基于聚类的位置数据库动态重组*

马 帅[1+], 王腾蛟[1,2], 唐世渭[1,2], 杨冬青[1], 高 军[1]

[1](北京大学 计算机科学技术系,北京 100871)

[2](北京大学 视觉与听觉信息处理国家重点实验室,北京 100871)

Dynamic Reorganization of Location Databases Based on Clustering

MA Shuai[1+], WANG Teng-Jiao[1,2], TANG Shi-Wei[1,2], YANG Dong-Qing[1], GAO Jun[1]

[1](Department of Computer Science and Technology, Peking University, Beijing 100871, China)

[2](National Laboratory on Machine Perception, Peking University, Beijing 100871, China)

+ Corresponding author: Phn: 86-10-62756374, E-mail: mashuai@db.pku.edu.cn

http://www.pku.edu.cn

Received 2002-07-01; Accepted 2002-09-11

Ma S, Wang TJ, Tang SW, Yang DQ, Gao J. Dynamic reorganization of location databases based on clustering. *Journal of Software*, 2003,14(5):963~969.

http://www.jos.org.cn/1000-9825/14/963.htm

Abstract: How to effectively organize and store the profile of moving objects in a mobile environment, which can effectively lower the paging and update cost, is an important problem in location management. Combining data mining into a mobile environment is a challenging research task, which has broad prospective applications. In this paper, a solution is provided to optimize the placement of location databases from the aspect of data mining. First a new hierarchical clustering algorithm is presented to cluster the moving log, then use the clustering results to dynamically reunite location databases, thus the paging and update cost can be lowered effectively.

Key words: clustering; data mining; location management; mobile object; location database

摘 要: 在无线移动计算环境中,如何合理地组织和存储移动对象(mobile object)的配置信息从而有效地降低查询和更新代价是位置管理(location management)中的一个重要问题.将数据挖掘应用到移动计算环境中是一项具有挑战性的研究课题,具有广阔的应用前景.从数据挖掘的角度出发,提出了一种优化位置数据库的解决方案.首先采用一种新的层次聚类算法对移动日志聚类,然后根据聚类的结果对位置数据库动态重组,从而有效地降低了查询和更新代价.

关键词: 聚类;数据挖掘;位置管理;移动对象;位置数据库

中图法分类号: TP391 **文献标识码**: A

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2002AA4Z3440 (国家高技术研究发展计划); the National Grand Fundamental Research 973 Program of China under Grant No.G1999032705 (国家重点基础研究发展规划(973)); the Foundation of the Innovation Research Institute of PKU-IBM of China (北京大学-IBM 创新研究院项目)

第一作者简介: 马帅(1975－),男,山东潍坊人,博士生,主要研究领域为数据库,信息系统.

在移动计算环境中,如何合理地组织移动对象(mobile object)的位置信息,从而有效地降低查询和更新代价是位置管理(location management)中的一个重要问题,也是当前的研究热点.位置数据库(location databases)[1]是存储移动用户配置信息(profile)的分布式数据库,其中移动用户的位置(location)是配置信息中的一项重要内容.位置数据库的组织结构中最流行的两种结构是双层结构(HLR/VLR)和多层次结构[2].在双层结构中,归属位置寄存器 HLR 存储所有所属移动用户的配置信息,包括移动用户的当前位置等;拜访位置寄存器 VLR 存储覆盖范围内当前移动用户的配置信息的拷贝.多层次结构扩展了 HLR/VLR 双层结构以便于系统的扩展和升级,在这种结构中,位于高层的位置数据库包含底层的位置数据库中移动用户的配置信息.

将数据挖掘应用到移动计算环境中是一项具有挑战性的研究课题,具有广阔的应用前景.如挖掘单个用户的移动模式[3]用于移动用户配置信息的合理分配;通过聚类[4]建立层次,提供更加有效的路由.近年来,为了减少移动用户的配置信息的查询和更新代价,提出了不同的位置数据库的管理模式,包括划分[5],但是文献[5]中的划分是针对单个用户的移动模式的,而且没有提供实现的算法;数据复制[6]考虑的同样也是单个用户的移动模式.当考虑整个位置数据的组织结构时,从单个移动用户的角度来看是不合理的.在本文中我们从数据挖掘的角度出发,考虑群体移动用户的移动模式,提出了一种新的层次聚类算法,并且将此算法应用到位置数据库的动态重组,实现其布置的优化.

本文第 1 节形式化地描述位置数据库的动态重组问题.第 2 节描述具体的聚类算法.第 3 节是总结.

1 动态重组问题的形式化描述

鉴于本文将针对多层次结构中位置数据库的布置优化展开问题的讨论,首先我们将进一步解释位置数据库的多层次结构.通常这种结构是树状结构(如图 1 所示,最下层的六边形代表区域或蜂窝,圆形代表位置数据库),在这种情况下,处于叶子结点的位置数据库为一个区域(zone)或蜂窝(cell)服务(为了方便地描述问题,下文中我们将用区域来代表区域或蜂窝),包含在这个区域注册的移动用户的信息.处于内部结点的位置数据库包含以该结点为根的子树中所有结点的移动用户的信息.由图 1 可以看出,区域 0 中移动用户的配置信息存放在位置数据库 4 中;而位置数据库 4,5,6 的移动用户的配置信息存放在位置数据库 1 中.位置数据库中移动用户的配置信息或者是指针,指向位于底层位置数据库的实际信息,或者是该用户实际的位置信息.

1.1 相关概念

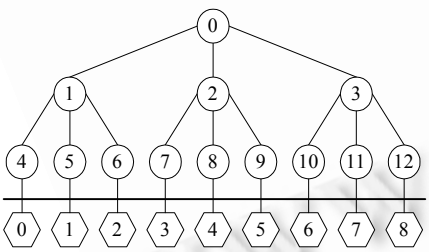

Fig.1 Multi-Tiers architecture
图 1 多层次结构

定义 1(最近共同祖先[1]). 在位置数据库的多层次结构中,最近共同祖先(least common ancestor)是指两个结点的共同祖先结点,并且该祖先结点在层次结构中与结点最接近.结点 i 和 j 的最近共同祖先仍然采用引用文献[1]中的表示方式 $LCA(i,j)$.如在图 1 中,尽管 0 和 1 都是结点 4 和结点 5 的祖先,但是最近共同祖先 $LCA(4,5)=1$.

定义 2(路径结点集). 在位置数据库的多层次结构中,路径结点集是结点的集合,以任意两个结点为起始结点和终止结点形成一条路径,该路径所包含的结点的集合称为路径结点集.结点 i 到 j 的路径结点集合(nodes of path)用 $NOP(i,j)$来表示,在树中任意两个结点是连通的,而且任意两个结点的路径有且仅有一条,因此,$NOP(i,j)$只与 i 和 j 有关.如路径结点集 $NOP(4,0)=\{4,1,0\}$.

定义 3(结点距离). 结点距离(distance of nodes)是指在位置数据库的多层次结构中,任意两个结点之间的距离.结点 i 和 j 的距离用 $DOP(i,j)$来表示,其值等于 $DOP(i,j)$中包含的结点的个数.如在图 1 中,结点 4 和结点 0 的距离为 $DOP(4,0)=3$.

1.2 形式化描述

在所有的结点中,维护移动用户的配置信息可以有效地降低查询代价,但却导致了每次移动操作都需要更新所有的位置数据库,从而增加了更新代价.将移动用户的配置信息有选择地在层次结构的结点中维护,可以有

效地降低更新代价.在这种情况下,当查询和更新操作时,只查询和更新包含位置数据库的结点,忽略其他结点.当一个电话发生在结点 i 和 j 之间时,只有查询位于结点 i 向上到结点 $LCA(i,j)$的路径上的结点 $NOP(i,LCA(i,j))$上的移动用户的配置信息,如当一个电话发生在区域 0 和 2 之间时,只需要查询位置数据库 0,4 和 1.

这样,当移动用户打电话时,查询(paging)代价为

$$COST_P(i,j)=DOP(i,LCA(i,j)), \tag{1}$$

i 和 j 分别表示呼叫移动用户和被呼移动用户所在区域所对应的位置数据库,因为查询 $DOP(i,LCA(i,j))$个结点就会找到所需移动用户的配置信息,在此我们采用需要查询的位置数据库的数目表示代价.

当移动用户从一个区域移动到另一个区域时,需要更新维护包含该用户的配置信息的位置数据库,更新代价由 3 部分组成,删除从位置数据库 i 到 $LCA(i,j)$上的该移动用户配置信息的代价 $COST_d(i,j)$;更新位置数据库 $LCA(i,j)$上移动用户的配置信息所需代价 $COST_u(i,j)$;从位置数据库 $LCA(i,j)$到 j 上增加该移动用户配置信息所需代价 $COST_i(i,j)$.容易得出更新(update)代价为

$$COST_U(i,j)=2*DOP(LCA(i,j),j)+1. \tag{2}$$

由上面的查询代价(1)和更新代价(2)可以看出,$LCA(i,j)$是影响查询和更新的一个重要因素.减少 $DOP(LCA(i,j),j)$或者 $DOP(i,LCA(i,j))$将会有效提高查询效率和降低更新代价.我们提出的解决方案是根据群体用户的移动日志,将区域聚类,使得移动用户在同一个聚类中的区域之间的移动尽可能地多,而在位于不同聚类中的区域之间的移动尽可能地少,然后我们根据聚类的结果动态地重新组合位置数据库.移动用户的移动日志记录了移动用户的移动模式,通常以〈*old_zone,new_zone*〉的形式存在.这样,问题就转化为如何根据移动日志将区域进行聚类.如果将位于同一个聚类中的区域对应的位置数据库安置在多层结构中的同一个结点下,则实际上降低了 $LCA(i,j)$与 i 或 j 的距离.此算法考虑群体移动用户的移动模式,从而在整体上提高了查询效率并降低了更新代价.我们将区域抽象为图的结点,如果移动用户在两个区域之间移动频繁,则它们之间对应有一条边.因而上面的问题形式地描述为,给定无向图 $G=(V,E)$和参数 $\alpha\in[0,1]$,生成图 G 的子图集 $SubGsets=\{G_1,G_2,\ldots\}$.我们用 $G_i[V],G_i[E]$分别表示子图 G_i 的点集和边集;$|\ |$表示集合的势,即集合中元素的个数.子图集 $SubGsets=\{G_1,G_2,\ldots\}$需要满足下面的条件:

(1) 对于$\forall G_i\in SubGsets$,满足 $G_i[V]\subseteq G[V]$和 $G_i[E]\subseteq G[E]$;

(2) 对于$\forall G_i,G_j\in SubGsets$,满足 $G_i[V]\cap G_j[V]=\varnothing$;

(3) 对于 $SubGsets=\{G_1,G_2,\ldots\}$,满足 $G_1[V]\cup G_2[V]\cup\ldots=G[V]$;

(4) 对于$\forall G_i\in SubGsets$,满足$|G_i[E]|\geq\alpha*C^2_{|G_i[V]|}$;

(5) 对于$\forall G_i,G_j\in SubGsets$,$G_i\cup G_j$后生成的图不满足上面的条件.

第(1)个条件可以保证数据的有效性,生成结果集是图 $G=(V,E)$的子图集;第(2)个条件保证生成的子图互不相交;第(3)个条件保证聚类的完备性;第(4)个条件通过参数 α 来控制生成子图的紧密程度,当 $\alpha=1$ 时,生成的子图是完全图;第(5)个条件保证生成的子图是极大子图,即结果集中任意两个子图合并后生成的图不满足前三个条件.

2 聚类算法

聚类是数据挖掘的重要方法之一,通过聚类可以解决上面提出的问题.空间聚类算法可以分为基于划分、层次和网格的算法[7].在许多层次聚类算法中将数据模型化为图,ROCK[8]采用稀疏图来模型化数据;CHAMELEON[9]采用 K-NEAREST 邻居图表示稀疏图;AMOEBA[10]采用 DELAUNAY 图来模型化数据.结合具体情况和数据特征,我们同样采用图来模型化数据,并且扩展这种模型化数据的方法使其应用到聚类算法中用

```
Procedure   Graph_Clustering(α){
Step 1:Graph=Build_Graph()
Step 2:ClusterInitialSet=Graph
Step 3:Do{
       CluserResultSet=Clustering(ClusterInitialSet,α)
       ClusterInitialSet=CluserResultSet
       }While(CluserResultSet.size<ClusterInitialSet.size)
}
```

Fig.2 Graph_Clustering

图 2 聚类算法

来表示聚类.图中的每个结点包含其所有的邻接点和与对应邻接点之间的关联度的信息.聚类中与图的某一结点相邻的结点的数目称为聚类与该结点的关联度.

算法的主要思想是采用循环优化的方法,首先生成初始图,然后把图的每一个结点当作一个聚类,采用自底向上(bottom-up)的层次聚类算法,每次聚类的结果集作为下一次聚类的输入,一直到聚类的结果不再改变为止,即到再也没有可以合并的聚类为止(如图 2 所示),下面我们将详细地描述聚类算法.

2.1 数据结构

```
Class  Node {
       Integer Id
       Set Vertex
       Set Neighbor
       Set Degree
       Integer NumOfVertex
       Integer NumOfEdge
}
```

Fig.3 Data structure
图 3 数据结构

图的结点和聚类统一采用下面的数据结构:*Id* 存储图的结点号或聚类号;*Vertex* 存储图的某个结点 *Id* 或者是某个聚类中的所有的结点的 *id*;*Neighbor* 用来存储 *Vertex* 的邻居结点;*Degree* 用来存储对应邻居结点的关联度,即聚类与各对应邻居的边的数目;*NumOfVertex* 存储结点 *Vertex* 的数目,对于图的结点来说为 1;*NumOfEdge* 存储结点 *Vertex* 内部边的数目,对于图的结点来说,结点 *Vertex* 的数目为 1,因此 *NumOfEdge* 为 0(如图 3 所示).

2.2 聚类判断标准

我们采用图趋向于完全图的程度来描述一个聚类内部结点的紧密程度,因为当一个聚类中 C_i 的结点联系紧密时,由 C_i 所形成的子图在图 *Graph* 中就越趋向于完全子图,这样我们采用聚类 C_i 中边的数目与由 C_i 的结点组成完全图的边的数目的比率作为判断一个聚类好坏的标准:

当$|C_i|$=1 时:$Goodness(C_i)\geq\alpha$.

当$|C_i|$=2 时:

$$Goodness(C_i)=C_i.NumOfEdge/C^2_{C_i.NumOfVertex}. \tag{3}$$

我们用 $Co_link(C_i,C_j)$表示两个聚类中相邻结点对的数目,即在图 *Graph* 中将两个聚类连接在一起的边的数目.这样,相应的两个聚类 C_i 和 C_j 合并后好坏的判断标准是

$$Goodness(C_i,C_j)=(C_i.NumOfEdge+C_j.NumOfEdge+Co\_link(C_i,C_j))/C^2_{C_i.NumOfVertex+C_j.NumOfVertex}. \tag{4}$$

只有满足 $Goodness(C_i,C_j)\geq\alpha$ 的两个聚类才能合并.我们在算法中通过阈值 $\alpha\in[0,1]$来控制聚类中结点的紧密程度.当图 *Graph* 中的结点联系比较少时,相应的,α 的值应设得小一些.反之,当图中的结点联系比较紧密时,相应的 α 的值应设得大一些..

2.3 主要过程

过程 *Build_Graph*(),首先通过统计移动日志〈*old_zone,new_zone*〉得到任意两个区域之间移动用户的移动次数,当移动次数大于一定阈值时认为两者在图 *Graph* 中存在一条边(算法中我们将阈值设为移动用户的平均移动次数),这样就生成了图的矩阵表示;然后采用图 3 所示的数据结构表示生成的图:对于图来说 *Vertex* 存储结点的 *Id*,*Neighbor* 存储结点的邻居,*NumOfVertex* 的值为 1,*NumOfEdge* 为 0,*Degree* 存储对应邻居结点的关联度,图结点与邻接结点的关联度都为 1.

过程 *Clustering* (*ClusterInitialSet*, α) 如图 4 所示.首先将聚类的结果集 *CluserResultSet* 赋为∅,然后逐个考虑初始聚类集 *ClusterInitialSet* 中聚类 C_i ($1\leq i\leq ClusteInitialSet.size$), 如果在

```
Procedure  Clustering(ClusterInitialSet,α){
Step 1:CluserResultSet=∅
Step2:i=1
Step3:If (i>ClusterInitialSet.size) Then Goto Step4 Else Goto Step5
Step4:Return CluserResultSet
Step5:Ci=ClusterInitialSet.elementAt(i)
      If there exisits Cj∈ClusterResultSet(1≤j≤ClusterResultSet.size) which
      fulfils Goodness(Ci,Cj)≥α and Co_neighbor(Ci,Cj) is the largest one Then
            Cj=Merging_Clusters(Ci,Cj)
      Else
            Generate a new cluster Ck=Ci(k=ClusterResultSet.size+1)
            ClusterResultSet=ClusterResultSet∪Ck
Step6:i++
Step7:Goto Step3
}
```

Fig.4 Clustering procedure
图 4 聚类过程

CluserResultSet 找到一个聚类 C_j ($1\le j\le ClusteInitialSet.size$)满足条件 $Goodness(C_i,C_j)$,并且两个聚类的共同的邻居 $Co_neighbor(C_i,C_j)$最大,则合并聚类 C_i 和 C_j,并用合并后的聚类代替 *CluserResultSet* 中的聚类 C_j;否则在 *CluserResultSet* 中增加新的聚类 C_k($k=CluserResultSet.size+1$).

过程 $Merging_Clusters(C_i,C_j)$合并聚类时有两个因素要加以考虑,一是合并的两个聚类的共同的邻居 $Co\_neighbor(C_i,C_j)$;另一个是合并的两个聚类之间的联系 $Co\_link(C_i,C_j)$.第 1 个因素,使得我们的聚类算法充分考虑了空间聚类的特性,即结点的邻居对聚类的最终结果的影响,使得聚类的合并趋向更加有利;第 2 个因素保证了合并聚类相互之间联系紧密.总之,需要在满足 $Goodness(C_i,C_j)\ge\alpha$ 的基础上,保证 $Co_neighbor(C_i,C_j)$最大的两个聚类合并,在算法中我们通过参数 α 加以控制聚类的紧密程度.

下面我们描述具体的合并操作,假定 $C=C_i\cup C_j$, 则:

$C.Vertex= C_i.Vertex\cup C_j.Vertex$,

$C.Neighbor= C_i.Neighbor\cup C_j.Neighbor - C.Vertex$,

$C.Degree= C_i.Degree\cup C_j.\ Degree - C.Vertex.Degree$,

$C.NumOfVertex=|C.Vertex|$,

$C.NumOfEdge=C_i.NumOfEdge+C_j.NumOfEdge+1/2*\ ((C_i.Neighbor\cup C_j.Neighbor)\cap C.Vertex).Degree$.

其中,合并后的聚类 C 的顶点是聚类 C_i 和 C_j 顶点的并集;聚类 C 的邻居是聚类 C_i 和 C_j 邻居的并集,然后去除那些合并后成为内部结点的邻居结点;聚类 C 与其邻居的关联度是聚类 C_i 和 C_j 关联度的并集(如果有相同的邻居,则对应关联度的和作为合并后的关联度),然后去除那些成为内部结点的关联度;聚类 C 内部结点之间边的数目 *NumOfEdge* 是聚类 C_i 和 C_j 的 *NumOfEdge* 之和,然后加上那些合并后成为内部结点的邻居结点的关联度之和的 1/2(由图论易知,边的数目是关联度的 1/2).

2.4 算法的复杂性分析

我们假定区域的数目为 m,移动日志的数目为 n.在过程 *Build_Graph*()中首先在 $O(n)$时间内统计生成 $m\times m$ 矩阵,然后在 $O(m^2)$时间内生成图 *Graph*.图的聚类 *Clustering*()可以在 $O(m^2)$内完成,因此整个算法 *Graph_Clustering*()的时间复杂性为 $O(n)+O(m^2)+O(m^3)$.通常算法中区域和移动日志的数量级相差很大,此时 *Graph_Clustering*()聚类算法的时间复杂性可近似为 $O(n)$.

容易看出,此算法的空间复杂性为 $O(m^3)$.

2.5 算法的正确性分析

下面我们证明聚类算法的正确性,即算法解决了在第 1.2 节提出的问题.由上面算法的描述可知,初始聚类集中的聚类是图 $G=(V,E)$中的结点,第 2.3 节的判断标准保证初始聚类集中的任意聚类满足 $Goodness(C_i)\ge\alpha$,容易看出,初始聚类集除了第 1.2 节中的条件(5)以外,其他条件都满足;此后每次循环的聚类合并保证 $Goodness(C_i,C_j)\ge\alpha$,这样就保证了第 1.2 节中除条件(5)之外的其他条件;算法的终止条件保证了条件(5),从而算法的最终聚类集满足第 1.2 节中问题的所有条件.综合上面分析可知,我们的聚类算法正确地解决了第 1.2 节提出的问题.

2.6 算法的实验分析

我们采用 Stanford 大学无线网络小组开发的仿真程序产生的测试数据 SUMATRA (Stanford University Mobile Activity TRAces).SUMATRA 包含两类数据:即 36 个区域的斯坦福学校内部无线环境的信号信息和 90 个区域的海湾地区的位置信息.实验的硬件环境是 IBM Netfinity 5500:两个 X86 Family 6 Model 10 Stepping 1 GenuineIntel~700Mhz 的 CPU,主存为 512M,硬盘为 20G;软件环境是:操作系统为 Microsoft Windows 2000 Server,算法采用 Java 编写.

从图 5 和图 6 中可以看出,参数 α 对算法执行效率的影响很小.我们在算法中通过阈值 $\alpha\in[0,1]$来控制聚类中结点的紧密程度.当图 *Graph* 中的结点联系比较少时,相应的 α 的值应小一些.反之,当图中的结点联系比较紧密时,相应的 α 的值应大一些.从图 7 中可以看出,区域个数对算法的执行效率有一定的影响,但随着数据量的增

加,它对算法的影响逐渐减少,这与第 2.4 节中对算法的复杂性分析是一致的;我们在图 8 中给出了算法执行的细节,其中区域的个数为 90,参数 α 为 0.4.图 8 说明了算法中生成图的部分占据了算法执行时间的主要部分,一旦生成图,对图聚类仅仅需要很少的时间.同时可以看出,由于区域的个数固定,对图聚类的执行时间接近于常量,这与第 2.4 节中对算法的复杂性分析也是一致的.

同时我们也可以看出,在大量移动日志的情况下聚类算法始终保持在秒级内完成,有着良好的性能,从而具有较好的扩展性.

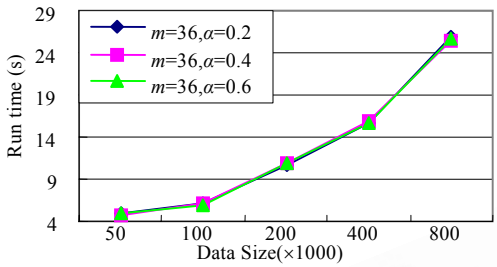

Fig.5 Effect of α to the algorithm's efficiency

图 5 参数 α 对算法执行效率的影响

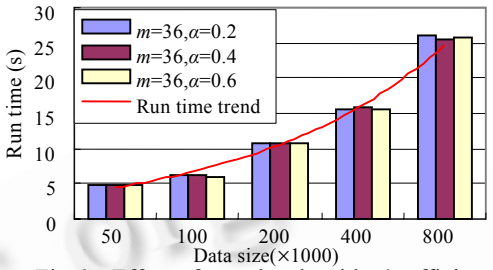

Fig.6 Effect of α to the algorithm's efficiency

图 6 参数 α 对算法执行效率的影响

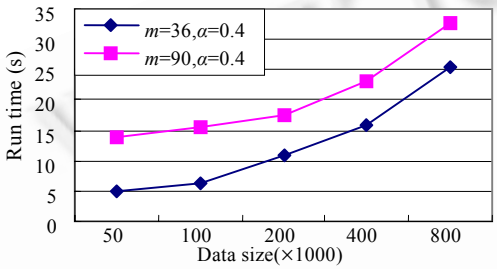

Fig.7 Effect of zone to the algorithm's efficiency

图 7 区域个数对算法执行效率的影响

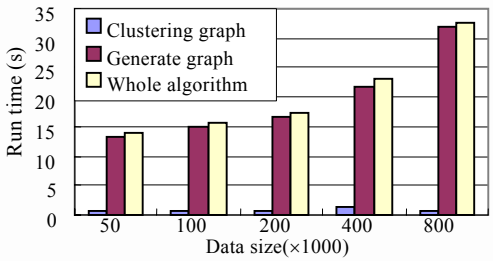

Fig.8 Algorithm's executing detail

图 8 算法执行细节

3 结 论

将数据挖掘应用到移动计算环境中是一项具有挑战性的研究课题,具有广阔的应用前景.本文的贡献在于,从数据挖掘的角度出发提出了一种优化位置数据库布置的解决方案;提出了一种新的层次聚类算法,此算法具有较好的扩展性,能够适应移动日志数据量特别大的需要.同时,通过此算法动态重组位置数据库可以有效地降低查询和更新代价.

致谢 IBM 信息基础设施技术组的陈捷博士后,宋国杰、遇辉等博士和牛兴雯硕士对于本文的完成给予了很大的帮助,在此一并表示感谢.

References:

[1] Pitoura E, Samaras, G. Locating objects in mobile computing. IEEE Transactions on Knowledge and Data Engineering, 2001,13(4): 571~592.

[2] Awerbuch B, Peleg, D. Concurrent online tracking of mobile users. ACM SIGCOMM Computer Communication Review, 1991, 21(4):221~233.

[3] Peng WC, Chen MS. Mining user moving patterns for personal data allocation in a mobile computing system. In: Proceedings of the 2000 International Conference on Parallel Processing. USA: IEEE Computer Society, 2000. 573~580.

[4] Banerjee S, Khuller S. A clustering scheme for hierarchical routing in wireless networks. In: Proceedings of the IEEE INFOCOM 2001, the Conference on Computer Communications, 20th Annual Joint Conference of the IEEE Computer and Communications Societies, Twenty years into the communications odyssey. IEEE Computer Society. 2001. 1028~1037.

[5] Badrinath BR, Imielinski T, Virmani A. Locating strategies for personal communications networks. In: Proceedings of the 1992 International Conference on Networks for Personal Communications. IEEE Computer Society, 1992. II.1-II.9.

[6] Shivakumar N, Jannink J, Widom, J. Per-User profile replication in mobile environments: algorithms, analysis, and simulation results. ACM-Baltzer Mobile Networks and Applications, 1997,2(2):129~140.

[7] Han JW, Kambr M. Data Mining Concepts and Techniques. Beijing: Higher Education Press, 2001. 335~395.

[8] Guha S, Rastogi R, Shim K. ROCK: A robust clustering algorithm for categorical attributes. In: Proceedings of the 15th International Conference on Data Engineering. IEEE Computer Society, 1999. 512~521.

[9] Karypis G, Han E-H, Kumar V. Chameleon: Hierarchical clustering using dynamic modeling. IEEE Computer, 1999,32(8):68~75.

[10] Estivill-Castro V, Lee I. AMOEBA: Hierarchical clustering based on spatial proximity using delaunay diagram. In: Forer P, Yeh AGO, He J, eds. Proceedings of the 9th International Symposium on Spatial Data Handling. HongKong: Study Group on Geographical Information Science of the International Geographical Union, 2000. 7a.26~7a.41.