第 22卷 第 3期 黑龙江大学自然科学学报 Vol. 22 No. 3
2005年 6月 JOURNAL OF NATURAL SCIENCE OF HEILONGJIANG UNIVERSITY June, 2005

空间数据库的索引技术

郭龙江[1,2], 李建中[1,2]
(1. 哈尔滨工业大学 计算机科学与技术学院, 黑龙江 哈尔滨 150001; 2. 黑龙江大学 计算机科学与技术学院, 黑龙江 哈尔滨 150080)

摘 要: 由于空间数据库中的数据量很大,因此空间数据库查询的开销一般要比关系数据库大,特别是查询语句的条件谓词中包含一些对空间数据操作的函数,计算这些函数的开销远比数值或字符串的比较要大。如果用顺序扫描的方法查询,则效率非常低。因此,为了提高查询效率,采用空间索引是十分必要的。目前人们的研究工作更多地集中在空间数据的多维索引的研究上。全面地总结了当前空间数据库领域中空间索引的研究进展,然后介绍了目前空间数据库中广为采用且比较新的 4种索引方法: (1) R树 (2) K-D树 (3) Quad树 (4) GiST。最后指出在空间数据库中的高维索引的研究是目前前沿研究的热点。

关键词: 空间数据;空间数据库;空间索引;高维索引
中图分类号: TU313 **文献标识码**: A **文章编号**: 1001-7011(2005)03-0288-06

1 引 言

近几年,人们对地理资源、地球环境以及地理信息进行研究时产生了大量的空间数据。与此同时,遥感、卫星图像处理、人体医疗成像等应用也提供了大量的空间数据信息。人们需要存储和管理大量的空间数据,同时还要在大量的空间数据上进行快速的查询和计算。目前的商用数据库管理系统 (DBMS)处理空间数据有一定的困难,因此人们提出了空间数据库的概念。由于空间数据库中的数据量很大,因此空间数据库查询的开销一般要比关系数据库大,特别是查询语句的条件谓词中包含一些对空间数据操作的函数,计算这些函数的开销远比数值或字符串的比较要大。如果用顺序扫描的方法查询,则效率非常低。因此,为了提高查询效率,采用空间索引是十分必要的。目前,人们的研究工作更多地集中在空间数据的索引研究上。

当前,人们已经提出了很多种多维索引技术。这些多维索引技术包括 Bang[7]文件,、网格文件[15]、hB树[14]、KDB树[16]、Paramid树[2]、四叉树[17]、R树[10]、R*树[1]、R+树、TV树和 VA文件[19],这些索引主要是用来对点的集合进行索引。能够同时处理区域数据和点数据的索引结构包括区域四叉树、R树和 SKD树。文献[9]讨论了针对由线性约束定义的区域如何用 R树进行索引。这些技术的一些变形和其它几种不同的技术也被提出来了, Samet的文章[18]处理了它们中间的很多问题。文献[8]是到目前为止最新的综述。文献[6]提出了线性化多维数据的 Hilbert曲线的使用。文献[4]讨论了空间连接的问题。为了能够适应更复杂的数据类型的索引,人们还提出了通用搜索树 GiST[11]。用户可以在自定义空间数据类型的基础上,自行定义树的插入、删除、修改、搜索操作,更有效地实现空间数据类型的查询。通用搜索树 GiST,它可以被实例化成各种树的索引。文献[13]讨论了 GiST的并发控制和恢复问题。文献[21]讨论了 GiST的并行化问题。文献[12]讨论了索引机制的复杂性,特别是针对区域查询。文献[3]讨论了高维索引这个问题。文献[5]是一个综述,总结并讨论了如何在基于内容的多媒体数据库中进行检索。最新的研究趋势是时空数据库的应用,文献[20]中讨论了跟踪移动对象等问题。

到目前为止还没有被一致认为是"最好"的空间索引结构。但是,应用最广泛的是 R树。R树有许多变

收稿日期: 2004-04-03
作者简介: 郭龙江 (1973-),男,博士研究生,讲师,主要研究方向:数据库,数据流上的数据挖掘技术,传感器网络, E-mail: guolongjiang@mail.banner.com.cn
通讯作者: 李建中 (1950-),男,教授,博士生导师,国家科学技术进步奖二等奖获得者, E-mail: lijz@mail.banner.com.cn

形,包括 Cell树、Hilbert R树、Packed R树、R*树、R +树、TV树和 X树。已经可以在商用的 DBMS中看到 R树索引。这是由于 R树相对简单,能同时处理点和区域数据,而且它的性能至少比那些更复杂的索引结构不差。

本文的内容安排如下:在第 2节中,讨论了 R -树的存储结构以及如何在 R -树上进行增、删、改操作以及搜索操作。第 3节给出了 K - D树的存储结构。第 4节给出了 Quad树的存储结构。第 5节介绍了 GiST树的存储结构以及函数操作接口。第 6节讨论了空间数据库的高维索引问题。

2　R -树

R树是处理空间数据的 B +树的改进,它像 B +树一样,是一个高度平衡的数据结构。R树的搜索码是区间的集合,一个区间是一维。可以把搜索码看成是一个被这些区间所包围的方框,方框的每一条边都和坐标轴平行。R树中搜索码的值将被称为边界框。

叶子节点中包含数据项。一个数据项是由 <n维方框, rid >对组成的,这里 rid标识一个对象,方框是包含这个对象的最小方框。作为特殊情况,如果数据对象是一个点而不是区域,那么这个方框就是一个点。非叶子节点包含的索引项的形式为 <n维方框,指向子节点的指针 >。在非叶子节点 N上的方框是包含所有以节点 N为子树根的所有数据对象的区域。

下图描述了数据对象和边界框在空间是如何分布的以及数据对象的分布所对应的 R树实例。

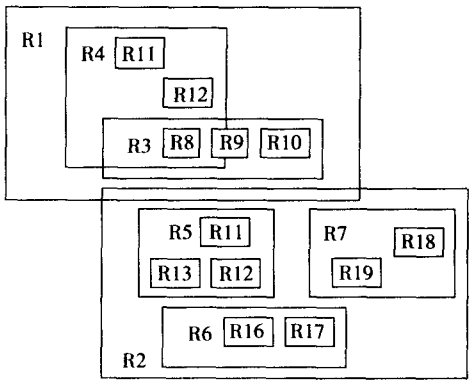

图 1　数据对象和边界框的例子

Fig. 1　An instance of data objects and their boundary

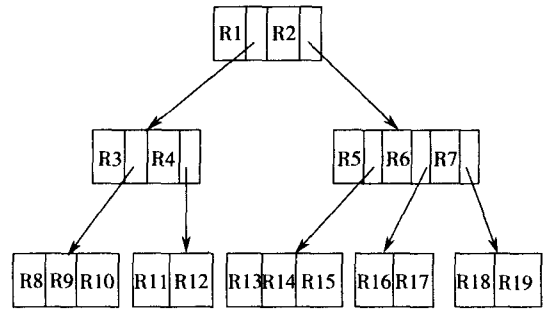

图 2　图 1中的数据对象所对应的 R树

Fig. 2　An R-tree constructed by data objects in figure 1

在实例树中有 19个区域。区域 R8～R19表示数据对象,同时在叶子级别上表现为数据项。区域 R1～R7表示树的内部节点的边界框。例如区域 R1是包含左子树空间的边界框,它包括数据对象 R8、R9、R10、R11和 R12。

一个给定节点的两个孩子的边界框可以重叠,例如,根节点的孩子边界框 R1和 R2是重叠的。这意味着一个满足所有边界框约束的给定数据对象可以包含在多个叶子节点中。但是,每个数据对象都精确地存储在一个叶子节点中,即使它的边界框落在多个高层节点对应的区域内。例如,考虑 R8表示的数据对象。它同时包含在 R3和 R4中,可以被放置在第一个或者第二个叶子节点中 (在树中从左到右)。这里选择将它插入到最左边的叶子节点中而没有插入到树中其他任何地方。

2.1　R -树的搜索操作

为了搜索一个点,需要计算对应该点的边界框 B,然后从树根开始查找。首先测试树根的每个孩子的边界框以确定它是否与查询框 B重合,如果重合就搜索以这个孩子为根的子树。如果树根的多个孩子的边界框都与 B重叠,那么就必须搜索所有相应的子树。这是与 B +树的一个重要差别:即使一个点也可能导致搜索沿着树的几条路径进行。当到达叶子节点时,检查叶子是否包含需要的点。当查询点所在的区域不被任何一个与叶子节点相对应的框覆盖时,就可能不会访问到一个叶子节点。如果搜索没有访问到任何叶子节点,那么查询点就不在索引的数据集中。

区域对象的搜索和区域查询的处理是类似的,都是首先计算需要的区域边界框,然后像搜索一个对象那样进行区域查询,当到达叶子节点后,检索属于该叶子的所有区域对象,并测试以确定它们是否与给定的区

域重叠 (或者被包含,这取决于查询)。需要注意的是即使对象的边界框与查询区域重叠,对象本身也可能不重叠!

例如,假设查询区域是表示对象 R9的边界框,希望找出查询区域覆盖的所有对象。首先,从树根开始,发现查询框与 R1重叠而不与 R2重叠。这样,就搜索左子树,而不搜索右子树。接着发现查询框与 R3重叠而不与 R4重叠,因此继续搜索最左边的叶子,找到对象 R9。

为了搜索一个给定点的最近的邻居,可以像搜索点本身一样进行处理。先检索作为搜索的一部分的叶子节点中的所有点,然后返回与查询点最近的点。如果没有访问到任何叶子节点,就以查询点为质心的一个小边界框代替查询点,重复搜索。如果还是不能访问到任何叶子节点,就增大框的范围,然后再次进行搜索,继续这个过程直到找到一个叶子节点为止。于是,在搜索的迭代中考虑了从叶子节点中检索到的所有点,然后返回与被查询点最近的点。

2.2 R-树的插入和删除操作

为了插入 rid为 r的数据对象,需要计算对象的边界框 B,然后将 <B, r>对插入到树中。从根节点开始遍历从根节点到叶节点的一条单独路径 (与搜索相比,搜索可能要遍历几条这样的路径)。在每一级选择这样的子节点:它的边界框只需最小的扩大 (按照面积的增长进行度量)就可以覆盖边界框 B。如果几个子节点都可以覆盖 B的边界框 (或者是为了覆盖 B的边界框需要相同的增长),那么从这些子节点中选择具有最小边界框的节点。

在叶子级插入对象,而且如果有必要还需要扩大叶子节点的边界框来覆盖边界框 B。如果需要扩大叶子节点的边界框,那么叶子节点的祖先的边界框就必须扩大,为什么呢?因为在插入完成以后,每个节点的边界框都必须覆盖所有后代的边界框。如果叶子节点没有空间以插入新的对象,就必须把节点进行分裂,然后把数据项重新分布到老的叶子节点和新的节点。同时必须调整老叶子节点的边界框,将新叶子节点的边界框插入到它的父亲中。当然,所有这些改变能够沿着树向上进行传播。

为了从 R树中删除一个数据对象,先执行搜索算法,并且可能还要检查多个叶子。如果对象在树中,就去掉它。原则上,尽量缩小包含对象的叶子的边界框以及所有祖先节点的边界框。在实际中,通常只是简单地将对象移走来实现删除操作。

3 K-D树

为了理解如何索引二维或高维的空间数据,首先考虑对一维数据中的点进行索引。树结构 (如二叉树和 B树)的操作是连续的把大的空间划分成一些较小的空间。例如:二叉树的每个内结点把一个一维区间划分成两个子区间。左区间中的点存储到左子树,右区间中的点存储到右子树。平衡二叉树中,每个划分的区间应该近似包含子树中的点的一半。类似的,二叉树的每一层把一个一维区间划分成多个部分。

根据上述直觉,能够为二维或高维空间创建树结构。K-D树是用于索引多维数据的早期数据结构之一。K-D树的每层都把空间划分为两个部分。划分沿着树的顶层结点中的一维进行,然后是下一层结点中的其它维,等等如此划分下去。划分尽量使子树中的结点有一半属于一个划分,另一半属于另一个划分。当结点中包含的点数少于叶子结点中包含的最大点数时停止划分。图 3显示了用 K-D树表示一个二维空间中的点集。每条线对应树中的一个结点,叶子结点中包含的最大点数设置为 1。线号表示对应结点出现在树中的层数。

为减少树的高度,把 K-D树扩展为 K-D-B树,就象把二叉树扩展为 B树一样。K-D-B树比 K-D树更适合索引二级存储器中的数据。

4 Quad树

二维数据的另一种表示方法是使用 Quad树。图 4显示了 Quad树对空间的划分。Quad树中的每个结点对应空间中的一个矩形区域。顶层结点对应整个目标空间。树中的每个非叶子结点把该结点对应的区域划分为四个大小相等的象限,每个象限有一个孩子结点与之对应。叶子结点包含点的数目介于零和某个定值之间。相应的,如果一个区域对应的结点包含的点数超过了这个最大值,则需要为这个结点创建孩子结点。在图 4的例子中,叶子结点中的最大点数为 1。

可以使用区域 Quad树 (region quadtrees)来存储数组信息。如果区域 Quad树中的结点覆盖的区域中所有数组元素的值都相同,则该结点是叶子结点。否则,该结点是内部结点,被进一步划分为四个等大小的子结点。区域 Quad树中的每个结点对应一个子数组的所有值。对应叶子结点的子数组,或者包含一维数组元素,或者包含多维数组元素,所有元素的值都是相同的。

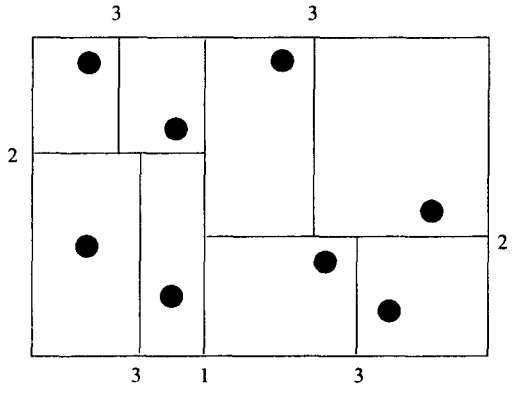

图 3　用 K-D 树表示一个二维空间中的点集

Fig.3　2-dimensional points organized in a K-D tree

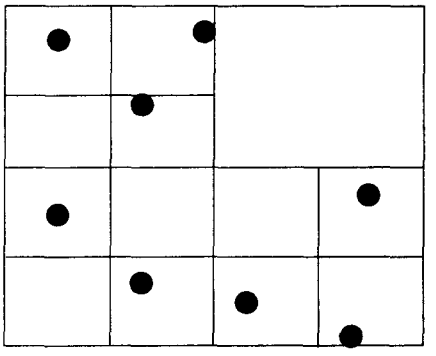

图 4　用 Quad 树划分二维空间

Fig.4　Using a Quad tree on the partion of 2-demensional space

5　GiST

B +树和 R树在许多方面是类似的:它们都是高度平衡的,搜索都是从根节点开始,然后向叶子节点处理,每个节点覆盖底层数据空间的一部分,同时节点的子节点覆盖与节点相关联的区域的子区域。当然两者还有很重要的差别,例如,在 B +树的表示中,空间是线性化的,但是在 R树中不是这样。它们的共同特征使它们在插入、删除、搜索甚至并发控制上有很大的相似之处。美国威斯康星大学 Joseph M. Hellerstein于 1995年在 VLDB会议上首先提出 GiST(Generalized Search Trees),称之为通用搜索树。GiST抽象出树索引结构的基本特征,并为插入、删除和搜索提供"模板"算法。GiST的核心思想是一个对象关系 DBMS可以支持多个模板算法,因此使得高级数据库用户实现特定的索引结构更容易 (例如 R树及其变形)而不需对任何系统代码做改动。实现扩充比从头实现新的索引算法花费要少很多。

许多特殊的搜索树都可以通过 GiST实现。可以说,GiST统一了所有不同的树结构,它是特殊搜索树的模板。GiST向用户提供一组函数接口,这些接口需要用户来实现,然后再注册到数据库系统中。这组函数既反映了用户自定义类型的结构和特点,也反映了对已用户自定义类型为索引关键字的 GiST树的基本操作。

GiST是一棵平衡树,它提供了一些模板算法用于周游、删除、修改、分裂、合并。与其它搜索树类似,叶结点存储 (key, ptr)对,key是索引关键字,ptr是指针,指向包含记录的数据块在磁盘上的地址。内部结点包含 (p, ptr)对,p是一个逻辑谓词,它描述用户要查找的数据是否在 ptr所指向的子树中。GiST的所有叶结点用链表连接起来。GiST结构见图 5。它有如下特性:

a) 根结点至少有 2个子女。

b) 每个内部结点包含的子女数记为 N, kM ≤N ≤ M。其中 k称为最小填充因子,2 /M ≤k ≤1 /2。实际上,每个 GiST结点包含 N个 (p, ptr)对,p是一个抽象谓词,ptr是一个指针,它指向一棵子树,子树的叶结点中包含符合谓词 p的记录。称 (p, ptr)是一个入口 (Entry),简记为 E。规定,在 GiST中所有结点大小相同,最多包含 M个入口。如果结点中含有少于 M个入口,则其余的位置空缺,以便其它的入口插入到 GiST中。

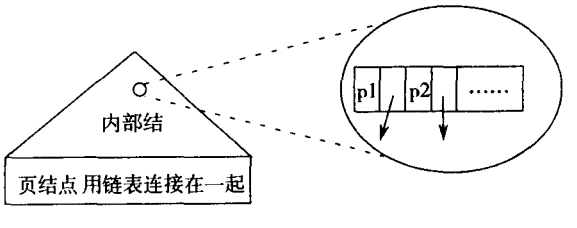

图 5　GiST 的结构图

Fig. 5　The structure of GiST

c) 对于叶结点上的每一个入口 E = (key, ptr), key

中存放记录的关键字,ptr指向真实的记录。

d) 所有的叶结点都处在同一层。叶结点所处的这一层规定为 0层,叶结点的父亲所处的一层是 1层,…。以此律推,子女所处的层数等于父亲所处层数减 1。

GiST本身提供了一系列操作算法:插入算法 (Insert)、分裂算法 (Split)、删除算法 (Delete)、搜索算法 (Search)。GiST的平衡特性由它的插入算法、分裂算法、删除算法保证。搜索算法是输入一个谓词 q,然后对树进行搜索,返回满足谓词 q的所有记录。插入算法是在 GiST中选择一个合适的位置,将 (key, ptr)插入到 GiST中。删除算法是将 (key, ptr)从 GiST中删除。为实现上述操作,类型的定义者和开发者还必须向数据库系统注册并且实现如下方法:

a) Consistent(E, q): 输入一个 Entry E = (p, ptr)以及一个查询谓词 q, 如果 p∧q为 false,则返回 false,否则返回 true。系统调用 GiST的搜索算法时,需要调用此函数,用来确定满足用户谓词 q的记录是否在 Entry E. ptr所指向的子树中。

b) Union(P): 输入一个 Entry的集合 $P = \{E_i \mid E_i = (p_i, ptr_i), i = 1, 2, \cdots, n\}$,返回一个谓词 r, 其中 $r = p_1 \vee p_2 \vee \cdots \vee p_n$。系统调用 GiST的分裂和合并算法时要调用此函数,来进行内结点之间的合并。

c) Compress(E): 输入一个 Entry E = (p, ptr),返回一个 Entry E' = (π, ptr),其中 π是 p的压缩形式。将谓词原封不动地存储在磁盘上可能会浪费空间,此函数的作用是:将 GiST中的谓词压缩后存储在磁盘上。

d) Decompress(E): 输入一个 Entry E' = (π, ptr) 其中 π是 p的压缩形式,返回一个 Entry E = (r, ptr),其中 $p \rightarrow r$。此函数的作用是:将压缩的 GiST在内存中解压缩。

e) Penalty(E_1, E_2): 输入 2个 Entry $E_1 = (p_1, ptr_1)$, $E_2 = (p_2, ptr_2)$,返回一个将 E_2 插入到以 E_1 为根的子树中所带来的惩罚值。惩罚值越小越好。系统调用 GiST的分裂和插入算法时要调用此函数。插入时,在 GiST中为插入的 Entry选择一个较优化的位置。分裂时,在 GiST中为分裂的内结点的另一半选择一个较优化的位置。

f) PickSplit(P): 输入一个 Entry集合 $P = \{E_i \mid E_i = (p_i, ptr_i), i = 1, 2, \cdots, M + 1\}$,把 P分裂成 2个集合 P_1 和 P_2,每个集合中的 Entry数目大于等于 kM。系统的分裂算法将调用此函数,用来分裂一个 Entry数目大于 M的内结点。

6 空间数据库的高维索引

在某些应用,比如基于内容的检索或者文本索引中,维数可能很大 (几十维是很常见的)。对这些高维数据进行索引很困难,需要新的索引技术。例如当在多于 12维的数据集进行单个点的搜索时,顺序扫描就比 R树索引扫描要快。

对高维数据集合进行最邻近查询是最常见的查询。对于高维数据存在这样一个潜在的问题:如果高维数据分布的比较广泛,当维数 d增加时,到最近邻居的距离 (从任何给定的查询点开始)变得越来越接近于到最远的点的距离。在这种情况下最邻近搜索是没有意义的。

在许多应用中,可以对高维数据进行索引而不会出现上面谈到的问题。最好在对高维数据进行最邻近查询之前,检查高维数据集合来保证最邻近查询是有意义的。这里给出一种检查的方法:随机地产生一些样本查询,测量每个样本查询中的查询点到最近点和最远点的距离,然后计算这些距离的比率,接下来取这些比率的平均值。如果这个平均值接近于 1,则可以断定最邻近查询是没有意义的。距离查询点最近点的距离和最远点的距离的比率被称为数据集的对比度。通过上述方法可以测量出一个数据集的对比度。在需要最邻近查询的应用中,首先应当通过数据的经验测试来保证数据集较小的对比度。

参考文献:

[1] BECKMANN N, KRIEGEL H P, SCHNEIDER R. The R* tree: An efficient and robust access method for points and rectangles[A]. Proc ACM SIGMOD Conf On the Management of Data[C]. 1990.

[2] BERCHTOLD S, BOHM C, KRIEGEL H P. The pyramid - tree: Breaking the curse of dimensionality[A]. In ACM SIGMOD Conf On the Management of Data[C]. 1998.

[3] BEYER K, GOLDSTEIN J, RAMAKRISHNAN R. When is "nearest neighbor" meaningful? [A]. Proc Int Conf on Database Theory[C]. 1999. 217 - 235.

[4] THOMAS BRINKHOFF, HANS - PETER KRIEGEL, RALF SCHNEIDER. Comparison of Approximations of Complex Objects Used for Approximation - based Query Processing in Spatial Database Systems[J]. ICDE, 1993, 40 - 49.
[5] FALOUTSOS C. Searching Multimedia Databases[M]. Content Kluwer Academic, 1996.
[6] CHRISTOS FALOUTSOS, SHARI ROSEMAN. Fractals for Secondary Key Retrieval[J]. PODS, 1989, 247 - 252.
[7] MICHAEL FREESTON. The BANG File: A New Kind of Grid File[J]. SIGMOD Conference, 1987, 260 - 269.
[8] VOLKER GAEDE, OLIVER GÜNTHER. Multidimensional Access Methods[J]. ACM Comput Surv, 1998, 30(2): 170 - 231.
[9] JONATHAN GOLDSTEIN, RAGHU RAMAKRISHNAN, URI SHAFT, et al. Processing Queries By Linear Constraints[J]. PODS, 1997, 257 - 267.
[10] ANTONIN GUTTMAN. R - Trees: A Dynamic Index Structure for Spatial Searching[J]. SIGMOD Conference, 1984, 47 - 57.
[11] JOSEPH M HELLERSTEIN, JEFFREY F NAUGHTON, AVI PFEFFER. Generalized Search Trees for Database Systems[J]. VLDB, 1995, 562 - 573.
[12] JOSEPH M HELLERSTEIN, ELIAS KOUTSOUPIAS, CHRISTOS H. Papadimitriou: On the Analysis of Indexing Schemes[J]. PODS, 1997, 249 - 256.
[13] MARCEL KORNACKER, MOHAN C, JOSEPH M HELLERSTEIN. Concurrency and Recovery in Generalized Search Trees[J]. SIGMOD Conference, 1997, 62 - 72.
[14] DAVID B LOMET, BETTY SALZBERG. The hB - Tree: A Multiattribute Indexing Method with Good Guaranteed Performance[J]. ACM Trans Database Syst, 1990, 15(4): 625 - 658.
[15] JÜRG NIEVERGELT, HANS HINTERBERGER, KENNETH C SEVCIK. The Grid File: An Adaptable, Symmetric Multikey File Structure[J]. ACM Trans Database Syst, 1984, 9(1): 38 - 71.
[16] JOHN T ROBINSON. The K - D - B - Tree: A Search Structure For Large Multidimensional Dynamic Indexes[J]. SIGMOD Conference, 1981, 10 - 18.
[17] HANAN SAMET. The Quadtree and Related Hierarchical Data Structures[J]. ACM Comput Surv, 1984, 16(2): 187 - 260.
[18] HANAN SAMET. The Design and Analysis of Spatial Data Structures[M]. Addison - Wesley, 1990.
[19] ROGER WEBER, HANS - JRG SCHEK, STEPHEN BLOTT. A Quantitative Analysis and Performance Study for Similarity - Search Methods in High - Dimensional Spaces[J]. VLDB, 1998, 194 - 205.
[20] OURI WOLFSON, PRASAD SISTLA A, BO XU, et al. DOMINO: Databases for Moving Objects tracking[J]. SIGMOD Conference, 1999, 547 - 549.
[21] 郭龙江,李建中,张兆功. 基于记录分布的并行一般搜索树: RD - GiST[J]. 计算机科学, 2002, 29(8.增刊 A): 263 - 266.

Indexing techniques in spatial databases

GUO Long-jiang[1,2], LI Jian-zhong[1,2]

(1. College of Computer Science and Technology, Harbin Institute and Technology, Harbin 150001, China; 2. College of Computer Science and Technology, Heilongjiang University, Harbin 150080, China)

Abstract: Due to a great quantity of data in spatial databases, so in general the cost of query in spatial databases is higher than in relational databases. Especially, when there are some functions, in predicate of query, which deal with spatial data, the cost of computing these functions is higher than the cost of comparing strings and numerical values. If query strategy is to scan sequentially spatial data, then the efficiency will be very low. For improving query efficiency, it is necessary to adopt spatial indexing techniques. Indexing techniques in spatial databases have gradually caused the attentions of many people. Research advance of indexing techniques for spatial databases is summarized, and then four new and often used indexing methods, including R - tree, K - D - tree, Quad tree and Generalized search tree, are introduced. Finally, it is pointed out that the high dimensional index is a hot research field in spatial databases.

Key words: spatial data; spatial databases; spatial index; high dimensional index