第11卷 第4期
1996年11月

环 境 遥 感
REMOTE SENSING OF ENVIRONMENT
CHINA

Vol.11, No.4
Nov., 1996

主存空间对象的索引方法

刘 东 李 琦 承继成

(北京大学遥感与地理信息系统研究所 北京 100871)

摘 要 空间索引关系到空间数据库和地理信息系统的整体性能。目前，随着计算机主存价格的迅速下降，发展主存空间数据库已经成为可能。主存空间数据库需要相适应的空间索引。本文设计两种面向主存的空间索引——主存网格索引和主存F_树索引，并对两者的性能进行比较。在多数应用环境下，F_树空间索引性能更优。

关键词 空间数据库，地理信息系统，主存空间数据库，空间索引，网格索引，F_树索引

1 引 言

空间索引是指依据空间对象的位置和形状，按一定顺序排列的一种数据结构，其中包含空间对象的概要信息如对象的标识、外接矩形及指向空间对象实体的指针。空间索引使空间操作能够快速访问操作对象，从而提高效率。

目前普遍认为，空间索引方法的采纳与否以及空间索引的性能优劣直接影响地理信息系统的整体性能，空间索引是地理信息系统的一项关键技术[1, 2]，因此各国的研究人员投入相当多的力量研究开发高效的空间索引[2—4]。常见的空间索引有R_树、R+_树、K_D_B树、Field树和Cell树等。各种空间索引都有一定的适用范围[5]。

综观目前的各种空间索引，它们多作如下假设：空间数据的数量巨大，其主体必然只能存在磁盘中，磁盘的读写速度比内存慢近3个数量级，且是按块读写的。现有空间索引的核心就是合理地划分二维数据空间，使落入每个子空间的空间对象能存储在相邻的磁盘块中。本文中统称这些空间索引为面向外存的空间索引。随着半导体存储器价格的迅速下降，当前个人计算机拥有几十兆乃至上百兆的主存已很常见，这样大的主存空间足以容纳绝大多数实际的空间数据库，事实上，目前很多GIS软件正是将空间数据全部装入主存来处理的。退一步考虑，即使空间数据库的规模大于上面谈到的规模，也有理由认为与操作相关的数据存储在主存中，毕竟空间数据是为人服务的，而人不会同时关心太多的细节。基于上述考虑，作者认为：在空间索引的研究中继续假设空间数据的主体存在于磁盘上是不合适的，目前应侧重研究主存内空间数据的索引方法——面向主存的空间索引。

计算机的主存和磁盘性质差别较大，主要表现在：

(1) 访问主存要比访问磁盘快3个数量级，这正是发展主存数据库的主要动力。

(2) 磁盘是面向块操作的存储设备，访问时有一个固定代价——寻址时间，因此数

收稿日期：1995年9月11日；收到修改稿日期：1996年6月11日

据在磁盘上的组织很重要，如果能将相关的数据放入相邻的磁盘块中，则可减少寻址时间；与此形成对照的是主存允许随机存取，访问时间与数据的组织基本无关[6]。

在面向外存的空间索引中，影响效率的主要因素是算法访问磁盘的次数；而在面向主存的空间索引中，影响效率的主要因素是算法的计算量。二者虽然有本质不同，但在空间划分、空间对象分割等方面还是可以互相借鉴。本文借鉴面向外存的空间索引，提出并实现两种面向主存的空间索引 —— 主存网格空间索引和主存 F_树空间索引，并用实际的空间数据集对它们进行测试和比较。

2　主存网格空间索引

主存网格空间索引是一种相对简单的空间索引，它规则地将二维数据空间划分为大小相等的网格，每个网格分配一个动态存储区，全部或部分落入该网格的空间对象的标识及外接矩形存入该网格的动态存储区内，其中存储外接矩形信息是为了使索引在执行查询时更好地筛选空间对象，而无须进行多级网格划分[7]。如图 1，点状空间对象只属于一个网格；而线状和面状空间对象则有可能属于多个网格。可见，主存网格空间索引是一种分割空间对象的索引方法。

主存网格空间索引的基本操作主要包括空间对象的插入、删除和查询等，其中查询包括点查询和域查询，点查询是搜索与指定点最近的空间对象；域查询是搜索与指定区域具有特定空间关系（如相交、包含等）的空间对象。点查询可以看成是域查询的特例。本文中主要以域查询的时间作为指标来衡量空间索引的性能，其中域查询要满足的空间关系是相交或包含，以下提到的查询都是这种域查询。下面分别介绍主存网格空间索引中空间对象插入和域查询算法，删除与插入类似。注意到空间索引只与空间对象的外接矩形和标识发生作用，因此在设计空间索引算法时无须考虑空间对象是点、线或面。

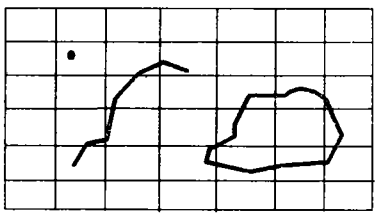

图 1　网格索引的空间划分

Fig.1　Space partition of grid index

插入算法

输入：空间对象的标识 ID 和外接矩形 MER 以及主存网格空间索引。

输出：加入上述空间对象后的主存网格空间索引。

GI1　确定 MER 全部或部分覆盖的网格集合 G。

GI2　对于任意网格 $g \in G$，将 ID 和 MER 加入其动态存储区内。

GI3　返回。

域查询算法

输入：查询窗口 W 和主存网格空间索引。

输出：满足查询条件的空间对象的集合 C。

GQ1　确定 W 全部或部分覆盖的网格的集合 G。

GQ2 对于任意网格 g∈G，检查其动态存储区内每一个空间对象的 MER 是否被 W 全部或部分覆盖，如果这样将其 ID 放入集合 C。

GQ3 整理集合 C，使其中的 ID 值不重复。

GQ4 返回。

3 主存 F_树空间索引

主存 F_树空间索引是自顶向下逐级划分空间的[5, 8]。如图 2，整个空间构成第一级划分；以后各级的划分与四叉树很类似，都是将上级区域（父区域）分为 4 个规则的子区域，不同之处在于每个子区域边长比四叉树子区域边长大，这样划分空间的好处是可以避免被四叉树子区域边界所切割的小空间对象进入上级区域，如图 2 中的椭圆和长方形。如果按四叉树方法划分空间，它们将属于第一级划分 —— 整个空间，而如果按 F_树空间划分它们就属于下级区域，从而使空间对象的分布在整个数据结构中趋于合理，有利于查询时快速排除不相关的空间对象。显然，当 c=0 时，F_树的空间划分就退化为四叉树的空间划分。

F_树空间索引的规模由它的级数（L）决定，级数越多，空间划分就越细致，总的区域数为 $\sum_{i=0}^{L} 2^i \times 2^i$。如果空间对象全部落入某一区域就称它属于该区域，如果空间对象同时属于不同级数的区域，就将它分配给级数最大的区域，这样的空间划分不会分割空间对象，这与主存网格空间索引形成鲜明对照；如果空间对象同时属于同一级的多个兄弟区域，如图 2 中的椭圆就同时属于区域 1 和 3，此时规定将这样的空间对象分配给区域号码较小的区域，于是图 2 中的椭圆就属于区域 1。

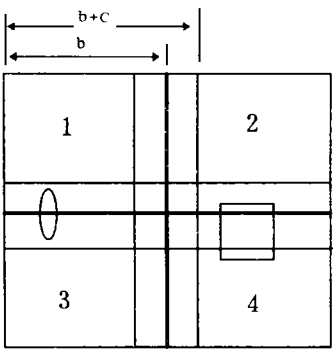

图 2 F_树索引的空间划分

Fig.2 Space partition of fieldtree index

主存 F_树空间索引的基本操作与主存网格空间索引相似，主要也包括空间对象的插入、删除和查询等；不同之处在于 F_树索外是逐级划分空间的，因而特别适于采用递归算法处理，下面给出插入和查询的递归算法。

插入算法

输入：空间对象的标识 ID 和外接矩形 MER 以及主存 F－树空间索引的一个区域 R。

输出：加入上述空间对象后的主存 F_树空间索引。

FI1. 如果 R 是最后一级区域，将空间对象的标识 ID 和外接矩形 MER 存入当前 R 的动态存储区内，然后转入 FI4。

FI2. 检查 R 的 4 个子区域{S1, S2, S3, S4}能否容纳空间对象，如果能就以参数（ID, MER, Si）调用插入算法（递归调用），然后转入 FI4；否则执行 FI3。

FI3. 将空间对象的标识 ID 和外接矩形 MER 存入区域 R 的动态存储区内。

FI4. 返回。

域查询算法

输入：查询窗口 W 和主存 F_树空间索引的一个区域 R。

输出：满足查询条件的空间对象的集合 C。

FQ1. 如果查询窗口 W 全部或部分落入区域 R，检查区域 R 的动态存储区内的所有空间对象，将全部或部分落入查询窗口 W 的空间对象的 ID 值放入集合 C 中；否则转入 FQ3。

FQ2. 设区域 R 的 4 个子区域分别为 S1、S2、S3 和 S4，分别以 (W, S1)、(W, S2)、(W, S3) 和 (W, S4) 调用域查询算法 (递归调用)。

FQ3. 返回。

4 实验结果及讨论

为验证上面两种主存空间索引的正确性及效率，我们采用有代表性的实际空间数据集合进行了测试，这里给出一些规律性的结果。以下测试结果用到的空间数据集合为西北某县土地利用图，数据空间的尺寸为 4000×4000，包含 5483 个弧段，1447 个多边形，数据量约为 1.8 兆。弧段外接矩形的平均大小为 40×40，空间对象在数据空间中的分布较为均匀，个别区域中空间对象的面积较大，这些区域约占总面积的十分之一。实验中的所有空间索引都是面向弧段的。实验系统用 C++ 语言实现，操作系统为中文 Windows 3.2 版，硬件环境为 486 /66 微机，主存 32 兆。下文提到的查询窗口是指边长为指定大小的正方形，查询时间为 500 次查询的总时间，其中每次查询窗口大小相同、位置随机分布。

4.1 参数的确定

从上两节的说明中可以看到，两种主存空间索引方法各有两个参数，主存网格空间索引需要指定水平和垂直方向的网格数 M 和 N，因实验数据是正方形且分布较为均匀，故只考虑一个方向的网格数 N；主存 F_树空间索引需要指定 F_树的级数和两相邻区域的重叠系数 d≡c /2b，b 和 c 的含义见图 2。

图 3 显示主存网格空间索引下查询时间随网格数 N 和查询窗口而变化。当网格数较小时，每个网格内的空间对象较多，小窗口查询时要依次检查这些空间对象，并排除其中多数空间对象，因而查询时间长；但在大窗口查询时，由于空间对象在不同网格内重复出现的概率比网格数大的空间索引小，查询

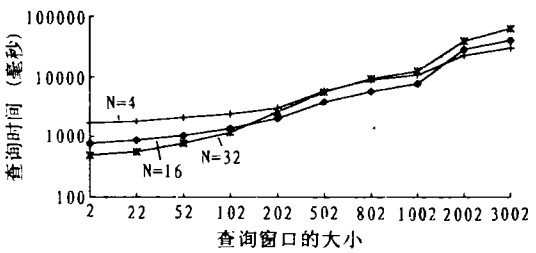

图 3　网格索引下查询时间随参数 N 的变化

Fig.3　Query time versus query window size for various N of grid index

时间反倒比网格数大的空间索引短。为了兼顾不同窗口大小的查询要求，一般折中地选N为8—16，即相当于每个网格的边长是空间对象外接矩形平均边长的5—10倍。

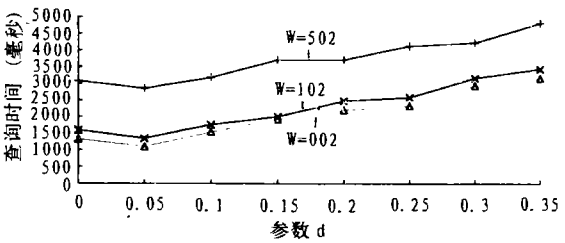

图4 F_树下查询时间与参数d的关系(L=3)

Fig.4 Query time versus parameter d of fieldtree index (L=3)

图4显示主存F_树空间索引下查询时间随参数d和查询窗口而变化（横坐标轴为参数d），可以看到查询时间在d=0.05时取极小值以后随着d增大而增大；当F_树退化成四叉树时(d=0)，性能不及d=0.05时的F_树，原因在于F_树的每个子区域都大于相应四叉树的子区域，且子区域之间有重叠，这样可以避免很多小的空间对象进入上级区域，从而使空间索引对空间对象的归类更加合理，减少查询时的判断次数，缩短查询时间。但d不是越大越好，d大到一定程度，下级区域的大小逐步逼近上级区域，上面讲到的优势逐步消失，查询时间开始上升。

图5显示主存F_树空间索引下的查询时间在不同级数L下随查询窗口的变化。可以看到：小窗口查询时，不同的级数查询时间差别很大。原因是级数小，F_树的规模就小，空间划分不够细致，每个区域有很多空间对象，小窗口查询是要排除其中的大部分，因而查询时间长，在这一点上与小网格数的网格索引相似；随着查询窗口的加大，需要排除的空间对象逐步减少，因而不同级数空间索引的查询时间差别逐渐减小，最后相差无几。从图5中还可以看到：L=5和7时查询时间差别不大，这说明当空间划分细致到一定程度后进一步的划分实际意义不大，空间划分的准则与空间网格索引相似，一般是最后一级区域的大小为空间对象平均大小的5—10倍。

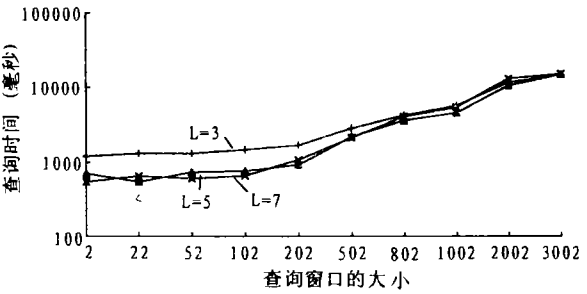

图5 F_树索引下查询时间与参数L的关系(d=0.05)

Fig.5 Query time versus query window size for various L of fieldtree index (d=0.05)

4.2 查询时间的比较

研究空间索引的目的在于提高空间操作的效率。空间索引的查询功能较好地代表了空间索引这方面性能。图6显示主存网格索引(N=16)、主存F_树索引(d=0.05, L=5)及不加空间索引时查询时间的比较，从中可以明显看到：主存F_树索引下的查询时间在所有窗口条件下都小于主存网格索引下的查询时间，约为后者的30%—50%，这表明主存F_树索引的查询性能明显优于主存网格索引。原因除前面提到的之外，

还注意到主存网格索引查询算法的 GQ3 是非常耗时的，主存 F_树索引的查询算法由于空间对象不可能重复出现在不同区域内而无须上述步骤，从而节省时间。这也从一个侧面反映了自顶向下逐级划分空间、不分割空间对象的空间索引要比一次性划分空间、分割空间对象的空间索引在查询性能上要好。从图 6 中还可以看到：有无空间索引查询时间相差 5— 50 倍(网格索引下大窗口查询时除外)。这充分显示空间索引的使用价值。

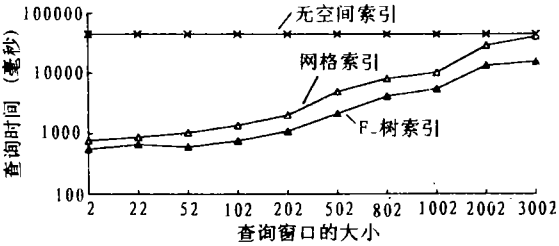

图 6　查询时间的比较

Fig.6　Comparison of query time

5　结　论

在计算机主存价格飞速下降的今天，有必要发展主存空间数据库及相应的主存空间索引方法。在给出的两种空间索引方法中，主存 F_树空间索引的性能都明显好于主存网格空间索引。

值得指出的是虽然强调发展主存空间索引，但并不排斥空间数据库的数据存储在外存设备中，通过面向主存的空间索引方法与面向外存的空间索引方法相结合，可构成处理海量空间数据的空间索引系统。

参　考　文　献

[1] 吴炳方，张明金，李新功. 地理信息系统的发展. 地理学报，1994，增刊.

[2] Ooi, B. C. Efficient Query Processing in Geographic Information Systems. Lecture Notes in Computer Science, Springer, Berlin, 1990, **471**.

[3] Hans－Werner, S., Widmayer, P. Spatial Access Structure of Geometric Databases. Lecture Notes in Computer Science, Springer, Berlin, 1992, **594**.

[4] Gunther, O., Bilmes, J. Tree－Based Access Methods for Spatial Databases: Implementation and Performance Evaluation. IEEE Trans. on Knowledge and Data Engineering Sep. 1991, 3(3).

[5] Frank, A., Barrera, R. The Fieldtree: A Data Structure for Geographic Information Systems. Design and Implementation of Large Spatial Databases. First Symposium SSD'89 July, Santa barbara, CA. 1989.

[6] Garcia－Molina, H., Salem, K. Main Memory Database Systems: An Overview. IEEE Trans. on Knowledge and Data Engineering, Dec. 1992, 4(6).

[7] 肖伟器，冯玉才，缪勇武. 空间对象数据库的网格索引机制. 计算机科学，1994，(10).

[8] Wenke Lee. Data Modeling and Management of Large Spatial Databases. Proceedings of the 3rd International Workshop on Geographic Information System, Beijing. Aug. 19－22, 1993.

作 者 简 介

刘 东，男，1968年6月生，1990年毕业于北京大学无线电系，获学士学位；1996年7月毕业于北京大学遥感与地理信息系统研究所，获博士学位。现为IBM中国研究中心从事计算机相关研究工作。

Indexing on Main Memory Spatial Object

Liu Dong Li Qi Cheng Jicheng

(*Institute of Remote Sensing & GIS, Peking University*)

Abstract Spatial Index is one of the key techniques that play an important role in spatial database and GIS. Currently, with the quick price reduction of computer main memory, It is possible to develop main memory spatial database. Main memory spatial database needs its own spatial indices. Two kinds of main memory oriented spatial indices have been put forward in this paper, the basic ideas of which can be traced back to disk oriented spatial index. These two main memory spatial indices are main memory grid index and main memory fieldtree index. Comparison of these two spatial indices has been made. Test results show that in most case the main memory fieldtree index gives better performance over the grid index.

Key words Spatial database, GIS, Main memory spatial database, Spatial index, Grid index, Fieldtree index