计算机科学2002Vol. 29№. 4

类属 B 树方法在 GIS 系统中的应用*)

Generalized B-trees are Applied in GIS

鞠时光[1]　马正华[2]

(江苏大学　镇江212013)[1]　(江苏石油化工学院　常州213016)[2]

Abstract　This paper introduces generalized B-trees. Then it describes how to design the dynamic type system based upon the generalized B-trees and apply to the GIS systems. In additional, in the paper, the work process and functions of generalized index mechanism are described.

Keywords　B-trees, GIS, Data structure

1　引言

自80年代以来，数据库新的应用领域不断出现。这些新兴的应用领域如：计算机集成制造系统 CIMS、多媒体系统、地理信息系统(GIS)等，要求支撑的数据库管理系统(DBMS)能存储和处理各种复杂对象；支持新的数据类型和操作；支持动态模式修改；存储和处理四维环境下的信息{x,y,z,t}；并能提供基于四维关系上的查询功能[5]。

实现这种系统的关键是：DBMS 的存储机制及索引机制必须能适用于任意的数据类型[3]。而传统的关系型 DBMS 只为数字和字符串两种数据类型内置了 B 树索引代码。文[1]在1986年提出了用户自定义数据类型的思想。这个思想为开发对象关系型商用数据库奠定了技术基础，如 Postgres 的索引系统[2]、扩展的 DB2系统[7]、支持可扩充的 B^+ 树和 R 树。

但 B^+ 树仅支持区域谓词(<，=，>)，R 树仅支持 n-d 区域查询(包含，被包含，相等)。例如，用 B^+ 树对一组电影进行索引，难以进行诸如"找出巩俐所曾演过的电影"这种查询。Informix 使用一种称为数据刀片(DataBlade)的标准软件模块，插入到数据库中，用以扩展 DBMS 的能力，来解决特殊数据的管理问题[8]。这种模块实际是一种类似于类库的面向对象的软件包，它封装了特定的数据类型。对每一个特殊的数据类型必须有一个相应的 DataBlade 模块。所以，一个完备的系统也许需要装入若干个 DataBlade 模块。而且，这种类型系统与 DBMS 是完全分离的。

我们使用 Visual C++4.0，在 Windows 平台上实现了以类属的 B 树为内核的 DBMS，并将其应用于管理地理信息系统。本文将系统地介绍类属 B 树模块的定义，以及基于其上的用户自定义数据类型、谓词及操作函数的接口方法，并举例说明了类属的 B 树在数据库搜索机制中所具有的重要作用。

2　类属 B 树的定义

索引是一种把关键字的值映射到一个或多个含有该值的记录上的组织数据的方法。我们实现的类属 B 树(记为 GBT)，可看作是 m-路查找树的特例[4]。它也是一种平衡树。就如其它索引树一样，GBT 存储一对(关键字 K_i，记录标识 RID)在它的叶结点中。这里 RID 指出了在这个数据页上相应的记录。树的内结点存储一对(关键字 K_i，子页指针 A_i)。

我们定义的这种 GBT 树模块具有以下一些特点：

(1)GBT 带有三个参数：KT，Less_Than，LessOrEqual。此处，KT 表示关键字的数据类型；Less_Than 表示逻辑上的比较谓词"<"运算；LessOrEqual 表示逻辑上的比较谓词"≤"运算。

(2)它最多有 m 棵子树，其根具有如下结构：

$(A_0, K_1, A_1, K_2, A_2, \cdots, K_n, A_n)$

其中：$A_i(0 \leqslant i \leqslant n \leqslant m)$是指向子树的子页指针，$K_i(1 \leqslant i \leqslant n \leqslant m)$是关键字，且 K_i 的数据类型为 KT。

(3)如果一个结点 N 的所有关键字有序地排列成 $K_1, K_2, \cdots, K_m$，则 N 的子树也存在一个有序的排列 $A_0, A_1, A_2, \cdots, A_m$，且

·在子树 A_0 中的所有关键字值的集合$\{K_0\}$满足：$\forall \{K_0\}$ LessOrEqual K_1；

·在子树 A_m 中的所有关键字值的集合$\{K_m\}$满足：K_{m-1} Less_Than $\forall \{K_m\}$；

·在子树 $A_i(0 \leqslant i < n)$中的所有关键字值的集合$\{K_i\}$满足：K_{i-1} LessOrEqual $\forall \{K_i\}$ Less_Than K_{i+1}。

(4)子树仍然是一个 GBT 树。

GBT 中，关键字 K_i 的数据类型并不像 B 树中的那样单一。它可以是用户定义的任一种数据类型。另外，比较谓词 Less_Than 、LessOrEqual 并不只是代表传统的 B 树中所具有的比较谓词"<"、"≤"，而是具有更广泛的含义。要使得 GBT 能正确地在用户定义的数据类型之上进行操作，我们必须让 GBT 知道这个新数据类型所具有的物理含义。即用户需定义新数据类型的表示法以及书写它的函数。同时定义与之相应的比较谓词(与新数据类型相匹配的比较操作函数模块)。为了使读者更容易理解下面将要介绍的用户自定义数据类型的过程，我们首先构造一组面向 GIS 应用领域的数据模型。

3　地理数据类型的定义

一个 GIS 系统需要存储三维空间中的对象。这些对象可以是点或其它任何形状。它的数据元素是二维或三维的。它可以用来描述房屋、道路、桥梁、管道和其它地物。通过对这些处理对象的分析与归纳，我们抽象出下面四种数据模型作为开发 GIS 应用系统所需使用的数据类型。

3.1　点模型 POINT

一个地物对于某个参照系其大小已无关紧要时，可将其抽象成一个点模型。如一个大学或一个工厂所占空间在小比例尺地形图上已无法表示其大小，只能用一个点来表示。其点

*)本课题的研究得到国家自然科学基金(No. 60173064)及江苏自然科学基金(No. BK99110)的资助。

模型可记为 POINT：

POINT(name,x,y,z,parameter)

这里 name 表示点的名称；x,y,z 为全球定位系统坐标；parameter 为点的参数；其中 name,parameter 均可缺省。

3.2 曲线模型 CURVE

一个对象实体在某个参照系中，其宽度已无关紧要，重要的是其在空间所刻画的轨迹。这种对象可抽象为曲线模型 CURVE：

CURVE(name,P_1,P_2,…,P_n,parameter)

其中，P_1,P_2,…,P_n 为曲线顶点，其数据类型为 POINT；parameter 为曲线的参数，用来定义线型、粗细、颜色等。

3.3 面模型 AREA

一个空间实体，在系统中，其厚度无关紧要。我们把这种对象定义为面模型 AREA。

AREA(name,P_1,P_2,…,P_n,P_1,parameter)

其中 AREA 表示一个平面，P_1,P_2,…,P_n 为该面封闭周线上的顶点，其数据类型为 POINT ；parameter 为面特征参数。由于面特征的多样化，可能需要一个复杂的数据结构才能描述这个特征。所以面特征参数可以是一个指针，去指向这个复杂的数据结构。

3.4 体模型 VOLUME

对一个空间实体，需表示出其三维表面特征，我们将这种对象定义为体模型 VOLUME；我们用有向的任意三角平面集合来定义其外表面

VOLUME{ name,A_1,A_2,…,A_n}

这里 A_i,i=1,2,…,n 为有向的空间任意三角平面。其数据类型为 AREA，其数据结构可用点-面表来实现。如图1所示的空间物体，可用图2所示的点-面表来表示。

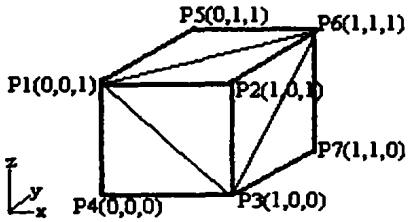

图1 空间实体例

点表：

Pi	x	y	z
P1	0	0	1
P2	1	0	1
P3	1	0	0
P4	0	0	0
P5	0	1	1
P6	1	1	1
P7	1	1	0
P8	0	1	0

面表：

Ai	Pm	Pn	Pk
A1	P1	P4	P3
A2	P1	P3	P2
A3	P2	P3	P6
…	…	…	…

图2 空间实体对应的点面表

以上四种数据模型可用来表示地理信息系统中所涉及到的地物。而在传统的 DBMS 中，没有一种数据类型能用来直接表示这些模型。为了便于用户开发其应用系统，我们提供给用户一种自定义数据类型的手段。可将应用系统所需的这些数据类型添加到 DBMS 的类型系统中。我们在下面的介绍中，以这四种数据模型作为应用实例，阐述如何基于 GBT 树来扩充基类。

4 GBT 树中比较谓词的定义

由于 GBT 树中，结点关键字 K_i 的数据类型是可变的，这就导致比较谓词必须随着关键字类型的变化而变化。即如果 GBT 模块中，关键字的数据类型 KT 是用户自定义的，则 GBT 模块中使用的比较谓词 Less_Than 和 LessOrEqual 也必须使用相应的用户自定义的比较谓词。所以，每当定义一个新的数据类型，同时必须定义一组相应的比较谓词函数。

表1中举例列出了几种数据类型所对应的比较谓词函数。第一行表示在传统的 B 树中，处理整型数使用的常规的比较谓词。GBT 一栏的第一行列出了在类属的 B 树模块中，用 Less_Than 作为其形式参数，来对应传统的 B 树中的比较谓词“<”。用 LessOrEqual 来对应传统的 B 树中的比较谓词“≤”。

表1

		数据类型	比较谓词	
B_trees		Inteqer	<	≤
GBT	形式参数	KT	Less_Than	LessOrEqual
	用户自定义类型	POINT	North_of	NorthOrSame
	用户自定义类型	CURVE	Shorter	ShorterOrSame
	用户自定义类型	AREA, VOLUME	Smaller	SmallerOrSame

当使用 GBT 进行操作时，如果索引关键字的数据类型 KT 为 POINT 类型时，则 GBT 模块中的形式参数 Less_Than 用 North_of 函数来置换。同样，如果索引关键字的数据类型 KT 为 CURVE 或 AREA、VOLUME 时，GBT 模块中的 Less_Than 参数用 Shorter 或 Smaller 来替换。而这些比较谓词函数 North_of、Shorter 或 Smaller 则由用户根据应用系统的实际需要来定义其物理意义，即用 VC 编写相应的函数。例如，基于地理信息系统应用的要求，我们需将 POINT 类型的数据在其 X 坐标的正方向上索引排序。在此意义上，我们编写出比较谓词函数模块 North_of 如下：

```
Boolean Function North_of(point_1,point_2: POINT)
  /* two parameters that have data types POINT */
{
 if (point_1.x > point_2.x)/* to suppose that the positive direc-
    tion of the axis x of coordinate system is north */.
 then return (true)
 else return (false)
 }
```

又如，根据应用需求，CURVE 类型的数据需依据曲线长度进行索引排序。在此意义上，我们可编写比较操作函数模块 Shorter 如下：

```
Boolean Function Shorter(line_1,line_2:CURVE)/* two parame-
ters that have data types CURVE */
  {
    a = length(line_1);
    b = length(line_2);
    if a<b then return(true)
         else return(fals)
    }
Float Function length(L:CURVE)
  {
    N = L.n; /* to obtain the vertex number of the curve */
    sum =0;
```

```
for i=1 to N-1
    sum=sum+distance(p_{i+1}-p_i);
return(sum);
}
```

这些比较操作函数模块，用户可借助于下节所述的类型构造器来定义，并将其置入数据库管理系统的类型系统中。

5 类型构造器

GBT 树作为 GIS 系统的索引机制，将类型系统向用户开放。允许用户定义的抽象数据类型加进其类型系统。然后，就像传统的系统提供的类型一样，被数据库的最终用户所调用。这样，数据库的输入和输出都有可能是用户自定义的数据类型的数据（例如 POINT 类型，CURVE 类型，AREA 类型等等）。所以，创建一个新的基类，需用户定义类型名、类型的存储信息和将该数据类型与 ASCII 字符互相转换的子程序。用户自定义的基类是完全封装的，通过定义在其上的函数来访问。

我们在数据库的 ORSQL 语言中，提供了一个用户自定义数据类型的工具，称为类型构造器。这个类型构造器包裹在 GBT 模块的外围。正如图3所示，类型构造器由四部分所组成：类型存储信息定义模块，比较谓词函数编辑模块，查询操作函数编辑模块和类型管理模块等。

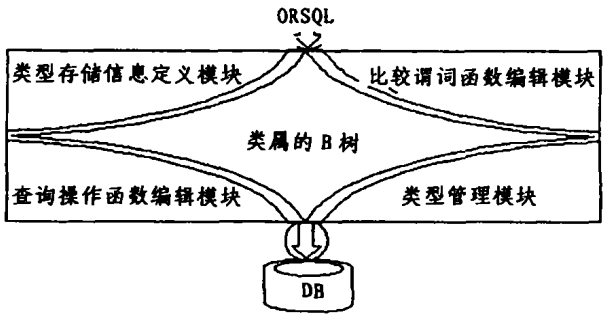

图3 类型构造器的组成

5.1 类型存储信息的定义

当用户要求定义新的数据类型时，启动类型构造器，屏幕上首先显示出定义类型存储信息的窗口。该窗口要求用户输入：类型名、类型结构及比较谓词函数名等。在图 4中，我们试图定义一个名为 POINT 的新类型。首先，如果有继承关系则给出新类型的父类型及调用权限。然后分成类型成员、比较谓词和操作函数三部分来定义。在类型成员定义中，逐个地定义其成员变量名、成员变量类型、调用权限，并将其添加到相应的显示窗中。类型的存储结构可用 VC 中的组合、集合、指针等来编写一个复合结构。该例中，我们将 POINT 类型定义为具有四个分量的组合结构。

在比较谓词定义中，给出相应于“<”和“<=”的比较谓词函数名“North_of”和“NorthOrSame”，以及相应的函数代码，系统将借助于 VC 编译器对用户所定义的函数代码进行语法检查、编译等工作。

图4中第三部分是关于操作函数的定义。这些操作函数是基于 POINT 类型之上，提供给最终用户的查询操作函数。这里，我们用 Contained 和 Above 二个操作函数作为示例。同样，在图4所示的窗口中，只定义其函数返回值的类型、函数名、函数所具有的参数以及该函数的调用权限。然后，将这个函数添加到类型系统中。同时，在显示窗中显示函数的信息。同法，可定义其它新类型。

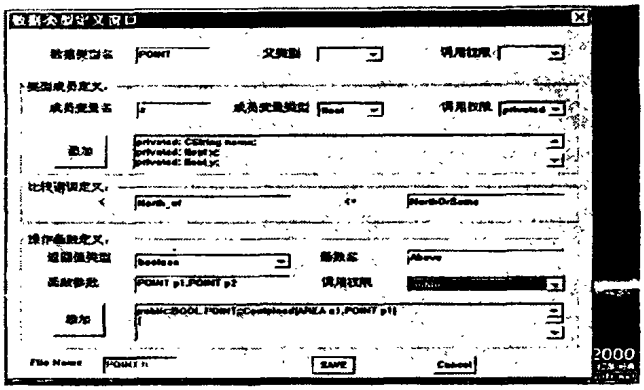

图4 定义 POINT 类型存储信息

5.2 查询操作函数的讨论

上一段我们提到，基于用户定义的新类型之上，需定义一些查询操作函数。好让最终用户在数据库查询程序中调用。编制代码的依据是函数在应用领域所代表的物理意义。例如，我们试图为地理信息系统的 POINT 类型引入二个查询操作函数 Above 。首先，我们需分析在地理信息系统中与 POINT 类型的数据相关的查询，并从中抽象出共性，其函数算法如下。

```
Boolean Function ABOVE(s1,s2)
/ * to retrieve the objects S1 that are located above of the objects S2.
The data type of S1 and S2 ∈(POINT,CURVE,AREA,VOLUME)
. And S1,S2∈flat files f. * /
(Processing algorithm)
  parse ORSQL and interpret it to operation-lines;
   interpret header of files f.
    while there are more entities s in the file
     begin
      next E(f,s);
      if test E(f,s)∈ s1 then E(f,s)-> T1;
      if test E(f,s)∈ s2 then E(f,s)-> T2;
     end
    while there are more entities s1 in T1
     begin
      next E(T1,s1);
      if test E(T1,s1)is up of ∀ E(T2,s2)
      then insert E(T1,s1)into the out-line;
     end
     To explain and put out the output-lines.
```

5.3 动态类型管理机制

在 GBT 中，为了使用户定义的数据类型能象系统内部提供的类型一样使用，我们开发了一种动态类型管理机制。这个机制的核心部分使用如图5所示的类型系统目录表。

```
...
用户数据类型目录表::数据类型名，类型来源，谓词函数目录起始，谓词个数；
用户谓词函数目录表::谓词名，谓词函数标识，谓词函数代码起始指针；
用户操作函数目录表::操作函数名，参数个数，返回值类型，函数代码起始指针；
...
```

图5 类型系统目录表的组成

每当用户定义一个新的类型，将对类型系统目录表的元素进行添加或修改。例如，当用户使用类型构造器定义了 POINT、CURVE、AREA、VOLUME 等新的数据类型时，在类型系统目录表中将记录下关于这些新类型的详细信息。正如表2－表4所示。

表2 用户数据类型目录表

数据类型名	类型来源	谓词函数目录起始	谓词个数
POINT	User-defined	1	2
CURVE	User-defined	3	2
AREA	User-defined	5	2
VOLUME	User-defined	7	2

表3 用户谓词函数目录表

谓词名	谓词函数标识	谓词函数代码起始指针
North_of	North_of	→
NorthOrSame	NorthOrSame	→
Shorter_Than	Shorter_Than	→
ShorterOrSame	ShorterOrSame	→
Smaller_Than	Smaller_Than	→
SmallerOrSame	SmallerOrSame	→

表4 用户操作函数目录表

操作函数名	参数个数	返回值类型	函数代码起始指针
Contained	2	boolean	→
Above	2	boolean	→
Length	1	float	→

这样，这些用户定义的数据类型就像内部构造的数据类型一样让应用程序使用。当用 GBT 树进行索引或对 ORSQL 查询程序进行编译时，均需建筑在这个类型系统目录表之上。

例1 假设在对象关系型数据库中，有一对象实例的集合。每个对象实例具有 n 个属性。这些属性可以是关系型的、对象型的或其它。其中，记录的关键码定为属性 i。并且，属性 i 的数据类型为 POINT 类型，其值如图6中所示(在此只列出属性 i 的数值)。

	属性1	属性2…	属性 i	…属性 n
对象实例1	…	(p_1,49.2,300.0,19.0,0)		…
对象实例2	…	(p_2,139.4,279.0,18.8,0)		…
对象实例3	…	(p_1,20.0,300.0,19.0,0)		…
对象实例4	…	(p_2,75.5,279.0,18.8,0)		…
对象实例5	…	(p_1,53.0,300.0,19.0,0)		…
对象实例6	…	(p_2,36.9,279.0,18.8,0)		…
对象实例7	…	(p_m,145.3,167.0,23.0,0)		…

图6 一组对象实例，索引关键码为其第 i 个属性域

当使用 GBT 对图6中所列对象关系型数据进行索引时，首先检查所用关键字的数据类型。当关键字的类型是用户定义的，如 POINT 类型，则 GBT 从类型系统中选择出 POINT 类型所对应的比较谓词，如 North_of 和 NorthOrSame。并用这些比较谓词函数去置换 GBT 模块中的形式参数 Less_Than 和 LessOrEqual。因而 GBT 树使用 North_of 和 NorthOrSame 函数作为比较谓词去索引以点类型 POINT 为关键字的一组数据。

通过 GBT 索引后，我们可得到一个如图7所示的3阶的 GBT 树。它实际是将这组对象按其第 i 个属性项的 X 坐标排序而获得。这就使对象关系型数据库能对任意类型的信息进行快速查询成为可能。

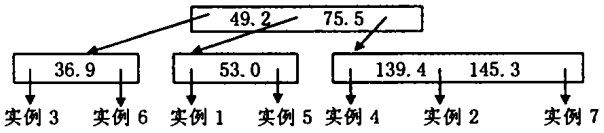

图7 对应图6中所示对象关系型数据的 GBT 索引树

结束语 我们将关系数据库技术与面向对象技术融合在一起，以地理信息系统作为应用领域，使用 Visual C++4.0在 Windows 平台上，实现了一个带有动态类型系统及其类属的索引机制的对象关系型数据库管理系统(ORDBMS)内核。

实现过程中，引进了类属 B 树的概念，将类型系统开放，使得类型管理机制及索引机制能支持用户自定义的数据类型、比较谓词和运行于这种类型之上的查询操作函数。并且，开发了类型系统目录表这种数据结构，使得类型管理机制，能够动态地将用户定义的数据类型、谓词、操作函数等登记到数据库管理系统中，就象内部构造的类型一样让最终用户使用。同时，类属的索引机制能够将对象关系型数据在任意的数据类型之上进行索引和排序。这就有效地解决了 GIS、多媒体、时空等数据库应用中，管理新兴数据类型的迫切要求。

感谢陈伟鹤、蔡涛、马汉达等对实现系统的贡献。

参 考 文 献

1 Stonebraker M. Inclusing of Abstract Data Type and Abstract Indexes in a Database System. IEEE Data Engineering, 1986. 262～269

2 Hellerstein J, Naughton J, Pfeffer A. Generalized search trees for database systems. In: proc. 21st Int'l conf. on very large databases. Zurich, Switzerland, Sept. 1995. 562～573

3 Kornacker M, Mohan C, et al. Concurrency & recovery in Generalized Search Trees. VLDB'97, 1997. 62～72

4 Kumar A, Tsotras V J, Faloutsos C. Designing Access Methods for Bitemporal Databases. IEEE Transactions on Knowledge and Data Engineering, 1998, 10(1): 1～20

5 Ju S-g, Chapa S, Chen W. Extensible motor of a object-relational DBMS: design and implementation. Technology of object-oriented languages and systems. IEEE computer society, 1999. 372～379

6 Anderson L, et al. Looking for the objects in object-relational DBMSs. ACM SIGPLAN Noties, 1997, 32 (10): 93～102

7 Stonebraker M. Object-relational DBMSs: the next great wave. Morgan Kaufmann Publishers, 1994

8 鞠时光. 对象关系型数据库管理系统的开发技术. 科学出版社, 2001, 3

2000 年中国科技期刊影响因子前 100 名 (只列计算机类)

名次 (73) 计算机科学 0.654　　名次 (76) 计算机集成制造系统-CIMS 0.649

(84) 计算机学报 0.631　　(100) 模式识别与人工智能 0.570

2000 年中国科技期刊总被引频次前 100 名 (只列计算机类)

名次 (74) 计算机学报 604　　名次 (88) 计算机科学 539

以上数据摘自“国家科学技术部发展计划司委托项目——2000 年度中国科技论文统计与分析”年度研究报告 (P. 126~127)