空间数据库索引管理系统的设计与实现*)

汪林林 马 锐
(重庆邮电学院软件学院 重庆400065)

The Design and Implementation of Spatial Database's Index Management System

WANG Lin-Lin MA Rui
(Chong Qing University of Post and Telecommuniction,Chongqing 400065)

Abstract By the special character and wide application of spatial data,it is needed to design a kind of spatial database system management system(SDBMS),index management is a core part of SDBMS,the performance of a database is decided by the management function of index largely,this paper defines from the structure of the minimum data item to index tree,index page's storage structure and memory management,designs and implements ultimately a spatial database index management system.

Keywords Index,Key-value,Node,B+-Tree,R-Tree

1. 引言

在应用日益广泛的计算机辅助设计(CAD)和地理信息系统(GIS)中,除了存放属性数据之外,更多的用于存储譬如机器零件位置、国家或城市的地理位置等信息的空间数据。在地理信息系统中,空间数据对象可以是个点、线、面,用来表示城市、交通线路和行政区域,因此空间数据属于多维数据,并且空间数据之间存在着一定的拓扑关系。鉴于空间数据的特征,普通的一维数据库管理系统已不适合空间数据的存储和管理,由此既能管理属性数据又能管理空间数据的空间数据库管理系统(SDBMS)应运而生。空间数据库与传统的一维数据库的区别在于空间数据的引入,相应地就需要一种不同的索引管理机制。本文就是为空间数据库设计一套索引管理方案。

2. 空间数据库管理系统的体系结构

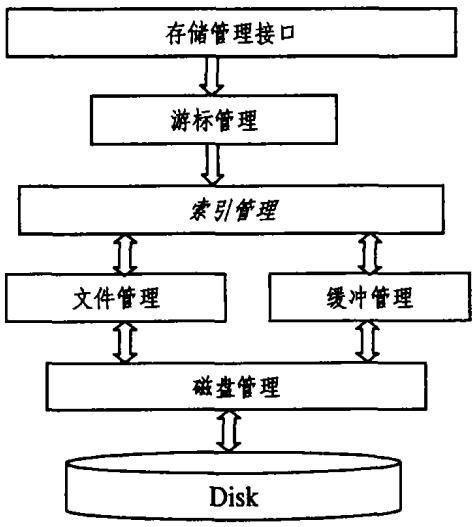

图1 SDBMS体系结构图

如图1所示,空间数据库管理系统的体系结构主要由磁盘管理、缓冲管理、文件管理、索引管理及其游标管理五大部分组成。其中磁盘管理主要负责磁盘文件、分区的管理,担负Buffer和磁盘间读写的任务;缓冲管理完成在buffer中维持经常使用的page,减小磁盘I/O操作次数的功能;文件管理担任顺序文件的管理及表空间中记录的管理;索引管理则主要负责为经常使用的数据提供快捷的访问方式;游标管理是为用户提供便利的记录级顺序文件检索机制。

由图可见,索引管理位于SDBMS中间层的位置,和下层交互处理来自上层的查询请求,本文以下几部分将详细描述如何设计一套索引管理系统。

3. 数据元素的表示

索引用的数据元素主要是码值,按照不同的要求检索,自然需要不一样的码值,那么就需要一种描述码值信息的数据结构——码值描述符。码值又常常是几个字段的组合,因此对于每个字段,需要定义字段描述符描述它的数据信息。字段描述符和码值描述符的具体数据结构如下:

3.1 字段描述符

在数据库中,一个属性即为一个字段,字段的描述符用来描述字段的特性,包括该字段在记录里的起始位置,长度,数据类型。定义如下:

```
typedefstruct{
int offset; /* 记录里字段的起始位置 */
int length; /* 该字段的长度 */
int type; /* 该字段的数据类型 */
}T_FIELDDESC;
```

3.2 码值描述符

一个码值可能由几个字段组成,在码值描述符里描述了字段个数,字段标志和每个字段的描述符。具体结构可描述为:

```
typedef struct {
INT4 NumFields; /* 字段个数 */
INT4 KeyFlags;    /* 码值标识位 */
T_FIELDDESC FieldDesc[MAXKEYFIELDS];
/* 每个字段的字段描述符,字段最多MAXKEYFIELDS个 */
```

*)重庆市科委攻关项目资助。汪林林 教授,硕士研究生导师,研究方向:数据库系统,GIS系统,计算机网络。马 锐 硕士研究生,研究方向:数据库系统,GIS系统,计算机网络。

} T_KEYDESC;

3.3 从记录中提取 key 值

当用户指定索引码值时，相应地生成码值描述符。对于每条记录按照码值描述符分析出索引码值包括的字段数，由每个字段的字段描述符即可提取 key 值。该 key 值就是我们要操作的对象，最基本的数据元素。

4. 索引空间数据库数据的两种树结构

空间数据库包括属性数据和空间数据，对于属性数据的索引采用 B^+-Tree，而对于空间数据的索引采用 R-Tree。下面简要介绍一下这两种树的结构。由于篇幅的限制，详尽的算法这里不作描述。

4.1 B^+-Tree

4.1.1 B^+-Tree 的结构及特征 B^+树把它的存储块组织成一棵平衡树，即从树根到树叶的所有路径都一样长。通常 B^+树有三层：根、中间层和叶，但也可以是任意多层。其结构如图2所示。B^+树结构的特征是：

·根结点中至少有两个指针被使用。所有指针指向位于 B^+树下一层的存储块。

·叶结点中，最后一个指针指向它右边的下一个叶结点存储块，即指向下一个键值大于它的块。在叶块的其他 n 个指针当中，至少有(n+1)/2个指针被使用且指向数据记录；未使用的指针可看作空指针且不指向任何地方。如果第 I 个指针被使用，则指向具有第 I 个键值的记录。

·在内部节点中，所有的 n+1个指针都可以用来指向 B^+树中下一层的块。其中至少(n+1)/2个指针被实际使用。

·假若我们以常规的画树方式来画 B^+树，任一给定节点的子节点按从左到右的顺序排列。那么，我们在任何一个层次上从左到右来看 B^+树的节点，节点的键值按非减的顺序出现。

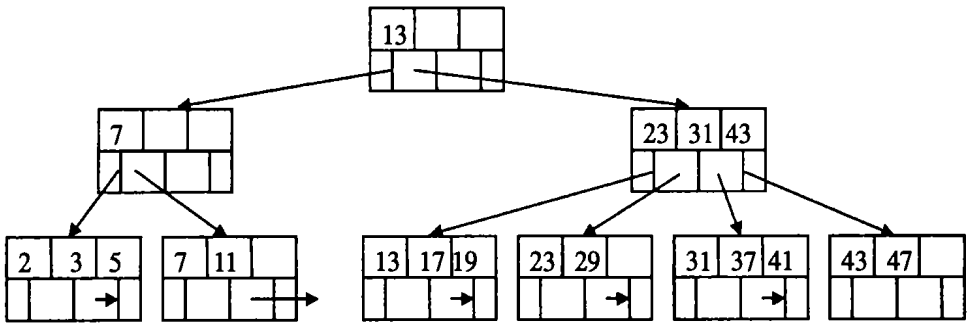

图2 B^+-Tree

4.1.2 B^+-Tree 的效率

B^+-Tree 使我们能实现记录的查找、插入和删除，而每个文件操作只需很少的磁盘 I/O。首先我们注意到，如果每个块容纳的键数 n 相当大，比如10或更大，那么，分裂或合并块的情况将会很少。此外，这种操作必需时，绝大多数时候都被局限在叶节点，因此只有两个叶节点和它们的父节点受到影响。所以，基本上可以忽略 B^+树重组的 I/O 开销。

然而，每次按给定查找键值查找记录都需要我们从根节点一直访问到叶节点以找到指向记录的指针。因为我们只读 B^+树的块，所以磁盘 I/O 数将是 B^+树的层数加上一次(对查找而言)或两次(对插入或删除而言)处理记录本身的磁盘 I/O。那么，B^+树到底有多少层?对于典型的键，指针和块大小来说，三层就足够了，除非数据库极大。

4.2 R-Tree

地理信息系统用一个二维的空间存储对象，对象可能是点或形状，来表示房子、路、桥、管道或其他物理对象，这些对象的集合就构成了一幅地图。GIS 系统里经常有这样的查询：范围查询，即找某一地理范围内满足某些特征的形状集合；最临近查询，即查找与给定点最近的点；where-am-I 查询，即点击一个点时，我们想知道这个点所在的区域。传统的查询方法当然也能得到结果，但效率低下，准确率也不是很高，因此引入了一种专门用于空间数据查询的树结构——R-Tree。

4.2.1 R-Tree 的结构及特征 R-Tree(区域树)是一种利用 B^+树的某些本质特征来处理多维数据的数据结构。另一方面，R-Tree 表示由二维或更高维区域组成的数据，被称为数据区。R-Tree 的叶子节点对应于某个区域，这个区域是把地图元素如点、线、面封装起来的最小边界图形(如图3所示区域图，这里两块阴影部分代表地图上的图形元素，它们分别被区域图形 R8，R9所封装)；内部节点也对应于某个区域，与叶子节点不同的是它所封装的对象不是地图元素，而是下一层子节点的区域图形(如图3的区域 R3，它是封装 R8，R9，R10的内部节点区域)。原则上，区域可以是任何形状，但在实际中，由于计算上的原因，一般采用矩形。图4是由图3所示区域图构造出来的 R-Tree。

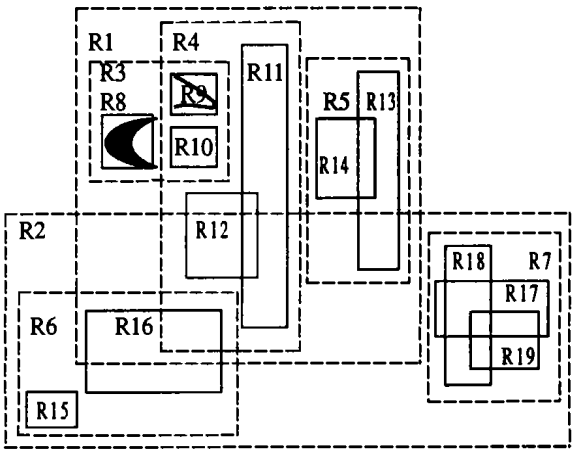

图3 区域图

如图4所示，每个节点页(块)可以存放若干个节点，本例取作3(关于索引页的问题将在第5节作详细介绍)，每个内节点的结构定义为[I，child-pointer]，其中 I 为边界矩形，child-pointer 为指向子节点页的指针；页节点的结构定义为[I，tuple-pointer]，I 同样为边界矩形，tuple-pointer 为指向数据元组的指针。

设 M 为一个节点页里允许装入的最大节点数目，m＝⌊M/2⌋为一个节点页装入的最小节点数目。

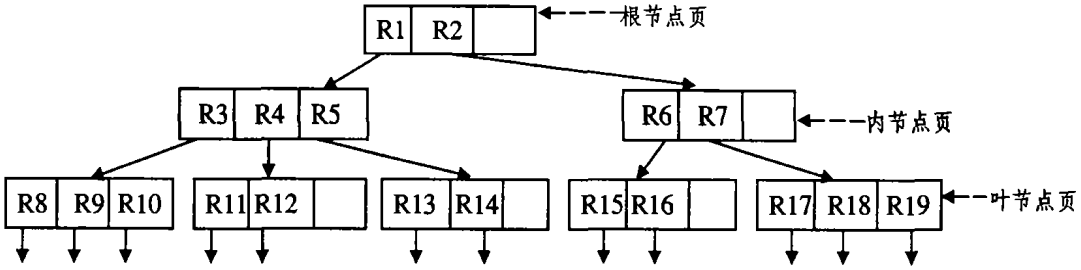

图4 R 树

R 树的结构特征是：

·除了叶节点页即为根节点页这种情况，每个节点页(包括叶节点页和内节点页)必须有 m～M 个节点。

·对于叶节点页里的每个节点[I,tuple-pointer]，I 是把二维实体包含住的最小边界矩形。

·对于内节点页的每个节点[I,child-pointer]，I 是把 child-pointer 指向的节点页里所有的节点区域都包含在内的最小边界矩形。

·如果根结点页不是叶子节点页，那么根节点页至少要有两个节点。

·所有叶子节点页都在同一层上，即 R 树是一种高度平衡树。

含有 N 个索引记录的 R 树高度最大为⌈logm N⌉，除了根节点页以外的其它节点页的空间利用率最坏为 m/M。

把每个节点页的节点数目限制为最小 m 是为了减小树的高度，提高空间利用率。

5. 索引页的存储结构

本索引系统是基于磁盘的，也就是说索引页存储在磁盘上，对索引页所作的修改全部写进磁盘。图5 所示的即为一个基于磁盘的索引页结构。Slot 数组标记从 slot[1] 开始，与 Entry 一一对应，值为每个 Entry 的偏移量。

索引页的大小一般取作1kB 或4kB，索引页的头信息存储在页末，数据信息从页头开始，向页中部动态增加，同时，slot 数组也动态增大，增长方向与 Entry 增长方向相反。

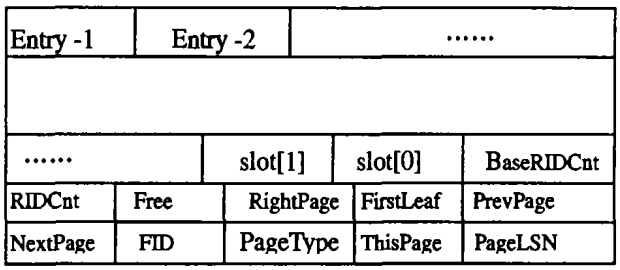

图5 B^+-Tree/R-Tree 索引页

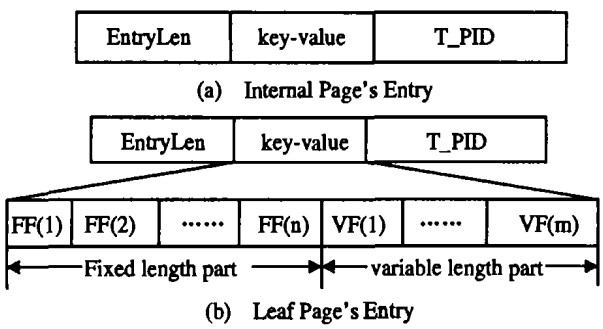

图6 B^+-Tree 的 Entry 结构

由图5定义可看出，B^+-Tree 和 R-Tree 的页结构基本一致，只是各自的 Entry 结构不同。如图6所示，对于 B^+-Tree，内部页 Entry 的结构中，用 EntryLen 标记 Entry 的长度，key-value 值为索引用的码值字符串，T_PID 为指向的下一个 IndexPage 的 ID 号；叶子页 Entry 结构中，前两个字段的含义与内部页相同，不同的是 T_RID 字段，它为指向的数据记录 ID 号。

R-Tree 中用 MBR(minium boundary rectangle)来表示区域，子区域。它的 Entry 结构如图7所示。

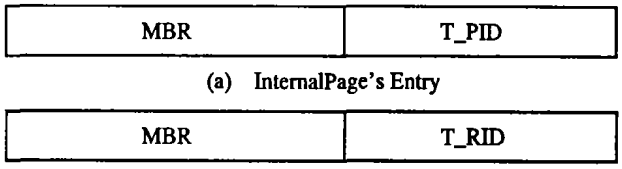

图7 R-Tree 的 Entry 结构

6. 索引表，分割表

由于一段时间内可能不只一个进程调用索引，为了免去每个进程都从硬盘索引文件读入的麻烦，引入了索引表的概念。索引表是为了减少磁盘 I/O 次数，驻留在内存中的记录索引信息的内存区域。

通过一个散列函数(散列键是32位的文件的 ID 号＋索引编号的值)，链接到索引链表上(如图8所示)，索引链表上的每个索引表记录着索引表的状态，该索引表被引用的次数，数据文件的索引，索引描述符，码值描述符以及链接用的左指针和右指针(如图8所示的放大部分)。

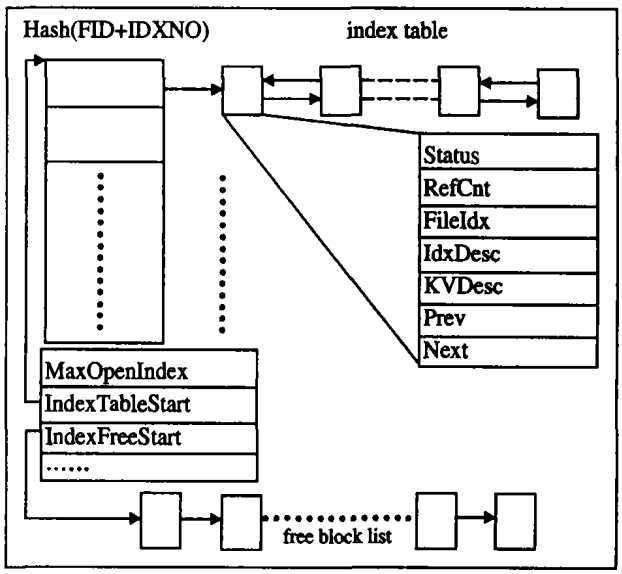

图8 分割表与索引表组织示意图

(下转第128页)

供对 VI 管理、队列管理、内存映射以及标准的网络通信的支持。软件 VI 结构在 NIC 固件(Myrinet),特殊的设备驱动器(ServerNet),或在标准网络驱动器的顶端增加一个中间驱动层(Gigabit Ethernet)来实现 VI 结构的各个功能。

参 考 文 献

1 Compaq. Intel and Microsoft Corporations, Virtual Interface Architecture Specification. Version 1. 0, 1997. http://www.viarch.org

2 Dunning D,et al. The Virtual Interface Architecture. IEEE Micro, Mar.-Apr. 1998. 66～76

3 Buonadonna P, Geweke A, Culler D. An Implementation and Analysis of the Virtual Interface Architecture. In: Proc. of SuperComputing Conf. Nov. 1998

4 Giganet Inc., cLAN Performance. http://www.giganet.com/products/performance.htm

5 Dubnicki C,Bilas A,Li K,Philbin J F. Design and Implementation of Virtual Memory-Mapped Communication on Myrinet. In: Proc. of the 11th Intl. Parallel Processing Symposium, April 1997

6 Eicken T V, Basu A, Buch V, Vogels W. U-Net: A User-level Network Interface for Parallel and Distributed Computing. In: Proc. of the 15th Symposium on Operating Systems Principles, 1995. 40～53

7 National Energy Research Scientific Computing Center (NERSC), M-VIA: A High Performance Modular VIA for Linux. http://www.nersc.gov/research/FTG/via Study of User-Level Communication Interfaces VIA(1)

(上接第121页)

分割表是用来管理索引表块和空闲块的内存结构。分割表块把内存分成若干索引表大小的块,它的头记录着索引表开始的位置及空闲块的开始、结束位置。需要新建索引表时,从空闲块链表中取下一块放到相应的散列表里。

为了减小分割表的长度,定时检测索引链表,如果索引表数目达到了索引链允许的最大值,对于引用次数为0的索引表,把它从索引表链中删掉并链入空闲块。

需要使用一个索引时,先打开内存中的分割表,如果在内存中,即可获得该索引的索引描述符及码值描述符。不在内存,就调入。

7. 一个 SDBMS 的索引管理系统设计流程

以上几部分详细地介绍了索引管理所涉及到的数据结构和管理办法,在这些基础上,下面将介绍一下空间数据库(SDBMS)索引管理系统的设计流程:

(1)数据结构的类定义,包括数据元素的定义、索引页结构的定义以及分割表、索引表的定义等。

数据元素的表示方法多种多样,数据元素的定义正是为了规范化数据元素的表示;索引页结构的定义规定了索引结构的存储格式,本文所示的格式不仅简单明了地说明了索引页和索引页之间的联系,而且极大地增强了存储空间的利用率;分割表、索引表是为了减少磁盘 I/O 的次数提出的用于管理内存中索引信息的数据结构。

(2)与 SDBMS 其它模块的接口(如与缓冲管理器的接口、与文件管理器的接口、游标管理器的接口)设计。

处理游标管理器发出的请求,并通过与缓冲管理器、文件管理器的互操作实现对磁盘上的数据进行读写。

(3)索引算法的设计。

包括属性数据的 B^+Tree 索引算法和空间数据的 R-Tree 算法设计。这是索引管理系统的核心,算法的好坏和查询的速度、精度直接相关,可以根据需求适当地对这两种算法作些改动。

以上是建立一个索引管理系统所必备的几点要求,那么当应用层提出请求按某一码值索引某一数据文件时,操作流程又是如何呢?

(1)按照码值建立码值描述符。

(2)为需要索引的数据文件构造 B^+Tree、R-Tree,创建索引文件。

(3)把该索引描述符加到文件管理器里管理所有索引的"索引类文件"中。

(4)当需要查看或数据文件更新时,首先查看内存中的分割表头的 IndexTableSem 字段,找到散列表地址,接着由 Hash(FID+IDXNO)得到散列值,由此找到相应的索引表链。下面分两种情况:

·如果该数据文件的索引表在这个链上,则:i)得到索引信息。得到该索引的索引描述符等信息,从而按照索引描述符提供的信息,查询或修改相应的 B^+-Tree 和 R-Tree;ii)修改索引表。把索引表的状态字段设为"active",同时引用次数字段加1。

·如果该数据文件的索引表不在这个链上,则:i)查文件管理器的"索引类文件",得到索引描述符并把该索引描述符加到散列表中,以便下次引用;ii)由索引描述符信息查询或修改 B^+Tree 和 R-Tree。

结束语 空间数据库的索引管理系统是数据库管理系统中一个非常重要的层面,它的设计好坏直接影响着整个数据库系统的性能。该文从空间数据的特性出发,结合传统数据库索引方法,在研究分析的基础上,设计并实现了一套既能管理空间数据又能更好管理属性数据的空间数据索引方案,该方案对于空间数据库的研究与实现有着一定的现实意义。

参 考 文 献

1 Garcia-Molina H, Ullman J D, Widom J. Database System Implementation, Pentice Hall, 2001. 3

2 Silberschatz A, Korth H F. Database System Concept. Pentice Hall, 1999

3 Zaniolo C,et al. Advanced Database System. Pentice Hall, 1997

4 Atzeni P, Ceri S, Paraboschi S, Torlone R. Database Systems Concepts, Languages and Architectures. McGraw-Hill Book Company 1999

5 Ramakrishnan R. Database Management Systems, WCB/McGraw-Hill 1998

6 Bach M J. The Design of the UNIX Operating System. Prentice Hall, 2000. 8

7 Beckman N, Kriegel H-P, Schneider R, Seeger B. The R*-Tree: An Efficient and Robust Access Methods for Points and Rectangles. In: Proc. 1990 ACM SIGMOD conf. pp. 322～331

8 Sellis T K, Roussopoulos N, Faloutsos C. The R+-Tree: A Dynamic Index for Mutidimensional Objects. In: Proc. Intl. Conf. On Very Large Databases, 1987. 507～518

9 Mohan C, et al. ARIES: A Transact ian Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging. ACM Transactionson Database Systems, 1992, 17(1)

10 Mohan C, Levine F. ARIES/IM: An Efficient and High Concurrency Index Management Method Using Write-Ahead Logging: [IBM Research Report RJSS46]. IBM Almaden Research Center, Aug. 1989