计算机科学1996 Vol.23 №.3
56－57

电子地图数据库存贮文件的设计

詹舒波 张其善
（北京航空航天大学 北京100083）

摘 要 BANG file is a balanced and nested grid file. It is limited in handling complex objects. An enchanced BANG file is presented in this paper according to the characteristics of digital map. The capability of creating access file for complex objects is developed in the enchanced BANG file, and a shortened query time is proved in the file.

关键词 Digital map, Access file, Vector Database, R-Tree, K-D-B Tree, BANG file.

1 前 言

电子地图具有实时、多任务、动态等特点[1]，要提高电子地图的性能、缩短地图显示和查询时间，设计出拓扑明确的矢量数据库及有序的数据库存贮结构是关键。有序的数据存贮结构，在电子地图信息查询时，使搜索路径短，减少磁盘数据的读取次数，可加快查询响应时间；在地图显示时，系统能进行有效的图形剪裁，只刷新相应区域，可提高地图显示速度。

我们平常遇到的存贮文件，如：顺序文件、各种树索引文件、hash 结构等，都是通过单个关键词建立的，包括倒序文件，它们统称常规存贮文件，主要应用于定长记录的常规数据库。矢量数据库和常规数据库有很大不同，矢量数据库用点、线、面表示对象在 n 维空间的位置[1]，对数据库的操作不仅有简单的单个记录信息查询和范围查询，还有最短距离计算、路径计算、部分匹配等。矢量数据库的存贮文件为 n 维结构。用常规的存贮文件处理空间对象，存在许多缺点，实际效果不能令人满意[2]，因此应使用新的技术对矢量数据进行管理。

近年来，国外学者提出应用空间基数分区（radix partition of space）对空间数据进行管理[6]，已得出了几种空间数据存贮结构。其中有代表性的有：R-树、K-D-B 树，BANG 文件结构等，但在实际应用中，这些结构都存在不同的缺点和局限，不适应在电子地图系统中应用。本文根据电子地图的特点，以 BANG 文件为基础增强 BANG 文件结构，增强对复杂对象的处理。它继承了 BANG 文件的优点，有容易实现、查询时间短等优点。

2 空间数据存贮文件

所谓空间存贮文件，就是按照一定的逻辑结构（如树，顺序），把空间数据组织成有序记录存放在磁盘上。空间数据存贮文件的逻辑结构分为两大类[5]：①按属性分类存贮。比较查询技术，如两维树、多维树等，就属于该类。②按区域分类存贮（也可称空间基数分区）。R-树、Grid 文件都属于该类。属性分类存贮是从一维平衡树基础上发展起来的，因为多维数据与一维数据有很大不同。在应用中该类存贮结构问题较多，在空间数据存贮文件中已较少使用。近年来区域分类存贮已越来越多地受到重视，较有代表性的有 R-树、K-D-B 树、BANG 树等，下面就对它们的性能进行论述，以对空间存贮文件有一定认识，为提出新的存贮结构打下基础。

2.1 R-树

R 树是通过对空间数据区域递推分区并以分区后的区域作为关键词建立起的存贮结构。R 树类似 B 树，为多级动态平衡树，适用于处理 n 维空间对象，整个结构无须周期性地重新组织，图1a、b 表示了 R 树的结构，从中我们可看出对象与分区的包含与覆盖关系。

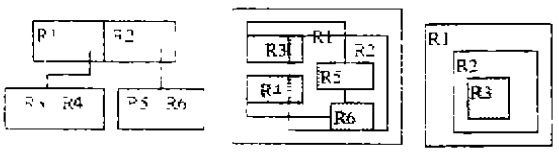

图1 R 树结构 图2

R-树中节点分为两种：叶节点和非叶节点。叶节点中包含记录形式为：(I, Data-Pointer)，其中：Data-Pointer 表示指针，指向对象在数据库的地址；I 为 n 维矩形。R-树规则规定：空间对象只有全部被 I 矩形包含，才存贮在记录中。非叶节点中包含记录形式为：(I, Child-Pointer)，其中：Child-Pointer 表示指针，指向子节点的地址；I 为 n 维矩形，完全包含了子节点的区域。

设M为每个节点包含的最多记录，m为节点包含的最少记录（一般m=M/2），则R-树具有如下特性：①每个根至少有两个孩子，除非它本身为叶；②每个非叶节点至少有m到M个孩子，除非它本身为根；③除非是根，每个叶节点包括m到M个记录；④所有叶节点在同一级上。

由上述规则可知：①R-树中节点空间利用率较高，至少为50%；②每个空间对象在R树中只表示一次，因此冗余小。③对象数据插入操作性能好。待插入对象从根开始，一次完成加入到叶节点。

但R树也有许多缺点。首先，由于R-树分区要完全包含区域中对象，为了表示空间区域中对象，R-树的分区将有重叠（如图1），重叠区域将发生在任意位置。当进行查询某一对象操作时，在最坏情况下，查找将搜索整个R树，如图1对R5物体的查询，因此查询速度慢。其次，当R-树叶节点溢出分裂时，象B树一样，该分裂可能导致其父节点溢出分裂，直到到达根节点。再次，R-树分裂算法对相互嵌套物体（图2）的操作将失败[6]，R-树的分裂规则为：分区区域小，尽可能远。这时R-树不能很好表示空间物体。

在电子地图中，图形放大剪裁、信息查询都应用查询操作，查询速度最为关键，因此R-树不适应在电子地图中应用。

2.2 K-D-B树

K-D-B树为多级平衡时，适用于处理n维点数据（简单对象）。K-D-B树通过把搜索区域分解成互不相交的子区域，并以这些子区域为关键词，为空间数据建立存贮文件[7]。图3表示了K-D-B树的结构。

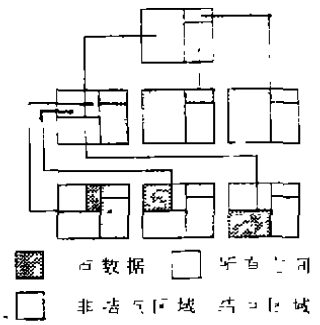

图3 K-D-B树结构

图4 K-D-B分裂图

K-D-B树中，节点分为两种：非叶节点和叶节点。非叶节点中包含记录形式为：(Region，Page-ID)。其中：Region为n维区域空间。n维区域可表示为$I_1 \times I_2 \times \cdots\cdots \times I_n$，其中：$min_i \leqslant x_i < max_i$，$0<i\leqslant n$；Page-ID类似指针，标识子节点的位置。叶节点中包含记录形式为：(Point，Location)。其中：Point表示n维点坐标；Location表示该点在数据库中的地址。

K-D-B树能较好地完成对n维点数据的查询等处理，但它也有许多不足，如节点的空间利用率不高、数据的插入、删除算法复杂、花费时间长等缺点。这些不足主要是由于K-D-B树的分裂规则造成的。

插入数据使叶节点溢出时，K-D-B树节点发生分裂。分裂将沿n维空间坐标之中的某一坐标进行。设原区域用$I_1 \times I_2 \times \cdots\cdots \times I_n$表示，其中$min_i \leqslant I_i < max_i$，选择$I_i$中的某一点$x_i$，$min_i \leqslant x_i < max_i$。分裂把原区域分解成两个子区域：

(1) $I_1 \times I_2 \times \cdots \times [min_j, x_j) \times \cdots \times I_n$　左区域

(2) $I_1 \times I_2 \times \cdots \times [x_j, max_j) \times \cdots \times I_n$　右区域

K-D-B树中区域不能相交，因此，当非叶节点的分裂线贯穿子节点区域时，其子节点要重新进行分裂计算，如图4，区域2点溢出分裂后，使区域3也分成2个区域。节点的这种被动分裂会破坏树的平衡，有可能使节点区域不包含任何点数据，指针为空，影响节点的利用率，同时节点的合并计算也很复杂，这使得K-D-B树的数据插入删除花费时间长。

K-D-B树不适合对复杂对象（线、面数据）的处理，树中数据是以n维点数据存贮的，因此不能充分利用线、面对象为连续区域这一特点。对复杂对象的查询，要映射成点数据的查询，这一映射过程也使查询更复杂、耗时长。

对主要以公路（线状）为对象的电子系统而言，K-D-B树显然也不满足要求。

2.3 BANG文件

BANG文件全称为平衡嵌套栅格文件，由Freeston最早提出[5]。其最初目的是设计一多维文件结构，以改进知识库的交互查询性能。近几年来，BANG文件在对n维点数据处理中，得到广泛重视。

BANG文件建立过程如下：通过递推分区，把空间区域分成许多栅格（子区域），每次分区都是按严格的二分法进行，即把区域分成大小相等的二个栅格（子区域），每个栅格被赋予一序号，并在目录中保留有一入口。目录是一逻辑数组，存贮在连续空间中，目录由目录项构成，每个目录项除包含栅格序号外，还包括与栅格序号相对应的指向存贮数据桶的指针。其示意如图5所示。具体操作性能见文[5]。

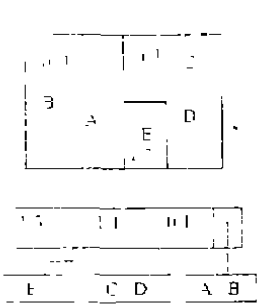

图5 BANG文件结构

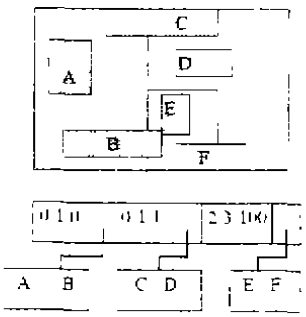

图6 扩展BANG文件结构

BANG 文件允许区域嵌套，目录利用率高。对点数据操作时，BANG 文件还具有如下优点：①空间对象在文件中无重复表示，冗余小。②对空间对象的操作，如删除、插入、查询所经路径短，因此运行速度快。③实现方便。

对复杂对象（线、面数据），BANG 是通过把它们映射成点数据进行管理的，性能并不理想。

Freeston 在文[6]中提出扩展 BANG 文件结构，其目的是改进 BANG 在处理线、面数据的性能。扩展 BANG 文件示意如图6所示。扩展 BANG 文件与原 BANG 的主要区别是目录表中对栅格序号的表示。为了处理复杂对象，扩展 BANG 文件目录采用如下格式：

[Page-Number， Point-Levl， Region-Prefix， Backet-Pointer]

其中：Page-Number：表示目录中对象覆盖的区域；Point-Level：表示目录中对象中心所在区域；Region-Prefix：为序列，表示 Point-Level 所在位置。其中：1表示栅格的右或上，0表示左或下；Backet-Pointer：表示与栅格相应数据桶的指针。

扩展型 BANG 文件中，每个对象在文件中只表示一次，冗余小；对点数据处理，扩展型 BANG 继承了 BANG 的优点，对复杂对象的插入操作速度快。但扩展型 BANG 文件也有明显的缺点，对复杂对象的查询在最坏情况下，将搜索整个文件。因此 BANG 文件及扩展型 BANG 文件也不适合在电子地图上应用。

3 增强型 BANG 文件

3.1 增强型 BANG 文件结构

在电子地图系统中，地图数据库以公路（线），地区（面）为主要研究对象。系统运行时，用户经常进行的操作为：窗口图形放大，信息查询等，而很少涉及对数据库的编辑。因此在设计地图数据库存贮文件时，数据库精确查询、范围查询等操作性能的优化应放在最先考虑的位置，而对数据库插入、删除、修改性能只稍加兼顾。根据电子地图上述要求，在 BANG 基础上，本文提出增强型 BANG 文件。增强型 BANG 文件和 BANG 文件一样，也采用严格的二分法，把空间区域递推分解成栅格，以目录形式进行管理。同样增强型 BANG 文件也遵守如下两条规则：①各栅格区的并集为原空间区域。②如果两个栅格相交，则其中一个栅格将完全被另一个包含。即允许栅格嵌套，但不存在部分相关。

但和 BANG 文件不同，增强型 BANG 文件不仅能对简单对象，还能对复杂对象（线、面）进行处理。简单对象与复杂对象不同，简单对象为零单位，在空间区域上只属于一个栅格，因此容易管理，而复杂对象为有形物体，可能跨越区域，属于多个栅格，因此管理复杂，为此增强型 BANG 文件规定：当复杂对象属于多个栅格时，增强型 BANG 文件相应的每个栅格都保留该对象的一个拷贝，结果如图7所示。

在增强型 BANG 文件中复杂对象可能有多个拷贝，这增加了数据库的冗余，为了尽可能地减少数据的冗余，就必须对增强型 BANG 文件分裂机制重新定义，与之相对应的合并机制、目录管理、插入等操作亦做相应改变。下面我们分别就这些方面进行讨论。

3.2 增强型 BANG 文件分裂与合并机制

向数据空间加入数据，引起某栅格溢出后，增强型 BANG 文件将进行分裂。分裂沿 n 维空间的某一坐标进行，把原栅格分成大小相等的两个子栅格。增强型 BANG 分裂机制与 BANG 不同，考虑到复杂对象引起的冗余，分裂坐标的选取原则为：

①使两个子栅格具有相同对象的数目尽可能小，这样可减少冗余；

②使两个子栅格尽可能平衡。

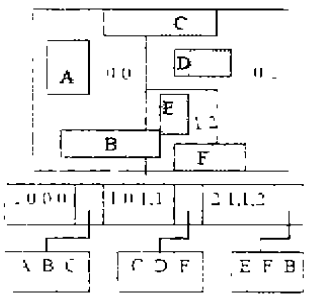

图 7 增强 BANG 文件结构

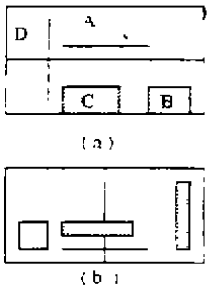

图 8 分裂机制

上述两个原则有时相互矛盾，这时就需要找到一折中方案。但优先考虑为减少子栅格中相同对象的数目。如图8a 所示的数据分布，我们将沿 y 轴分裂，这样只有线段 D 在两子栅格有拷贝；而不沿 X 轴，因为如果沿 X 轴分裂，曲面 C、线段 A 都将在子栅格中有拷贝，尽管此时两子栅格平衡。但对图8b，我们选 X 轴分裂，使两子栅格尽量平衡，因为沿 y 轴，两子栅格严重不平衡，并且不利于进一步分区。

数据删除后，某些栅格将进行合并，变成一个栅格，以提高贮存空间的利用率。考虑到数据冗余，增强型 BANG 文件合并按如下步骤进行：

①栅格与自己包含的子栅格合并。合并从包含的最小子栅格开始，如果合并后数据溢出或没有子栅格，操作失败，继续运行步骤②，否则成功退出。

②栅格与孪生栅格合并，如果失败，则运行步骤③，否则成功退出。孪生栅格指当栅格按严格两分法分区时，结果的两子栅格为孪生栅格。

③栅格与直接包含自己的栅格合并。标识成功与否退出操作。

3.3 增强型BANG文件目录管理

和其它栅格文件一样，增强型BANG文件目录的管理占有重要位置，它的主要任务为：维护栅格与数据桶之间的动态关系[2]。除此之外，增强BANG目录管理还完成如下功能：①把空间坐标映射成栅格序号，进行数据的存取；②为目录建立索引。当目录项多时，为了缩短目录查找时间，为目录项建立索引，加速查询时间。

增强BANG目录由多个目录项构成，目录项结构为：

(Level1,Level2,…,Leveln,Region-n1,Region-n2,…,Region-nn,Backet-Point)

其中：(Level1,Level2,…,Leveln)表示栅格相对全体空间的大小，Level i 为 i 方向的分区数；(Region-n1,Region-n2,…,Region-nn)表示栅格在 n 维空间中的序号；Backet-Pointer；表示与该栅格相应数据桶的指针。

增强型BANG文件采用的栅格序号与BANG不同，以二维空间为例，栅格序号如图9。与BANG文件相比，该栅格序号定义更直观，序号与分区顺序无关，同时由于采用字符计算，而非原先的位操作，使目录映射更简单，运行速度更快。

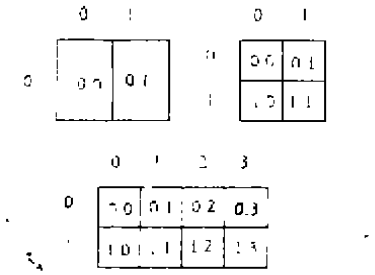

图9 序号定义

以二维空间为例，我们可推出坐标(x,y)与栅格序号的映射关系。设DX,DY为空间数据的最大值，为了计算简单，它常常取接近实际值的整数。level x,level y为x,y方向上的分区数。则(x,y)所在栅格序号(nx,ny)为：

$$nx=\left\lfloor \frac{x}{DX}\ 2^{level\,x} \right\rfloor \qquad ny=\left\lfloor \frac{y}{DY}\ 2^{level\,y} \right\rfloor$$

⌊X⌋表示取X的整数

增强型BANG文件中，目录项较小时，可直接用栅格序号匹配方法进行目录的检索，但在目录项较多时，常应用hash方法以栅格序号为关键字为目录项建立索引文件，实现快速寻址。

目录hash算法采用除取余的方法，使孪生栅格、相邻级的相同序号的hash值相近或相等，这样有利于范围检索。目录hash溢出处理采用分布式溢出空间[7]，使目录在磁盘上存贮空间连续。

以二维空间为例，当指定一点(x,y)时，查询包含该点的目录项算法如下：

```
;Sub Get-Dir,查找目录项
;输入参数坐标(x,y)
;输出目录项中的栅格序号
level:=Get—level( );得到分区总数
For D:=level—1 to 0 do begin;在不同级上查询
  nx:=Get—Region x(DX,x);
  ny:=Get—Region y(DY,y);
  i:=hash(D,nx,ny);栅格序号的 hash 值
  For J:=0 to num do begin
  K=Match (Dir[i,j],D,nx,ny);是否存在该栅格
  IF(K:=1)return dir[i,j]
  End For
End For
end
```

查询目录中栅格序号，首先查询包括该点的最小栅格区，如果没有找到(k≠1)，它就找上一级(level-1)包含该点的栅格，直到找到为止。

算法中，hash( )为以栅格序号(D,nx,ny)为关键字的hash值。

Match( )判断栅格(D,nx,ny)是否存在。如果存在则返回1，否则为0。

参 考 文 献

[1]詹舒波、张其善，电子地图综述(待发表)

[2]J. Nievergelt, The Grid File: An Adaptable, Symmetric Multikey File Structure, ACM Trans. On Database System, March 1984

[3]A. Guttman, R-Trees: A dynamic index structure for spatial search, Proc. ACM SIGMOD Conf. on Management of Data, 1984

[4]J. T. Robinson, The K-D-B Tree: A Serach Structure for Large Multidimensional Dynamic Indexes, Proc. ACM SIGMOD Conf. on Management of Data, 1981

[5]M. W. Freeston, The Bang File: A New Kind of Grid Fille, Proc. ACM SIGMOD Conf. on Management of Data, 1987

[6]M. W. Freeston, A well-behaved file structure for the structure of spatial objects, Proc. lst Symp. On the Design of Large Spatial Database (1989)

[7]郑若忠等，数据库原理与方法，湖南科学技术出版社

[8]J. Nievergelt: 7±2 Criteria for Assessing and Comparing Spatial Data Structures, same to [6]