第35卷 第22期 计 算 机 工 程 2009年11月
Vol.35 No.22 Computer Engineering November 2009
·软件技术与数据库· 文章编号：1000—3428(2009)22—0091—03 文献标识码：A 中图分类号：TP311

# 基于树结构的数据库设计方案及应用

罗 舒，曹 旻
(上海大学计算机工程与科学学院，上海 200072)

**摘 要：**在需求不明确的情况下，用传统的数据库设计方法组织数据比较困难，数据层次性较差，且无法设计出一个通用的库，造成设计周期长、代码不易重用等问题，针对该问题提出一种新的基于树的数据库设计方法，该方法简单、直观、易于数据的组织，提高数据库设计的灵活性和通用性，在水库移民补偿金信息管理系统中得到较好的应用。

**关键词：**树结构；基于树的数据库；水库移民补偿金计算

# Tree Structure-based Database Design Method and Its Application

LUO Shu, CAO Min
(School of Computer Engineering and Science, Shanghai University, Shanghai 200072)

**【Abstract】** It is difficult to organize data by using traditional database design method when the requirement is uncertain. Traditional database design method has insufficient support for describing the level relation between or among data. The issues associated with designing and implementing a flexible and efficient information management system are not adequately addressed. This paper presents a novel database design approach which is based on tree structure. The intuitional description for data level makes it simple and pellucid. The flexibility and versatility of the approach provide not only reuse of database but also reuse of software architecture. The well use in the information management system for reservoir immigration compensation shows that the design method is feasible.

**【Key words】** tree structure; database based on tree; reservoir immigration compensation calculation

随着计算机技术的飞速发展，数据库技术也得到了迅速的发展，关系型数据库以其理论完整、结构简单、操作容易、存取有效等优点广泛应用于教育、金融、保险等行业[1]。但是，用传统的数据库设计方法来组织数据，在需求不是很明确的情况下，无法设计出一个通用的库，因而造成设计周期长、代码不易重用等问题，同时，传统的数据库设计方法所组织的数据的结构层次性较差，使得对实际问题的分析处理较复杂。因此，寻求有效和实用的数据库设计方法，提高数据库设计的灵活性和通用性显得非常必要。本文提出了一种基于树的数据库设计方法，并将其应用到水库移民补偿金信息管理系统中。

## 1 基于树的数据库设计方案

为了控制严重而频繁发生的水、旱等自然灾害，许多地区都建造了水库，水库工程同时也造成了大量的水库移民。在移民搬迁工作中如何充分保障移民的各项权益，杜绝挪用移民搬迁补偿金现象的发生，提高移民搬迁补偿金拨付效率，是政府移民工作的宗旨。为此，本文建立了水库移民补偿金信息管理系统。

### 1.1 系统主要功能及设计中的问题

本文开发的水库移民补偿金信息管理系统对某水库加坝扩建工程中涉及的移民和工商企业、专项迁建等工程的搬迁和补偿信息进行管理，使得广大公众和企业能查询移民及非移民的各项补偿政策和标准、所获得的补偿金及其各项实物指标、银行补偿金的动态发放情况；移民部门能对移民补偿信息进行整理、入库、维护；各级领导能对移民资金的使用情况、移民补偿金的发放情况进行汇总和动态监管。

根据本系统的需求，将其划分为5大模块：公众信息服务子系统，业务管理子系统，汇总分析子系统，信息发布子系统和系统管理子系统。其中，业务管理子系统是很重要的一个模块，它除了提供对移民基本信息、项目实施计划等数据的录入、添加、删除、修改和查询以外，还需要计算出每户移民应得的补偿金。

补偿金分为2个方面：实物补偿和移民安置。移民安置所需支付的补偿金根据移民户的贫困程度和被安置的地点来发放，不需要进行复杂的计算。实物补偿的补偿金需要根据每家每户的不同情况进行一系列复杂的计算才能获得。它分为3项：房屋，土地和果木，即根据移民户原有的房屋面积、土地数量和果木的数量、品种等进行折价。移民户实得的实物补偿金需要先计算出这 3 项的值，再进行指定计算才能得出。

然而房屋、土地和果木也都需要根据它们下面的子项计算得出，这些子项怎么分，决定了数据库中的每一张表如何设计，即每一种实物下面的子项就是数据表中的具体字段。以移民的果木为例，每户移民一般拥有不同种类的果木，每种果木又可分为大树、中树、小树、幼苗，不同的品种、不同的大小，其补偿标准不一样。如表1所示，若有4种果树，每种可分为大、中、小、幼4类，则按传统的数据库设计方法所设计的果木表中就会有16个字段，即大龙眼、中龙眼、小龙眼、龙眼幼苗等。

**基金项目：**上海市重点学科建设基金资助项目(J50103)
**作者简介：**罗 舒(1986－)，女，硕士研究生，主研方向：软件工程，分布式计算，面向对象技术；曹 旻，副教授、博士
**收稿日期：**2009-06-18 **E-mail：**shuluo@shu.edu.cn

**表 1　果木情况调查表(部分)**

| 果木 | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 龙 眼 | | | | 荔 枝 | | | | 枇 杷 | | | | 香 蕉 | | | |
| 大 | 中 | 小 | 幼 | 大 | 中 | 小 | 幼 | 大 | 中 | 小 | 幼 | 大 | 中 | 小 | 幼 |

但在开发的过程中，遇到了一个致命的问题，即设计数据库时，需求还不是非常明确。例如，在表 1 中，是否每种果木一定分为大、中、小、幼尚不确定，可能荔枝有大、中、小、幼之分，而香蕉只有大、小之分。子项分类不明确使得无法编写统一的表操作代码。当然这种情况有 2 个简单的解决办法：(1)表项最小化，将果木的大小分至最细，即尽管香蕉只有大、小之分，但在数据表中依然保留中香蕉、香蕉幼苗 2 个字段，从而确定了表的结构；(2)存储过程，即通过存储过程来实现对表的操作，表结构发生变化时只需要修改存储过程，而不用修改代码。可即便用简单的方法解决了上述的问题，新的问题又来了：补偿项不确定。因为不同的地区具有的果木种类不一样，有的地区可能只有荔枝、香蕉，有的地区却只有龙眼、枇杷。项的不确定性相应带来的是数据库设计中表结构的不确定性，因此，在表结构无法确定的前提下，之后的代码工作将无法继续。这个问题很难用“表项最小化”或“存储过程”的方法来解决。例如，有的地区实物补偿项包括房屋、土地和果木，而另外一些地区的实物补偿可能包括房屋、土地和蔬菜等，不同的地区，项的结构有可能发生较大的变化。类似的问题还有补偿标准不明确，如房屋补偿时是否考虑装修级别、房屋折旧率如何确定等。补偿计算方法的不确定性更无法用“表项最小化”或“存储过程”的方法来解决。

如果采用一般的设计方法，由于项无法确定，就无法设计一个通用的库，因此也不能编写通用的库操作函数，只能针对一个地区设计一种类型的库，为其编写操作代码。这样做，无论是库设计、软件结构设计、数据结构设计，还是类的域和方法设计、实现等，都不具备可重用性，进而造成开发代价大、周期长等问题。为了解决这一矛盾，提出一种基于树形结构的数据库设计方案。树形结构能充分地体现各补偿项之间的层次关系，如图 1 所示，树层次充分体现了实物补偿层次。将树结构存储在数据库的数据表中，则增加或删除某个补偿项时，只需要增加或删除树节点，也就是在数据表中增加或删除一条记录即可，而无须改变基本表的结构。

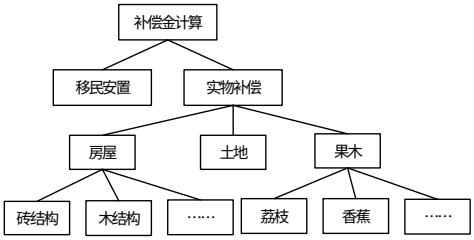


**图1　实物补偿层次图和树结构**

### 1.2　基于树的数据库设计思想

树是一种非常重要的非线性数据结构，一般适合于描述具有层次结构的数据。树结构的定义如下[2]：

**定义 1** 树(tree)$T$ 是一个包含 $n(n\geq 0)$个数据元素(在树中每个数据元素用一个节点表示)的有限集合。并且有：

(1)当 $n=0$ 时，$T$ 称为空树。

(2)如果 $n>0$，则树有且仅有一个特定的被称为根的节点，树的根节点只有后继，没有前驱。

(3)当 $n>1$ 时，除根节点以外的其余节点分为 $m(m>0)$个互不相交的非空有限集 $T1, T2, \cdots, Tm$，其中，每一个集合本身又是一棵非空树，并且称它们为根节点的子树。

**定义 2** 树的术语

(1)节点：在树中每一个数据元素及指向其子树根的分支称为一个节点。

(2)节点的度：一个节点的子树数目为该节点的度。

(3)终端节点：在树中，度为 0 的节点为终端节点或叶子。

(4)非终端节点：在树中，度不为 0 的节点为非终端节点或分支节点。

(5)孩子和双亲：在树中，节点 $p$ 的子树的根称为节点 $p$ 的孩子，反之，这个节点 $p$ 称为其孩子的双亲(父亲)。

(6)祖先：在树中，从根节点到节点 $p$ 所经的分支上的所有节点称为节点 $p$ 的祖先。

(7)子孙：在树中，以某节点 $p$ 为根的子树中的所有节点都称为节点 $p$ 的子孙。

例如，在图 1 中，补偿金计算是树的根节点，房屋、果木是分支节点，它们的父节点都是实物补偿，荔枝、香蕉等是叶节点，实物补偿的孩子有 3 个：房屋，土地和果木。

由树定义和图 1 可以看出，树及其相关术语与实物补偿的层次和结构非常相似，因此，它适合于描述系统的需求。例如，实物补偿的孩子有 3 个：房屋，土地和果木，而移民户实得的实物补偿金就是这 3 项的和。但是，在计算过程中，房屋、土地的计量单位、计算公式完全不一样，而树定义中则要求节点是同一种类型。为了满足实际的系统需求，对树的定义进行了扩展，建立了基于树的模型。

如图 2 所示，规定所有的节点都有一个值和一个计算表达式，值表示该节点的补偿金，计算表达式则说明了值的计算方法。所有的叶子节点都有一个或多个属性，这些属性可能是单价、数量等，每一个叶子节点的属性都可以不同，但属性的值都是已知的，存放在数据库中，是进行补偿金计算的原始数据。例如，假设图 2 中叶子节点 $D$ 表示砖结构房屋，其补偿金与它的面积、单价、装修级别、折旧年限有关，需要用这些参数经过计算所得，所以 $D$ 节点有以上 4 个属性以及关于这 4 个属性的一个计算表达式。$G$ 节点表示果木中的荔枝幼苗，补偿金计算取决于幼苗的单价和数量，则 $G$ 节点只有 2 个属性：单价和数量。所以，在扩展的树模型中，$D$ 的度数为 4，而 $G$ 的度数为 2，每个属性的类型不同。

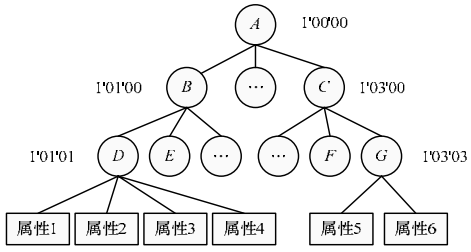


**图2　扩展的树模型**

## 2　基于树的数据库的实现

选择树来表示每一种补偿对象，树的节点表示项。为了便于动态修改树，需要将树结构存储在数据库中，这样，增加或删除某个补偿项时，只需要增加或删除树节点，也就是在数据表中增加或删除一条记录即可，无需改变表的结构。为了在数据库中存储树结构，需要保存节点在树形结构中的位置以及各节点之间关系。有 2 种方法：父节点法和编码法。父节点法是在节点表中以外键的方式存储每个节点的父节

点。这种方法的优点是设计简单，能满足用户的所有需求，缺点是数据量很大时，查询效率很低。本系统采用的是编码法，即按树的层次编码，每层 2 位编码，则每个节点最多可有 99 个孩子，如图 2 所示，这个最大孩子数目对于一般的补偿项足够了。存储时将每个节点的编码与节点信息同时保存在节点表中，所设计的树节点表如表 2 所示。

**表 2　节点信息表**

| 节点标识 | 编码 | 名称 | 计算公式编号 |
|---|---|---|---|
| 1 | 1’00’00’00 | 实物补偿 | 1 |
| 2 | 1’01’00’00 | 房屋 | 1 |
| 3 | 1’02’00’00 | 果木 | 1 |
| 4 | 1’01’01’00 | 砖结构 | 2 |
| 11 | 1’02’01’01 | 荔枝幼苗 | 3 |
| … | … | … | … |

表 2 所示的节点信息表中存储了树中所有节点的信息，计算补偿金时，首先在数据库中查询该表，根据查询的结果，按照编码确定各节点在树中的位置，然后把它们拼装成一棵树。拼装树的源代码是统一的，与树的深度、树中节点数、每个节点的孩子数等无关，因此，无论图 1 或图 2 所示的补偿项是否发生变化，拼装树的源代码都不需要修改，它是可重用的。

图 2 所示的树模型中的属性叶节点可以进行同样的处理，但是，为了提高数据表的查询效率，本文设计了表 3 所示的属性表和表 4 所示的节点属性表，用于存放每个节点所具有的属性。添加和删除节点的属性时，只需在表 4 中添加或删除一条记录，不需要改变表 4 的结构。例如，新的补偿标准中对于砖结构房屋不需要考虑房屋年限和折旧率，只需要删除表 4 中的第 4 行数据，即节点属性标识为 4 的那一行数据，表 4 的结构不发生变化。数据表的字段不变化，所有关于表操作的源代码就不需要进行任何修改。

**表 3　属性表**

| 属性标识 | 属性名称 |
|---|---|
| 1 | 单价 |
| 2 | 面积 |
| 3 | 装修级别 |
| 4 | 折旧年限/率 |
| 5 | 数量 |
| … | … |
| … | … |

**表 4　节点属性表**

| 节点属性标识 | 节点标识 | 属性标识 |
|---|---|---|
| 1 | 4 | 1 |
| 2 | 4 | 2 |
| 3 | 4 | 3 |
| 4 | 4 | 4 |
| 5 | 11 | 5 |
| 6 | 11 | … |
| … | … | … |

移民户的实物补偿相关的基本信息保存在表 5 所示的节点属性值表中，它只有 3 个字段，包含了移民户的所有“实物”。例如，从表 5 中可以得到如下信息：移民户 A 有 90 $m^2$ 的砖瓦结构的房屋，该房屋的装修级别为 1 级(全装修)，折旧年限为 5 年(折旧率为 90%)，单价为 1 000 元/$m^2$，还有 100 株荔枝幼苗等。

**表 5　节点属性值表**

| 节点属性值标识 | 移民标识 | 节点属性标识 | 属性值 |
|---|---|---|---|
| 1 | A | 1 | 1 000 |
| 2 | A | 2 | 90 |
| 3 | A | 3 | 1 |
| 4 | A | 4 | 5 |
| 5 | A | 5 | 100 |
| … | … | … | … |

按照这种设计方案，需求树的添加、删除等操作只与数据表中的数据有关，与表结构无关。另外，添加和删除补偿项时也不影响计算部分的源代码。例如，如果表 1 中的 1 号计算公式为 $\sum_{i=1}^{n}\text{child}_i$，表示所有孩子节点的值相加，则代码中只要设计出获得孩子节点的函数 getChildren()，再把所求的孩子节点的值全部相加即可。节点的孩子都存储在表 2 中，孩子的数目不会影响计算公式的解析和计算过程，所以添加和删除补偿项时计算部分的源代码是不变的。还有，如果提供了计算公式的解析函数，则计算方法的变化只需要修改数据库中的值，而不需要修改源代码。

## 3　方案论证

如果采用传统的设计方法，即表 1 所示的顺序存储方式，在数据表中每户移民只有一条数据，而采用新的树形结构设计方法，每户移民会有多条数据。根据需求，移民大约有 4 000 户，每户移民都具有房屋、土地和果木。每种实物中又至少包含 2 层分级，按照树的层次和节点数估算，表 5 的数据量大约有 80 万，可谓海量。如果将这 80 万条数据都存储在一张表中，可能会影响到查询效率和程序运行的速率。如何存储数据，是将所有数据全部存储到一张表中，还是按村、镇分不同的表来存储，针对上述树型数据库存储结构的设计，为了选择一个合适的存储方式，做了如下的实验测试。

(1)测试软硬件条件

软件：操作系统为 Windows 2003 Server，数据库为 SQL Server2000。

硬件：工作站为 Dell L170，CPU 为 1.83 GHz，内存为 1 GB。

(2)测试方案设计思想

在数据量很大时，建立适当的索引可以极大地提高查询效率，因此通过建立索引的方式进行实验。

(3)数据库设计

为了验证本文的设计方案，使用表 2~表 5 进行实验，并在表 5 的“移民标识”字段上建立聚类索引。

(4)测试步骤和结果

1)在表 2 中插入 1 万条数据：用时约 15 s。

2)在表 3 中插入 1 万条数据：用时约 14 s。

3)在表 4 中插入 1 万条数据：用时约 17 s。

4)在表 5 中插入 100 万条数据：用时约 30 min 42 s。

5)从表 5 中根据移民标识查询 2 万条数据，用时约 1 s。

6)从表 2~表 5 中联合查询 2 万条数据，用时约 3 s。

7)在表 5 中重新插入 800 万条数据，用时约 2 h 43 min1 s。

8)从表 5 中根据移民标识 2 万条数据，用时约 1 s。

(5)实验结论

从以上实验可知，尽管系统的数据量很大，但使用聚类索引极大提高了查询效率，即使是多表联合查询，速度也很快。就算数据量增加到 800 万，也不会影响查询速度，因此，可以确定核心设计方案为将所有数据全部存储到一张表中。

## 4　结束语

水库移民补偿金信息管理系统主要用于计算需要支付给移民的补偿金，因为它的数据具有很强的层次关系以及在数据库设计阶段需求的不明确性，采用树形结构设计方案，这一方案使得该系统具有很强的通用性和可重用性，不仅限用于一个水库，还可用于各个地区的各个水库。同时，它也适用于动态修改补偿项、补偿标准等情况。

**参考文献**

[1] 李昭原, 罗晓沛. 数据库技术新进展[M]. 北京: 清华大学出版社, 1997.

[2] 缪淮扣, 顾训穰, 沈　俊. 数据结构——C++实现[M]. 北京: 科学出版社, 2004.

编辑　索书志