第33卷 第18期 计 算 机 工 程 2007年9月
Vol.33 No.18 Computer Engineering September 2007
·软件技术与数据库· 文章编号：1000—3428(2007)18—0106—03 文献标识码：A 中图分类号：TP391.72

协同 CAD 系统中的并发控制机制

赵 蓉，史维峰，郑 超
（西北大学计算机科学与技术系，西安 710069）

摘 要：并发操作是协同 CAD 系统中至关重要的问题。该文提出了并发操作控制机制的设计原则，阐述了基于语义的并发控制机制利用嵌套事务来解决传统图档事务并发控制中的数据一致性低、中间版本不易维护等问题，结合实际给出了博士 CAD 系统中并发控制的系统模型，并作了相应的说明。

关键词：并发控制；事务处理；CAD；语义

Concurrency Control Mechanism in Cooperative CAD System

ZHAO Rong, SHI Wei-feng, ZHENG Chao
（Dept. of Computer Science and Technology, Northwest University, Xi'an 710069）

【Abstract】Concurrent manipulation control scheme is crucial for CAD system. This paper proposes several design considerations for concurrency manipulation control scheme, and introduces the nested transaction of a semantic-based concurrency to solve the problem of lower consistency and difficulty in managing middle version and so on. The model for concurrent DorCAD system is described.

【Key words】concurrency control; transaction treatment; CAD; semantics

随着计算机网络技术的发展和市场竞争的加剧，制造业对CAD系统提出了更高的要求：对用户的协同能力，大数据量复杂对象的处理能力，维护数据一致性和完整性的能力等。计算机支持的协同工作(computer supported cooperative works, CSCW)适应了信息社会中人们工作方式的群体性、交互性、分布性、协作性特点，因此，利用CSCW技术将单机版的CAD系统开发成支持协同工作的CAD系统是一个显著的发展趋势[1]。

实现协同 CAD 系统的关键技术之一是对共享数据的并发操作，即协调多个独立的用户在同一时间对同一个共享对象进行访问而不会导致不一致的状态，这是并发控制的目的，也是目前研究的重点。

一个简明、高效的并发控制策略对CAD系统至关重要，直接影响用户对协同CAD系统的使用效果。传统的并发控制方法[2~5]包括双阶段锁(two phase locking, 2PL)、时间戳机制(timestamp ordering, TSO)、UNDO/REDO法、操作转换、多版本机制、多粒度机制及刘新福等提出的在CSCW 环境下利用向前变换重构对象的操作历史记录的串行化并发控制。以上方法从不同角度提出了并发控制策略，但都没有很好地捕捉工程设计中丰富的语义信息，导致了数据的一致性差且并发率低。

本文提出了一种基于事务语义的并发控制策略并将其应用于博士 CAD 系统中。阐述了事务分类、语义锁和冲突协调等关键问题，其目的在于不仅使用户能自由地操控共享数据，还保证了共享数据的一致性，并在此前提下完成用户的操作，响应时间短、并发程度大。

1 设计原则

与传统的分布式数据库不同，协同CAD系统是直接面向用户的分布式系统[5]，它既强调用户操作的独立性，又看重用户之间的协调与合作。因此，协同CAD系统中的并发操作控制机制需要考虑以下要素：

(1)一致性

共享实体的状态在任一时刻对任意用户来说都是相同的，它是整个机制的先决条件，优先于其他所有的设计要素，包括操作对象 ID 一致性、结果一致性、因果一致性和用户意图一致性。

(2)交互性(人人交互)

它是并发性和响应性二者的有机结合。在通常情况下，不论采用何种并发控制机制，根据角色、权限和其他因素的不同，每个用户都可能对共享实体执行一些操作，这些操作既可以是同步的也可以是异步的，应使用户在其可以承受的时间延迟下得到响应，并且对共享实体进行有效操作后尽快通知其他用户端。

(3)协作模式的多样性

协作者之间是选择同步和异步交叉进行还是纯同步或是纯异步的模式。不同的协作模式对并发控制要求的侧重点有所不同，因此，在设计并发控制的机制时要充分考虑协作过程中模式的改变，从而采用相应的并发策略。

(4)低通信流量

并发控制不应过快增加通信流量，否则将导致协同设计系统通信性能的急速降低。

(5)用户的行为习惯和心理状态

每个用户在设计过程中都具有独占性的特点，即不喜欢被别人打断或者总想保留自己的设计意图。为了避免操作冲突，应尽可能使用户自己察觉到彼此的所有操作。

作者简介：赵 蓉(1982－)，女，硕士研究生，主研方向：计算机网络，CAD；史维峰，教授；郑 超，助理工程师

收稿日期：2006-09-30 **E-mail：**bearloving00@163.com

(6)简单性和实用性（人机交互）。鉴于用户的多样性，应尽量避免设计一些不实用的功能，以致增加延迟，降低系统的响应程度。

2 博士 CAD 的并发控制机制

2.1 系统模型

协同设计系统常采用 2 种数据共享模型：集中式和全复制式[6]。集中式的所有同步都在server端实现，严格执行WYSIWIS（你见即我见），易于维持一致性的要求，但是对通信流量引起的网络延迟十分敏感。全复制式没有服务器和客户机之分，每个客户端都保有一个本地拷贝，每个用户实际上是对其本地拷贝进行操作，最大化地响应了用户的需求，但是难以保证各拷贝的一致性。为了更好地利用二者的优势，采用了集中式和全复制式混合方式部署共享数据：复制式针对图形设计过程，用户可以根据自己所需建立本地拷贝并对其进行独立操作，其他用户也可以看到其设计过程，每次的设计结果都真实地传到其他用户，在此过程中设计用户可以得到其他协作者的意见，实现了本地最优响应和交互性；集中式针对设计结果的统一性而言，将设计的中间版本或是最后版本提交至服务器进行统一管理，使得不同用户在特定时期访问的是共享对象的同一版本，便于下一步工作的开展，维护了共享对象的一致性。

如图 1 所示，客户端执行人机交互处理、操作管理和网络通信功能，其中用户并发操作管理器控制用户本地操作，修改本地共享对象拷贝，同时与服务器协同处理并发操作控制。服务器端对并发操作加以确认，形成最终版本并通知客户端更新共享对象视图。

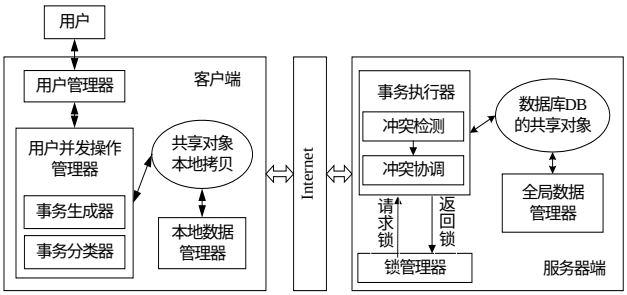

图 1 并发控制机制的网络体系结构

2.2 基于事务语义的并发方案

基于语义的并发控制方法主要分为 2 大类：基于事务语义的并发控制和基于抽象数据类型语义的并发控制[7]。前者利用了应用程序中的数据操作及事务语义，包括快照验证、嵌套事务、多层事务等方法；后者利用了模式定义中类型及操作方法的语义，包括可交换性、序列依赖、可恢复性等方法。由于协同事务的复杂性和持久性，本机制中采用嵌套事务来封装用户的各种操作，主要解决了以下几个方面的问题：事务分类及结构，事务提交，语义锁和冲突协调。

2.2.1 事务分类

现有的基于事务的并发控制方法对所有的事务不予分类，这就隐蔽了不同事务自身的语义信息，降低了系统的并发性。本文将事务分为“观察”事务和“行为”事务 2 种。前者封装的是用户的观察操作，它们的执行对数据库的状态不造成任何影响，而后者封装用户的各种修改操作，例如放大、旋转、复制等，行为事务的执行会改变数据库的状态。并发控制机制主要是针对行为事务展开。这样减少了观察事务和行为事务间的干扰，提高了系统的并发度。

2.2.2 事务结构

系统采用嵌套的事务结构开发事务内部的并发性。由于一个事务可能包含很多子事务，这些子事务也可能繁衍出更多的子事务，从而形成了一个任意深度的事务树，由此产生了嵌套事务总的结构模型，嵌套事务定义如下：

T 表示一个嵌套事务树上的事务构成的集合。

定义 1 对于任意的事务 t1, t2 属于 T，P(t1,t2)表示 t1 为 t2 的父事务，t2 为 t1 的子事务；Fp(t1,t2)表示 t1 是 t2 的祖先事务，t2 是 t1 的后代事务。

定义 2 存在 t 属于 T，若(P(t,t2) 且 P(t,t2))，则称 t1 和 t2 为兄弟事务，记为 B(tl,t2)。

定义 3 存在 t 属于 T，若 Fp(t,t2)和 Fp(t,t2)同时存在，则称 t1 和 t2 为同一家族事务。

定义 4 对于任意一个事务 ti 属于 T，若不存在 P(ti,t)，则称 t 为顶层事务；若不存在 P(t,ti)，则称 ti 为叶事务。

如图 2 所示，本文用一个事务树来表示嵌套事务的构成，用树中的节点表示事务，用节点间的连线来表示相关事务间的父子关系。在这个事务树中，语义被加入到协同事务的顶层，由此一个嵌套事务树可以被认为是由根事务作为外部接口，由根事务的所有子事务的子事务树组成的集合。

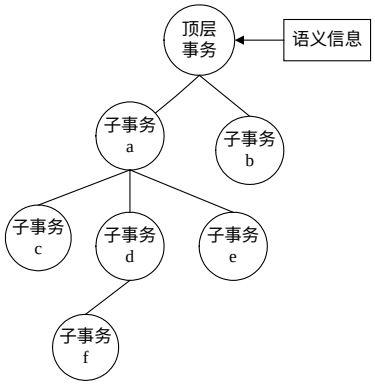

图 2 嵌套事务树

2.2.3 事务提交

在实际的设计过程中，行为事务间的并发程度并不高，因此采用乐观方法，即在事务提交之前，用户对本地拷贝的本地操作无须等待直接响应，用户之间可以看到彼此的行为但是不能相互干扰，只有在事务提交时才进行冲突检测。系统收到不同的用户端所递交的事务树，执行冲突协调器来管理事务的并发：

(1)If Trans==Tr

博士 CAD 系统能快速响应所有的观察事务系统，服务器将最新版本的视图传递给客户，在观察事务的执行过程中，若共享对象发生了变化，系统会及时通知用户更新视图。

(2)If Trans==Tw

根据语义锁上的语义信息，如果是同一对象的不同属性的操作，进行版本合并；如果是同一对象同一属性的相同操作，保留其中一个版本而舍弃其他的版本；如果是同一对象同一属性的不同操作，进行冲突协调，可以采用多种方法，如时间戳机制或是语义可交换性方法。

2.2.4 语义锁

在博士 CAD 系统中，传统的数据库管理采用了事务锁和数据锁。这 2 种锁都不用于锁定对象，而是用于冲突检测。事务锁用于表明有应用层使用的执行具体功能的对象上的语义信息；数据锁用于表明包含几何信息的对象上的语义信息。

对于嵌套事务而言，事务锁采用下面的规则：对每个事务的执行而不是对每一个原子操作请求一个锁。当一个子事务执行完之后，如果父事务被异常终止，则子事务持有的锁被释放；如果其父事务存在并被提交，锁被父事务继承，在这种情况下，锁被父事务或祖先事务保留，在保留锁层次之外的事务不能获得其任意冲突锁，但保留者的子事务能够得到，只有当顶层事务执行结束后才释放它的锁。

数据锁作用于共享的数据对象上，标识出共享对象最终的语义操作，便于系统向服务器提交最新版本时判断是否进行版本的合并。

2.2.5　冲突协调

博士 CAD 系统提供了 3 种方式协调“ 行为 ”事务间的冲突，并且这 3 种方式是按照顺序原则执行的，即当系统监测到冲突时，首先系统自身进行协调，若不能消除冲突则由冲突双方进行协商，如果还不能达成一致，则交由第三方仲裁。

(1)系统自动协调

在这种模式下，系统根据预制的规则，自动处理冲突。默认的冲突处理规则可以有多种选择，例如对于事务锁的冲突可采用事务语义的可交换性原则或是时间戳的方法来协调事务的执行顺序；对于数据锁的冲突可以根据抽象数据类型的可交换性矩阵进行协调。此外，用户还可以自定义默认的冲突处理规则。

(2)冲突双方协商

当系统不能自动协调冲突时，会启动及时通知功能告知用户进行协商，用户可以使用系统中的实时对话功能进行交谈。

(3)第三方仲裁

当冲突双方不能达成一致时，可以向上级领导汇报，由上级领导决定最终的方案。

2.3　系统的实现

图 3 为博士 CAD 系统的用户登录界面,通过输入用户名和密码进行用户身份确认，输入服务器和端口号与服务器建立连接。

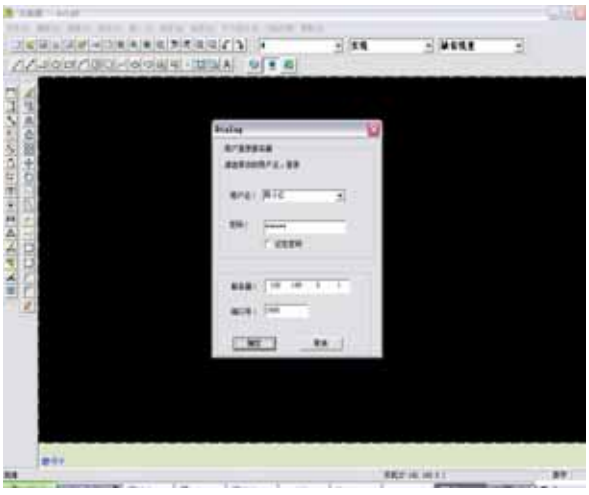

图 3　客户-服务器建立连接

连接成功后，一方面，各个用户可以根据自己所需建立本地拷贝并对其进行独立操作，提高了系统的响应性；另一方面，其他用户也可以看到其设计过程，设计用户可以交流意见，提高了系统的交互性。如图 4 所示，用户 a 可以察觉到用户 b 对共享对象的操作。最后将设计的中间版本或最终版本提交至服务器进行统一管理,保证了共享对象的一致性。

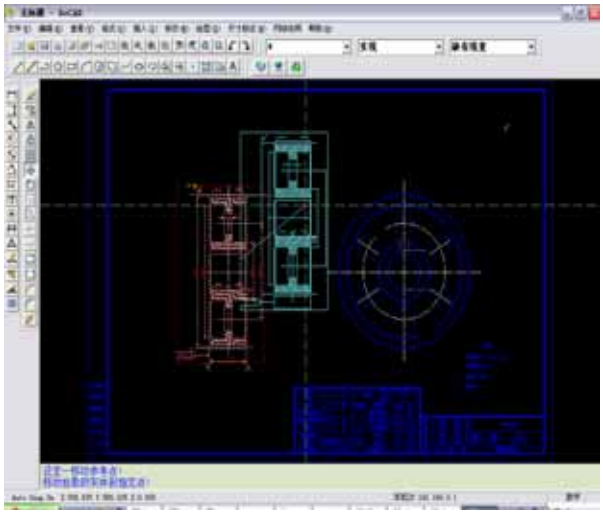

图 4　客户端的共享视图

3　总结与展望

一致性、并发性和响应性是并发控制机制中 3 个至关重要但又相互冲突的需求。本文针对博士 CAD 系统提出的并发控制机制较好地实现了协同操作的并发控制，提高了数据的一致性。本机制的最大特点在于充分利用了图形系统中丰富的语义信息，较快地响应本地请求，同时利用灵活的冲突协调最大化并发性，从而有效地在三者之间找到了平衡点。

参考文献

1 张秉森, 王　钰. 计算机辅助设计教程[M]. 北京: 清华大学出版社, 2005: 13-25.

2 Ellis C A, Gibbs S J. Concurrency Control in Groupware Systems[C]// Proceedings of 1989 ACM SIGMOD International Conference on the Management of Data, Portland, Oregon. 1989: 387-423.

3 Ellis C A, Rein G L. Groupware Some Issues and Experiences[J]. Communications of the ACM, 1991, 31(1): 32-68.

4 Liu F, Luh P, Moser B. Scheduling and Coordination of Distributed Design Projects[J]. Annals of CIRP, 1998, 47(1): 111-114.

5 Brayner A, Harder T, Ritter N. Semantic Serializability: A Correctness Criterion for Processing Transactions in Advanced Database Applications[J]. Data & Knowledge Engineering, 1999, 31(1).

6 Bagley J D. The Behavior of Adaptive Systems which Employ Genetic and Correlation Algorithms[J]. Dissertation Abstracts International, 1967, 28(12): 256-287.

7 Jun W, Gruenwald L. Semantic-based Concurrency Control in Object-oriented Databases[J]. Jour. of Object-oriented Programming, 1998, 10(8): 194-223.