 查询,会出现结果不全面和不准确的情况.

所以,综合上面的介绍对比,本文提出的 ST-Rtree 索引在索引大小、构建代价上和已有索引相似,但是在范围查询和 *k*NN 查询时,查询效率都很高.

5.2 对比实验

上一节已经对各种已有索引的代价及其适合的查询类型进行了详细分析,本文采用 RT-tree 的索引思想作为对比,对于 RT-tree 索引,它是先根据数据对象的空间信息建立一个二维的 R-tree 空间索引,然后在每个 R-tree 节点上汇总以这个节点为根的子树的时间信息.对于时间信息的存储,如果在每个节点上汇总所有子节点的全部时间信息,在遍历 RT-tree 的时候,会产生时间信息过滤效率过低,尤其是层数越低的节点,存储的时间信息越多,过滤的时间代价越高.本文采用在每个节点上只存储时间信息的最大值和最小值的 RT-tree 索引,这样做过滤的时间效率提高了,但是可能导致一些无用的遍历,因为仅仅根据存储的时间范围过滤,不能保证以这个节点为根的子树中一定包含最优的结果,可能还要回溯,所以效率也没有本文提出的 ST-Rtree 索引的效率高.

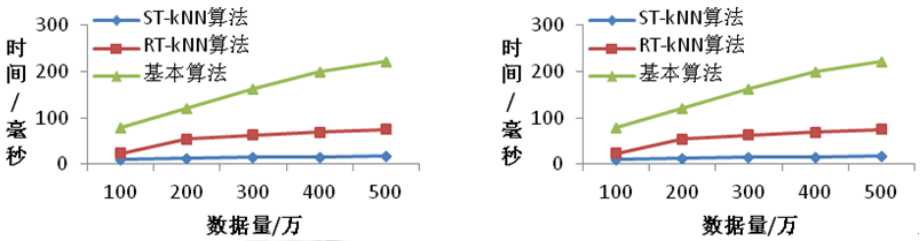
同时,本文还设计了基本的对比实验,即:不给数据建立任何索引,逐条查找并计算相似度,从而找到 *k* 个最近似的结果,将这个基本实验和采用 RT-tree 索引的实验作为对比实验,来和本文提出的查找算法进行对比.

5.3 实验设置

采用两个数据集,一个是真实微博数据集,抽取了 500 万条微博数据,其中,每条微博数据都带有地理位置坐标和时间标签;然后,将地理位置和时间标签进行处理,转化成(x,y,t)这样的三元组.另一个采用合成数据集,包含 1000 万个时空对象,随机抽取国内的地理位置坐标,然后随机分配时间标签,组成三元组.首先给这些三元组对象建立 ST-Rtree 索引,然后进行查询测试.实验时,我们设定调节权重的参数α为 0.6,查询个数 *k* 为 8.实验环境如下:主机采用 Intel(R) Core(TM) i7 CPU,3.07GHz 双核处理器,内存容量为 8G,操作系统为 64 位 Windows7.

5.4 实验结果和分析

图 4 显示了查询时间随着数据量改变的变化趋势,实验时,我们设定参数α为 0.6,*k* 值为 8.



(a) 微博数据集 (b) 合成数据集

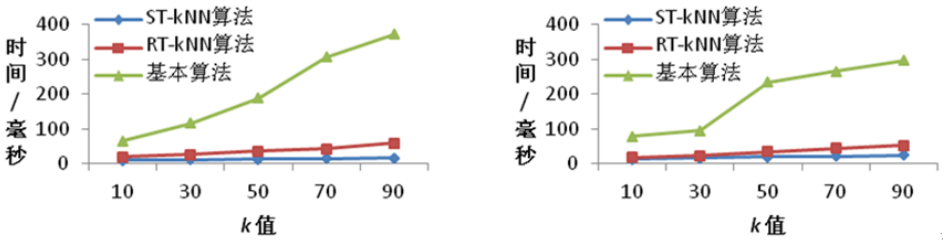
Fig.4 Query time effect of data sizes

图 4 数据量大小对查询时间的影响

我们可以看出:无论是微博数据集还是合成数据集,ST-*k*NN 查询算法、RT-*k*NN 查询算法和基本查询算法总体趋势都是随着数据量的增大,查询时间增加;并且 RT-*k*NN 查询算法和基本算法的查询时间明显大于 ST-*k*NN 算法的查询时间,说明本文提出的 ST-Rtree 索引和 ST-*k*NN 查询算法是很有效的.然而我们可以看到:对于 ST-*k*NN 查询,随着数据量的增大,它的查询时间增加的并不是很明显.这是因为在建立 ST-Rtree 索引的时候,ST-

Rtree 的高度随着数据量的大小呈现对数级增长,所以查询时间增长的慢.假设数据量大小为 N,ST-Rtree 的每个节点的最大容量为 t、最小为 $\left\lceil \frac{t}{2} \right\rceil$,那么 ST-Rtree 的最大高度为 $h=\log_{\left\lceil \frac{t}{2} \right\rceil}\frac{N}{2}+1$,则 ST-*k*NN 查询的最长时间代价为 $\left(\log_{\left\lceil \frac{t}{2} \right\rceil}\frac{N}{2}+1\right)k$.在实验中,我们设定 ST-Rtree 的每个节点的最大容量为 4,最小为 2.图 4(a)中,当数据量从 100 万增长到 500 万时,根据上面的计算我们发现,ST-Rtree 的高度从 12 增长到 14,变化不大,所以我们看到, ST-*k*NN 查询时间增长比较慢.

图 5 显示的是查询时间随着 k 值改变的变化趋势,测试的数据量为 100 万个,参数α设为 0.6,k 值从 10 变化到 90.从图 5(a)、图 5(b)中查询时间的变化趋势可以看出:随着 k 值的增大,无论是微博数据集还是合成数据集的查询时间都增加得很明显,ST-*k*NN 查询算法的查询效率优于其他算法.

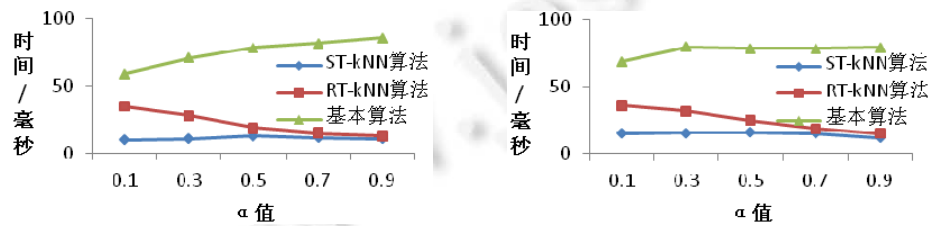


(a) 微博数据集 (b) 合成数据集

Fig.5 Query time effect of k

图 5 k 值对查询时间的影响

图 6 显示的是调节权重的参数α对查询时间的影响.这里,测试的数据量也为 100 万个,查询个数 k 设为 8.在两个数据集的测试结果中我们都可以看到:ST-*k*NN 查询算法和基本查询算法,参数α对它们的查询时间几乎是没有影响的.但是对于 RT-*k*NN 查询算法,从图 6 中可以看出它的查询时间是随着参数α的增大而减小,参数α增大,表示空间因素的权重增加了,说明用空间因素作为相似性查询的主要因素使查询效率提高了;反之,正如第 5.2 节中分析的,因为 RT-tree 索引中的时间信息在每个节点上的存储是用一个时间范围表示的,不够精确,所以查询效率降低了.参数α只是一个用户可以根据自己的偏好随意设定的值,应该尽量保证对查询效率没有影响,因此,RT-*k*NN 查询算法在这方面的性能不好.



(a) 微博数据集 (b) 合成数据集

Fig.6 Query time effect of α

图 6 α值对查询时间的影响

图 7 显示的是用微博数据集构建 ST-Rtree 索引和 RT-tree 索引时需要的时间随着数据量改变的变化趋势.实验时设定的数据量为 100 万~500 万,参数α为 0.6,k 值为 8.随着数据量的增大,构建索引的时间也在增加.同样的,对于构建 ST-Rtree 和 RT-tree 需要的空间进行了测试,从图 8 中可以看到:随着数据量的增大,构建索引需要的空间也在增大.对于 ST-tree 索引和 RT-tree 索引,可以看出,构建它们需要的时间和空间基本是相同的,但是从

上面几个图显示的实验结果来看,ST-tree 索引的效率比 RT-tree 高.

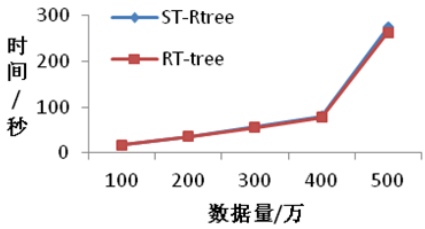


Fig.7 ST-Rtree's contruction time effect of data sizes

图 7 数据量对构建 ST-Rtree 时间的影响

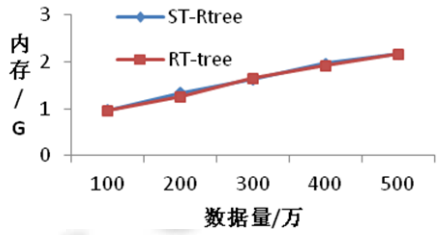


Fig.8 ST-Rtree's contruction memory cost of data sizes

图 8 数据量对构建 ST-Rtree 所需内存的影响

6 结束语

本文通过对带有地理位置标签和时间标签的数据对象进行映射变换,然后给映射变换后的数据对象建立 ST-Rtree 索引,并设计了在 ST-Rtree 索引上的精确查找算法,从而找到了 top-*k* 个离用户给出的查询点在时间和空间上都很近的对象,解决了用户关于带有时空标签信息的查询需求.本文提出的查询处理方法主要是对信息对象的时间和空间变量进行过滤查询,该方法很好地融合了时间和空间两个因素,提出了新的时空数据的混合索引,提高了查询效率.同时,本文提出的索引和查询算法能够帮助用户从网上各种带有时间标签和地理位置标签的信息中快速且准确地查询到想要的信息,为时空文本混合信息的查询研究做了铺垫.互联网上每天都产生大量的各种团购信息、微博信息等等,这些信息种类繁多,但是用户有时候只关心与其联系紧密的某类信息.为了防止用户想要查询的相关结果被大量不相关的信息淹没,下一步,将在本文的研究基础上加入文本信息的过滤功能,并通过研究设计融合时间、空间、文本这 3 个因素的索引,实现对这 3 个因素的过滤查询,以便更符合用户的查询需求.

References:

[1] Chen L, Cong G, Jensen CS, Wu D. Spatial keyword query processing: An experimental evaluation. In: Proc. of the 39th Int'l Conf. on Very Large Data Bases (VLDB 2013). VLDB Endowment, 2013. 217–228. [doi: 10.14778/2535569.2448955]

[2] Cao X, Chen L, Cong G, Jensen CS, Qu Q, Skovsgaard A, Wu D, Yiu ML. Spatial keyword querying. In: Atzeni P, Cheung D, Sudha R, eds. Proc. of the ER 2012. LNCS 7532, Springer-Verlag, 2012. 16–29. [doi: 10.1007/978-3-642-34002-4_2]

[3] Chen YY, Suel T, Markowetz A. Efficient query processing in geographic Web search engines. In: Proc. of the Int'l Conf. on Management of Data (SIGMOD 2006). New York: ACM Press, 2006. 277–288. [doi: 10.1145/1142473.1142505]

[4] Guttman A. R-Trees: A dynamic index structure for spatial searching. In: Proc. of the Int'l Conf. on Management of Data (SIGMOD'84). ACM Press, 1984. 47–57. [doi: 10.1145/602259.602266]

[5] Chen C, Li F, Ooi BC, Wu Sai. TI: An efficient indexing mechanism for real-time search on Tweets. In: Proc. of the Int'l Conf. on Management of Data (SIGMOD 2011). New York: ACM Press, 2011. 649–660. [doi: 10.1145/1989323.1989391]

[6] Cao X, Cong G, Jensen CS. Retrieving top-*k* prestige-based relevant spatial web objects. In: Proc. of the 36th Int'l Conf. on Very Large Data Bases (VLDB 2010). 2010. 3(1-2): 373–384. [doi: 10.14778/1920841.1920891]

[7] Zhang D, Chee YM, Mondal A, Tung AKH, Kitsuregawa M. Keyword search in spatial databases: Towards searching by document. In: Proc. of the 25th Int'l Conf. on Data Engineering (ICDE 2009). Shanghai: IEEE Computer Society, 2009. 688–699. [doi: 10.1109/ICDE.2009.77]

[8] Magdy A, Mokbel MF, Elnikety S, Nath S, He Y. Mercury: A memory-constrained spatio-temporal real-time search on microblogs. In: Proc. of the 30th Int'l Conf. on Data Engineering (ICDE 2014). Chicago: IEEE Computer Society, 2014. 172–183. [doi: 10.1109/ICDE.2014.6816649]

[9] Skovsgaard A, Sidlauskas D, Jensen CS. Scalable top-*k* spatio-temporal term querying. In: Proc. of the 30th Int'l Conf. on Data Engineering (ICDE 2014). Chicago: IEEE Computer Society, 2014. 148–159. [doi: 10.1109/ICDE.2014.6816647]

[10] Chen L, Cong G, Cao X, Tan KL. Temporal spatial-keyword top-*k* publish/subscribe. In: Proc. of the 31st Int'l Conf. on Data Engineering (ICDE 2015). Seoul: IEEE Computer Society, 2015. 255–266. [doi: 10.1109/ICDE.2015.7113289]

[11] Theodoridis Y, Sellis T, Papadopoulos AN, Manolopoulos Y. Specifications for efficient indexing in spatiotemporal databases. In: Proc. of the 10th Int'l Conf. on Scientific and Statistial Database Management. Capri: IEEE Computer Society, 1998. 123−132. [doi: 10.1109/SSDM.1998.688117]

[12] Theodoridis Y, Vazirgiannis M, Sellis T. Spatio-Temporal indexing for large multimedia applications. In: Proc. of the 3rd IEEE Int'l Conf. on Multimedia Computing and Systems. Hiroshima: IEEE Computer Society, 1996. 441−448. [doi: 10.1109/MMCS.1996.535011]

[13] Nascimento MA, Silva JRO. Towards historical R-trees. In: Proc. of the ACM Symp. on Applied Computing (SAC'98). New York: ACM Press, 1998. 235−240. [doi: 10.1145/330560.330692]

[14] Song Z, Roussopoulos N. *K*-Nearest neighbor search for moving query point. In: Proc. of the 7th Int'l Symp. on Advances in Spatial and Temporal Databases (SSTD 2001). London: Springer-Verlag, 2001. 79−96. [doi: 10.1007/3-540-47724-1_5]

[15] Lazaridis I, Porkaew K, Mehrotra S. Dynamic queries over mobile objects. In: Proc. of the 8th Int'l Conf. on Extending Database Technology (EDBT 2002). Springer-Verlag, 2002. 269−286. [doi: 10.1007/3-540-45876-X_18]

[16] Benetis R, Jensen CS, Karciauskas G, Saltenis S. Nearest neighbor and reverse nearest neighbor queries for moving objects. In: Proc. of the Int'l on Very Large Data Bases. New York: Springer-Verlag, 2002. 229−249. [doi: 10.1007/s00778-005-0166-4]

[17] Cong G, Jensen CS, Wu D. Efficient retrieval of the top-*k* most relevant spatial Web objects. In: Proc. of the Int'l Conf. on Very Large Data Bases (VLDB 2009). VLDB Endowment, ACM Press, 2009. 337−348. [doi: 10.14778/1687627.1687666]

[18] Felipe ID, Hristidis V, Rishe N. Keyword search on spatial databases. In: Proc. of the 24th Int'l Conf. on Data Engineering (ICDE2008). Cancun: IEEE Computer Society, 2008. 656−665. [doi: 10.1109/ICDE.2008.4497474]

[19] Lu J, Lu Y, Cong G. Reverse spatial and textual *k* nearest neighbor search. In: Proc. of the 2011 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2011. 349−360. [doi: 10.1145/1989323.1989361]

[20] Mokbel MF, Ghanem TM, Aref WG. Spatio-Temporal access methods. IEEE Computer Society Technical Committee on Data Engineering, 2003,26(2):40−49.

[21] Zhou AY, Yang B, Jin CQ, Ma Q. Location-Based services: Architecture and progress. Chinese Journal of Computers, 2011,34(7):1155−1171 (in Chinese with English abstract).

[22] Zhu SP, Zhao JJ. Research of spatio-temporal access method. Computer Technology and Development, 2008,18(7):56−59 (in Chinese with English abstract).

附中文参考文献:

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

[22] 祝蜀平,赵瑾瑾.时空数据库索引方法研究.计算机技术与发展,2008,18(7):56−59.

李晨(1991－),女,辽宁鞍山人,硕士生,主要研究领域为查询处理.

寇月(1980－),女,博士,副教授,CCF 会员,主要研究领域为实体搜索,数据挖掘.

申德荣(1964－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为分布式数据管理,数据集成.

聂铁铮(1980－),男,博士,副教授,CCF 会员,主要研究领域为数据质量,数据集成.

朱命冬(1985－),男,博士,讲师,CCF 会员,主要研究领域为相似性查询,分布式处理,数据分析.

于戈(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库,大数据管理.