空间数据库磁盘管理器的研究与设计

范建伟　汪林林
（重庆邮电学院软件学院　重庆400065）

Research and Design on the Disk Manager of Spatial Database

FAN Jian-Wei　WANG Lin-Lin
(Software Institute of Chongqing University of Posts and Telecommunications, ChongQing 400065)

Abstract The disk manager of spatial database provides uniform storage and management for spatial data and attribute data in GIS applications. To investigate it will have effect on creating storage manager of spatial database and SDBMS with lore's property right for our country. This paper investigates and analyzes the Oracele's spatial data model, and on the basis of it , it puts forward a new method about disk storage of spatial data and has it designed.

Keywords Spatial database, Geographical information system(GIS), Disk manager, Page manager, Binary large object (BLOB)

1. 前言

空间数据库[1,3~5]在GIS[2]中的作用与地位越来越重要，并将成为GIS产业之争的主导者之一。空间数据信息不仅有一般属性信息的特征，还具有数据量大、多尺度等特性。因此为了高效地存储和管理大容量的空间数据，以提高现有数据的利用效率和共享程度，研究和设计空间数据库磁盘管理器是十分必要的，也是进行空间数据快速访问和处理的基础。该研究将填补目前我国空间数据库及其磁盘管理器研究的空白，并将对建立我国具有自主知识产权的GIS基础软件、空间数据库管理系统产生十分积极的作用。随着数据库处理能力的增强、GIS应用需求的增加和数据量的日趋庞大，建立和使用真正意义的空间数据库管理系统已是大势所趋。

2. 空间数据

2.1 类型

Oracle空间数据库支持三种基本的空间几何类型：1. 点（例：天上的星星）；2. 线（例：道路）；3. 多边形（例：行政区域边界）。

2.2 模型

Oracle提供的空间数据模型包含元件（基本的几何类型）、几何体和层（Layer）。层是由几何体或几何体对象构成，而几何体又是由三种基本的几何类型构成，因此，它又被称为层递式结构模型。

（1）*元件（点、线和多边形）* 是构成空间几何物体的最基本模块，元件的每个坐标是用X（横坐标）和Y（纵坐标）来表示。点包含一对X，Y坐标；线由两对X，Y坐标构成；多边形包括多条有向线段，即包含多对X，Y坐标，这些坐标顺时针或逆时针定义在多边形的周围。

（2）*几何体* 由一组原始的元件构成，用来描绘空间几何特性。每个几何体都有一个唯一的标识符（GID）。使用GID将几何体中的一组元件互相联系。

（3）*层* 是具有相同属性的各种各样的几何体的集合。例如：某一层在GIS中包含地形学的特征；某一层也许是用来描绘人口密度的；而另一层也许是用来描绘某个区域中的道路和桥梁（用点和线表示）的；每层的几何体对象和它们的空间索引都储存在标准表中。

2.3 数据库结构

Oracle是用四个数据库表（见图1）来存储空间数据和为空间数据建立索引的，这四个表通常称作为层。Oracle使用自定义的函数使这四个表互相联系。

Tablel-1 〈layername〉_SDOLAYER

SDO_ORDCNT	SDO_LEVEL	SDO_NUMTILES	SDO_COORDSYS
〈number〉	〈number〉	〈number〉	〈varchar〉

Tablel-2 〈layername〉_SDODIM table or view

SDO_DIMNUM	SDO_LB	SDO_UB	SDO_TOLERANCE	SDO_DIMNAME
〈number〉	〈number〉	〈number〉	〈number〉	〈varchar〉

Tablel-3 〈layername〉_SDOGEOM table or view

SDO_GID	SDO_ESEQ	SDO_ETYPE	SDO_SEQ	SDO_X1	SDO_Y1	…	SDO_Xn	SDO_Yn
〈number〉	〈number〉	〈number〉	〈number〉	〈number〉	〈number〉	…	〈number〉	〈number〉

范建伟　硕士研究生。汪林林　教授，从事GIS系统、数据库系统、计算机算法分析、网络等方面的研究。

Tablel-4 〈layername〉_SDOINDEX table

SDO_GID	SDO_CODE	SDO_MAXCODE	SDO_GROUPCODE	SDO_METAK
〈number〉	〈raw〉	〈raw〉	〈raw〉	〈raw〉

图1 空间数据结构

各个表的每列定义如下：

·〈layername〉_SDOLAYER：SDO_ORDCNT 列表示在〈layername〉_SDOGEOM 数据库表中每行有几列用来存放坐标值。SDO_LEVEL 列表示进行索引时分割这个图层的次数。SDO_NUMTILES 列表示划分〈layername〉_SDOGEOM 数据库表中每个几何对象所需 TILES(TILE 索引法)的个数。SDO_COORDSYS 列指定你所用的坐标系统的名称(直角坐标，极坐标等)。

·〈layername〉_SDODIM：SDO_DIM 列表示几何对象的维数，第一行是1，然后每行递增。SDO_LB 列表示该维坐标的下限(例如纬度不低于-90)。SDO_UB 列表示该维坐标的上限(例如经度不超过180)。SDO_TOLERANCE 列表示两个点相距的距离不超过该列的值可以被视为同一个点。SDO_DIMNAME 列指出该维的名称(例：X，Y 坐标，经度，纬度)。〈layername〉_SDOGEOM：SDO_GID 列表示某图层中几何对象的唯一数字标志符。SDO_ESEQ 列枚举了几何对象中的每个元件(点、线、多边形)。SDO_ETYPE 列表示每个元件的几何类型(1代表点，2代表线，3代表多边形)。SDO_SEQ 列记录组成该元件的数据在每行的顺序。SDO_X1列表示第一个坐标的横坐标的值。后几列类似。

·〈layername〉_SDOINDEX：SDO_GID 列与上面定义的相同。SDO_CODE 列表示该层中的某个几何对象所覆盖的 TILE 的标志符。SDO_MAXCODE 用来在使用可变 TILE 进行索引时描绘当前象限最小的那个逻辑 TILE。SDO_GROUPCODE 是 SDO_CODE 的前缀。SDO_META 提供足够的信息来找到某个 TILE 的边界。

3. 存储空间数据的方法

空间数据信息不仅有一般属性信息的特征，还具有数据量大、多尺度等特性。Oracle 用了四个数据库表来存储一个层的信息，其数据量非常大，而 BLOB 可以一次分配多个页来存储大量的二进制数据，即可以实现对 Oracle 中层的存储，因此为了高效地存储和管理大容量的空间数据，采用 BLOB (Binary Large Object 二进制大对象)方式进行存储。BLOB 在磁盘上的结构如图2所示。

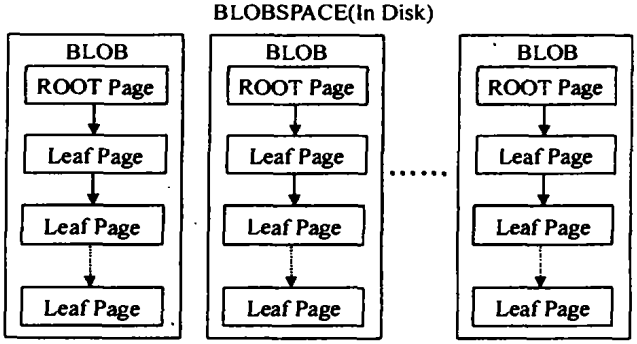

图2 BLOB 在磁盘上的数据结构

3.1 BLOB 的数据结构(BLOB 存储在磁盘上)

(1)BLOB 空间是由若干个 BLOB 组成，BLOB 空间和空间里的 BLOB 对象都有自己的 ID(BLOBSpaceID 和 BLOBID)。

(2)每个 BLOB 对象由许多的 Page 组成，每个 Page 有自己的 ID。BLOB 对象中的第一个 Page 是 root Page，root Page 通过它的数据结构中的 NextPage 指针指向第一个 leaf Page，第一个 leaf Page 通过它的数据结构中的 NextPage 指针指向第二个 leaf Page，依次类推，所有的 Page 组成一个链表。空间数据就存放在 Leaf Page 中。

4. 存储空间数据的磁盘页管理器的设计

page 是 Disk Manager[6~15]管理对象中最小的单位，是磁盘和 Buffer 之间交换数据的最小单位。Block 是物理上邻接的多个 Page 集合，是给文件(File)分配、回收空间的最小单位。文件是由多个 Block 构成，理论上是 Recorder(记录)存放的空间。Segment 是物理上连续的部分 Disk Partition 或全部 Disk Partition，或一般 OS 上的文件。Database 由一个或多个 Segment 构成，理论上是一个 Disk Object，每个 File 都属于一个 Database。

4.1 数据页

Data Page 是 File(文件)中数据实际存放的地方，由存放 Data(数据)的数据区域和存放 Page 自己属性的 Header 部分构成。

4.1.1 Data Page 的数据结构(C/C++描述)

```
Typedef struct tagPageHeader{
        Short nTotalCount; /* 该页中能存放记录的总个数 */
        Short nCurr; /* 该页目前存放记录的个数 */
        Short nDeleted; /* 该页已删除记录的个数 */
        Int nLink; /* 溢出的页数，没有溢出页值为-1 */
        Long nAllocSpace; /* 该页已分配空间的大小 */
        Long nFreeSpace; /* 该页空闲空间的大小 */
        Long nOffset; /* 该页中存放下一个记录的偏移量 */
}PAGEHEAD;
typedef struct tagPage{
        PAGEHEAD ph;
        Char Body [1];
}PAGE;
```

4.1.2 Data Page 的数据结构的示意图(如图3)

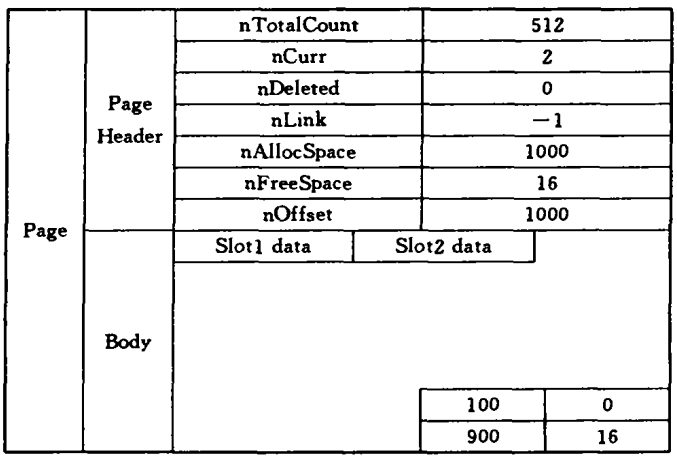

图3 Data page's Data Structure

4.1.3 Page Manager 的数据结构的设计

设计思想：

(1)一个 Disk Object 物理上是由一个或多个 Segment 构成，这里我设定一个 Disk Object 包含256个 Segment；一个 Segment 是由一个或多个 Block 构成，也设定一个 Segment 包含256个 Block；一个 Block 是由邻接的多个 Page 构成，也设定一个 Block 包含256个 Page。

(2)一个 Disk Object 逻辑上是由一个或多个文件构成(这里文件是指数据库系统的表空间的概念)，这里设定一个 Disk Object 含有其相应的 Page Info File 和 Page File：

Page Info File 存储了每个 Block 所包含的256个 Page 的指针(偏移量)，如果该 Disk Object 占用了几个 Page，那么 Page Info File 就存储了这几个 Page 的指针，其他的偏移量(指针)被赋予-1，来表示没被该 Disk Object 使用。

Page File 是由该 Disk Object 占用的那几个 Page 本身构成，各个 Page 顺次排列。

(3)Segment，Page，Block 和 File 之间的关系如图4所示。

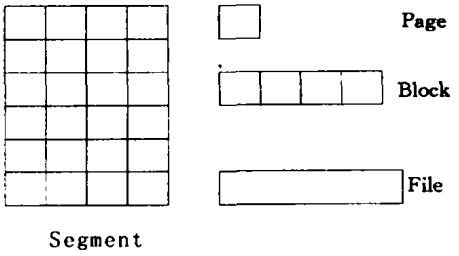

图4 Segment，Page，Block 和 File 的关系

这样就把 Page Manager 的各个部分有机地结合起来了，各部分之间紧密联系，如图5所示。

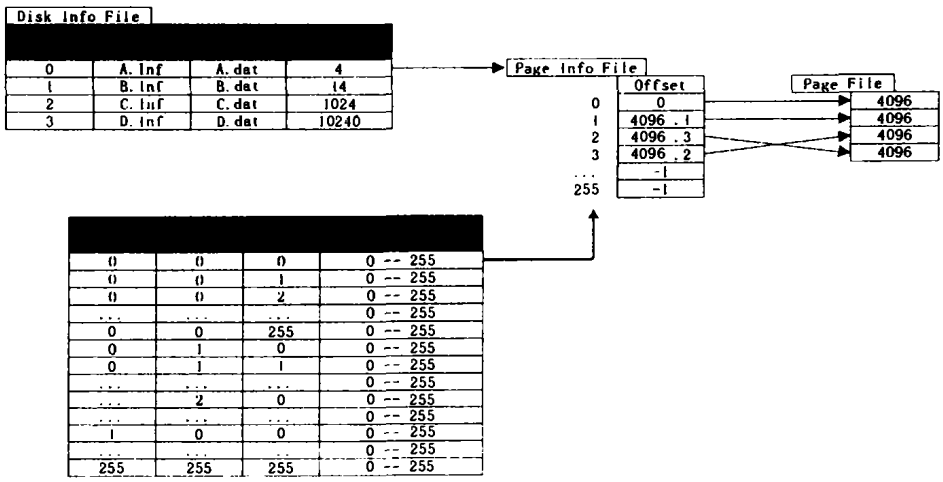

图5 Page Manager's Structure

4.2 具体编程实现的设计

4.2.1 Page Class 的设计(用 C/C ++描述)

```
class CPage
{
public:
        CPage();
        CPage(long nPageSize);
        CPage(long nBasePageSize, int nCount);
        ~CPage();
public:
        //Member Functions
        int Alloc();
        int Free();
        int AttachPage(PAGE * pPage);
        int DetachPage(PAGE * pPage);
        //Data Read/Write, Delete Functions
        long ReadData(long nSid, void * pCont);
        long WriteData(void * pCont, long nSize);
        long DeleteData(long nSid);
        //Base Data Manipulation Functions
        long CopyData(long nOffset1, long nOffset2, long nSize);
        long ModifyToSmallData(long nIndex, void * pCont, long
        nSize);
        long ModifyDataInAllocSpace(long nIndex, void * pCont,
        long nSize);
        ModifyDataAfterPageAdjusting ( long nIndex, void *
        pCont, long nSize);
        //Data Modify, Move Functions
        long ModifyDataInPage(long nSid, void * pCont, long
        nSize);
        //Gabage Collection Functions
        long PageAdjusting(void);
        //Page Attribute Functions
        long IsWritable(int nSize);
        long ReadSlotData(int nSlot, SLOT * pSlot);
public:
        //Attributes
        long m_nPageSize;
        long m_nBodyLoc;
        PAGE * m_pPage;
        PAGEHEAD * m_pPageHead;
        SLOT * m_pSlot;
};
```

4.2.2 功能函数的分析与设计

(1)当我们对数据页中的数据进行操作时(读数据、写数据、修改数据和删除数据)，首先我们要寻找到这个记录，根据 PageID 和 SlotID 就可以找到记录，然后再进行对数据的操作(ReadData()，WriteData()，DeleteData())。

PageID 和 Slot 的数据结构：(用 C/C ++描述)

```
Typedef struct{
    Short DiskNo; //disk 的编号
    Short SegmentNo; //Segment 的编号
    Short BlockNo; //Block 的编号
    Int PageNo; //Page 的编号
}PID;
Typedef struct{
    Short Offset; //该 Slot 所对应数据存放的起始位置
    Int Size; //该 Slot 所对应的数据的大小
}SLOT;
```

(2)当一页已经被记录写满时，我们就用 Alloc()函数申请新的一页，如果某页因为删除等操作，致使该页没有任何记录存在，为了有效地利用磁盘空间，我们调用 Free()函数把该页释放，以供重新被申请。

(3)每次进行页分配的时候，为了提高系统的效率，总是一次性分配多页，如果数据页已满，则调用 AttachPage()函数来增加页以便存放数据。如果该页中原来的记录完全被删

(下转第117页)

sachusetts Institute of Technology, 2001
23 Joachims T. Text Categorization with Support Vector Machines: Learning with many Relevant Features. Machine Learning, Lecture notes in computer science, 1998, 1398: 137～142
24 Leopold E, Kindermann J. Text Categorization with Support Vector Machines-How to Represent Texts in Input Space ?. Machine Learning, 2001, 46(1/3)423～444
25 Jonsson K, Kittler J, Li Y P, Matas J. Support vector machines for face authentication. Image and Vision Computing, 2002, 20: 369～375
26 Li Yongmin, Gong Shaogang, Sherrah J, Liddell H. Multi-view Face Detection Using Support Vector Machines and Eigenspace Modelling. In: Proc. Intl. Conf. on Knowledge-based Intelligent Engineering Systems and Allied Technologies, Brighton, UK, Sep. 2000. 241～245
27 Huang J, Shao Xuhui, Wechsler H. Face Pose Discrimination Using Support Vector Machines(SVM). In: Proc. of 14th Intl. Conf. on Pattern Recognition, 1998
28 Wang Yingjie, Chua C-S, Ho Y-K. Facial feature detection and face recognition from 2D and 3D images. Pattern Recognition Letters, 2002, 23: 1191～1202
29 Oren M, Papageorgiou C, Sinha P, et al. Pedestrain detection using wavelet templates. In: Pros. of CVPR'97, Puerto Rico, 1997
30 Hearst M A, Scholkopf B, Dumais S, et al. Trends and controversies-support vector machines. IEEE Intelligent Systems, 1998, 13(4): 18～28
31 Lu Chunyu, Yan Pingfan, Zhang Changshui, Zhou Jie. Face recognition using support vector machine. In: Proc. of ICNNB'98, Beijing, 1998. 652～655
32 Mayoraz E N. Multiclass Classification with Pairwise Coupled Neural Networks or Support Vector Machines. Artificial Neural Networks - ICANN'2001, Lecture notes in computer science, 2001, 2130: 314～321
33 卢增祥，李衍达. 交互支持向量机学习算法及其应用. 清华大学学报，1999，39(7)：93～97
34 阎辉，张学工，李衍达. 应用SVM方法进行沉积微相识别. 物探化探计算技术，2000，22(2)：158～164
35 闻芳，卢欣，孙之荣，李衍达. 基于支持向量机(SVM)的剪接位点识别. 生物物理学报，1999，15(4)：734～739
36 Morris C W, Autret A, Boddy L. Support vector machines for identifying organisms--a comparison with strongly partitioned radial basis function networks. Ecological Modelling, 2001, 146(1-3): 57～67
37 Hadzic I, Kecman V. Support Vector Machines Trained by Linear Programming: Theory and Application in Image Compression and Data Classification. In: Proc. of 5th Seminar on Neural Network Applications in Electrical Engineering, 2000. 18～23
38 Roobaert D, et al. View-based 3D object recognition with Support Vector Machines. In: Proc. IEEE Neural Networks for Singnal Processing Workshop. 1999
39 Pontil M, Verri A. Support Vector Machines for 3D Object Recognition. IEEE transactions on pattern analysis and machine intelligence, 1998, 20(6): 637～647

(上接第95页)

除，则调用 DetachPage()函数来丢弃该页，但不释放，以便需要新增加页的时候直接获取，而不必去调用 Alloc()函数去分配页。这样就大大节约了系统处理的时间。

(4)当某页删除了某些记录以后，页中就剩下了很多空闲的空间，如果没有处理这种现象，将会有很多的磁盘空间被白白地浪费，所以调用 PageAdjusting()函数来弥补这一不足之处。

(5)当向某页进行写记录时，首先调用 IsWritable()函数，看该页是否有记录长度与 Slot 长度的总和大小的空间，如果有，就返回真，否则，返回假。

结束语 本文设计的健壮的统一管理空间数据和属性数据的、通用的、具有自主知识产权的空间数据库磁盘管理器将大大减少后继系统开发的费用，避免重复设计，其通用性不仅保证了该产品可为多种系统开发或应用开发所用，从而节省了开发成本，而且提高了 GIS 等应用中的空间数据存储和检索效率。整个设计采用模块化系统设计和程序设计方法，提高了代码的可重用性，增强了系统的开放性、可靠性及稳定性。

参 考 文 献

1 Ralf Hartmut Guting. An Introduction to Spatial Database Systems. The VLDB Journal, 1994, 3(4)
2 Abel D J. SIRO-DBMS: A Database Tool Kit for Geographical Information Systems. Intl. J. of Geographical Information Systems, 1989, (3): 103～116
3 Abdelmoty A I, Williams M H, Paton N W. Deduction and Deductive Databases for Geographic Data Handling. In: Proc. 3rd Intl. Symposium on Large Spatial Databases, Singapore, 1993. 443～464
4 Aref W, Samet H. Extending a DBMS with Spatial Operations. In: Proc. 2nd Intl. Symposium on Large Spatial Databases, Zurich, 1991. 299～318
5 Egenhofer M, Frank A, Jackson J P. A Topological Data Model for Spatial Databases. In: Proc. First Intl. Symposium on Large Spatial Databases, Santa Barbara, 1989. 271～286
6 Stonebraker M. the design of the POSTGRES storage system, EECS Department, University of California, Berkeley, Ca., 94720
7 Weikum G, Zabback P, Scheuermann P. Dynamic File Allocation in Disk Arrays: [Technical Report]. ETH Zurich, 1990
8 Carey M J, et al. Storage Management for Objects in EXODUS. In: Kim, W., Lochovsky, F. H., eds. Object Oriented Concepts, Databases, and Applications, Addison-Wesley, 1989
9 Livny M, Khoshafian S, Boral H. Multi-Disk Management Algorithms, ACM SIGMETRICS Conf., 1987
10 Moad J. Relief for Slow Storage Systems. Datamatton, 1990, 36(17)
11 Patterson D A, Gibson G, Katz R H. A Case for Redundant Arrays of Inexpensive Disks(RAID), ACM SIGMOD Conf., 1988
12 Reddy A L, Banerjee P. An Evaluation of Multiple-Disk I/O Systems. IEEE Trans. on Computers, 1989, 38(12)
13 Weikum G, Zabback P, Scheuermann P. Dynamic File Allocation in Disk Arrays: [Technical Report]. ETH Zurich, 1990
14 Method for allocating computer disk space to a file of known size. IBM Technical Disclosure Bulletin 27, 10B Mar. 1985. 6260～6261