

文章编号: 1007-6301 (2001) 增刊-0069-09

地图数据库研究

王宇翔¹, 张 燕², 杨崇俊¹

(1. 中国科学院遥感应用研究所, 北京 100101;

2. 中国科学院地理科学与资源研究所, 北京 100101)

摘要: 本文在分析地图数据特征的基础上, 针对如何提高地图空间查询效率这一难题, 从多种地图索引方法中选择了 R⁺ 树索引方法, 实现了 R⁺ 树索引的算法, 最后给出了一个包括地图数据存储和索引、查询功能的地图数据库实例。

关 键 词: 地图数据; 地图索引; 地图查询; 地图数据库

中图分类号: P208; P283.7 **文献标识码:** A

1 地图数据的特征

地图数据是描述与地理位置、地理空间关系有关信息的载体。地图数据是对现实世界中的地物和地貌特征的抽象。地图数据除了具有一般数据的特征之外, 还具有一些区别于其它数据的特性。地图数据的特征主要包括:

(1) 空间性。这是地图数据最主要的特性。它描述了空间物体的地理位置、形状和周围地物的空间位置拓扑关系。例如描述一条河流, 一般数据侧重于河流的流域面积, 水流量, 枯水期等, 而地图数据则侧重于河流的位置、长度、发源地等和空间位置有关的信息, 以及河流与流域内其它的距离、方位等空间关系。空间性是地图数据区别于其它数据的标志特征。

(2) 多尺度与多态性。不同的观察尺度具有不同的比例尺和不同的精度, 同一地物在不同的比例尺下就会有形态差异。最典型的例子有: 就形态而言, 任何城市在地图中都占据一定范围的区域, 因此可以认为其是面状地物, 但在比例尺比较小的地图数据库中, 城市是作为点状地物来处理的。

(3) 多时空性。地图数据具有很强的时空特性。一个地图数据库中的数据源既有同一时间不同空间的数据系列, 也有同一空间不同时间序列的数据。不仅如此, 地图数据库会根据实际需要而采用不同尺度对地图进行表达。地图数据是包括不同时空和不同尺度数据的集合。

收稿日期: 2001-05; **修订日期:** 2001-08

基金项目: 973 计划项目 (G200077904); 国家自然科学基金项目 (40071065); 中国科学院地理科学与资源研究所创新前沿项目 (CX10G-D00-03-02)

作者简介: 王宇翔 (1975-), 男, 理学硕士, 现在中国科学院遥感所攻读博士。研究方向为网络地理信息系统、分布式空间数据库等。

2 地图数据存储方式的演变

早期的地图数据以文件方式存储,有的文件格式还保存地图数据的拓扑关系。例如早期 Arc/Info 所采用的“节点—弧段—多边形”的数据模型。这种数据模型处理单个对象的能力很弱,修改一个对象的时候,会牵涉到其他的对象。由于需要维护拓扑关系,不便于对地图数据的更新修改。由于 CAD 软件的制图功能强大,大量的 GIS 数据是存储在 CAD 文件中的,如 Autodesk 的 DWG 文件格式和 Microstation 的 DGN 格式。但是 CAD 系统所管理的数据文件一般情况下都比较小,而且没有存储空间拓扑关系,无法满足海量地图数据量的要求。

以地图数据表示的地物不仅具有空间信息,而且具有很多的非空间的附属信息。如城市的人口、国民生产总值等,这些构成了地理元素的属性信息。在早期的文件中,由于存储地图数据和属性数据所采用的格式不同,在 GIS 系统中一般都将其分开存储。这样,维护两者的一致性并进行一体化管理便成为必须解决的问题。

随着面向对象思想的出现和面向对象方法学的应用,人们开始用面向对象的思想来进行地图数据模型的设计。按照面向对象思想,每种地物都可以被抽象为某一类具有公共属性的对象,如点、线、面等,具体的地物则是该对象的一个实例,它还具有自己的属性。各种对象分层管理。这样就解决了地图数据与属性数据的一体化管理。

早期的文件都是针对专门的软件的,各个 GIS 软件厂商都有自己的文件格式,无法共享数据。随着 GIS 的发展,出现了一些数据格式标准化组织,提出了统一地图数据编码的思想,以便在不同 GIS 系统之间进行数据交换。但各种文件格式之间的数据转换依然费时费力,这就导致了 GIS 系统的建设成本居高不下。对于 GIS 厂商来说,由于需要文件格式的升级和向下兼容,导致开发负担大大加重。

由于地图数据的特殊性,它要处理的数据都具有空间特征和空间关系,难以在关系模型中表达。利用关系数据库来存储地图数据在理论与技术上都遭遇了暂时的困难。这个阶段,有些系统开始利用关系数据库来存储属性数据,而地图数据保持原有文件结构不变,通过在地图数据和属性数据之间建立关联的方法架起二者的桥梁。

纵观地理信息系统的发展,最初的数据是适应特定的系统的,每一个系统都拥有自己的数据格式。并有特定的数据结构来适应程序和算法,这时考虑的重点是系统的专有性;随着数据量的增多,数据共享需求的提出,数据的重要地位越来越突出,各个系统开始考虑数据的统一问题,并且提出了一些改良方案。最后初步形成了目前集各种技术之大成的地图数据库系统,标志地图数据库技术和地图数据库系统的初步成熟。而且,基于关系数据库或者对象关系数据库的地图数据管理正在逐步成为 GIS 发展的潮流。

空间数据库最关键的技术是对空间数据的存储、查询和空间索引。

3 空间查询与索引

3.1 空间查询的分类

对地理信息的空间查询可以归纳为:区域查询、点查询、空间连接和最近邻查询 4 种。

区域查询是指在空间中给定一个区域(通常为矩形),检索出所有在空间中与该区域相交的空间对象。点查询是在空间中给定一个点,检索出所有包含该点的空间对象。点查询是区域查询的一个特例,当给定的矩形收缩成一个点时,区域查询退缩成一个点查询。空间连接是按照包含空间谓词的几何属性将两个数据集中的空间对象进行合并,空间谓词可能为“相交”、“包含”和“有一定距离”等。最近邻查询是检索出与指定对象距离最近的空间对象。

3.2 空间索引综述

为快速响应用户提交的空间查询要求,地图数据库系统不仅要属性数据作很好的索引,更要求对地图数据作空间索引(Spatial Index),以便提高各种空间操作的效率。与一般的数据库系统相比,地图数据库中空间对象的表达形式复杂。数据量大,各种空间操作不仅计算量巨大,而且大多具有面向邻域的特点。如果能在各种空间操作之前对操作对象作初步的筛选,则可大大减少参加空间操作的空间对象数量,从而缩短计算时间,提高查询的效率。空间索引就是为此而设计的。

所谓空间索引就是指依据空间对象的位置和形状或空间对象之间的某种空间关系,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。作为一种辅助性的地图数据结构,空间索引介于空间操作算法和空间对象之间,通过它的筛选,大量与特定空间操作无关的空间对象被排除,从而提高空间操作的效率。

空间索引方法的采纳与否以及空间索引性能的优劣直接影响地图数据库和地理信息系统的整体性能,空间索引方法是地图数据库和地理信息系统的一项关键技术,有必要将空间索引当作地图数据库体系结构中重要一环加以研究。因此各国研究人员投入相当多的力量研究开发高效的空間索引方法。著名的商业数据库厂商在支持地图数据时也采用了空间索引的方法。常见的空间索引一般是自顶向下、逐级划分空间的各种树结构空间索引,比较有代表性的包括BSP树、K_D_B树、R_树等。下面对它们作简单介绍。

早期的空间索引方法没有顾及外存按页存储的特性,因而不适宜用来处理海量的地图数据。这类方法中比较典型的是BSP树(Binary Space Partitioning Tree,二值空间划分树)。BSP树是一种二叉树,代表了逐级将空间一分为二的过程,图1显示了二值空间划分的过程及相应的BSP树。BSP树能很好地与地图数据库中空间对象的分布情况相适应。但一般来说,BSP树的深度较大,对各种操作均有不利的影响。

K_D_B树是较早提出的面向外存的空间索引结构,K_D_B树是BSP树向多维空间发展的一种形式,K_D_B树对多维空间中的点进行索引,具有较好的动态特性,删除或添加空间对象可以很方便地实现,无需周期性地调整索引自身结构。K_D_B树的缺点

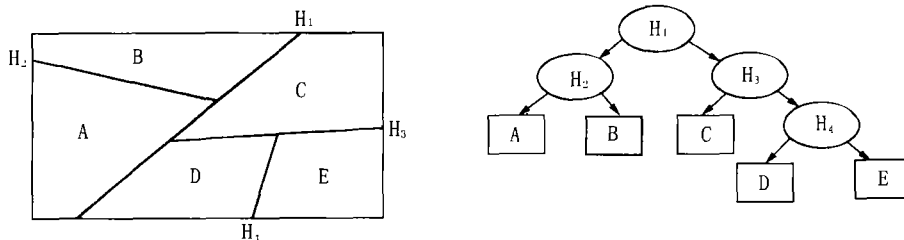


图1 二值空间划分及相应的BSP树

Fig. 1 Binary Spatial Split and Correspond BSP tree

是不直接支持占据一定空间范围的空间对象,如二维空间中的线和面。可以通过空间映射或变换的方法部分地解决这个问题,空间映射或变换就是将 d 维空间中的区间变换到 $2d$ 维空间中的点,这样,利用点索引结构就有能对区间进行索引,原始空间中的区间查询就转变为高维空间中的点查询。但这样的方法有一定的缺点:首先在高维空间中的点查询比原始空间中的点查询困难得多;其次,经过变换,原始空间中相邻的区间可能在点空间中距离相当遥远。在实际应用中,这些问题会严重影响空间索引的性能。

与 K_D_B 树不同, R 树空间索引直接对空间中占据一定范围的空间对象进行索引。 R 树空间索引也是 B 树向多维空间发展的一种形式,它的结构和对空间的划分如图2。 R 树的每一个节点 N 都对应着磁盘页 $D(N)$ 和区域 $I(N)$,如果节点不是叶节点,则该节点的所有子节点的区域都在区域 $I(N)$ 的范围之内,且存储在磁盘页 $D(N)$ 中;如果节点是叶节点,那么磁盘页 $D(N)$ 中存储的将是区域 $I(N)$ 范围之内的一系列子区域,子区域紧紧围绕空间对象,一般可以是空间对象的外接矩形。

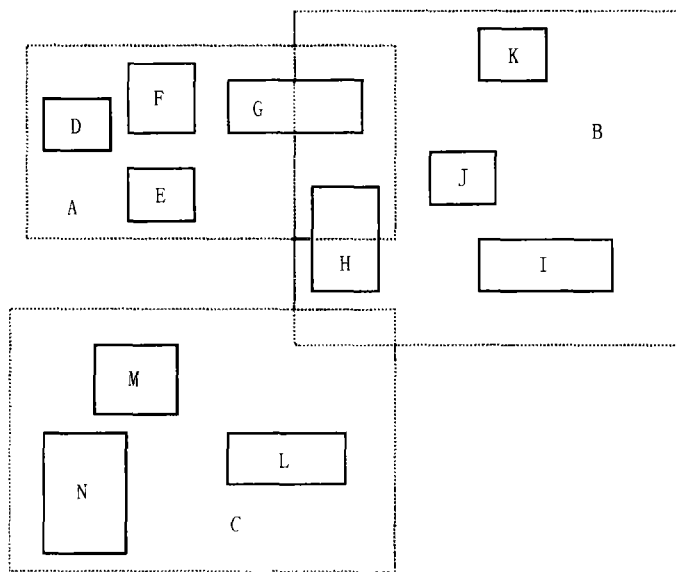


图2 R 树划分的几个矩形

Fig. 2 Some Rectangles Organized into R_tree

与 B 树的性质相近, R 树中每个节点所能拥有的子节点数目是有上限和下限的。下限保证索引对磁盘空间的有效利用。子节点数目小于下限的节点将被删除,它的子节点被分配到其余的节点中;设立上限的原因是因为每个节点只对应一个磁盘页,如果节点要求的空间大于一个磁盘页,那么这个节点就一分为二,子节点被分配到这两个新的节点中。

R 树有一个重要的特点就是兄弟节点对应的空间区域可以互相重叠,图2所示的矩形组织成 R 树见图3,这样的特性使 R 树比较容易进行删除和插入操作,但却使空间搜索的效率降低。

因为区域之间有重叠,可能要对多条路径进行搜索后才能得到最后的结果。这样的不足促使人们研究区域之间没有重叠的索引方法,这就发展了 R^+ 树。 R^+ 树对空间的划分如图4。

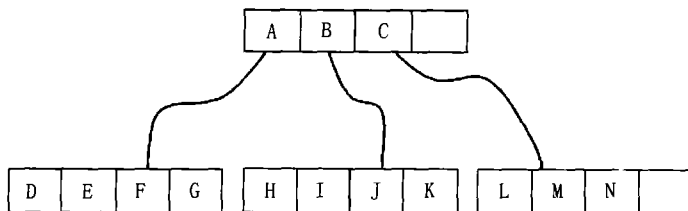
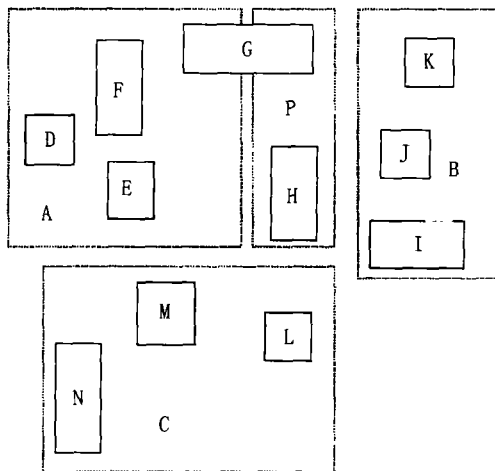


图3 图2所示矩形的R_树

Fig. 3 R_tree for The Rectangles of Fig. 2

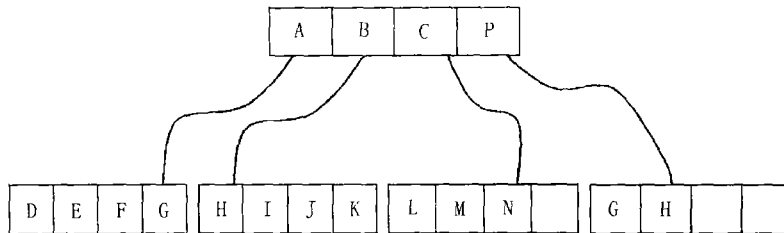
图4 图2的几个矩形组织成R⁺树时的划分Fig. 4 The Rectangles of Fig. 2 Grouped to Form R⁺_tree

3.3 R⁺树索引

3.3.1 R⁺树索引的特征

在R⁺树中,兄弟节点对应的空间区域没有重叠,这样划分空间可以使空间搜索的效率提高。图2的矩形在R⁺树中划分为图4,图4的矩形组织成R⁺树见图5。覆盖区域的大小和叠置是影响R树搜索性能的重要因素。某一层的R树的覆盖范围是能包含这一层的所有节点覆盖范围的最小范围。叠置是某一层的两个和多个节点的公共区域的总和。显然,有效的R树搜索要求覆盖范围和叠置区域最小。最小的覆盖范围减少了节点的死区域。死区域是不存放几何信息的区域。如果一个查询窗口落在K个相互叠置的区域内,那么最坏的情况下,要遍历K个子节点,因此降低了搜索效率。实际表明,无叠置情况只有在数据点是预先知道的情况下出现,并且对R树使用紧缩技术,搜索效率会显著的提高。但是,如果采取分割矩形的方法会避免在中间节点之间出现叠置。当较低层出现矩形叠置时,把矩形分割成几个子矩形。矩形是包围空间实体的最小矩形。避免叠置的同时增加了空间范围,增加了树的深度。空间范围的增加在整个树中呈现对数分布。深度的增加远小于搜索多个短路径的开支。

R⁺树可以看作是K_D_B树在多维区域的扩展,使它能处理零维以外的空间对象。它的改进表现在覆盖区域的减少,某一层的区域不必覆盖整个初始化区域。而且和R_树比较,R⁺树表现了非常好的查找性能尤其是点查询。

图 5 图 4 的矩形组织成的 R^+ 树Fig. 5 The R^+ tree Built for Fig. 4

R^+ 树算法的特征可以概括如下:

叶节点: (oid, RECT)。oid 是对象标识符, 指向数据库中的对象, RECT 是描述数据对象的最小外接矩形。在二维空间中, 一个数据项矩形描述如下: (Xlow, Xhigh, Ylow, Yhigh) 表示了矩形的左下角和右上角。

中间节点: (p, RECT), 其中 p 是指向它的子树的指针。M 是一个叶节点和中间节点的最大数据项数。 R^+ 树有以下特征:

(1) 对于一个中间节点的每一个数据项 (p, RECT), p 指针所指的子树包含一个矩形 R, 当且仅当 R 被 RECT 所包含。唯一的例外是 R 是叶节点的矩形, 这时 R 必须和 RECT 相交。

(2) 对于一个节点的两个数据项 (p_1 , $RECT_1$) 和 (p_2 , $RECT_2$), 这两个矩形不相交。

(3) 根节点至少有两个子节点, 除非它是叶节点。

(4) 所有的叶节点在同一层。

3.3.2 R^+ 树索引算法

R^+ 树索引的算法包括 R^+ 树的建立、节点的插入、删除和对 R^+ 树的查询。其中最重要的是插入和删除。下面分别讲述插入和删除算法。

3.3.2.1 插入算法 (Insert) 在插入时, 由于 R^+ 树每个节点有最大数据项数目的限制, 如果超过时要对节点进行分裂, 因为各个兄弟节点之间的区域不重叠, 所以在插入时一个矩形可能要分为多个子矩形。插入算法是由插入 (Insert)、节点分裂 (Split node) 和分割 (Partition) 3 部分组成。

第一步 插入:

输入: 根节点是 R 的 R^+ 树和要插入的矩形 IR

输出: 插入 IR 后的 R^+ 树

方法: 找到 IR 要插入的位置, 然后插入到相应的叶节点

步骤:

(1) 找中间节点

如果 R 不是叶节点, 对于 R 的每一个数据项 (p, RECT); 比较 RECT 与 IR 是否相交, 如果相交, 调用 Insert (CHILD, IR), CHILD 是 p 指针指向的节点。

(2) 插入到叶节点

如果 R 是叶节点, 把 IR 插入到 R 中。如果插入后, R 的数据项数目超过了 M, 则调用节点分裂算法 SplitNode (R)。

当一个节点的数据项溢出时,需要用节点分割算法产生两个新节点。由于两个子节点在空间上不能重叠,所以找到一个很好的分隔线。找分隔线的算法和紧缩算法中使用的方法很相似,在分割部分详细描述。分割的向下传播是重要的。以图 6 为例: A 是 B 的父节点, B 是 C 的父节点,如果 A 节点必须被分割,则 B 和 C 也必须被分割。

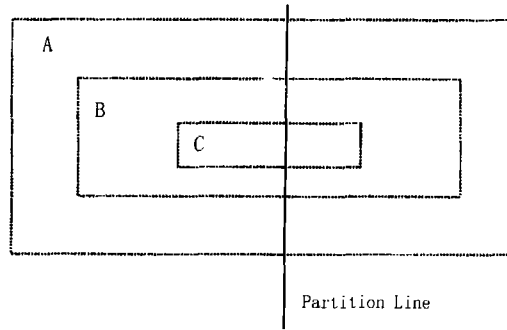


图 6 递归性节点分裂

Fig. 6 Recursive Node Splitting

这是由 R^+ 树的特性 1 决定的。 R^+ 树要求矩形 R 不应该在以 A 为根结点的子树发现,除非 R 被 A 节点的矩形所包含。因此,和分隔线相交的节点(除非它是叶节点)必须被递归的分割,直到它是叶节点。因为叶节点的对象不分割。

第二步: 分裂节点 (Split Node)

输入: 节点 R

输出: 新的 R^+ 树

方法: 找到分割线, 创建两个新节点, 如果需要, 向上和向下传播分割。

步骤:

(1) 调用 Partition 算法, 让 (p, RECT) 和节点 R 相关联

S_1 和 S_2 表示分割以后的矩形。创建两个节点 $n_1 = (p_1, \text{RECT}_1)$ 和 $n_2 = (p_2, \text{RECT}_2)$, 它们是由 R 分割后产生:

$$\text{RECT}_i = \text{RECT} \cap S_i, (i = 1, 2)$$

(2) 如果要加入的节点 p_k 不被分割后的矩形包含 $\text{RECT}_k \neq \text{RECT}_k \cap \text{RECT}_i (i = 1, 2)$; 如果 R 是叶节点, 把 RECT_k 加入到两个节点中; 否则继续使用节点分割算法递归的对子节点分割。

(3) 分割向上传播, 如果 R 是根结点, 创建新的根结点有两个子节点; 否则让 P_R 作为 R 的父节点, 把 R 用 n_1 和 n_2 代替。如果 P_R 超过了 M 个数据项, 继续分割。

第三步: 分割区域 (Partition region)

输入: 矩形集合和最小数据项数

输出: 包含第一子矩形的 R 节点和包含其它节点的另外的子矩形

方法: 把一个矩形分割为两个子矩形

步骤:

(1) 如果整个区域包含的矩形个数小于等于最小数据项数, 则返回。

(2) 计算整个区域的最小 X, Y 值。

(3) 沿 X 方向扫描, 找到 X 方向的分割值 C_x

(4) 沿 Y 方向扫描, 找到 Y 方向的分割值 C_y

(5) 用 C_x 和 C_y 的最小值分割区域, 产生两个子矩形, 把第一个子矩形内的元素赋给 R 节点, 剩余的矩形作为另外的子矩形返回。

R^+ 树和 R 树的插入算法的不同在于,要插入的矩形可能要加入到多个叶节点,因为它可能被分割成几个子矩形。溢出的结点被分割,分割即向父节点传播也向子节点传播。当父节点被分割后,子节点必须重新组织。

3.3.2.2 删除算法 (Delete)

从 R^+ 树中删除一个矩形和 R 树比较相似,先定位必须删除的矩形然后删除。

输入: 根结点是 R 的 R^+ 树和要插入的矩形 IR

输出: 删除 IR 后的 R^+ 树

方法: 找到 IR 的位置, 然后从相应的叶节点删除

步骤:

(1) 找中间节点

如果 R 不是叶节点, 对于 R 的每一个数据项 (p , $RECT$); 比较 $RECT$ 与 IR 是否相交, 如果相交, 调用 $Delete(CHILD, IR)$, $CHILD$ 是 p 指针指向的节点。

(2) 叶节点

如果 R 是叶节点, 把 IR 从 R 中删除。调整包括了剩下子矩形的父节点。显然大量删除后, 存储空间利用率会显著下降。 R^+ 树应该周期性的重新组织以获得较好的性能。

4 地图数据库实例

根据上面描述的算法, 我们设计了一个地图数据库 (GSD)。在 GSD 中, 地图数据是按照图层来组织的。把每一个空间位置的信息从逻辑上划分为图层。每一个图层中包含了不同的空间实体。由于地图数据的不确定性, 即一个地理元素的点的个数不确定, 普通的字段无法满足其需要, 目前多数大型数据库支持长二进制字段, 它允许存储任意长度的二进制数据; 这样把地图的点坐标信息存入长二进制字段。整个表的结构设计见表 1。在用 R^+ 树建立空间索引时, 需要几何体的最小外接矩形信息, 所以要存储 (X_{min} , X_{max} , Y_{min} , Y_{max}) 信息。图形元素的标识号是主关键字, 每一个图形元素的标识号码是唯一确定的。同时它和对应的非空间属性表中的表示号码对应, 是非空间属性表的外码, 这样保持了数据库的参照完整性。图形元素的类型设计参考 OPENGIS 规范, 定义常用的图形元素类型 (图 7)。

表 1 地理数据表

Tab. 1 Geographic data table

字段名	类 型
图形元素标识号	整 形
X 最小值	双精度
X 最大值	双精度
Y 最小值	双精度
Y 最大值	双精度
图形元素类型	自定义
图层名	字符串组
图形坐标信息	长二进制

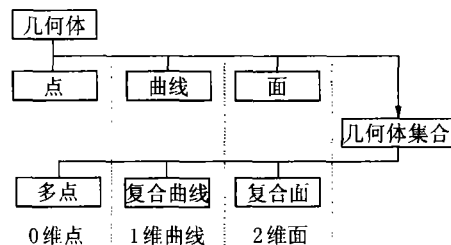


图 7 OpenGIS 定义的地理元素类型

Fig. 7 Geographic Element Types by OpenGIS

5 结论与展望

地图数据库是存储和管理地图数据的有效方法,特别是对于海量地图数据,但关键技术在于地图数据的索引。索引不仅可以针对地图还可以处理遥感图象和三维图形。 R^+ 树索引是目前最先进的索引技术,但没有看到国内的地图数据库使用这种方法。目前 GML 语言是存储地图数据的新方法,如何与 GML 结合是需要深入研究的问题。

参考文献:

- [1] Antonm Guttman. R-trees: A dynamic index structure for spatial searching[J]. *ACM (Association for Computing Machinery)*, 1984, **6**: 47-57.
- [2] Zhang Zhiqiang. Integrating attribute and space characteristics in choropleth display and spatial data mining[J]. *Geographical Information Science*, 2000, **14**(6): 6543-566.
- [3] Robert Lsurini. Spatial multi-database topological continuity and indexing: a step towards seamless GIS data interoperability[J]. *Geographical Information Science*, 1998, **12**(4): 373-402.
- [4] 陈述彭, 鲁学军, 周成虎. 地理信息系统导论[M]. 北京: 科学出版社, 1999.
- [5] Hanan Samet. Spatial Data Structure[M]. VLDB, 1985.
- [6] www.digitalearth.net.cn
- [7] www.cs.umb.edu

Research of geographic Spatial Database

WANG Yu-xiang¹, ZHANG Yan², YANG Chong-jun¹

(1. Institute of Remote Sensing Applications, CAS, Beijing 100101, China;

2. Institute of Geographic Sciences and Natural Resources Research, CAS, Beijing 100101, China)

Abstract: In this article, properties of geographic spatial database are analyzed. Then an algorithm of R^+ tree spatial index is implemented for spatial index, to improve the spatial search efficiency. At last, an example of spatial database is given.

Key words: Geo-spatial data; Spatial index; Spatial query; Spatial database