深度优先算法在创建树形结构中的应用研究

唐青松
(四川文理学院 计算机学院,四川 达州 635000)

摘 要:为了让软件系统可以对树结构进行灵活管理,对相关学者提出的生成动态树结构的方案进行改进,给出了以数据表自关联的方式对节点信息进行存储,提出了在存储状态下的父节点、兄弟节点、叶子节点等节点类型的定义。使用深度优先非递归算法抽取节点信息,并按照树结构方式对节点进行排序,依据排序结果以及节点类型生成树结构,实现了一种具有很好可移植性、可扩充性和可维护性的无限级动态树。最后,将动态树植入学校管理系统,通过实验证明,植入该树结构之后系统具有界面结构性强、信息层次清晰、用户操作简单等优点。

关键词:深度优先算法;管理信息系统;关系数据库;树形结构
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2014)09-0226-04
doi:10.3969/j.issn.1673-629X.2014.09.053

Application and Research on Tree Structure Based on Depth-first Algorithm

TANG Qing-song
(College of Computer, Sichuan University of Arts and Science, Dazhou 635000, China)

Abstract: In order to make the software system can be flexible management of tree structure, modify the scheme to generate dynamic tree structure proposed by relevant scholars, give the data table autocorrelation method to store node information, put forward the definition in the storage state of the parent nodes, sibling nodes, leaf nodes. The use of depth first algorithm to extract the node information, and nodes are sorted according to the tree structure, based on the ranking results and spanning tree structure and node type, achieve a infinite order dynamic tree with good portability, extensibility and maintainability. Finally, put the dynamic tree into the school management system, the experiment results show after implantation of the tree structure, the system has the advantages of strong structural information interface, clear information layer and simple operation for user.

Key words: depth-first algorithm; MIS; relational database; tree structure

0 引 言

树是由一个集合以及在该集合上定义的一种关系构成的一类非线性数据结构,其中,集合中的元素称为树的节点,所定义的关系称为“父子”关系,“父子”关系在树的节点之间建立了一个层次结构,这种层次结构具有界面结构性强、层次清晰、使用方便等特点。树结构在人类社会中应用广泛,比如国家的科层式管理、企业的营销管理等;而在计算机软件系统开发中,其地位更加显著,比如操作系统中的文件系统,数据库系统中信息类型的组织形式等。因此,对树形结构的构建和遍历一直是信息系统开发领域的关注热点。

当前,树形结构主要有静态树和动态树两种形式。静态树中的各节点固定并且关系明确,在算法分析设计中,对该结构各节点遍历容易,但对树节点的更新和层次的修改都有很大的局限性;动态树中的各节点可以随时添加或删除,交互性强,易于维护。动态树的优势明显,很多学者都对动态树进行了研究,并给出了解决方法。如文献[1]实现了树型控件,但对单表数据表的设计还可以改进,而采取 4 张数据表就会存在树的深度很难延伸等问题;文献[2]在. net 环境下使用 TreeView 控件实现树状图,但是如果没有良好的底层数据支撑,就会出现文中提及的当树节点过多时,服务器将会占用大量资源而影响整个系统运行的问题。也有开发人员使用 Ajax 实现树状图,但存在不能很好遍

收稿日期:2013-11-07 **修回日期**:2014-02-11 **网络出版时间**:2014-07-17
基金项目:国家自然科学基金资助项目(61152003);四川省教育科研基金项目(13ZB0103)
作者简介:唐青松(1980-),男,硕士,讲师,研究方向为网络数据库、软件技术与理论。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20140717.1231.037.html

历子树的问题,当删除一个子树根节点时,其所有子节点将不能完全删除,这些节点将一直驻留在数据表中[3]。文中将对底层数据表进行改进,用“深度优先遍历非递归算法”来访问树节点,以克服现有动态树研究和应用中的不足,使动态树形结构更具有可扩充性、可移植性和可维护性。

1　树结构中各节点的存取

1.1　树节点的存储

树形结构的动态更新是来源于节点信息的变化,只要树形结构能够按节点信息的变化而变化,就产生了动态效果,不限使用什么数据库,只要在数据库表中存入相关节点信息并提供改变这些数据的方法。文献[4]给出了两个关系数据表,其中主表保存树结构的父节点数据,子表用于保存树结构的孩子节点[4]。根据文献[4]的论述,可以分析得知,其关系结构主要存在以下问题:

(1)不易于对树结构进行维护,该关系结构主要表现在,树节点的顺序难以调整。如:假设在数据表中保存 A、B、C 三个父节点信息,从数据表中提取节点信息时,只能按照序号,节点名称方式进行排序取出各节点信息,不能达到节点顺序任意调整的效果,从而导致在生成树结构过程中加载节点信息的时候不能按照预想的结果展示。

(2)树节点的“父子”关系不明确,从两张表的关系结构来看,二表是根据节点名称进行关联的,由此得知,当节点名称有重复值的情况出现时,其子节点在关联查询的时候将出现二义性。

为达到实现树形结构能实现任意改变形态的效果,要求节点信息的数据来源必须准确合理[5]。因此,对树节点信息存储结构的设计需要改进。结合理论分析,改进了数据表的关系,见表 1。

表 1　改进后的树节点存储方式

字段名	数据类型	长度	主键	说明
id	int	4	是	自动递增
node_name	varchar	50	否	节点名称
node_order	int	4	否	排序值
node_pid	int	4	否	父节点号

数据表设计为单表,使用当前节点的父节点编号与上级节点建立层次关系,由此可以实现无限级层次的树结构。数据表中的编号是树节点的唯一标识,其值从 1 开始依次递增;节点名称为字符型类型,以保存节点信息;排序值是用于指定在树中相同深度时节点的排列顺序;父节点编号用于指定树结构中节点的层次关系,与主键关联,规定父节点编号不能与自身节点的编号相同。根据以上存储形式,结合树型数据结构,给出以下定义:

定义 1:在存储的所有节点中,如果父节点编号为 0,称该节点为树根节点。存储的节点中若有多个父编号为 0 的节点,则说明存储了多棵树。

定义 2:在存储的节点中,父节点编号值相同的节点为兄弟节点。

定义 3:指定数据表中的某一节点,然后搜索父节点编号与该节点编号相同的其他所有节点,如果搜索结果为空,则该节点为叶子节点,如果搜索到的结果不为空,则这个(些)节点为该节点的孩子节点。

1.2　节点的提取与维护

为方便与数据库建立联系,使用面向对象程序设计语言设计节点类(设名为 Node),该类与以上关系建立映射(ORM)[6],即节点类的属性与字段一一对应,建立映射后,数据表中存储的一条记录代表树结构一个节点对象。

建立查询事务,主要方法如下:

(1)提取节点 getNode(int id),根据传递的编号值(id),在数据表中查询与该编号相同的节点,如果查询的结果为空,则该节点不存在,返回空对象(null);否则,使用查询结果的唯一记录的各字段值初始化节点对象并返回该对象。

(2)根据定义 1 判断该节点是否是根节点 isRoot(Node node),如果是根节点返回结果“真”,否则返回“假”。

(3)根据定义 3 判断该节点是否是叶子节点 isLeef(Node node),如果是叶子节点,返回结果“真”,否则返回“假”。

(4)根据定义 3 提取孩子节点 getChildren(Node node),如果该节点不是叶子节点,按排序值递增(递减)的方式提取该节点的所有孩子节点,并存储到一个堆栈中。

(5)存储节点 saveNode(Node node),初始化节点对象的属性值(除编号),添加到数据表中。

(6)修改节点信息 modefyNode(Node node),在数据表中对当前节点进行修改。

(7)删除节点 deleteNode(Node node),先遍历该节点子树,将遍历出来的所有节点从数据表中删除。

(8)提取该节点的下一个访问节点 getNextNode(Node node),用于对树结构的遍历。

2　深度优先算法创建已存储的树

2.1　深度优先求解方法

树的遍历是按照某一规则,按照对树中的每个节点各做一次且仅做一次的访问。常见的算法主要有深

度优先算法和广度优先算法,通过以上算法对树节点的遍历,将把树的各节点按照一定的顺序排列,然后存储在一个堆栈中[7-9]。

深度优先遍历树的过程如图 1 所示。从根节点 *A* 开始,然后访问 *A* 节点下的第一个子节点 *B*,接下来访问 *B* 节点的子节点 *E*,再访问 *E* 节点的子节点 *H*,*H* 节点为叶子节点,则访问 *H* 节点后,访问 *H* 节点父节点的下一个兄弟节点,如果父节点没有下一个兄弟节点,则访问祖父节点的下一个兄弟节点,继续上述过程,直到访问的下一个节点与根节点相同,则所有节点访问完成[10-13]。根据以上方法,图 1 中各节点访问序列为:*A-B-E-H-F-C-D-G*。

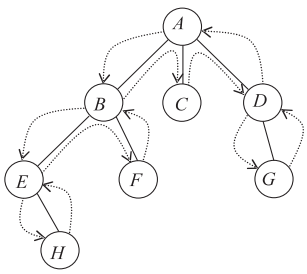

图 1 深度优先访问节点过程

2.2 深度优先算法访问树节点

树节点存储在数据表后,重要的是如何将节点取出,以还原成预先设定的树形结构。用户在存储树节点时,其节点存入的顺序是随意的,因此访问树节点的就是将这些存储在数据表中的节点进行合理的排序,使之与树形结构展开后的节点顺序一致[14]。通过对深度优先算法的分析,该方法能有效地解决(见图 2),其算法步骤如下:

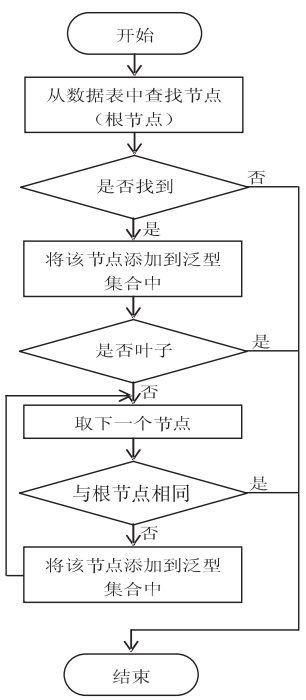

图 2 深度优先访问树节点流程图

Step1:定义泛型 List 类的对象 list;

Step2:取出根节点 getNode(int id);

Step3:如果根节点为空,则为空树并退出,否则将该节点对象添加到 list 中;

Step4:深度优先遍历该节点的下一个节点 getNextNode(Node node),判断与根节点是否相同;

Step5:若节点与根节点不相同,将该节点对象添加到 list 中,转向 Step2 继续;

Step6:若节点与根节点相同,则成功创建树结构并退出。

2.3 构建动态树的关键技术

使用树节点创建树结构,就是对树节点实现一种特殊的排序,文中根据图 1 算法流程使用非递归搜索方法查找树节点,并将访问到的节点存储在堆栈中,最后依次取出堆栈中的节点,即可实现对树节点的排序[15]。在搜索树节点过程中应注意以下问题:

(1)搜索下一个节点时,注意避免死循环的出现。由图 1 访问树节点的箭头流向可以看出,当访问到 *H* 节点时,下一个节点应该是节点 *F*,如果在程序中没有做出合理的判断,将会出现 *A-B-E-H-B-E-B* 的死循环;另外,用户不能以手动的方式直接在数据表中删除节点数据,这样会导致查找下一个节点时出现断点,从而导致死循环。

(2)生成树的时候如何控制各节点前的图标,使之具有很强的层次关系,显示效果与 Windows 系统下的资源管理器类似。有效的解决方案是在访问每一个节点时,首先判断该节点是否是所有兄弟节点的最后一个节点,如果是兄弟节点的最后一个,则使用节点结尾标记标签图标,否则使用连接图标的标签,然后判断该节点是否是叶子节点,如果是叶子节点,则使用叶子节点对应的标签,如果是一个父节点,则使用父节点对应的标签。另一种有效的方案是,将遍历出来的节点数据作为数据源,然后使用系统开发工具中的控件和一些开源框架实现,如 TreeView、JQuery 提供的 tree 控件等。

3 设计实现及测试结果

高校学生管理系统的开发技术已很成熟,很多高校都使用计算机软件实现对学生的管理,当前大多管理软件都将学生的一些学籍信息封装在学生数据表中,如学生所在系科、专业、年级、班级等。显而易见,这些信息可以根据学校的组织机构,抽象成一棵树(如图 3 所示),对学生信息数据表进行改进(见表 2),如果需要提取学生相关信息,只需要根据外键查询树节点,然后依次找出该节点的上级节点就可以得到信息。

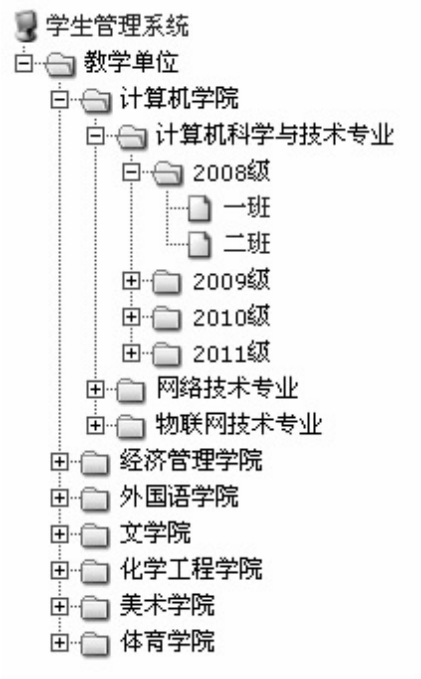

图 3　学校教学单位组织树结构

表 2　改进后的学生表结构

字段名	数据类型	长度	主键	说明
stu_no	varchar	10	是	学生学号
stu_name	varchar	10	否	学生姓名
node_id	int	4	否	树节点编号
…	…	…	…	其他信息

由表 2 的结构得知,使用这种结构的最大优点是减少数据冗余,增强数据的可维护性。将学生信息表结构修改后,对软件的更新操作,只需要稍加修改系统中关于维护和统计的 SQL 代码即可,易于实现。限于篇幅,不再展示修改后的程序代码。

通过改进后的实验表明,将组织抽象成树结构,以树的节点作为信息节点能有效地处理各种复杂的数据信息,给用户提供良好的层次结构,方便了系统操作人员对数据的统计、更新等操作。

4　结束语

文中针对当前一些学者研究动态树的基础上进行改进,通过实验结果表明,使用自关联数据表存储树节点模式,可以减少数据冗余,提高无限级动态树的创建和管理效率。使用非递归方法生成树状图,实现了在删除树的某一节点时,可以将该节点对应的子树所有节点一起删除,解决了无关的树节点驻留在数据表中的问题。同时,该方法可在其他动态 Web 技术中实现,以提高树结构的可扩充性和可维护性。

参考文献:

[1] 李俊飞,陈　皓,赵卫东. 树形结构数据输入输出控件的设计与实现[J]. 计算机工程与设计,2011,32(9):3054-3058.

[2] 魏　斌,马继辉,牛　虎. 基于递归算法的树型结构图的设计与实现[J]. 计算机应用与软件,2011,28(1):96-98.

[3] 龚建华. 深度优先搜索算法及其改进[J]. 现代电子技术,2007,30(22):90-92.

[4] 张华兴,孙　毅,单继宏. 网页树形结构自动生成研究[J]. 计算机系统应用,2009,18(7):62-66.

[5] 李　睿,曾俊瑀,周四望. 基于局部标签树匹配的改进网页聚类算法[J]. 计算机应用,2010,30(3):818-820.

[6] 耿　飙,宋余庆,梁成全,等. XML 文档到关系数据库映射方法的研究[J]. 计算机应用研究,2010,27(3):951-954.

[7] 徐华结,吴仲城. 基于 Web 服务实现的环境监测数据采集平台[J]. 计算机技术与发展,2013,23(6):237-240.

[8] Wang Xingbo. Study on non-recursive and stack-free algorithms for preorder traversal of complete binary trees[J]. Computer Engineering and Design,2011,32(9):3077-3081.

[9] Li Lihong, Xu Yuanfei. Study on parameters optimization of support vector machines based on DFS[J]. Computer Simulation,2011,28(7):216-219.

[10] Zhang Xin, Liao Pin, Guo Bo. Depth-first search algorithm for mining frequent closed itemsets[J]. Journal of Computer Applications,2010,30(3):806-809.

[11] Odkerk S. A proposal for an XML confidentiality label and related binding of metadata to data objects[R]. [s. l.]:[s. n.],2010.

[12] Blazic A J, Saljic S. Confidentiality labeling using structured data types[C]//Proc of 2010 fourth international conferences on digital society. [s. l.]:[s. n.],2010:182-187.

[13] Ghinita G, Karras P, Kalnis P, et al. A framework for efficient data anonymization under privacy and accuracy constraints[J]. ACM Transactions on Database System,2009,34(2):1-47.

[14] Kenig B, Tassa T. A practical approximation algorithm for optimal k-anonymity[J]. Data Mining and Knowledge Discovery,2012,25(1):134-168.

[15] Aho A V, Hopcroft J E, Ullman J D. The design and analysis of computer algorithms[M]. [s. l.]: Addison Wesley/Pearson,2005.

深度优先算法在创建树形结构中的应用研究

作者： 唐青松， TANG Qing-song
作者单位： 四川文理学院 计算机学院,四川 达州,635000
刊名： 计算机技术与发展 ISTIC
英文刊名： Computer Technology and Development
年，卷(期)： 2014(9)