基于双向链表的产品协同设计版本存储模型①

刘国军, 杨宏志

(延安大学附属医院 CT 诊断科, 延安 716000)

摘 要: 针对产品协同设计的版本管理的版本存储问题进行研究. 在分析了目前已有的版本增量存储和完整存储技术的基础上, 提出了一种将完整存储和逆增量存储相结合的产品协同设计版本存储模型. 将版本的存储结构按照其父子关系定义为一个双向版本链表, 当设计过程产生新的版本时, 采用反复迭代和插入的方式, 将其存放在版本链表结构中, 以实现协同设计版本的快速存储、节约存储空间和提高版本存储的安全性.

关键词: 协同设计; 版本管理; 完整存储; 逆增量存储; 链表

Version Storage Model of Product Collaborative Design Based on Doubly Linked List

LIU Guo-Jun, YANG Hong-Zhi

(CT diagnosis Branch, Affiliated Hospital of Yan'an University, Yanan 716000, China)

Abstract: The study is on version storages issues of the version management in the product collaborative design. Based on the analysis of the current version incremental storage and complete storage technologies, this study poses a new version storage model of product collaborative design which will combine inverse incremental storage with complete storage. The storage structure of version is defined as a two-way version of the list according to their parent-child relationship, when the design process produces a new version, it will be stored in the list structure of version by the way of repeated iteration and insertion, in order to achieve the fast storage, save the storage space and improve the security of the version storage in the product collaborative design process.

Key words: collaborative design; version management; complete storage; inverse incremental storage; linked list

协同设计是指利用计算机技术、多媒体技术和网络通信技术, 由两个及两个以上的设计群组人员, 针对某一复杂的产品设计目标, 通过相互协作、互相协同地完成设计任务的现代产品设计方法[1]. 在产品协同设计过程中, 不同的设计阶段会产生不同的设计版本, 如何保证产品协同设计版本的正确性、安全性和节省存储空间, 已经成为产品协同设计过程版本管理研究的核心问题之一.

版本存储是协同设计版本管理的重要组成部分. 一方面, 不论是横向协商过程还是纵向迭代过程, 所产生的版本都不是最优版本, 在最终设计方案产生之前, 需要对设计过程中的所有版本进行存储, 以便于比较和回顾; 另一方面, 在设计过程所产生的多个版本中, 蕴含着诸多经验和教训, 对中间版本进行有效的存储和归纳, 将有助于对设计知识库进行扩充, 实现智能化设计. 目前关于产品协同设计的版本存储的研究主要有: 潘翔等[2] 提出的版本存储模型采用统一管理、分散处理的策略, 将版本数据存放在索引区和信息数据区, 提高了检索效率. 王维丽等[3]从版本存储的角度讨论了版本管理的方法, 给出了一个混合版本存储模型, 利用该存储模型可以通过选择恢复点保证版本能够恢复, 同时通过选择的控制节点的版本信息存储, 提高了系统的可靠性和安全性以及系统的空间利用率. 李祥等[4]提出了一种混合存储模型, 将集中式存储和分布式存储相结合, 使是版本存储具有更好的柔性. 徐保民等[5]从空间和时间效率以及协同工

① 收稿时间:2013-03-22;收到修改稿时间:2013-04-27

作的需要考虑, 采取存差策略大大节省了存储空间. 王红丽等[6]提出了一个混合式版本管理存储模型, 主要是将正增量存储和逆增量存储相结合, 该模型提高了基板本和最新版本获取效率, 节约了存储空间, 但是基板本和最新版本一旦破坏, 中间版本将无法修复, 中间版本的获取依赖基板本和最新版本, 不便于取得中间版本的效率和安全性不高. 付喜梅等[7]基于 STEP 数据交换标准, 提出两个差值存储模型, 通过构造版本存储阈值函数, 将差值存储与完整存储相结合, 有效实现了空间效率和时间效率的平衡.

完整存储技术是将版本信息完整地存放在数据库中, 其版本存取速度快, 但要占用大量的存储空间. 大多数情况下版本之间存在着继承关系, 新版本的生成, 仅有少量的新内容增加, 所以完整存储会产生了大量的冗余信息. 增量存储技术是只存储基板本和后继版本之间的差异, 由基版本和后继版本的差异构成了新的版本, 这种存储策略节省了大量的存储空间, 可以减少冗余信息的产生. 但是, 随着版本链的不断增长, 版本在差异恢复上将花费大量的时间, 版本的存取速度将大大下降. 同时由于版本之间存在着差异修复的关系, 一旦基版本被破坏, 其它后继版本将无法恢复. 本文从版本存储控制的角度, 提出了一种将完整存储和逆增量存储相结合的版本存储模型, 该模型集中了完整存储和增量存储的优点, 改善了存储空间的利用率, 同时又提高版本存取速度和安全性.

1　协同设计版本存储模型分析

1.1 版本数据结构定义

为了能对版本进行统一的管理, 便于历史版本的存储、检索、回溯, 将对单个版本的数据结构如图 1 所示的形式进行定义.

版本号	父版本	子版本	生成者	生成日期	存储位置	存储类别

图 1　版本数据结构图

各字段功能描述如下:

版本号: 版本的唯一编号.

父版本: 指向该版本前驱版本的存储地址.

子版本: 指向该版本后继版本的存储地址.

生成者: 版本的生成者.

生成日期: 版本提交系统时的日期.

存储位置: 该版本数据的存储位置.

存储类别: 1)完全存储; 2)逆增量存储

其中: 版本号和生成日期能够唯一确定一个版本.

1.2 版本的增量存储模型

产品协同设计版本的增量存储模型[8]可以定义为: $V_i = V_0 + \Delta_i|_{条件}$, 其中 V_0 是基版本, $\Delta_i|_{条件}$ 是 V_i 版本相对于 V_0 版本通过比较得到的差异信息. 这种增量存储模型中所有后继版本都和基版本 V_0 进行比较, 求取它和 V_0 之间的差异. 其优点是实现该增量存储模型算法比较简单, 版本的存取速度较快. 在一定程度上减少了信息的冗余, 节约存储空间. 其缺点是随着版本链的增长, 后继版本离基板本越来越远. 后继版本和基版本的差异信息越来越多, 这样将在比较差异和存储差异上花费大量的时间. 所以当版本链增长到一定的长度, 该模型的存储效率大幅降低, 且基板本一旦破坏其它后继版本将无法修复, 安全性不高. 于是许多学者对版本的增量存储模型进行了改进, 提出了正增量存储模型和逆增量版本存储模型.

① 正增量存储模型[8]可以定义为: $V_{i+1} = V_i + \Delta_i|_{条件}$, $i \geq 0$, i=0 时 V_0 为基版本, $\Delta_i|_{条件}$ 是在 V_i 基础上通过比较得到的差异信息. 后继版本仅存储与前驱版本之间的差异. 优点: 极大的减少信息冗余, 提高了空间利用率. 缺点: 提取中间版本的完整信息时就需要从基版本开始逐个比较修复差异, 才能恢复得到该版本, 随着版本链的增长, 花费在版本之间差异修复的时间明显增多, 存取效率明显下降, 且基板本一旦破坏其它后继版本将无法修复, 安全性不高.

② 逆增量存储模型[9]可以定义为: $V_{i+1} = V_i + \Delta_i|_{条件}$, $i \geq 0$, i=0 时, V_0 为基版本, 即 V_i 存储 V_i 和 V_{i+1} 之间的差异 $\Delta_i|_{条件}$. 前驱版本只存储与后继版本的差异, 最新版本完整存储. 优点: 极大的减少信息冗余, 提高了空间利用率, 可以直接获取最新版本; 缺点: 当版本链增长到一定的长度, 提取中间版本的完整信息时就需要从最新版本开始逐个和前驱版本进行比较修复差异, 才能恢复得到该版本, 而且一旦最新版本被破坏, 所有前驱版本将无法恢复, 安全性不高.

2　完整存储和逆增量存储相结合存储策略

通过对以上版本模型的优缺点分析, 从节省存储空间, 提高检索效率和安全性, 这三个方面考虑, 提

出一种完整存储模型和逆增量存储模型相结合的协同设计版本存储模型. 该混合模型的存储策略是基于双链表结构实现, 具体存储策略如下:

① 将版本按照其父子关系形成双向版本链表 L, 父版本对应前驱版本, 子版本对应后继版本.

② 将基版本完整存储作为双向版本链表 L 的表头.

③ 取一个临界值为 C, H: 相对于基版本的长度. 利用双向链表的性质计算该版本的前驱的个数可以得到它相对于基版本的长度.

④ 当 H≥C 时, 该版本完整存储, 把该版本取为新的基版本, 反复迭代,根据版本间父子关系依次插入 L 中. 当 H<C 时, 前驱版本存储与后继版本的差异, 后继版本完整存储, 根据版本间父子关系依次插入 L 中.

按照以上策略我们可以看到, 在双向版本链表 L 中, 不但节省了存储空间, 可以直接获取基版本和最新版本, 而且还有中间版本的完整存储, 不怕基版本和最新版本损坏, 无法取得中间版本, 从而提高了安全性. 利用双向链表的性质, 版本之间存在着双向指针可以向上或向下进行检索, 有效实现版本的回溯. 如果要获取的中间版本是完整存储版本, 那就直接获取, 如果要获取的中间版本是逆增量存储版本, 利用后继版本指针找到离它最近的一个完整存储的后继版本, 利用前驱版本指针查找其前驱版本, 再以该后继版本为基板本和其前驱版本进行逐级差异修复. 即可获得该版本, 提高检索效率.

3 完整存储和逆增量存储相结合的存储策略实现算法

在版本操作技术中, 只有版本插入和版本删除, 会影响版本链的长度, 下面从版本插入来描述完整存储和逆增量存储相结合的实现算法. 插入新版本有两种情况. 一种是要插入的版本无前驱版本, 则将其设置为基板本, 并且完全存储该版本. 另一种是要插入的版本有前驱版本, 根据前驱版本是完整存储还是逆增量存储, 再计算要插入的版本的存储方式. 算法实现流程:

① 设置插入版本信息以及临界值.

② 判断插入版本是否有前驱版本.

③ 如果没有前驱版本, 设该插入版本为基板本, 完整存储, 设置历史版本关联, 插入版本结束. 否则, 计算前驱版本相对于基板本的长度.

④ 如果前驱版本相对于基板本的长度大于给定临界值, 前驱版本完整存储. 否则, 前驱版本逆增量存储. 完全存储插入版本, 设置历史版本关联, 插入版本结束.

4 版本存储模型的实例分析

下面用一个简单的实例来说明产品协同设计过程中, 采用完整存储和逆增量存储相结合的存储策略的有效性. 图 2 是根据逆增量版本存储模型的定义, 对 V_0~V_5 进行存储得到的. 其中 ΔV_0~ΔV_4 为逆增量存储, V_5 为完整存储.

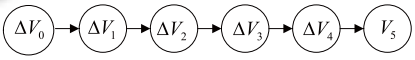

图 2 逆增量版本存储模型图

根据本文提出的完整存储和逆增量存储策略, 对 V_0~V_5 存储进行改进. 设 V_0 为基板本, 完整存储为 V_0, 设置临界值为 2, V_0 相对长度为 0, 小于临界值, 所以对其增量存储为 ΔV_0, V_1 相对 V_0 的长度为 1, 所以对其增量存储为 ΔV_1. V_2 相对 V_0 的长度为 2, 所以对 V_2 完整存储, 并把 V_2 选为新的基板本. V_3 相对于 V_2 的长度为 1, 所以把 V_3 增量存储为 ΔV_3, V_4 相对于 V_2 的长度为 2, 所以对 V_4 完全存储, 并把 V_4 选为新的基板本. 因为 V_5 为最新版本, 根据逆增量存储的原理, V_5 完全存储. 如果一个版本的分支数大于 1 时, 每个分支也按照完整存储和逆增量存储策略进行存储. 这样我们得到了图 3 所示的双向版本链模型.

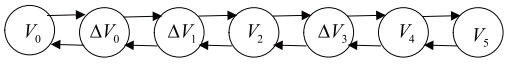

图 3 双向版本链存储模型图

双向版本链存储模型, 在一定程度上节省了大量的存储空间. 完整存储了、、和, 不怕破坏而无法修复其它中间版本, 而在逆增量版本存储模型中一旦破坏其它版本将无法修复. 因此, 双向版本链存储模型提高了版本的安全性. 假设现在要获取, 在逆增量版本存储模型中, 就得从开始经过四次(~)差异修复才能得到. 而在双向版本链存储模型中, 则只需在的基础上直接进行差异修复得到. 因此, 双向版本链存储模型提高了检索效率.

(下转第 154 页)

4 结语

本文通过在 BibSonomy 和 del.icio.us 两个数据集上进行实验, 证明了社区发现算法可以提高推荐结果的准确性, 从而证实社区发现算法可以在一定程度上将语义相近的标签聚合到一起. 社区生成算法有广泛应用, 下一阶段的研究会将本文中没有涉及到的社区发现算法应用到标签聚类的生成中, 并和之前的结果比较, 以进一步提高个性化推荐的效果.

参考文献

1 Andriy S, Jonathan G, Bamshad M, Burke R. Personalized recommendation in social tagging systems using hierarchical clustering. Proc. of the 2008 ACM Conference on Recommender Systems. Lausanne, Switzerland, ACM, 2008: 37–48.

2 Newman MJ, Girvan M. Finding and evaluating community structure in networks. Physical Review, E 74, 036104, 2006.

3 Kleczkowsi A, Grenfell BT. Mean-field-type equations for spread of epidemics: The‘small world’model. Physica A 274, 1999: 355–360.

4 Wagner A, Fell D. The small world inside large metabolic networks. Proc. R. Soc. London B 268, 2001: 1803–1810.

5 Camacho J, Guimera R, Amaral LAN. Robust patterns in food web structure, Phys.Rev. Lett. 88, 228102, 2002.

6 Girvan M, Newman MEJ. Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA, 2002, 99 (12): 7821–7826.

7 Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of community hierarchies in large networks. J. Stat. Mech. 2008, 10: 10008.

8 万里,廖建新,王纯.基于社会网络信息流模型的协同过滤算法.吉林大学学报(工学版),2011,41(1):275–28.

(上接第 131 页)

5 结束语

本文将完全存储模型和逆增量模型相结合, 提出基于双向链表的产品协同设计版本存储模型, 通过设置临界值和版本相对基板本的长度来进行比较, 确定完整存储还是逆增量存储, 节约了存储空间, 提高了存取效率和安全性.

参考文献

1 于海斌,朱云龙.协同制造.北京:清华大学出版社, 2004:142.

2 Dix A, Rodden T, Sommerville I. Modeling versions in collaborative work. Software Engineering IEEE Proceedings, 1997, 144(4): 194–206.

3 曹祥,郑国勤,胡毓宁.协同设计环境下的版本管理模型.计算机工程与应用,2001,37(15):61–63.

4 李祥,周雄辉,阮雪榆.注塑模协同设计过程中的版本管理研究.模具技术,2000,(5):18.

5 徐保民,徐爱琴,李峰.协同编辑器中版本管理的设计与实现.计算机工程与应用,2002,38(5):134–136.

6 王红丽,孙长嵩,李钟隽.一个混合式版本管理存储模型.2006 年北京地区高校研究生学术交流会——通信与信息技术会议论文集.北京.北京邮电大学出版社.2006,1644–1647.

7 付喜梅.基于 STEP 的协同设计版本存储控制策略.计算机工程,2008,34(24):61–66.

8 付喜梅,陈家新.协同设计中版本存储控制策略的研究.微计算机信息(管控一体化),2006,22(4-3):105–108.

9 Westfechtel B, Munch BP, Conradi R. A layered architecture for version management. IEEE Trans. on Software Engineering, 2001, 27(12).