一种基于P2P的空间数据索引方法

刘 丹[1] 谢文君[2]
(华中师范大学信息技术系 武汉 430079)[1] (国家基础地理信息中心 北京 100830)[2]

摘 要 提出一种基于分组Chord#的P2P空间数据索引,并给出该索引结构下的空间查询以及路由恢复方法。测试表明,这种分布式索引的维护代价较低,利用其进行空间查询具有较好的可扩展性。分组的增加可以减少查询跳数,但对于查询的总开销,存在一个最优的分组成员个数。另外,提出的基于空间接管的路由恢复机制可以较好地应对节点失效的问题,增强了系统的可用性。

关键词 P2P,分组,空间查询,空间数据索引
中图法分类号 TP393 **文献标识码** A

Spatial Data Index Method Based on P2P

LIU Dan[1] XIE Wen-jun[2]
(Department of Information Technology, Huazhong Normal University, Wuhan 430079, China)[1]
(National Geomatics Center of China, Beijing 100830, China)[2]

Abstract This paper put foward a spatial data index based on group chord. At the same time, it provided a kind of spatial query under this index frame and routing recovery. Test results present that this distributed index's maintenance cost is lower. So making using of this index to progress spatial index has scalability. The increase of the size of groups can decrease the query hops. But for the overall spending of this query, there is the most optimized size of group. In addition, this paper also put forward a routing recovery mechanism based on the space-taken, which can deal with the problems of peer failing efficiently and can reinforce the usability of the system as well.

Keywords P2P, Group, Spatial query, Spatial data index

1 引言

空间数据获取技术、网络、空间数据库等技术的发展,使得网络上的空间数据资源得到了极大丰富。如何有效地管理和组织这些地理上广泛分布的海量空间数据资源,成为目前亟待解决的难题。P2P系统在规模性、可用性、可扩展性等方面表现出来的优势使我们看到,利用P2P技术组织管理空间数据资源为GIS提供大规模数据共享平台,是一个很好的选择与发展方向。

空间查询是空间数据管理必须提供也是最重要的功能之一,如何在P2P环境下支持空间查询近年吸引了研究人员的关注并出现了一些研究成果。这些成果都是利用某种P2P环境下的分布式空间索引来定位与查询相关、裁减与查询无关的网络节点。索引的构造思路大都为将集中式空间数据索引技术通过某种映射关系移植到P2P覆盖网络中。比较有代表性的有:P2PR-tree[1]是将R树[9]移植到一种层次化的P2P覆盖网络中,Distrubuted-qudatree[2]是将MX-CIF四叉树[10]移植到Chord[3]中,SkipIndex[4]是将KD[11]树移植到Skip Graph[12]中,ASPEN[5]是将网格文件索引移植到CAN[6]中,SCRAP[7]是将空间填充曲线(如Hilbert曲线[13])移植到某种基于DHT的覆盖网络中(如Chord)等。已有的相关工作也存在一些缺陷,其主要集中在单点失效、负载均衡、可扩展性等方面。总体上看,SkipIndex在各方面都有较好的表现,但也存在着查询效率有待提高、没有一个很好的节点失效处理机制等方面的问题。

本文在SkipIndex的基础上,提出一种基于分组Chord#[8]的分布式空间数据索引,并给出该索引结构下的空间查询以及路由恢复方法。

2 应用场景与系统构造

本文假定数据源节点之间共享的空间数据属于同一维度、同一坐标空间下,并主要针对二维空间下的空间数据(可以根据需要扩展到二维以上)。另外,假定每个节点针对本地存储的数据维护了一个或多个本地空间数据边界矩形(Local SpatialData Mininum Bounding Rectangle, LSD-MBR)。这些同一坐标空间下的LSD-MBR形成"天然的空间数据分片",其分别代表存储了不同区域空间数据的节点或者相同区域但不同专题空间数据的节点。基于上述应用场景,将针对网络中的数据源所存储的空间数据建立分布式索引转换为针对这些粗粒度的LSD-MBR建立分布式索引。

本文将系统中所有的节点依据功能划分为路由节点和数据节点,分别用 *Route_peer* 和 *Data_peer* 表示。其中 *Data_*

到稿日期:2011-09-12 返修日期:2011-11-23 本文受华中师范大学中央高校基本科研业务费项目(CCNU11A01041)资助。
刘 丹(1978-),男,博士,讲师,主要研究方向为GIS、P2P环境下的空间数据检索。

peer 指的是仅负责根据本地数据计算查询的数据源节点，而 *Route_peer* 指的是从 *Data_peer* 中挑选出来的负责根据索引信息路由查询消息的数据源节点。另外，系统中每个数据源节点将 LSD-MBR 的中心作为自己的“代表点”，其主要目的是通过代表点来决定节点在数据空间中的位置，从而决定在覆盖网络中的位置。系统的构造过程描述如下：

(1)第一个数据源节点加入，从功能角度上说，它是一个 *Route_peer*。

(2)其他的数据源节点通过系统中任何一个已经存在的节点加入系统，并且它们都向 *Route_peer* 注册关于自己的地址、LSD-MBR、代表点等信息。

(3)当 *Route_peer* 发现维护的节点数量(包括自己)超过预先设置的某个门限值时(用 *Load_Max* 表示)，对整个数据空间进行分裂，从分裂出去的节点中选择一个节点作为 *Route_peer*，并将该节点的地址告知所有分裂出去的节点。这样，整个数据空间就一分为二，分别包含了若干数量的数据源节点。对于空间分裂需要说明的是，为了尽可能使两个子空间所包含节点的代表点远离分割线(减少 LSD-MBR 在分裂后跨越子空间的可能性)，分裂准则为：将原空间中所有代表点的坐标按每一维进行排序，并选出最小值与最大值，两者差值最大的维称为分裂维，分辨器的值取两者的平均值。比如有(2,7)，(4,2)，(6,12) 3 个代表点，那么在 y 轴(第二维)进行分裂，分辨器的值为 7。如果 LSD-MBR 跨越了一个以上的子空间，则将 LSD-MBR 与其他子空间相交部分的范围信息发送给相应子空间中的 *Data_peer*；如果相交部分的范围内没有包含数据，则不需要发送相关信息。

(4)重复上述过程，随着数据源节点的不断加入，数据空间最终被分裂成若干个子空间。如图 1 所示，灰色的点代表 *Route_peer*，黑色的点代表 *Data_peer*，虚线矩形代表节点的 LSD-MBR，*Load_*Max 为 2。

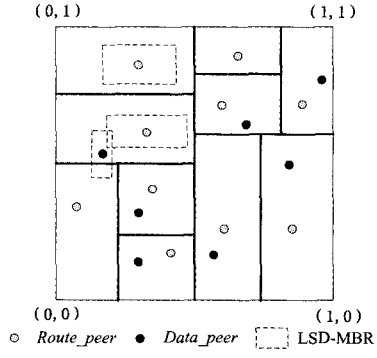

图 1　系统构造示意图

3　分组 Chord# 下数据源节点的组织

Chord# 是在 Chord 的基础上改进而来的，与 Chord 最大的不同在于，它的 *finger* 表基于节点标识符空间，而 Chord 则基于资源标识符空间 。其节点 *finger* 表中，第 i 项 *finger*[i]定义为：

$$finger[i]=\begin{cases} successor, & i=0 \\ finger[i-1].\ finger[i-1], & i\neq 0 \end{cases} \quad (1)$$

与 Chord 相比，Chord# 减少了 *finger* 表项，降低了 *finger* 表的更新开销，并保证查找资源的跳数开销为 O(log$N$)(N 为系统中节点总数)等。

本文利用层次化的结构(即分组 Chord#)来组织网络中的空间数据源节点。从覆盖网络的角度来看，每个子空间代表一个分组，每个 *Route_peer* 负责一个相对应的子空间。*Route_peer* 充当每个分组中超级节点的角色，*Data_peer* 则充当客户节点的角色(它们之间可以角色互换)。*Route_peer* 在覆盖网络中的位置由该节点所在分组的 *ID* 决定。*Data_peer* 属于哪个分组由该节点的“代表点”属于哪个 *Route_peer* 负责的子空间决定。

(1)子空间 *ID* 分配

由于数据空间是高维对象，因此需要采用某种方式对数据子空间进行 *ID* 分配，从而将它们映射到一维的 Chord# 环中。*ID* 分配方式描述如下。

假设子空间 *ID* 由 N 位构成，则整个系统最多可以容纳 2^N 个子空间。在图 2 所示的例子中，子空间 *ID* 由 4 位构成，因此最多可以容纳 16 个子空间。子空间 *ID* 的初始值(即第一个子空间 *ID*)为全 0，随着整个空间的分裂，对子空间 *ID* 进行相应调整。调整规则为：当系统在某个子空间中的某一维上进行分裂时，该维较小值区域的新子空间 *ID* 维持原来未分裂子空间的 *ID* 不变，该维较大值区域的新子空间 *ID* 则在下一个需要分裂的 *ID* 位数(在本文中定义为 *Next_Split*)上置 1。*Next_Split* 初始值为 1，最大值为 N，每分裂一次加 1。举例来说，在进行了如图 2 所示的第①次分裂后，系统分为 2 个子空间。由于分裂是在 x 轴上进行的，因此左边子空间 *ID* 保持不变，右边则为 1000。当系统进行了第②次分裂后，由于分裂是在 y 轴上进行的，因此右边上面部分的子空间 *ID* 为 1100，右边下面部分的子空间 *ID* 保持不变，依次类推。

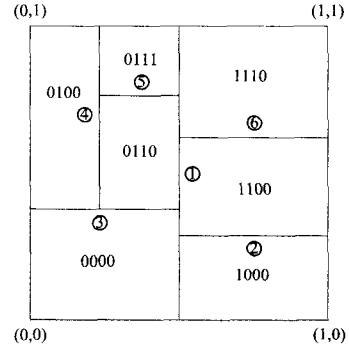

图 2　空间分裂历史示意图

这样每当系统发生分裂产生新的子空间的时候，这个新的子空间的 *Route_peer* 就可以根据前面所述的 *ID* 分配方式得到一个 *ID*，并据此按照 Chord# 的节点加入协议加入到系统中。最终由各个子空间的 *Route_peer* 形成一个顶层的环状网络。

(2)路由表构造

为了保证分组 Chord# 中查询路由的效率以及适应系统中节点的动态变化，需要为系统中的每个节点构造路由表。描述如下。

每个 *Route_peer* 以递归的形式维护系统中另外 m 个 *Route_peer* 的地址信息(假设系统中有 N 个 *Route_peer*，则 $2^m=N$)。另外，还需要维护一定数据量的后继 *Route_peer* 地址信息和本组内所有 *Data_peer* 的地址列表 *Data_peerlist*，如表 1 所列。

表 1　Route_peer 路由表定义

符号	定义
finger[i]	finger[i−1],finger[i−1],0<i≤m
successor	finger[0]
predecessor	本节点的前一个 Route_peer
successorlist	后继 Route_peer 列表
Data_peerlist	Data_peer[j],0≤j≤Load_max−1

Route_peer 定期将更新的路由表信息告知本组内所有 *Data_peer*(*Data_peerlist* 除外)。*Data_peer* 维护 *finger*[*i*] 的主要目的是,使查询消息直接通过其它 *Route_peer* 转发出去,而不需要每次都由本组 *Route_peer* 转发,从而降低对本组 *Route_peer* 的依赖。*Data_peer* 维护 *successorlist* 的主要目的是,本组 *Route _ peer* 失效后可根据 *successorlist* 重新加入新的组。

(3)性能分析

根据前文描述,在包含 N 个节点的 *Chord*$^{\#}$ 中,并且在已知目的地节点 *ID* 的前提下(如从节点 0000 将消息转发给目的地节点 1100),查询步长(跳数)为 O(log$N$),平均查询步长为 1/2 O(log$N$),更新每个节点的路由表开销(消息数)为 O(log$N$)。那么可以得出,在包含 N 个节点,每个分组的节点个数为 M,并且已知目的分组 *ID* 的前提下,分组 Chord$^{\#}$ 的查询步长为 O(logN/M)(由于路由消息在分组之间转发,分组的个数为 N/M),平均查询步长为 1/2 O(logN/M),查询开销为 O(logN/M)+M(当查询定位到分组后,分组内的 *Data_peer* 收到查询消息的开销为 M),更新每个分组的路由表的开销为 O(logN/M)+M(当每个分组的 *Route_peer* 更新路由表后,需要以广播的形式告知分组内的 *Data_peer*,开销为 M)。

4　分组 Chord$^{\#}$ 下的空间查询与路由恢复

(1)点查询

点查询的分组定位步骤为:当系统中的任一节点提出点查询 q 时,如果该节点发现 q 属于本组的子空间范围内,则将查询转发给本组的 *Route_peer*(如果节点本身就是 *Route_peer*,则不需要转发),由 *Route_peer* 向本组所有节点广播这个点查询。如果该节点发现 q 不属于本组的子空间范围内,则通过本地维护的分裂史信息,将查询利用 Chord$^{\#}$ 路由协议转发给在该节点看来其子空间包含了 q 的分组。之所以用“在该节点看来”来表示,是因为系统中的每个节点并不知晓全局的子空间分布状态,因此该分组可能不是真正包含 q 的分组。该分组收到查询后,继续按照上述步骤解析查询,直到查询被转发给其子空间真正包含了 q 的分组。如算法 1 所示,*LocalGID* 表示提出查询的节点所属的分组 *ID*,q[*splitdim*]表示 q 在 *LocalGID* 当前位上的分裂维上的坐标,*pos* 表示在 *LocalGID* 当前位上的分裂维上的分裂坐标,*DesGID* 则表示最终得到的下一步转发目标分组,初始值为全 0。

算法 1　PointRouting(q)

```
1. for i=1 to Next_Split do
2.    c=ith bit in LocalGID
3.    if(c=0&&q[splitedim])≤pos or
        c=1&&q[splitedim])>pos then
4.       DesGID[i]=c
5.    else
6.       DesGID[i]=~c
7.       return DesGID
8.       break
9.    end if
10. end for
```

(2)区域查询

区域查询的分组定位步骤为:当系统中的任一节点提出区域查询 *qrange* 时,如果该节点发现 *qrange* 被本组的子空间范围包含,则将查询转发给本组的 *Route_peer*(如果节点本身就是 *Route_peer*,则不需要转发),由 *Route_peer* 向本组所有节点广播这个区域查询。如果该节点发现 *qrange* 不被本组的子空间范围包含(可能相交也可能不相交),则通过本地维护的分布式搜索树,将查询利用 Chord$^{\#}$ 路由协议转发给那些在该节点看来其子空间与 *qrange* 相交或包含了 *qrange* 的分组。这些分组(可能不止一个)收到查询后,继续按照上述步骤解析查询,直到查询被转发给其子空间真正与 *qrange* 相交或包含了 *qrange* 的那些分组。如算法 2 所示,*LocalArea* 表示本地分组所属的子空间,*RemoteArea* 表示 *Route_peer* 维护的局部搜索树中的非本地分组子空间,*newarea* 表示 *RemoteArea* 和 *qrange* 相交的区域。

算法 2　AreaQuery(qrange)

```
1. if(LocalArea contains qrange)
2.    perform local query(qrange)
3.    else for all RemotoArea in local KDB tree
4.          if(RemoteArea. isintersect(quange)) then
5.          newarea=intersected(RemoteArea,qrange)
6.          forword areaquery(newarea) to RemoteArea
7.          end if
8.          end for
9. end if
```

(3)基于空间接管的路由恢复

在本文的分组 Chord$^{\#}$ 环境下,*Route_peer* 的失效会引发以下问题:本组内所有正常的 *Data_peer* 不能贡献存储在本地并且符合空间查询条件的结果,以及即使失效的节点经过一段时间恢复正常,也有可能不能重新回到系统中,并且新的节点也有可能不能加入系统(数据空间的丢失)。因此需要有相应的处理机制来应对 *Route_peer* 失效。

基于子空间接管的路由恢复方法的关键就是当系统中的某个 *Route_peer* 发现自己的后继节点失效后如何处理。基本思路为:系统中的每个 *Route_peer* 不仅维护 r 个后继节点的地址,还维护它们所在分组对应的子空间范围。当节点 p 发现后继节点失效后,首先将失效节点的后继节点作为自己新的后继节点,然后将失效节点所管理的子空间发送给新的后继节点,即始终保持整体数据空间的完整性。

在算法 3 中,*takeAreanum* 的初始值为 0(并不是每次调用路由恢复算法时都将其置 0,而是在收到了后继节点成功接管的确认消息后置 0),用来计算需要接管的失效节点个数。*temptakeArea* 的初始值为空,用来存储当前需要发送给后继节点接管的子空间信息。

算法 3　routing recovery

```
//takeAreamum 的初始值为 0,temptakeArea 的初始值为空
1. if(finger[0] failed && takeAreanum<succList. lenght)
2.    takeAranum++
3.    finger[0]=succList[takeAreanum]
```

```
4.  for i=0 to takeAreanum-1 do
5.      temptakeArea.add(succList[i].Area)
6.  end for
7.  send temptakeArea to finger[0]
8. end if
```

5 实验与分析

本文所有的仿真实验都是基于 P2P 仿真平台 PlanetSim 以及 Java SDK 1.6，实验程序运行环境为 IntelP4 3.0G，内存 512M。测试分为两大部分，第一部分为对分组 Chord# 进行测试，第二部分为对分组 Chord# 下的空间查询进行测试。

第一部分包含 3 个测试。第一个测试将基于 Chord# 的分组式覆盖网络与 SkipIndex 的底层覆盖网络 SkipGraph 进行比较。在分别由 2^{10}，2^{11}，2^{12}，2^{13}，2^{14}，2^{15} 个节点组成的网络中，随机选取源节点和目标节点，对查询从源节点到目标分组内的节点所经历的跳数进行测试对比。从图 3 可以看到，在每个组的大小为 1 的情况下，基于 Chord# 的分组式覆盖网络在平均查找跳数方面至少优于 SkipGraph 40%。

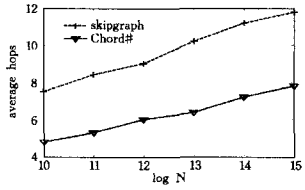

图 3 Chord# 与 SkipGraph 的平均查询跳数对比图

第 2 个和第 3 个测试为在不同规模的覆盖网络下将分组的大小分别设置为 1,8,64,256，然后随机选取源节点和目标节点，对查询从源节点到目标分组内的节点所经历的跳数和所花费的消息数进行测试。从图 4 可以看到，由于查询步长是由 *Route_peer* 的个数决定的，分组越大，节点总数一定的前提下 *Route_peer* 个数就越少，因此基于分组 Chord# 的覆盖网络的查询步长随着每个分组大小的增加而减少。从图 5 中可以看到，由于当查询到达目标组后需要在组内进行广播才能确保组内节点收到查询，随着分组大小的增加，组内广播带来的消息数增长幅度远远大于跳数减少的幅度，因此基于 Chord# 的分组式覆盖网络的查询开销随着每个分组大小的增加而增加。

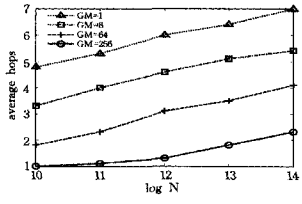

图 4 不同分组大小下的查询跳数

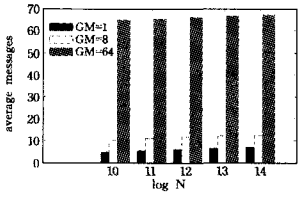

图 5 不同分组大小下的查询开销

第二部分包含 3 个测试。第一个和第二个测试为随机选取源节点和数据空间范围内的查询点，对查询从源节点到目标分组所经历的跳数和所花费的消息数进行测试。如图 6 所示，随着系统中节点数量的增加，点查询步长并没有显著增长，表明系统有较好的可扩展性。另外，随着分组内节点数量的增加，点查询步长逐步减少。根据前面的分析可以得出，点查询开销为 $O(\log N/M)^2+M$。如图 7 所示，点查询的消息开销随着 M 的增加表现为一种由大变小然后再变大的形式。从查询开销的理论分析和实际测试中能够得出，针对点查询开销存在一个最优分组成员个数。

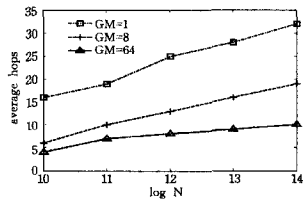

图 6 不同分组大小下的点查询跳数

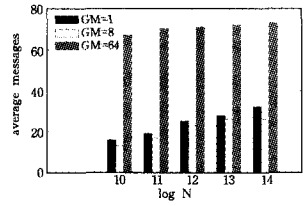

图 7 不同分组大小下的点查询开销

第三个测试为在不同规模的覆盖网络和分组大小下，随机生成其大小为整个数据空间 1% 的区域查询，并对查询所花费的消息数进行测试。从图 8 可以看出，分组大小为 8 的时候，消息开销最优；分组大小为 1 的查询开销在查询范围很小的时候和分组大小为 64 的情况差距不大，但随着查询范围的增加，前者的查询开销增长幅度高于后者。这是因为随着查询范围的增加，在分组间转发的消息数对查询开销的影响要大于在分组内多播的消息数对查询开销的影响。

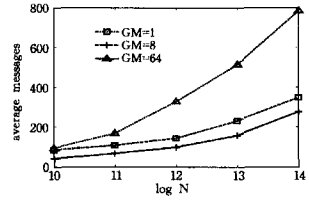

图 8 不同分组大小下的区域查询开销

结束语 提出 P2P 环境下基于分组 Chord# 的分布式空间索引，并描述了如何利用这种索引进行 P2P 空间查询。

测试表明，这种分布式索引的维护代价较低，利用其进行空间查询具有较好的可扩展性。分组的增加可以减少查询跳数。但对于查询的总开销，存在一个最优的分组成员个数。另外，提出的路由恢复机制可以较好地应对 *Route_peer* 失效的问题，增强了系统的可用性。

由于组内多播会导致很多无效的查询消息（不可能贡献查询结果的节点收到查询消息），因此，在下一步的工作中重点考虑如何在组内通过索引将查询消息进行再次过滤，从而实现减少查询开销的目的。另外，完善路由恢复机制，给出在路由恢复情况下的空间查询方法，并测试其对查询成功率、节点加入成功率等因素的影响也是下一步需要考虑的内容。

参 考 文 献

[1] Mondal A, Yi Li-fu, Kitsuregawa M. P2PR-Tree: An R-Tree-Based Spatial Index for Peer-to-Peer Environments[C]// Proceedings of the 9th International Conference on ExtendingDatabase Technology(EDBT '04). Heraklion, Crete, Greece, 2004: 516-525

[2] Tanin E, Harwood A, Samet H. Using a distributed quadtree index in peer-to-peer networks[J]. The VLDB Journal, 2007, 16(2): 165-178

[3] Stoica I, Morris R, Karger D, et al. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Application[C]// Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. San Diego, California, United States, 2001: 149-160

[4] Zhang Chi, Krishnamurthy A, Wang R Y. SkipIndex: Towards a Scalable Peer-to-Peer Index Service for High Dimensional Data

[R]. TR-703-04. Princeton University, 2004

[5] Wang Hao-jun, Zimmermann R, Ku W-S. ASPEN: An Adaptive Spatial Peer-to-Peer Network[C]// Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems. Bremen, Germany, 2005: 230-239

[6] Ratnasamy S, Francis P, Handley M, et al. A Scalable Content-Addressable Network[C]// Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. San Diego, California, United States, 2001: 161-172

[7] Ganesan P, Yang B, Garcia H. One Torus to Rule them All: Multidimensional Queries in P2P Systems[C]// Proceedings of the 7th International Workshop on the Web and Databases(WebDB '04). Pairs, France, 2004: 19-24

[8] Schutt T, Schintke F, Reinefeld A, et al. Structured Overlay without Consistent Hashing: Empirical Results[C]// Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid Workshops (CCGRIDW'06). Singapore, 2006: 8

[9] Guttman A. R-trees: A Dynamic Index Structure for Spatial Searching[C]// Proceedings of the 1984 ACM SIGMOD International Conference onManagement of Data. Boston, Massachusetts, United States, 1984: 47-57

[10] Samet H. The quadtree and related hierarehieal data struetures [J]. ACM Computer Surveys, 1984, 16(2): 187-260

[11] Bentley J L. Multidimensional binary search trees used for associative searching[J]. Communication of ACM, 1975, 18(9): 508-517

[12] Aspnes J, Shah G. Skip Graphs[J]. ACM Transactions on Algorithms, 2007, 3(4): 37

[13] Faloutsos C, Roseman S. Fractals for Secondary Key Retrieval [C]// Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '89). Philadelphia, Pennsylvania, United States, 1989: 247-252

(上接第 181 页)

制也不例外。所以在考虑系统安全性的同时，需要兼顾信息损失和代价两个因素对系统的影响。本节从 OLAP 系统实际应用的特殊性，给出了 QCS 方法的两种性能优化策略。

(1)优化历史推理集。基于查询的推理控制方法大多和历史查询集大小有关，系统的查询响应效率会随着查询集的增大而降低，QCS 方法也不例外。因为大多数用户一般不会基于一个长时间、大量的查询历史来推导敏感数据，所以在安全许可情况下，可定时删除过期历史查询集，或设定优化阈值来限制查询集大小等。当然这些策略必须对用户透明，以避免被用户恶意利用。

(2)基于角色的推理控制。基于角色的访问控制(RBAC)模型本质是把权限指派给角色而不是单个用户，所以，引用这种思想，在 QCS 方法中以角色为单位来维护查询历史推理集，这样既减少系统的性能负担，又可防止同角色里的用户间的串谋推理威胁。

结束语 本文着重讨论 OLAP 系统多维数据集的多维推理控制问题。提出的 QCS 方法能有效预防多维推理威胁，有较好的多维推理威胁检测效率。其算法的计算复杂性只与历史推理集长度有关，与底层数据集大小无关。QCS 方法是基于查询的动态推理控制，在提高系统安全性的同时牺牲了一定的在线查询响应时间。鉴于静态推理方法在这方面的优势，今后，我们将结合静动态推理控制各自的优势做进一步的优化研究；另外，本文的 QCS 方法没有考虑不同用户或角色间的串谋推理威胁，如何防止串谋推理威胁也是今后的研究重点。

参 考 文 献

[1] Hua M, Zhang S, Wang W, et al. FMC: An approach for privacy preserving OLAP[J]. Lecture Notes in Computer Science, 2005, 3589: 408-417

[2] Agrawal R, Srikant R, Thomas D. Privacy Preserving OLAP [C]// Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. Baltimore: Maryland, 2005: 251-262

[3] Zhang N, Zhao W. Privacy-Preserving OLAP: An Information-Theoretic Approach [J]. Knowledge and Data Engineering, 2011, 23(1): 122-138

[4] 周海清，陈启买，刘海. 基于数据立方体的数据仓库安全控制[J]. 计算机工程，2010，36(10)：152-154

[5] 毛奇正，柏文阳，刘奇志. 元数据相关推理研究[J]. 计算机科学，2004，31(11)：86-88

[6] Wang L, Wijesekera D, Jajodia S. Lattice-based Inference Control in Data Cubes [J]. Journal of Computer Security, 2004, 12(5): 655-692

[7] Wang L, Li Y, Jajodia S, et al. Preserving Privacy in On-Line Analytical Processing(OLAP)[J]. Advances in Information Security, 2007, 29: 147-167

[8] Denning D E, Schlorer J. Inference controls for statistical databases [J]. IEEE Computer, 1983, 16(7): 69-82

[9] Chin F, Ozsoyoglu G Y. Auditing and Inference Control in Statistical Databases[J]. IEEE Transactions on Software Engineering, 1982, 8(6): 574-582

[10] Chin F. Security Problems on Inference Control for Sum, Max, and Min Queries[J]. Journal of the ACM, 1986, 33(3): 451-464

[11] Kleinberg J, Papadimitriou C. Raghavan PAuditing Boolean Attributes[J]. Journal of Computer and System Sciences, 2003, 66(1): 244-253

[12] Wang L, Li Y, Wijesekera D, et al. Precisely Answering Multi-Dimensional Range Queries without Privacy Breaches[J]. Computer Security, 2003, 2808: 100-115

[13] Wang L, Li Y, Wijesekera D, et al. Cardinality-Based Inference Control in Data Cubes[J]. Journal of Computer Security, 2004, 12(5): 655-692

[14] Sung Y, Liu Y, Xiong H, et al. Privacy Preservation for Data Cubes[J]. Knowledge and Information Systems, 2006, 9(1): 38-61

[15] Li Y, Lu H, Deng R H. Practical Inference Control for Data Cubes[C]// Proceedings of IEEE Transactions on Dependable and Secure Computing. Canana: IEEE Computer Society, 2006: 115-120

[16] Gray J, Bosworth A, Bosworth A, et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals[J]. Data Mining and Knowledge Discovery, 1997, 1(1): 29-53

[17] Donnellan T. Lattice Theory[M]. Pergamon Press, Oxford, 1968